The Design and Evaluation Process

Joseph Jofish Kaye, Jarmo Laaksolahti, Kia Höök, and Katherine Isbister

Abstract The purpose of this chapter is to describe the design and evaluation process in the light of affective interaction. With a starting point in user-centred design we will explore what additional problems or opportunities become important when designing for affective interaction with computer systems. This chapter also provides a historical background to HCI ending with what is sometimes named the third wave of HCI – that is, designing for aesthetic, emotional experiences with and through technology.

1 User-Centred Design

Software design has traditionally been viewed as a linear process in which development proceeds stepwise from an application description, through a well-defined requirement specification and system design based on the specification, to a finished product. However, this model of software development (also known as the waterfall model, see e.g. Sommerville, 1992) has a number of problems associated with it. For instance, its linear structure does not allow for changing requirement specifications or system designs once they have been decided upon. Furthermore the requirement gathering and system design activities of the waterfall model seldom include representatives of the users that in the end will be using the system, meaning that their needs and wants are not taken into account. Instead the product, once finished, is imposed upon the users who are expected to adapt to the new situation and accommodate the system into their situation.

In contrast, user-centred design (UCD) is an iterative approach to systems development that puts users and questions regarding their needs, wants, and usage situations in focus. User-centred design is not a method per se but rather a design

J.J. Kaye (✉)
Nokia Research Center, Palo Alto, CA, USA
e-mail: jofish.kaye@nokia.com

philosophy that can have many incarnations, all sharing the same underlying principles (Preece et al., 1994):

- To make user issues central in the design process
- To involve users in the design process
- To carry out early testing and evaluation
- To do design iteratively

UCD recognizes the fact that users and usage contexts differ and that, consequently, systems should be designed with that in mind. Rather than forcing users to adapt to accommodate a system, the system is designed from the start to suit its users. The basic idea that systems should be designed based on what users need and want is a notion that permeates UCD methods. In practice this means paying extensive attention to users' needs, wants, and limitations in each step of the design process. It requires designers to analyse and foresee how users are likely to use a system and to test results in real-world tests with actual users. Such testing is necessary as it is often difficult for designers to understand what experiences users of their design go through.

The basic principles of UCD have been standardized in the ISO standards, 'ISO 13407:1999 Human-centred design process', and developed in 'ISO TR 18529: Ergonomics – Ergonomics of human–system interaction – Human-centred lifecycle process description'. These standards aim to provide guidance for achieving quality in use by incorporating UCD activities throughout the life cycle of interactive computer systems.

Figure 1 illustrates the four human-centred design activities that ISO 13407 suggests should take place during a system development project. First studies are performed to gain insight about contexts in which the system will be used and requirements in terms of users' needs, wants, and practices. This step may also include specifying organizational requirements that are of importance in a workplace. After analysing the results of the studies, conclusions and design ideas drawn from it are used to guide and fuel the design work. Eventually a design prototype is produced, which is tested with actual users to assess how well it meets the design intentions, which in turn are based on users' needs and wants. The produced prototypes can have various forms including scenarios and sketches showing broad functionality, paper- or screen-based mock-ups that simulate aspects of functionality, to fully working prototypes that represent the full functionality of the system. This development cycle is iterated several times during the design process allowing misconceptions to be clarified – and improvements based on user feedback to be incorporated into the design along the way.

While ISO 13407 describes the general outline of a UCD development cycle, there is great diversity in how the principles of UCD have been implemented. Consequently UCD processes differ along a number of dimensions, such as level of user involvement and control in the design process or methods used in the different steps. Methods such as participatory design advocate user control and involvement in the whole design process and often incorporate end-users into the design

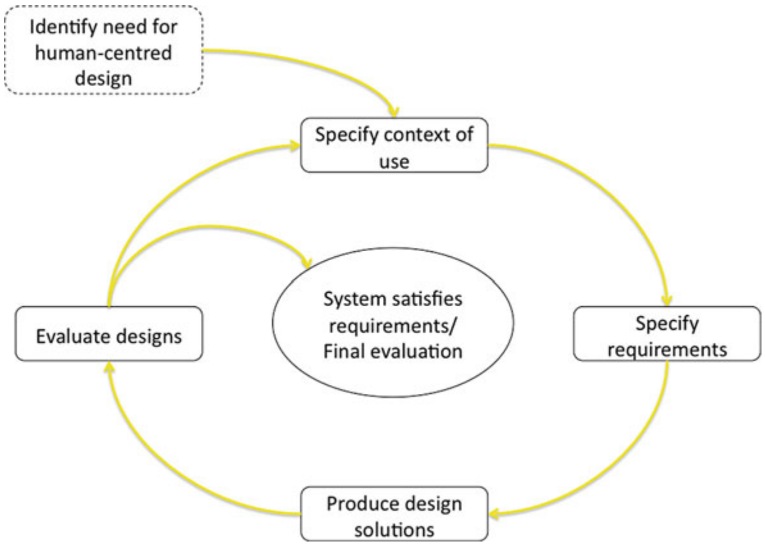
Identify need for human-centred design
Specify context of use
System satisfies requirements/ Final evaluation
Evaluate designs
Specify requirements
Produce design solutions

