FP48 User Centered Design: What, Why, and When

Matthias Rauterberg, Technical University Eindhoven

What is User Centered Design (UCD)?

The User Centered Design (UCD) approach comprises a set of several steps, methods and tools designed to assist engineers and developers in addressing the issue of usability in design of interactive systems. The UCD approach combine to assist in the process of collating design information obtained using a variety of user oriented data gathering techniques. The essence of the UCD approach is that it provides a structure to assist the developer in assuring that relevant design issues have been considered in a user oriented manner. Individual methods and tools may be visited and revisited a number of times in an iterative design process (Rauterberg and Strohm, 1992). The UCD approach directs developers to the questions that must be answered if products are to be successfully applied in their intended market, and focuses on the importance of taking user requirements into account. Consequently UCD should be seen as a methodology for collating design material rather than a design model per se. From this perspective UCD acts as a framework rather than being a detailed design method in its own right. Design usually involves a number of common activities. Typically these include: (1) a problem definition phase, (2) the development of a functional specification, (3) a building phase, and (4) a testing or evaluation phase. Many engineers and designers will intuitively recognize this sequence even though they may not actually use this terminology.

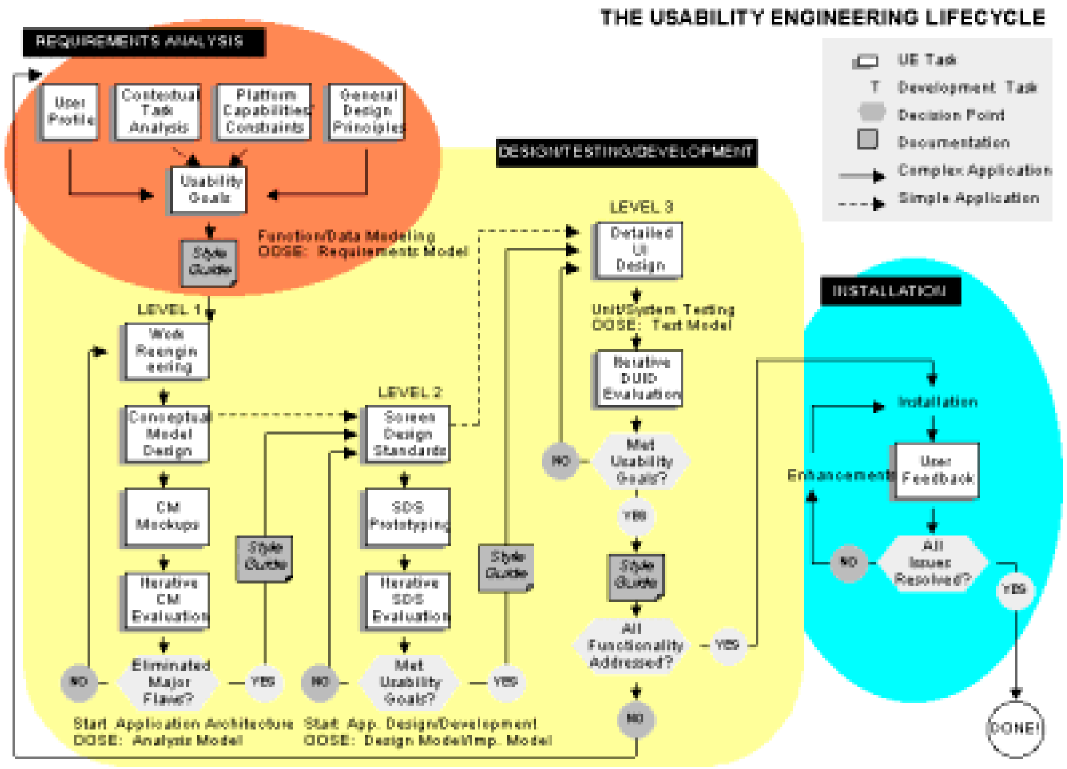

Fig. 1: The Usability Engineering Lifecycle (reprint from Mayhew, 1999)

Why is UCD important?

Several cases in industry show clearly why a UCD approach can improve the business by improved usability.
Case-1 ‘IBM website’: On IBM's website, the most popular feature was the search function, because the site was difficult to navigate. The second most popular feature was the 'help' button, because the search technology was so ineffective. IBM's solution was a 10-week effort to redesign the site, which involved more than 100 employees at a cost estimated 'in the millions USD.' The result: In the first week after the redesign, use of the 'help' button decreased 84 per cent, while sales increased 400 per cent (Tedeschi, 1999).
Case-2 ‘Online shopping’: A report showed that 39 per cent of test shoppers failed in their buying attempts because e-commerce sites were too difficult to navigate. Additionally, 56 per cent of search attempts failed. Creative Good offered the striking revelation that a dollar spent on advertising during the 1998 holiday season produced $5 in total revenue, while a dollar spent on user centered design improvements yielded more than $60 (Enos, 2000).
Case-3 ‘Stock exchange’: After the New York Stock Exchange upgraded its core trading systems using user-centered design techniques, productivity rose dramatically and users’ error rates fell by a factor of 10 even though workloads more than doubled (Gibbs, 1997).
Case-4 ‘Software products in general’: In a study of around 8,000 software development projects conducted by over 300 American companies, the Standish Group found that only 16 per cent of projects were successful (completed on time and on budget, with all features and functions as initially specified). The three main reasons for project failure were: (1) Lack of user input, (2) incomplete requirements and specifications, (3) changing requirements and specifications (Standish Group, 1995).
Around 63 per cent of software projects exceed their cost estimates. The top four reasons for this are: (1) frequent requests for changes from users; (2) overlooked tasks; (3) users’ lack of understanding of their own requirements; (4) insufficient user-analyst communication and understanding (Lederer and Prasad, 1992).
Only 33% of the maintenance effort is spent for debugging, and 67% of the maintenance effort therefore is needed for changing the systems (e.g. changed requirements caused by users). Adequate effort in the early development stages reduces the often cost- and time intensive repair and correction tasks in the maintenance phase (Rauterberg and Strohm, 1992).
It is about 40–100 times more expensive to fix problems in the maintenance phase of a program than in the design phase (Boehm, 1981).
Systems designed with usability engineering have typically reduced the time needed for training by around 25 per cent. User-centered design typically cuts errors in user-system interaction from five per cent down to one per cent. Without user-centered design, a user interface typically has around 40 flaws that can slow users and lead to errors (Landauer, 1995).
Revenues for one Digital Equipment Corporation product that was developed using user-centered design techniques increased 80 per cent for the new version of the software, and customers cited usability as the second most significant improvement (Wixon and Jones, 1995).

