

Extreme Evaluations – Lightweight Evaluations for Software Developers

Michael Gellner & Peter Forbrig

University of Rostock, Department of Computer Science, Software Engineering Group, Albert-Einstein-Str. 21, 18051 Rostock, Germany

{mgellner, pforbrig}@informatik.uni-rostock.de

Abstract: In most of the cases usability evaluations are done by usability experts. Employing such experts requires a certain size in business. Nevertheless users do not tolerate hard to use tools. So in a lot of small and middle sized companies developers are forced to learn handling usability aspects. This is not much easier than teaching usability engineers how to develop software. The usability evaluation process and its requirements also miss usable attributes. As a solution a light weighted usability evaluation model for software developers is created.

Keywords: usability engineering, usability testing, usability evaluation, interactive systems

1 Modeling Usability Evaluations and Usability Engineering

One of the most comprehensive views is given by Mayhew (Mayhew 1999). As shown in Figure 1 her lifecycle considers all aspects from the first steps to the successful installation. Fulfilling the requirements of this lifecycle will lead to a complex organizational structure. Further this process model does not suit well if several alternations are probable. These requirements are hard to fulfill for a

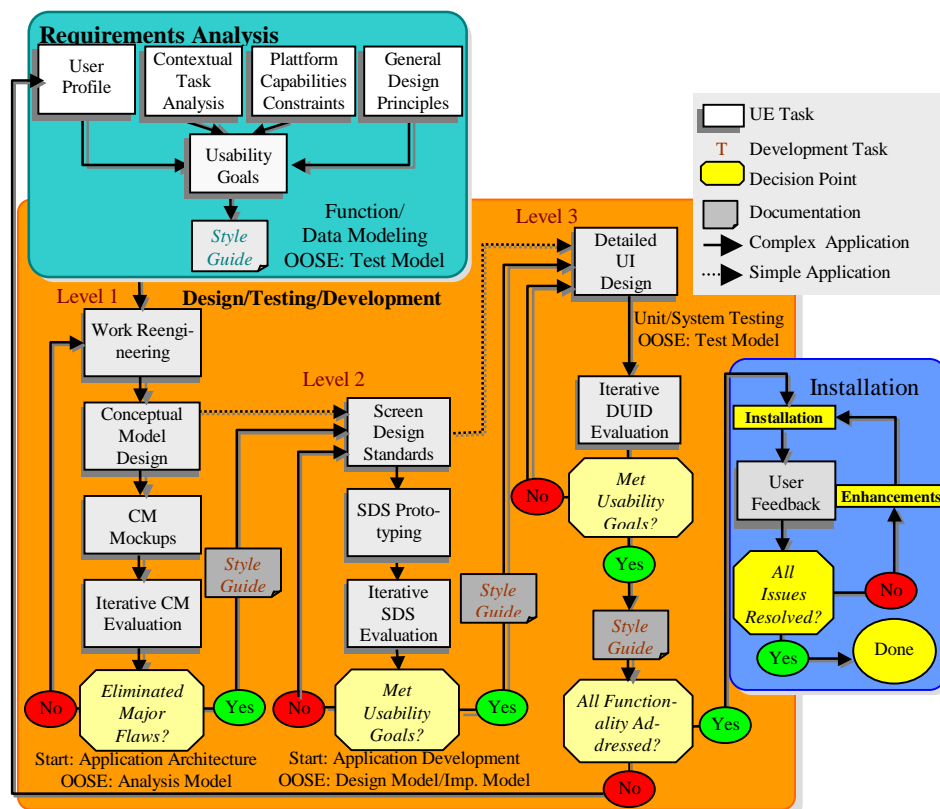


Figure 1: Usability Engineering Lifecycle (Mayhew 1999)

small or middle sized company. A whole staff is necessary to manage the various tasks. Mayhews lifecycle is directed to usability experts or deciders in a bigger environment that want to install an all including usability department. This approach seems not to be applicable for a small team of developers.