Fig. 1 ISO 13407 Human-centred design process: design activities

team (Ehn, 1992), while other methods are content with bringing in end-users as test subjects during the development cycle. Some methods use ethnography as a means to intimately understand the end-users situation, while others are content with using market research as a starting point. The 'right' combination of methods to use is ultimately determined by the characteristics and available resources for each development project.

With the increasing number of applications, such as computer games, whose primary purpose is to provide experiences rather than make us more efficient at tasks, UCD is faced with new challenges. Instead of task-completion time or productivity the important aspects for such applications are the experiences they provide and the emotions they evoke. While traditional HCI tools are well equipped for answering the first type of questions, they are less well equipped for answering the second type of questions. As of yet we do not know how to capture subjective experiences and affective involvement in such a way that it can provide feedback into the design process. Within the research community, this has led to an increasing interest in finding methods for designing and evaluating non-task-related aspects of application usage (Blythe et al., 2004; McCarthy and Wright, 2004). A UCD perspective and development philosophy is a good starting point as such but it will have to be extended to include methods that can specifically deal with designing for qualities such as flow, pleasure, fun, excitement, or fear while at the same time keeping users involved in the process.

The bulk of this chapter is concerned with such methods and will describe them and situate them in the general UCD cycle above. But before we present any methods, we will look at a historical perspective on design and evaluation from a human–computer interaction view in Sect. 2.2 to understand its origins and future directions, and have a look at issues related to design and evaluation outside research in Sect. 2.3 and finally discuss the particular challenges associated with designing for affective interaction in Sect. 2.4.

2 Evaluation: A Historical Perspective

The methods and approaches of evaluation in HCI have changed along with the field of computing itself. In particular, as different sets of users have different limiting factors on their ability to accomplish their tasks using computers, they require different forms of evaluation necessary to recognize and evaluate where those limitations occur. By tracing the history of these varying forms of evaluation in the field, we can understand how the current approaches to evaluation in HCI have come to be and further understand prospects for the future evaluation in the field, and in particular the role of affect in that evaluation.

2.1 Evaluation by Engineers

Computers were originally experienced as fundamentally mechanical and electrical machines. Vacuum tubes broke on a regular basis, and evaluation in the sense that it is used today in HCI did not exist – the closest in the published papers of the time is 'reliability' – because the biggest factor in determining how well a computer worked was how long it would do so. For example, John von Neumann brings up reliability of vacuum tubes and memory storage seven times in his draft report on the EDVAC in 1945 (von Neumann, 1945). This is an excellent example of a limiting factor on users accomplishing their tasks: limitations like the difficulty of entering commands were dwarfed by the simple fact that the computer itself would frequently cease to function after half an hour of continuous use.

2.2 Evaluation by Computer Scientists

By the 1950s, evaluation of computers was being done by computer scientists. The users were themselves programmers, although they were now freed of the necessity of understanding how the hardware itself worked, and had a level of abstraction provided for them by programming languages, assemblers, compilers, and debuggers. 'Reliability' remained an important metric: Holt (1960) uses the term to refer to how ready for use a particular piece of hardware or software is: '... some sort of reliability (or usability, or state-of-completeness) code.' But at this stage, we start to see a meaning of 'evaluation' that we can recognize; it means 'testing' or 'appropriateness for task'. The first use of 'evaluation' in this sense is Israel's (1957) paper, in which the focus is the development of a set of pre-packaged input data that can be fed to a program as part of a testing regime. Similarly, in talking of evaluation, Calingaert (1967) defines throughput, turnaround, and availability as fundamental measures of performance. Lucas (1971) refers to three major purposes for evaluating the hardware and software performance of computer systems: selection evaluation (deciding what to buy), performance projection (figuring out how fast it will run once you have bought it), and performance monitoring (figuring out how fast it runs now you own it).

These notions of evaluation emphasized the speed of the computer. This is not evaluation in the way it is commonly used in HCI, because by the late 1970s the computer was no longer necessarily the slowest part of the interface; the computer started spending a lot of time waiting for the human rather than the other way around. But at this earlier stage, the limiting factor was the speed of the machine. What is important about this stage is that it was formative in the way that computer scientists saw evaluation, and this approach was influential for many years thereafter. The evaluation is in MIPS (million instructions per second), megahertz, and megabytes, terms that computer scientists understood as giving them information about appropriateness for and speed of task execution.

