Engineering Joy

Marc Hassenzahl, Andreas Beu,
and Michael Burmester, *User Interface Design GmbH*

Joy of use has become a buzzword in user interface design although serious attempts at defining it remain sparse. The authors propose systematic methods of taking into account one of its main determinants, hedonic quality, and its complex interplay with usability and utility as a step toward truly engineering the user experience.

Over the last 30 years usability has become an acknowledged quality aspect of a wide variety of technical products, ranging from software to washing machines. The concept of usability has been accompanied by the assumption that usability can be engineered. Clearly, the aim of usability engineering is to devise processes to assure that products are usable.[1] Although usability engineering is still a first-generation field, some of its basic ideas are widespread and reach back to practices in industrial design[2] and the "golden rules" of John D. Gould and Clayton Lewis.[3] The key principles include: the analysis of a product's intended context of use (user skills and needs, task requirements, and physical, social, and organizational context) at the beginning of development; user participation throughout the development process; early prototyping; usability evaluation; and continuous revision based on evaluation data.[4] As the field's methods have evolved, they have changed the concept of usability from a narrow product-oriented quality attribute to the broad concept of quality of use, that is, "that the product can be used for its intended purpose in the real world."[5]

However broad the latest definition of usability is, it recently acquired a new associate, the so-called *joy of use*. The notion of joy of use is instantly appealing, though its actual meaning is hard to grasp. In 1997, Bob Glass said, "If you're still talking about ease of use then you're behind. It is all about the joy of use. Ease of use has become a given—it's assumed that your product will work." However, joy of use is extremely hard to define. As Glass said, "You don't notice it, but you're drawn to it."[6]

The way the term joy of use is employed in general computer and human–computer-interaction literature reveals three perspectives on the issue:

- *Usability reductionism* supposes that joy of use simply results from usable software and that the answer to the question of how to design for enjoyment is already known. The only problem is how to put usability engineering into practice. So, joy of use appears to be just a natural consequence of excellent usability. This perspective discounts the qualitative differences between simply doing a job and enjoying doing a job.

- *Design reductionism* reduces joy of use to a quality that graphical and industrial designers add to software. Designers "possess the [..] skills that combine science and a rich body of experience with art and intuition. Here is where 'joy' and 'pleasure' come into the equation: joy of ownership, joy of use."[7] This perspective assumes that joy of use is concerned more with superficial than with deeper qualities, such as interaction style and functionality. Therefore, it fails to acknowledge the complex interplay of visual, interactional, and functional qualities.
- *Marketing reductionism* reduces joy of use to a simple marketing claim. This opinion is comparable to the perception of usability at its advent: user-friendliness. It is mainly a claim with no substance.

None of these perspectives seems satisfactory. Given that our aim is to design enjoyable software systems, we should take the analysis of joy of use as seriously today as we took ease of use yesterday.

Why consider enjoyment in software design?

The most basic reason for considering joy of use is the humanistic view that enjoyment is fundamental to life. Glass said, "I believe that products of the future should celebrate life! They should be a joy to use. I predict that joy of use will become an important factor in product development and product success."[8]

Although some might readily agree with this view, others object on the grounds that there is a radical difference between leisure and work. The former calls for idle enjoyment, the latter for concentrated work. Erik Hollnagel has voiced a perspective against connecting emotions (such as enjoyment) with software design.[9] He argues that human–computer interaction is basically about efficiency and control, and that emotions interfere with these attributes. For example, one might make decisions based on highly subjective, emotional criteria not suitable for the rational work domain. Somewhat cynically he states, "Affective interfaces may serve a therapeutic purpose, [to] make the user feel better."[9] We, on the other hand, believe that the users' well being always matters, especially in a work domain. Technology acceptance research has demonstrated the positive effects of perceived enjoyment or fun in work settings. For example, in one study, when people enjoyed a software product, their acceptance and satisfaction increased.[10] The impact of user-perceived enjoyment on acceptance (one important determinant of productivity) nearly equaled that of user-perceived usefulness. In another example, providing an enjoyable workplace for call center agents was assumed to sustain the quality of customer service and even increase it throughout the day.[11] So, in certain work positions (those requiring "emotion work," such as a call center agent or hotel receptionist), enjoyment might have an important effect on work quality instead of solely serving a therapeutic purpose. There are other cases where joy or fun plays a role as a software requirement, for example, where learning is the main system function.[12]