1. Know The user
 - a. Individual user characteristics
 - b. The user's current and desired tasks
 - c. Functional analysis
 - d. The evolution of the user and the job
2. Competitive analysis
3. Setting usability goals
 - a. Financial impact analysis
4. Parallel design
5. Participatory design
6. Coordinated design of the total interface
7. Apply guidelines and heuristic analysis
8. Prototyping
9. Empirical testing
10. Iterative Design

Figure 2: Usability Engineering Model (Nielsen 1993)

Similar to Mayhews *Usability Engineering Lifecycle* is Nielsens *Usability Engineering Model*, see Figure 2 (Nielsen 1993). At the first view this looks like a list that has to be worked off. This is not right. Nielsens model contains circles and feedback paths, also, but it mentions only the components of usability engineering. The context is assumed to be known by the usability engineer. The correct application does not result in a kind of lightweight model. On the other hand the mentioning of the components only makes it hard for developers to apply this model.

Another model that is often shown in that context is the *Star Life Cycle* from Hartson and Hix, see Figure 3 (Preece 1994). In contradiction to Nielsens

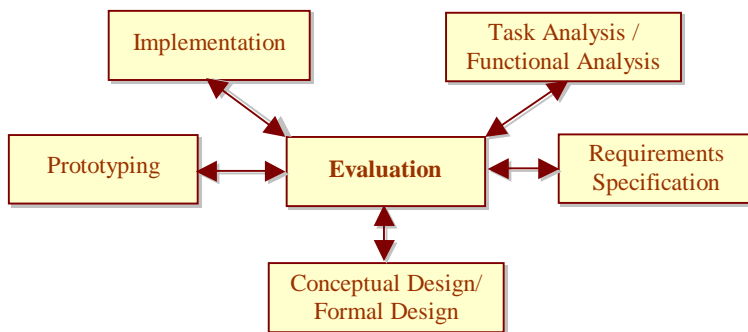


Figure 3: Star Life Cycle [Pre 94]

model the Star Life Cycle offers structural information beneath the components. This model is also not really helpful for developers: It shows only the steps that are known from the development models around the term »Evaluations«. This explains not much about how an evaluations could be conducted easily. Hartson and Hix do not consider the Star Life Cycle as a model for usability evaluations or for usability engineering. The focus is on an alternative to the waterfall or spiral model for development purposes since those models even do not mention evaluations or the term »usability«. So the Star Life Cycle is no solution for the problem mentioned above.

Further approaches to model usability engineering, usability evaluations or usability testing are shown by Rubin (Rubin 1994), Constantine and Lockwood (Constantine and Lockwood 1999), Dumas and Redish (Dumas and Redish 1999) and others. The approaches that show usability testing only are not discussed separately since this is only one aspect we want to cover.

2 Terminology

Before our approach is presented some terms should be determined. The first thing that is interesting is the *weight* of a model or a process. Some processes like *eXtreme Programming* (XP) or Hacking are considered as lightweight models. The counterpart to these one are models like the waterfall and the spiral model, also the Rational Unified Rrocess (RUP). The difference is the flexibility the process allows to its users (Eckstein 2000). Applying the waterfall model means to march one time from the first phase to the last one. Errors are hard to correct since it is allowed to go back only one stage if necessary. Practice shows that is often helpful to go back further. As a second criterion the quota of work is taken that counts for fulfilling the requirements of the model in relation to the work that is done to fulfill the project. Applied on the mentioned usability models Mayhews usability lifecycle and Nielsens usability engineering model are heavy weighted models. Our presented model offers a lot of liberties (see section *The Eight Phase Pattern*). It contains structural information but does not hinder to go back if it is necessary. This enables to decide nearly free how to combine phases. Since nearly no

work has to be done to fulfill the models requirements we tend to classify it as a lightweight model. The next topic are the usability terms: *Usability engineering* includes every activity that is related to any stage of the product development to improve the usability properties. The term stands for a kind of a superset of workings and measures around this issue. For the purpose of evaluating results such models are too broad. Sub terms are *Usability Evaluation*, *Usability Inspection* and *Usability Testing*. Usability evaluation is one component of usability engineering that is realized by usability inspections and usability testing. Further components of usability engineering are *task analyses*, *requirements engineering* and others that are not considered here. Usability Testing means to test artifacts with testing persons. The artifacts can be software prototypes but also early paper and pencil mock-ups. An usability inspection is done by comparing artifacts against requirements, checklists or giving them to an expert review. See Figure 4 for an all-embracing view.