A key example of this interface paradigm can be found in the influential conference paper by Royce (1970), in which he discusses the waterfall method of organizing large software programming tasks. He discusses the importance of testing software and the role of the *customer*, but the customer is entirely a business entity; there is absolutely no notion of *users*. His notion of evaluation is that it is entirely concerned with the task of programming, and one that harkens back to the engineering notion of reliability: 'Most errors are of an obvious nature that can easily be spotted by visual inspection [of the code].' Once again, the computer is the turf of engineers, of mathematicians, and of these new breeds of programmers and computer scientists, and they are the ones doing evaluation.

2.3 Evaluation by Experimental Psychologists

The situation changes around the end of the 1970s and beginning of the 1980s; probably the earliest example is the 1976 SIGGRAPH Workshop on User-Oriented Design of Interactive Graphics Systems (Treu, 1977), although there was little work published in this vein again until the beginning of the 1980s. At this point, the first signs of what we might recognize as 'end-users' show up. These are people who are using the computer as a tool to solve problems. They no longer understand what the computer is doing in the same manner as engineers and programmers do, and they do not care. They are still mainly government and corporate users, and they are still doing single, focused tasks. The speed of execution of these tasks is becoming the metric by which systems are measured rather than the speed of the calculations that enable that execution. But there was a change in the air of computing; the human was becoming an essential part of the human–computer equation, and the focus in the discourse of the uses of computing was shifting from a discussion based purely around the technology itself to a discussion about the role of the computer and the human working together. As such, experimental psychologists and cognitive scientists were starting to enter the field of human–computer interaction, bringing with them an approach to evaluation and a set of problems they are particularly equipped to solve, based on experimental and psychophysical approaches to human–computer interaction. So in publications of the time, we see an emphasis on, for example, the ergonomics of the keyboard, the legibility of different colours of text, and the role of timing in input and output. Grudin (1990) writes:

> Perceptual issues such as print legibility and motor issues . . . arose in designing displays, keyboards and other input devices . . . [new interface developments] created opportunities for cognitive psychologists to contribute in such areas as motor learning, concept formation, semantic memory and action. In a sense, this marks the emergence of the distinct discipline of human-computer interaction.

So what of evaluation at this point? Experimental psychology and cognitive science bring with them an experimental approach towards the evaluation of systems; they are used to test users in laboratory experiments to determine psychophysical parameters, such as response times to stimuli and the like. However, they are entering a field that already has computer scientists and their notions of evaluation as the dominant paradigm, and the end result is a notion of evaluation that combines the system focus of computer science with the experimental approach of cognitive science. The limiting factor becomes the ways and speed in which users can instruct the computer to do what they want.

2.4 Evaluation by Usability Professionals

As the 1980s progress, we see the advent of the personal computer and the development of something that starts to look like a coherent field of human–computer interaction. The key difference to previous stages is that the computer itself was no longer the site for action, but rather the interaction itself. Shackel (1997) writes, 'In the beginning, the computer was so costly that it had to be kept gainfully occupied for every second; people were almost slaves to feed it' (p. 977). Shneiderman (1979) describes the relationship between computer and user as one of symbiosis with implications of human and computer on an equal footing, but by this stage the human is, at least ideally, no longer the lesser player in the relationship.

In 1981, the personal computer (PC) enters the stage, bringing with it a change from the mainframe and minicomputer concept of one computer for many users to initially one computer for one user and the foundation for one-to-many human-to-computer relationships of ubiquitous computing and the many-to-many relationships of computer-supported collaborative work. To the corporation, the PC meant that each employee could have their own computer sitting on their desk, for their exclusive use: a one-to-one person-to-computer ratio for the first time. There was also increase in the number of computers in the schools and in the homes, encouraging the development of the notion of a computer for discretionary use rather than use required by a job. In general, this profusion of computing power had a number of effects on the way the computer was used, who was using it, and what they were using it for. For example, the level of expertise that was necessary to have access to computing power decreased significantly. No longer could the end-user be expected to go through a training course or walk themselves through three-ring binders of instructions; it became necessary to start to think of delivering software that was easy to use and accessible by lay people. The limiting factor in using a computer to accomplish a task started to be the difficulty to the human of making sense of the computer. The human interface rose to the top of issues that the industry needed to address to make their product more successful in the marketplace.

In 1984, the release of the Macintosh computer was accompanied by a book, the Apple Human Interface Guidelines, which set out the specifications for the human-focused graphical user interface, which defined the ‘look and feel’ of the Mac interface. This was necessary for two reasons. Firstly, the graphical user interface was a novel approach to human–computer interaction; software developers were simply not used to designing for it. It was not possible to just release development software and assume that developers would do the right thing with it. Secondly, the Macintosh project was set up in such a way so as to encourage the development of third-party software for the platform. External developers were a key part of how the new platform was going to be used by people everyday.[1] Apple made the Macintosh operating system ‘look and feel’ and ‘ease of use’ key parts of the way that the computer was presented to the world.