Acknowledging the positive effects of enjoyment does not necessarily imply knowledge of how to design enjoyable software. The primary question is: What do we actually have to do to design for joy of use? Advocates of usability reductionism would answer: "Nothing! Just provide useful functionality so the users can easily operate the software." This view emphasizes software's role as a tool for accomplishing a task and focuses on task-related qualities (usability and utility). It has been shown, however, that *hedonic* qualities, that is, task-unrelated qualities, can also play a role. For example, including hedonic components (task-unrelated graphics, color, and music) increased an information system's enjoyment and usage.[13] Similarly, the perception of hedonic quality (task-unrelated aspects such as novelty or originality) substantially contributed to the overall appeal of software prototypes for process control tasks[14] and different visual display units—a standard CRT, a LCD Flat-screen, and a computer image projected on the desktop.[15] Both studies demonstrate that task-related and -unrelated quality aspects seem to compensate for each other from the user's perspective. In other words,

The most basic reason for considering joy of use is the humanistic view that enjoyment is fundamental to life.Pullquote

Designers need to introduce novelty with care. User interfaces that are too novel and unfamiliar are likely to evoke strong adequacy concerns instead of hedonic quality perceptions.

extremely usable but tedious software might be as appealing to a user as an extremely unusable but thrilling one. Thus, exploring and understanding the nature of hedonic quality as a software requirement and, furthermore, the dependence between hedonic quality and task-related quality (utility and usability) is a valuable road toward designing for joy of use.

The driving forces behind hedonic quality

The definition of hedonic quality as task-unrelated quality is clearly too broad to guide design. The driving forces behind the scene might be more specific qualities such as the need for novelty and change and the need to communicate and express oneself through objects.[16]

Though these may not be the only needs, they exemplify the two-edged nature of the forces behind hedonic quality. One part is directed inward, concerning the individual's personal development or growth; the other part is directed outward, concerning social and societal issues. If users perceive a software product as potentially capable of satisfying the need for personal growth and status, it has hedonic quality. The perception of hedonic quality (or lack of it) will affect the user's preference for a given software product.

Need for novelty and change

Several areas of research have found evidence of a general human need for novelty and change. Daniel Berlyne, for example, states that our central nervous system is designed to cope with environments that produce a certain rate of stimulation and challenge to its capacities.[17] We reach best performance at a level of optimal excitement, where neither overstimulation nor monotony are present. The same notion exists in Mihaly Csikszentmihalyi's optimal experience concept.[18] Optimal experience or flow describes the state when somebody is completely wrapped up in an activity. The crucial determinant is the certainty that the activity is challenging but attainable—it has the optimal level of excitement. In a home automation system evaluation study,[19] we found that individuals with a technical job background reported the system to be of low hedonic quality compared to individuals with a non-technical job background. We suppose that technically educated individuals are more likely to possess knowledge about existing home automation system functionality, so they don't find the functionality excitingly new. Conversely, for individuals with non-technical job backgrounds, the system provided the means to do things they could not do before. Indeed, the focus during system design was on usability and visual design rather than on adding exciting new functionality. In this aspect, the experiment did not address the technically oriented users' need for challenge and stimulation. Strikingly, taking the need for novelty and change into account might unavoidably imply a reduction of usability. Usability and joy of use might be partially incompatible, because the former requires consistency and simplicity, whereas the latter requires surprise and a certain amount of complexity.[20]

Designers need to introduce novelty with care. User interfaces that are too novel and unfamiliar are likely to evoke strong adequacy concerns instead of hedonic quality perceptions.[21] What is needed is a way to determine an optimal level of novelty.