When and how does UCD help?

Let us now look at the UCD approach in more detail and consider how user orientation can be embedded within it (Mayhew, 1999).
(1) Problem definition is where a designer will identify a problem and begin to conceptualize a potential solution (e.g. a product or service). This process can be broken down into a series of inter–related activities which differ in purpose and level of detail required. These might include background research to establish the nature of the intended market; the extent of competition posed by similar products already in existence; the need to adhere to standards; estimates of the cost of development of the new product, and so on. It is at this early stage of the process that designers may find greatest use of the UCD methodology's methods and tools. The majority of UCD tools may be utilized at this stage. Briefly, the Environmental Context and Product Environment tools allow designers to refine a general understanding of the problem that is to be explored and to examine how a product may fit into the wider context in which end users live. The main purpose of these tools is to force developers to consider the wider implications of how the product will be supported, and also document some of the likely implications of these factors.
(2) Functional Specification: Once some understanding of the users and their activities has been obtained, it is necessary for developers to move to the more creative process of developing a detailed specification for a product to satisfy users needs. UCD assists in the process of refining a functional specification for products i.e. what should be implemented, but does not explicitly address technical details of the non-functional specification. The UCD framework assists in this definition process with a Product Attribute Matrix (PAM), which assists developers in cross–referencing the desired features of a product (as revealed through User and Activity Analyses) with its actual features (as suggested through Product Analysis and Environmental Context considerations). This allows an initial analysis of the likely success of a product in meeting the user requirements. The results of this comparison are then used to create a Requirements Summary (RS) and Design Summary (DS), which may be taken through into the build process.
(3) Build: Building the product follows the specification process. UCD does not provide assistance in the management of this process as this is largely a technical activity.
(4) Test: Once a physical prototype or product exists, UCD directly supports the planning of testing and evaluation activities through the usability evaluation tools and testing methods. These assist in the planning of testing activities, in summarizing the results of such testing, and in recording any actions needed in the form of design modifications. The emphasis is on evaluation of the functionality of products and the degree of match between the capabilities of the product and the needs of users.

Summary: UCD works (IOP-MMI, 2002).

References

Boehm, B W (1981) *Software Engineering Economics*. New Jersey: Prentice-Hall.

Enos, L (2000) Report: E-Holiday Glitches Could Cost $15B. *E-Commerce Times,* October 17, 2000. Downloadable from http://www.ecommercetimes.com/

Gefen, D and Straub, D (2000) The Relative Importance of Perceived Ease of Use in IS Adoption: A Study of E-Commerce Adoption. *Journal of the Association for Information Systems*, Volume 1, Article 8, October 2000.

Gibbs, W W (1997) Taking Computers To Task. *Scientific American*, July 1997.

IOP-MMI (2002) *UCD works* [interactive CD]. Downloadable from http://www.ipo.tue.nl/homepages/mrauterb/presentations.html

Karat, C (1993) Usability Engineering in Dollars and Cents. *IEEE Software,* 10(3), 88-89.

Landauer, T K (1995) *The Trouble with Computers*. MIT Press.

Lederer, A L and Prasad, J (1992) Nine Management Guidelines for Better Cost Estimating. *Communications of the ACM*, 35 (2), 50–59.

Mayhew, D (1999) *The Usability Engineering Lifecycle*. Morgan Kaufmann.

Rauterberg, M and Strohm, O (1992). Work Organization and Software Development. *Annual Review of Automatic Programming*, 16(2), 121-128.

Standish Group (1995). *Technical Report*. Downloadable from http://www.scs.carleton.ca/~beau/PM/Standish-Report.html

Tedeschi, B (1999) Good Web Site Design Can Lead to Healthy Sales. *New York Times,* August 30, 1999.

Wixon, D and Jones, S (1991) *Usability for fun and profit: A Case Study of the Design of DEC RALLY.* In Rudisill, M., C. Lewis, P. G. Polson and T. D. McKay (eds.) *Human-Computer Interface Design* (3-35). Morgan Kaufmann.

Contact: *g.w.m.Rauterberg@tue.nl*