2.5 Evaluation for the Social

As the personal computer became rapidly more popular, they entered the workplace in droves. But when computers were placed into office settings, to peoples’ surprise, it did not produce the increases in efficiency that were thought would be naturally brought about by such clearly advanced technologies. Many people using many computers would be, one would have thought, a simple matter of multiplying the effects of one person using one computer. But in practice the difference between many people-to-many computers and one person-to-one computer turned out to be as big a difference as that between one person-to-one computer and many people-to-one computer. Software specifically written to support existing ways of getting work done turned out to be hard to use and got in the way of accomplishing tasks rather than facilitating it. How, then, to solve the new problems produced by a situation in which many people in the workplace were using computers to do their own work, and yet desperately needed to communicate with others in the course of doing so?

Attempts to answer this question resulted in the development of what has become a separate field of computer-supported collaborative work, or CSCW. In 1984,

[1] See, e.g., http://www.folklore.org/StoryView.py?project=Macintosh&story=Inside_Macintosh.txt (visited 2010-05-30).

Paul Cashman and Irene Grief organized an interdisciplinary workshop to discuss the development of computer-related tools for the workplace, which was followed by the first full conference, CSCW'86, 2 years later (Greif et al., 1987). Previous efforts as part of the 'office automation' movement had hit somewhat of a dead end with a focus on building technical tools rather than understanding the environment in which they were placed (Grudin, 2005). Schmidt (1991, p. 10) uses the following quotation from Barber et al. (1983) to summarize the problem:

> In all these systems information is treated as something on which office actions operate producing information that is passed on for further actions or is stored in repositories for later retrieval. These types of systems are suitable for describing office work that is structured around actions (e.g. sending a message, approving, filing); where the sequence of activities is the same except for minor variations and few exceptions[. . .] These systems do not deal well with unanticipated conditions (Barber et al., 1983, p. 562).

So how did this new field, with its emphasis on the social in the workplace, deal with the problem of evaluation? In his thesis, Martin Ramage spends an entire chapter summarizing the various methods used in CSCW (1998). He divides CSCW evaluation into four types. The first, 'effects of a co-operative system in an organization', is concerned with examining the effects of a system once it has been introduced. The second, 'formative evaluation of CSCW technology', is evaluation that happens in the course of building technical systems, with the aim of developing them further and making them more useful and appropriate for the users – a notion very similar to the notions of evaluation we have been discussing in HCI. The third, 'conceptual development', is the kind of evaluation done only in a research context to evaluate the concepts that underlie the system and determine if they might be applicable. The fourth is 'what software should I buy?' Each of these categories is clearly a form of evaluation, but each requires a very different approach to the situation; as such, Ramage proceeds to synthesize a method of evaluation he refers to as systematic evaluation for stakeholder learning, or SESL, which involves multiple stakeholder evaluations of the socio-technical system, emphasizing the role of evaluation as a collaborative opportunity for learning within the organization.

2.6 Evaluation of Experience

The most recent approach to evaluation in HCI involves the evaluation of HCI focused on experience rather than tasks. Evaluations of experience-focused HCI are typically open-ended and encourage interpretation rather than eliminating ambiguity (Sengers and Gaver, 2006). Even more so than previous methods, they encourage looking at the human lived experience of the technology, rather than any kind of emphasis on the computer itself, and the researchers in this area draw from approaches that encourage this perspective, including ethnography, game design, industrial design, cultural theory, as well as art and aesthetic practices. Users are people going about their lives, generally outside of the workplace, communicating with friends, playing games, expressing and enjoying themselves. The field of experience-focused HCI has a strong theoretical grounding, with works such as McCarthy and Wright's *Technology as Experience* (2004) that leverages the work of pragmatist philosophers Dewey and Bakhtin, and Dourish's (2001) *Where the Action* that takes a phenomenological approach to HCI, recognizing the ways in which HCI engages with twentieth-century dissatisfaction with the adequacy of Descartian approaches to knowing and understanding. This is a field of research that is actively under development and has not developed agreed-upon approaches for evaluation, but there is early work that suggests likely approaches and demonstrates the emphasis on the role of design that is at the heart of approaches to experience-focused HCI.