Need to communicate and express oneself through objects

This need addresses the social dimension of using software. Robinson states that the artifacts people choose to use can be interpreted as statements in an ongoing "dialog" people have with other people in their environment.[22] We should not underestimate the fact that using a product can evoke a state of importance. Being an expert at something that others do not understand, being able to afford something that others cannot afford, or possessing something that others desire are strong driving forces. To give an anecdotal example from our experience: The home automation system mentioned earlier had a user interface designed to be as non-intimidating as possible in order to encourage use by people with low computer expertise. The strategy succeeded. Usability tests with elderly, non-computer-literate individuals showed an astonishingly low number of severe usability problems. However, one participant with a more sophisticated technical background complained about the visual design. He said it

Figure 1. Semantic differential for measuring hedonic quality.

Outstanding	☐	☐	☐	☐	☐	☐	☐	Second-rate
Exclusive	☐	☐	☐	☐	☐	☐	☐	Standard
Impressive	☐	☐	☐	☐	☐	☐	☐	Nondescript
Unique	☐	☐	☐	☐	☐	☐	☐	Ordinary
Innovative	☐	☐	☐	☐	☐	☐	☐	Conservative
Exciting	☐	☐	☐	☐	☐	☐	☐	Dull
Interesting	☐	☐	☐	☐	☐	☐	☐	Boring

looked like a "children's book" and that his friends would laugh at the system's apparent lack of professionalism. Thus, designers need to develop user interfaces with status needs in mind.

Techniques for engineering hedonic quality

There is an explicit difference between knowing that hedonic quality could play a role in designing interactive systems and actively accounting for it. The latter requires practical methodical support for both design (techniques for gathering and analyzing hedonic requirements) and evaluation (metrics and techniques to measure hedonic quality). As long as you understand their advantages and disadvantages, the following techniques can fit into a design process for interactive systems.

A semantic differential for measuring perceived hedonic quality

A well-known technique for measuring how people perceive and evaluate objects is the semantic differential. The differential we employ comprises seven pairs of adjectives that characterize hedonic quality's presence or absence, evaluated on a seven-point rating scale. Each pair of extremes corresponds to opposing adjectives, such as good–bad, interesting–boring, or clear–confusing. Once the participants rate the software on each characteristic, we calculate a hedonic quality "value" by summing or averaging the ratings. Figure 1 shows the semantic differential we typically use for measuring hedonic quality[14,15,19] (note that the verbal anchors used in these studies were originally in German).

We can apply the differential throughout the design process for interactive systems, from the evaluation of early mock-ups or prototypes to fully operational systems. It has various advantages: the usability engineer does not require a special training for using the differential, the participants can quickly and easily fill it in, and the statistical analysis is straightforward. The characteristics are high-level and deal with subjective user perceptions—that is the "quality is in the eye of the beholder." This makes the differential applicable to various software products without needing to adjust it to the product's special features.

The differential's general applicability is also one of its major disadvantages. Although it can show the extent to which users regard a piece of software as hedonic, the underlying reasons (determinants of hedonic quality or lack thereof) remain unknown. However, it is exactly the understanding of the underlying reasons that proves to be most important for stimulating and improving a software product design, especially when it comes to a premature construct such as hedonic quality. Another problem associated with the nature of hedonic quality is the solely operational definition that the differential provides. Without a theoretically solid definition, there is always the danger of missing an important facet of hedonic quality.