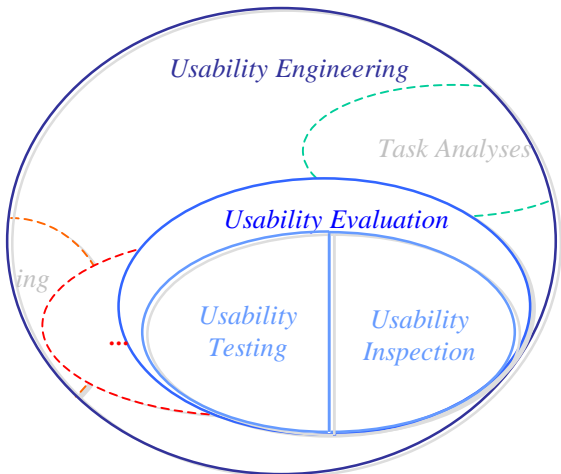


Figure 4: Layers of Usability Terms

The last term that should be explained is «eXtreme». The term *eXtreme Programming* (XP) is created by Beck in 1998 (Beck 1998). The consultant Beck became engaged in the *Payroll Project* at Chrysler, a project that should substitute the 15 different payroll systems that were running concurrently to that time. The project suffered on classical software engineering problems. This could be turned around

with a set of innovative methods Beck introduced. XP breaks with some established rules. The following shown approach is not as radical for usability engineering as Beck's approaches are for software engineering. The term extreme is adapted to *extreme evaluations* since our approach wants to enable more agile workflows, also.

3 The Eight Phase Pattern

To be understood easily by developers the approaches is designed as a model that is known in that domain. Although the former waterfall model due to Boehm has some disadvantages it is really easy to understand and well known by software developers. For usability evaluations it is no problem to purge the prevention to go back only one phase (Gellner 2000). Our eight phased model is shown in Figure 5.

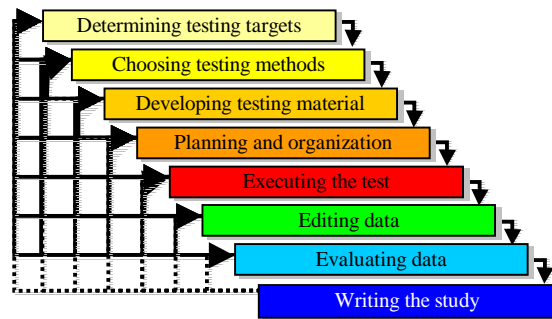


Figure 5: Eight Phase Pattern

Since it is part of a pattern language for extreme usability evaluations it is called *Eight Phase Pattern*. The pattern aspect can be ignored in this context. For further information see (Gellner 2003a). Beneath the information, what subtasks has to be fulfilled also structural information are given. It is possible to go back as far as necessary at every time. In comparison to software development one cycle through the Eight Phase Pattern is a smaller task

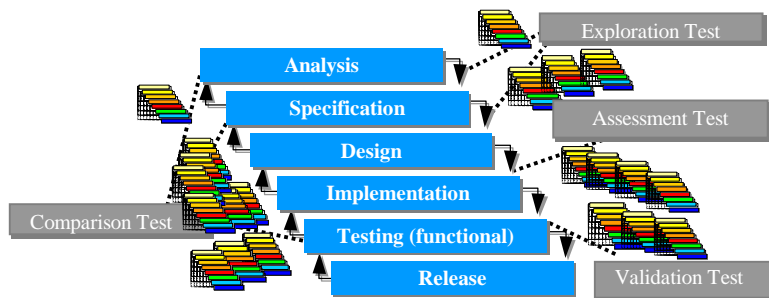


Figure 6: Integrating the Eight Phase Pattern

than one cycle in waterfall model for development. There one cycle stands for developing a whole release. Figure 6 shows the relations between the Eight Phase Pattern and the waterfall based development (based on). In the same way the integration of tests can take place by other development approaches (spiral model, incremental model, object oriented model etc.).