For example, at CHI 2006, Isbister and colleagues (2006) presented an 'affective evaluation tool', taken to be focused on a nonverbal, body-based approach that encourages UCD values and be practical, portable, flexible, cross-culturally valid, and fun to use. The tool centres around the use of eight hand-held white clay figurines that are manipulated by the user to express emotions non-verbally in the course of interacting with a technological system. Similarly, Joseph Kaye et al.'s work with intimate objects emphasizes open-ended responses to cultural probe-influenced questionnaires about users' interactions with the technology and the social systems the technology is intended to influence (Kaye et al., 2005; Kaye, 2006). William Gaver references the role that commentary and discourse can play in other cultural situations by appropriating media such as documentary film and journalism for the evaluation of technological systems (Gaver, 2007). In contrast to the rest of these examples, in an excellent example of relating ideas from a new intellectual approach to existing work in the field, Charlotte Wiberg has shown how it is necessary and possible to modify traditional usability metrics to evaluate the experience of using websites designed for entertainment (Wiberg, 2005).

The emphasis shown here on open-ended and qualitative evaluations that engage both user and designer in interpretation of their affective, lived experience with the technology seems likely to be a hallmark of experience-focused evaluation and demonstrates the importance of the material covered in this report.

3 Design and Evaluation in the Real World

After the brief review of the history of evaluation in human–computer interaction, we will next contrast design and evaluation practices in product development and research: What are key differences between the two perspectives?

Researchers and practicing commercial designers, which we are calling 'real world', do tend to approach design and evaluation differently, probably due to a mixture of differences in end product goals and in the working cultures that they inhabit. We can readily see the tensions and challenges of communicating between these worlds in its evolution and iteration at the CHI conferences in recent years, which made a dedicated effort to reach out to more 'real-world' practitioners with practical tutorials and more opportunities to exhibit commercial design methods and results.[2] But how do 'real-world' and research design practices differ in detail? Let us break down this issue into the activities of design as outlined in the previous section (cf. Fig. 1).

3.1 Problem Definition and Time Frame

When defining a problem and a time frame, 'real-world' designers carry out their work within the constraints of commerce: of shipping a purchased product to end-users. Most real-world design efforts come with an extensive set of constraints within which the designer must work. It includes available budget for the project, a set time frame for releasing the product, market constraints in terms of what is considered acceptable for the target user group, and resource constraints in terms of team expertise and equipment available. Real-world project teams typically have members whose roles are assigned according to established areas of expertise (e.g. visual design, interaction design, usability testing, and project management). Most real-world teams are comprised of individuals who have demonstrated expertise and prior experience with the roles they are assigned. Team members are often asked to join in producing and committing to reasonable time estimates for the completion of projects.

Designers in research contexts also have constraints, but of a different sort. Time frames are linked to conference calendar cycles as well as the 'publish or perish' imperative of the tenure track in academia, and internal reporting cycles of research labs. Budgets are a by-product of institutional constraints (e.g. industry lab budgets, or academic grants and available infrastructure and resources) and are often not as clearly demarcated on a project-by-project basis as commercial design budgets. Team members may or may not have prior experience in a given role, depending upon the researchers' backgrounds and the experience level of student or intern participants. Often in academic contexts, projects have a secondary aim of providing apprenticeship to budding researchers who will be growing their skills as they work, which also shifts the nature of the project management role. Defining a problem in the research context proceeds in quite a different manner than in the commercial world. A researcher's aim is not simply a polished end product but also (and sometimes instead) some sort of methodological and/or theoretical contribution to the ongoing research dialog – what Zimmerman et al. call 'transformative design' (Zimmerman et al., 2007). This makes a big difference in the choices that researchers make as the process proceeds, as will be outlined in the following.

3.2 Involvement of Users

Regarding user involvement, 'real-world' designers usually work on very tight timelines and also with a high degree of dependency for success upon the satisfaction

[2] See, for example, the call to the design community for CHI 2007: http://www.chi2007.org/community/design.php (visited 2010-05-30).

of end-users with what they do. The product of their efforts must satisfy a predetermined economically viable user group, and efforts will be made on most larger scale efforts to get the designs in front of some subset of that market to ensure that the team is on the right track. Involving users is typically done at the early conceptual state, with focus groups or other methods for gleaning market acceptability of a product concept, and then at mid-to-late stages of product development, with usability testing. Real-world designers can call upon the services of in-house or consultancy-based experts to recruit members of the target user group, for fees that are often substantial. If the project is being completed for a larger company with existing customers, the designer can probably draw upon the user base as well as aggregated data about use patterns. Typically user involvement is limited to focus groups and usability testing sessions. So one could say that the accuracy of the population selection can be quite high for real-world designers, but the range of involvement is rather restricted. There are of course exceptions, as documented in papers about participatory design efforts (e.g. Puri, Byrne et al., 2004; Kristensen et al., 2006). In recent years, some commercial design firms have begun to use and to market their use of a wider range of tactics for involving users.[3]