Repertory grid technique

A way to overcome the differential's problems is the *repertory grid technique* (RGT).[23,24] Georg Kelly assumes that individuals view the world (persons, objects, events) through personal constructs. A personal construct is a similarity–difference dimension comparable to a semantic differential scale. For example, if you perceive two software products as being different, you might come up with the personal construct "too colorful—looks good" to name the opposed extremes. On the one hand, this personal construct tells something about you, namely that too many colors disturb your sense of aesthetics. On the other hand, it also reveals information about the products' attributes. From a design perspective, we are interested in differences between software products rather than differences in individuals, so we focus on what the personal constructs of a group of users might tell us about the products they interact with.

RGT deals with systematically extracting personal constructs. It consists of two steps: construct extraction and product rating. For construct extraction, we present individuals with a randomly drawn triad from a software products set, marking the "design space" we are interested in. They must an-

Table 1

Example constructs[21]

	Pole A	Pole B
1	Does not take the problem seriously	Takes the problem seriously
2	Inappropriately funny	Serious
3	Non expert-like	Technically appropriate
4	All show, no substance	Technology-oriented
5	Playful	Expert-like
6	Has been fun	Serious (good for work)

swer in what way two of the three products are similar to each other and different from the third. This procedure produces a construct that accounts for a perceived difference. The people then name the construct (for example, playful–serious, two-dimensional–three-dimensional, ugly–attractive) indicating which of the two poles they perceive as desirable (having positive value). We repeat the process until no further novel construct arises. The result is a semantic differential solely based on each individual's idiosyncratic view. In the product rating step, we ask people to rate all products on their personal constructs. The result is an individual-based description of the products based on perceived differences.

Designers can apply RGT in various forms throughout the user-centered design process for interactive systems. A promising application might be "diagnostic benchmarking," for example, comparing your current, future, and competitors' websites.[25] We recently used RGT to explore the differences between design studies for control room software resulting from a parallel design session.[21] Table 1 shows some example constructs from this study. These constructs illustrate that the participants were concerned about the adequacy of some of the designs for a work domain. At least two different views became apparent: some participants believed that control room software must look serious (constructs 1–4), maybe to induce trustworthiness (constructs 1 and 4) and perceived control (construct 3). Other participants acknowledged the hedonic quality of some designs (and the enjoyment they derived from them) (construct 5) but emphasized the dichotomy between leisure and work (construct 5 and 6). This illustrates the rich information that we can obtain by RGT. In the study just mentioned, we extracted 154 constructs from 10 participants, covering topics such as quality of interaction and presentation, hedonic quality, and adequacy concerns (participants' belief about the extent to which the prototype is suitable for the task).

RGT has a number of advantages:

- It is a theoretically grounded[23] and structured approach, but nevertheless open to each participant's individual view. The focus on individual (personally meaningful) constructs is a clear advantage over the semantic differential. The differential can only measure what we define to be hedonic quality. In other words, the participants must use our constructs (the scales we provide), regardless of whether they are meaningful to them and cover the topics relevant to them.
- RGT is more efficient than comparable open approaches such as unstructured interviews. Focusing on the personal constructs as data denotes a significant reduction in the amount of data to be analyzed compared to transcribing and analyzing unstructured interviews. This is especially important in the context of parallel design, or benchmarking, when many alternatives are under study.
- Personal constructs have the potential to be design-relevant. The whole approach is likely to generate different views on software products, embodying various individual needs and concerns in relation to the product and its context of use. This again is something the semantic differential neglects.
- The basic method lends itself to the application of almost any set of software products.

The method's main disadvantage is the amount of effort invested. While we can use the semantic differential as an add-on to a regular usability test or as an online questionnaire, an RGT study is a self-contained method in which the experimenter needs considerable training. Another disadvantage is that RGT relies on comparisons and its application is therefore confined to situations where at least four alternatives are available.

Shira interviewing

Rainer Wessler, Kai-Christoph Hamborg (University Osnabrück), and Marc Hassenzahl have recently developed a new analysis method that avoids at least the multiple-alternative problem associated with the RGT. *Structured hierarchical interviewing for requirement analysis* (Shira)[26] is an interviewing technique that seeks to explore the meaning of product attributes such as "controllable," "simple," "impressive," or "innovative" for a specific software application in a specific context of use.