Research-based designers often use a sample of convenience (college students or fellow researchers) due to a lack of budget for and attention paid to the importance of targeting representative populations. It is also the case that a researcher may not be exactly sure, yet, what population will be best served by a new blue-sky idea for an interface. Thus, one could say that research designers tend to involve a less varied and ‘accurate’ set of users but also that this may not be the main point of soliciting feedback, at least in some cases. Research designers feel much more free to try out experimental methods for eliciting feedback and participation from end-users (e.g. cultural probes, Gaver et al., 2004, or the sensual evaluation instrument, Isbister et al., 2006). These methods may provide the designer with a wider range of input from users, offering richer fodder for design. However, such input must also be interpreted with care, as the methods often will not have been subjected to many applications and rigorous discussion among designers as to their efficacy in supporting design practice.

4 Brainstorming

As explained, ‘real-world’ designers usually work on a very tight time frame, and idea generation must take place rapidly. Some design consultancies have developed processes for rapid and effective generation of the maximum number of best quality ideas. These firms may work in collaboration with their client companies and can also pull in designers working on other projects for consultancy.

[3] For example, IDEO’s method cards (http://www.ideo.com/work/item/method-cards/; visited 2010-05-30).

Researchers are typically able to use longer time frames in the ideation phase that brings them to a final project concept, interweaving this work with completion of other projects. As with user involvement, researchers often feel freer to use more experimental methods such as bodystorming (Oulasvirta et al., 2003) to obtain a wide range of concepts for their designs. They may consider the evolution of new techniques and tools for brainstorming to be a part of their research (e.g. Bastéa-Forte and Yen, 2007).

5 Prototyping

'Real-world' designers use prototypes to serve several purposes: to help the team converge upon a working solution, to engage users in order to get corrective/confirmatory feedback, and to showcase for internal stakeholders (e.g. managers) to get approval and provide reassurance. The third purpose is quite important for commercial prototypes and can sometimes lead to substantial time spent polishing and carefully delineating what will be shown. Since real-world designers are working under time pressure, they will avoid throw-away efforts wherever possible – prototyping in commercial contexts may therefore be more conservative compared to the latitude often offered in research contexts. Some commercial designers make use of paper prototypes, wire frames, and other techniques that mitigate the early effort spent on polishing prototypes that leads to more design inertia.

Researchers vary widely in their use of prototypes – in many cases a prototype (rather than a polished product) is the end result of a research cycle. Most research prototypes are not nearly as robust and polished as commercial counterparts; however they are likely to be far more adventurous in terms of core concepts that are manifested in the design itself, for example exploring the benefits of new input modalities well before they are commercially viable (e.g. eye tracking – see Merten and Conati, 2006). Research prototypes usually do not have to stand up to hard use by outsiders or naysayers, which may lead to less being learned through exposing prototypes to users than in commercial contexts. However, there is also active research into prototyping as an activity in itself – feeding back valuable data to those in commercial practice about how and when to use low-fidelity versus high-fidelity prototypes, for example (e.g. Lim et al., 2006).

5.1 Testing with Users

'Real-world' designers, as mentioned in Sect. 3.2, typically engage in early focus group-style testing and mid-to-late-stage usability testing, with an emphasis on making sure that they test with exemplars of the target user group. Both focus group and usability testing are often done by outside companies or by internal departments that specialize in these processes, working from the belief that the designers themselves may have biases that get in the way of being able to truly test and critique their work, even though some commercial designers have questioned this bifurcated mode of working.[4] Commercial developers typically also allocate a significant amount of time to late-stage 'bug-fixing' testing of their releases.

Researchers may employ similar methods, though more likely with a convenience sample rather than a representative sample of users. As for the previous activities, they also spend time innovating user testing methods as well (as mentioned in Sect. 3.2).

5.2 Polishing and Sharing Results

A 'real-world' designer's efforts usually result in a product released to the public, which must stand up to unmonitored use. Thus, a great deal of time and money in the development cycle is spent polishing the product and ensuring that it will be reliable and able to withstand repeated use. Real-world designers sometimes write papers or deliver talks at design conferences (such as CHI, the ACM Conference on Human Factors in Computing, and DIS, Designing Interactive Systems Conference, conference series), but they may not be at liberty to share innovative details of their process if their company considers them a business advantage. In the end, commercial designers gauge their success by whether the product satisfies end-users and is successful in the marketplace.

As mentioned above, researchers often stop with a reasonably polished prototype, which may not need to be usable in an unattended fashion and may only be viewed by other relatively sympathetic and context-sharing experts. Researchers' products thus rarely achieve the level of polish and robustness of commercial designs. However, researchers must be prepared to explain and present the larger value of their work – exploring/revealing new terrain in terms of research contribution. A research prototype is meant to 'demonstrate a research contribution' (Zimmerman et al., 2007, p. 495); so publication and dissemination is an important part of the design process for researchers.