Shira starts from a pool of attributes covering usability aspects (such as "controllable") and hedonic qualities (such as "innovative"). We first introduce participants to a possible software application and its intended context of use—for example, a home automation system or software for writing one's diary. In a second step, we ask the participants to select an attribute from the pool that is important to them with regard to the software (Figure 2 shows an example dealing with the attribute "simple"). Starting from the attribute, they then list software features that would justify attaching that attribute. By repeatedly answering questions such as "what makes a home automation system seem innovative to you," they will generate a list of features that contain context and the attribute's software-specific determinants (for example, "user-friendly" and "not patronizing"). The resulting list comprises the *context level*. In the third step, the participants must produce recommendations for each entry in the context level suggesting how the actual design could address the feature (for example, "adaptive, learning, intelligent system that works more or less independently and requires little attention from the user"). We call this the *design level*. The result is a hierarchical, personal model of attributes that are important to the participants with regard to specified software product, what these attributes actually mean to them, and how they can be addressed by the design.

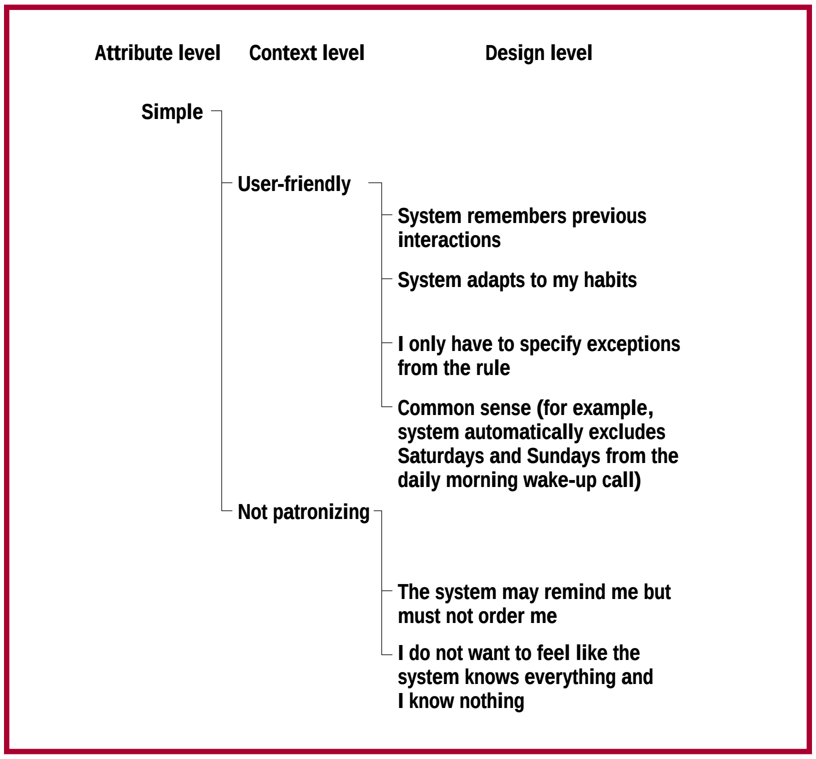
Shira is a systematic way to get in-depth data and detailed insights into an individual's expectations of a specified software system. Its hierarchical representation facilitates getting a better idea of central and peripheral aspects (attributes, features, or design recommendations). In particular, Shira has the power to gather hedonic requirements by using hedonic attributes as stimulation. By integrating personal models into a group model, we obtain a rich body of information about the system's design space.

Shira is especially suited to gather information at early stages of the design process for interactive systems. However, it might also be possible to evaluate software at a later stage regarding how it fits the user's expectations.

Shira is still at an early development stage. It is too early to assess advantages and disadvantages. However, from our preliminary experience with the technique, it seems to provide detailed design-relevant data in a structured form that facilitates interpretation and integration of multiple personal perspectives.

Usability and utility are basically about how well software supports people in getting their jobs done. However, task-unrelated qualities can play a crucial role. Traditional usability engineering methods are not adequate for analyzing and evaluating hedonic quality and its complex interplay with usability and utility. The techniques we have suggested might significantly broaden usability engineering practices by shifting the focus to a more holistic

Figure 2. Portion of a personal model gathered by using Shira.


Attribute level
Context level
Design level
Simple
User-friendly
System remembers previous interactions
System adapts to my habits
I only have to specify exceptions from the rule
Common sense (for example, system automatically excludes Saturdays and Sundays from the daily morning wake-up call)
Not patronizing
The system may remind me but must not order me
I do not want to feel like the system knows everything and I know nothing

perspective on human needs and desires. In the future, we might see usability engineering evolving toward more complete user experience design—one that encompasses the joy of use.

Acknowledgments

This article was partly funded by the German Ministry for Research (BMBF) in the context of INVITE (01 IL 901 V 8). See www.invite.de for further information. We are grateful to Uta Sailer for her helpful comments on an earlier draft of the article.

References

1. J. Nielsen, *Usability Engineering*, Academic Press, Boston, San Diego, 1993.
2. H. Dreyfuss, *Designing for People*, Simon & Schuster, New York, 1995.
3. J.D. Gould and C.H. Lewis, "Designing for Usability: Key Principles and What Designers Think," *Comm. ACM*, vol. 28, no. 3, Mar. 1985, pp. 300–311.
4. *Human-Centred Design Processes for Interactive Systems*, ISO-13407: 1999.
5. N. Bevan, "Usability is Quality of Use," *Proc. HCI Int'l 95*, Lawrence Erlbaum Associates, Mahwah, N.J., 1995 pp. 349–354.
6. www.sun.com.au/news/onsun/oct97/page6.html (current Dec. 00).
7. D.A. Norman, *The invisible computer*, MIT Press, Cambridge, Mass., 1998.
8. B. Glass, "Swept Away in a Sea of Evolution: New Challenges and Opportunities for Usability Professionals," *Software-Ergonomie '97. Usability Engineering: Integration von Mensch-Computer-Interaktion und Software-Entwicklung*, R. Liskowsky, B.M. Velichkovsky, and W. Wünschmann, eds., B.G. Teubner, Stuttgart, Germany, 1997, pp. 17–26.
9. E. Hollnagel, "Keep Cool: The Value of Affective Computer Interfaces in a Rational World," *Proc. HCI Int'l 99*, vol. 2, Lawrence Erlbaum Associates, Mahwah, N.J., 1999, pp. 676–680.
10. M. Igbaria et al., "The Respective Roles of Perceived Usefulness and Perceived Fun in the Acceptance of Microcomputer Technology," *Behaviour & Information Technology*, vol. 13, no. 6, 1994, pp. 349–361.
11. N. Millard et al., "Smiling through: Motivation at the User Interface," *Proc. HCI Int'l '99*, vol. 2, Lawrence Erlbaum Associates, Mahwah, N.J., 1999, pp. 824–828.
12. S.W. Draper, "Analysing Fun as a Candidate Software Requirement," *Personal Technology*, vol. 3, no. 1, 1999, pp. 1–6.
13. N. Mundorf et al., "Effects of Hedonic Components and User's Gender on the Acceptance of Screen-Based Information Services," *Behaviour & Information Technology*, vol. 12, no. 5, 1993, pp. 293–303.
14. M. Hassenzahl et al., "Hedonic and Ergonomic Quality Aspects Determine a Software's Appeal," *Proc. CHI 2000 Conf. Human Factors in Computing*, ACM Press, Addison-Wesley, New York, 2000, pp. 201–208.
15. M. Hassenzahl, "The Effect of Perceived Hedonic Quality on Product Appealingness" *Int'l J. Human–Computer Interaction*, submitted for publication.
16. R.J. Logan et al., "Design of Simplified Television Remote Controls: A Case for Behavioral and Emotional Usability," *Proc. 38th Human Factors and Ergonomics Soc. Ann. Meeting*, 1994, pp. 365–369.
17. D.E. Berlyne, "Curiosity and Exploration," *Science*, vol. 153, 1968, pp. 25–33.
18. M. Csikszentmihalyi, *Beyond Boredom and Anxiety*, Jossey-Bass, San Francisco, 1975.
19. M. Hassenzahl et al., "Perceived Novelty of Functions—A Source of Hedonic Quality," *Interfaces*, vol. 42, no. 11, p. 11, 2000.
20. J.M. Carroll and J.C. Thomas, *Fun. SIGCHI Bull.*, vol. 19, no. 3, 1988, pp. 21–24.
21. M. Hassenzahl and R. Wessler, "Capturing Design Space from a User Perspective: The Repertory Grid Technique Revisited," *Int'l J. Human-Computer Interaction*, vol. 12, no. 3/4, pp. 441–459.
22. L. Leventhal et al., "Assessing user interfaces for diverse user groups: evaluation strategies and defining characteristics," Behaviour & Information Technology, vol. 15, no. 3, 1996, pp. 127–137, and references therein.
23. G.A. Kelly, *The Psychology of Personal Constructs*, vols. 1–2, Norton, New York, 1955 (reprinted by Routledge, 1991).
24. F. Fransella and D. Bannister, *A Manual for Repertory Grid Technique*, Academic Press, London, 1977.
25. M. Hassenzahl and T. Trautmann, *Analysis of Web Sites with the Repertory Grid Technique*, submitted for publication.
26. R. Wessler et al., *Orientation, Understanding and Decision-Making—a User-Centred Approach to Guide the Design of Prototypes*, submitted for publication.