Researchers sometimes use final prototypes to help engage the community in a critique of existing practice: 'Design researchers engaged in critical design create artefacts intended to be carefully crafted questions. These artefacts stimulate discourse around a topic by challenging the status quo and by placing the design researcher in the role of a critic. The Drift Table offers a well known example of critical design in HCI, where the design of an interactive table that has no intended task for users to perform raises the issue of the community's possibly too narrow focus on successful completion of tasks as a core metric of evaluation and product success' (Zimmerman et al., 2007, p. 496, with reference to Gaver et al., 2004).

[4] See, e.g., Lane Becker (2004) 90% of all usability testing is useless (http://www.adaptivepath.com/publications/essays/archives/000328.php; visited 2010-05-30).

6 Third-Wave HCI and Evaluation Challenges

In this chapter, we have tried to provide a historical background to HCI in order to arrive at a better understanding of the challenges that affective interaction poses to design and evaluation methods. In our historical background, we have argued that the overall challenge has moved from early concerns with reliability and efficiency of the machine itself to subsequent concern with people as a part of the complete system of human and machine, to finally seeing people as collaborators in workplaces in the CSCW field. With the advent of seeing computers also as tools for play, ludology and experience design have emerged as important design goals. And it is here that we find affective interaction and its role.

We have also painted a particular picture through our focus on a UCD cycle, ending with the 'third wave of HCI' perspective. When user experience is dealt with from this stance, it becomes a problem of *interaction*. Emotion is seen as constructed in the dialogue between user and system – not as something that exists a priori in the user – and the design challenge is to design for such experiences to happen. As we shall discuss in subsequent chapters, this is a that stance differs from the more AI-oriented perspective that is still more directed at looking at the problem as a usability problem. The traditional AI view is that it should be possible to recognize some aspects of the users' emotions and then improve the efficiency of the system from this perspective.

In the following chapters, we will explore the influences of these approaches to understanding the nature of human computer interaction, and their implications for the involvement of users in the design cycle, for the kinds of systems that we aim to design, and for the evaluation metrics appropriate for such study.

References

Barber GR, de Jong P, Hewitt C (1983) Semantic support for work in organizations. In: Mason REA (ed) Information processing 83. Proceedings of the IFIP 9th world computer congress, Paris, 19–23 Sept 1983. North-Holland, Amsterdam, pp 561–566 (cited in Schmidt 1991)

Bastéa-Forte M, Yen C (2007) Encouraging contribution to shared sketches in brainstorming meetings. In CHI '07 extended abstracts on human factors in computing systems (San Jose, CA, USA, April 28–May 03, 2007). CHI '07. ACM, New York, NY, 2267–2272. DOI= http://doi.acm.org/10.1145/1240866.1240992

Blythe MA, Overbeeke K et al (eds) (2004) Funology: from usability to enjoyment. Kluwer, Boston, MA

Calingaert P (1967) System performance evaluation: survey and appraisal. Commun ACM 10(1):12–18

Dourish P (2001) Where the action is. MIT Press, Cambridge, MA

Ehn P (1992) Scandinavian design: on participation and skill. In: Adler PS, Winograd TA (eds), Usability: turning technologies into tools. Oxford University Press, New York, NY, pp 96–132

Gaver WW (2007) Cultural commentators: non-native interpretations as resources for polyphonic assessment. Int J Hum Comput Stud 65(4):292–305

Gaver WW, Boucher A et al (2004) Cultural probes and the value of uncertainty. Interactions 11(5):53–56

Gaver WW, Bowers J, Boucher A, Gellerson H, Pennington S, Schmidt A, Steed A, Villars N, Walker B (2004) The drift table: designing for ludic engagement. In CHI '04 Extended Abstracts on Human Factors in Computing Systems (Vienna, Austria, April 24–29, 2004). CHI '04. ACM, New York, NY, 885–900. DOI= http://doi.acm.org/10.1145/985921.985947

Greif I, Curtis B et al (1987) Computer-supported cooperative work (panel): is this REALLY a new field of research? In: Proceedings of the SIGCHI/GI conference on human factors in computing systems and graphics interface. ACM Press, Toronto, ON

Grudin J (1990) The computer reaches out: the historical continuity of interface design. In Proc. CHI '90, p264. ACM, New York, NY, 261–268. DOI= http://doi.acm.org/10.1145/97243.97284

Grudin J (1994) Computer-supported cooperative work: history and focus. IEEE Comput 27(5): 19–26

Grudin J (2005) Three faces of human computer interaction. IEEE Ann Comput 27(4):46–62

Holt A (1960) Over all computation control and labelling. Commun ACM 3(11):614–615