About the Authors

Marc Hassenzahl studied psychology and computer science at the Technical University, Darmstadt. For the past five years he has worked as a freelance usability consultant for clients such as the Federal Statistical Office in Germany and as a usability engineer at Siemens Corporate Technology – User Interface Design. Currently, he is working at User Interface Design GmbH in Munich. He is involved in projects ranging from usability evaluation of automation software to user interface design for computer chip design tools. His research interests are appealing user interfaces, especially hedonic qualities, and related new analysis and evaluation techniques. Contact him at User Interface Design GmbH, Dompfaffweg 10, 81827 Munich, Germany; marc.hassenzahl@uidesign.de.

Andreas Beu studied mechanical engineering at the University of Stuttgart. In the past six years, he has led numerous user interface design projects that have employed a user-centred design approach, mainly for complex industrial applications. He worked as a usability engineer at GSM GmbH, a spin-off company of the Fraunhofer-Institute for Industrial Engineering (IAO) in Stuttgart, and at Siemens Corporate Technology – User Interface Design. Since April 2000, he is working at User Interface Design GmbH in Munich. He is interested in user interface design for small displays, wearable computers, and augmented reality systems. Contact him at User Interface Design GmbH, Dompfaffweg 10, 81827 Munich, Germany; andreas.beu@uidesign.de.

Michael Burmester studied psychology at the University of Regensburg in southern Germany. He started his career as a researcher at the Fraunhofer Institute for Industrial Engineering (IAO) in Stuttgart. In 1997, he joined Siemens Corporate Technology – User Interface Design as a usability consultant and researcher for usability engineering. Since March 2000, he has been head of the Munich office of User Interface Design GmbH, a software and usability consultancy company. Results and experiences of his research and consultancy work are published in over 40 scientific and technical papers. Contact him at User Interface Design GmbH, Dompfaffweg 10, 81827 Munich, Germany; michael.burmester@uidesign.de.