Isbister K, Höök K, Sharp M, Laaksolahti J (2006) The sensual evaluation instrument: developing an affective evaluation tool. In: Grinter R, Rodder T, Aoki P, Cutrell E, Jeffries R, Olson G (eds) Proceedings of the SIGCHI conference on human factors in computing systems, Montréal, QC. ACM Press, New York, NY, pp 1163–1172

ISO/IEC 13407:1999 (1999) Human-centred design processes for interactive systems. International Organization for Standardization, Geneva

Israel DR (1957) Simulation techniques for the test and evaluation of real-time computer programs. J ACM 4(3):354–361

Kaye JJ (2006) I just clicked to say I love you: rich evaluations of minimal communication. In CHI '06 Extended Abstracts on Human Factors in Computing Systems (Montréal, Québec, Canada, April 22–27, 2006). CHI '06. ACM, New York, NY, 363–368. DOI= http://doi.acm.org/10.1145/1125451.1125530

Kaye JJ, Levitt MK, Nevins J, Golden J, Schmidt V (2005) Communicating intimacy one bit at a time. In CHI '05 Extended Abstracts on Human Factors in Computing Systems (Portland, OR, USA, April 02–07, 2005). CHI '05. ACM, New York, NY, 1529–1532. DOI= http://doi.acm.org/10.1145/1056808.1056958

Kristensen M, Kyng M et al (2006) Participatory design in emergency medical service: designing for future practice. In: Proceedings of the SIGCHI conference on human factors in computing systems, ACM, Montréal, QC

Lim Y-K, Pangam A et al (2006) Comparative analysis of high- and low-fidelity prototypes for more valid usability evaluations of mobile devices. In: Proceedings of the 4th Nordic conference on human–computer interaction: changing roles, ACM, Oslo, Norway

Lucas H (1971) Performance evaluation and monitoring. ACM Comput Surv 3(3):79–91

McCarthy J, Wright PC (2004) Technology as experience. MIT Press, Cambridge, MA

Merten C, Conati C (2006) Eye-tracking to model and adapt to user meta-cognition in intelligent learning environments. In: Proceedings of the 11th international conference on intelligent user interfaces. ACM, Sydney

Oulasvirta A, Kurvinen E et al (2003) Understanding contexts by being there: case studies in bodystorming. Pers Ubiquit Comput 7(2):125–134

Preece J, Rogers Y et al (1994) Human–computer interaction. Addison-Wesley, Reading, MA

Puri SK, Byrne E, Nhampossa JL, Quraishi ZB (2004) Contextuality of participation in IS design: a developing country perspective. In Proceedings of the Eighth Conference on Participatory Design: Artful integration: interweaving Media, Materials and Practices – Volume 1 (Toronto, Ontario, Canada, July 27–31, 2004). PDC 04. ACM, New York, NY, 42–52. DOI= http://doi.acm.org/10.1145/1011870.1011876

Ramage M (1998) The learning way: evaluating co-operative systems. Lancaster University, Lancaster, UK

Royce WW (1970) Managing the development of large software systems: concepts and techniques. In: Proceedings of IEEE WESTCON, Los Angeles, CA

Schmidt K (1991) Riding a tiger, or computer supported cooperative work. In: Bannon L, Robinson M, Schmidt K (eds) Proceedings of the 2nd European conference on computer-supported cooperative work (ECSCW '91), 25–27 Sept 1991, Amsterdam, The Netherlands. Kluwer, Norwell, MA, pp 1–16

Sengers P, Gaver B (2006) Staying open to interpretation: engaging multiple meanings in design and evaluation. In: Proceedings of the 6th conference on designing interactive systems, ACM, University Park, PA

Shackel B (1997) Human–computer interaction – whence and whither? J Am Soc Inf Sci 48(11):970–986

Shneiderman B (1979 Dec) Human factors experiments in designing interactive systems. IEEE Computer, Los Alamitos, CA, pp 9–19

Sommerville I (1992) Software engineering. Addison-Wesley, Redwood City, CA

Treu S (ed) (1977) User-oriented design of interactive graphics systems. In: Based on ACM/SIGGRAPH workshop conducted, 14–15 Oct 1976, Pittsburgh, PA. ACM Press, New York, NY

von Neumann J (1945) First draft of a report on the EDVAC, contract No. W-670-ORD-492, Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, PA. Springer, pp 383–392

Wiberg C (2005) Affective computing vs. usability? Insights of using traditional usability evaluation methods. In: CHI 2005 workshop on innovative approaches to evaluating affective systems, Vienna, Austria

Zimmerman J, Forlizzi J Evenson S (2007) Research through design as a method for interaction design research in HCI. In: Proceedings of the SIGCHI conference on human factors in computing systems, San Jose, CA. ACM Press, New York, NY, pp 493–502