In contradiction to some other models the Eight Phase Pattern contains no component that limits the usage in external projects. The scenario of external consultants or usability experts is covered as well as in-house evaluation.

4 Applications

An important motivation for developing the Eight Phase Pattern was the fact that none of the existing models was suited well nor to categorize products for supporting usability evaluation neither to develop them. So the Eight Phase Pattern can be seen as a process pattern (take it as is and word due to it) and as a development concept (take it and cover every phase with tools). Niensens model (see Figure 2) for example contains several points (participatory design, prototyping and others) that are hard to map into software tools, whereas every point in the Eight Phase Pattern can be seen as a component in a workflow management tool (represented in the most primitive case at least as wizards).

In comparison to software development we are nearly speechless if tools are methods have to be judges. In software development we can easily assign a tool to a certain phase in the waterfall view (IDE → implementation, UML-Painter → analyzing, specification, CASE Tool → spanning phases). This works even if the waterfall is not the underlying model. The Eight Phase Pattern enables such a communication for usability evaluations (see Figure 7).

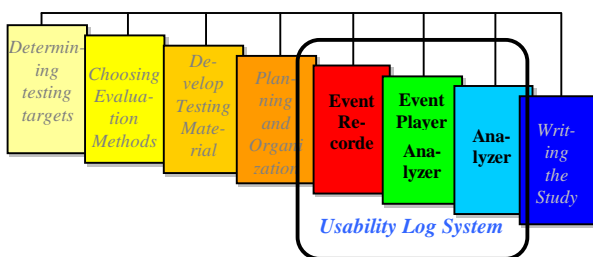


Figure 7: Categorizing Tools

It is also possible to analyze the costs that was caused by the works for each phase, see Figure 8.

This data can be used to compare with other labs to increase efficiency. Further it allows to see what areas or phases are not covered with tools.

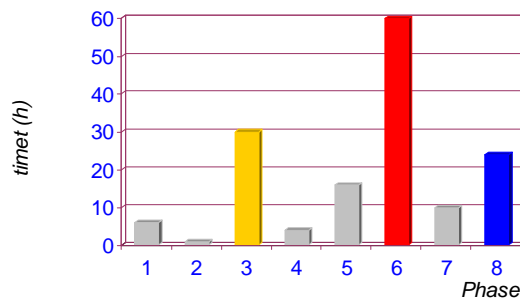


Figure 8: Rating Performance per Phase

Our intention to find approaches for software support led to a set of tools. Until now there are approaches and solutions for the phases 2 to 7. Tools that support phase embracing evaluations are considered as *Computer Aided Usability Evaluation Tools (CAUE)*. This term again is created similar to the term *Computer Aided Software Engineering (CASE)*.ⁱ Our most powerful approach *ObSys* combines the methods

- *Event Logging* (using predefined short cuts manually to save observations)
- *Event Recording* (capturing automatically the message queue of an operating system)ⁱⁱ
- *Video Recording*
- *Screen Capturing*

Especially event recording has a high potential to automate product or prototype based usability evaluations (As known by usability experts it is to late to start with evaluations when there are pre-releases. But this is not the only method we recommend.). The events are saved in databases and can be processed with SQL to find striking datasets (see Figure 9).

More concisely is our visualization called *MouseMap*. This multidimensional representation allows to watch the events graphically. The direction of mouse moves is visualized color gradient, clicks are pointed as round dots and the speed is

ⁱ In literature there is also mentioned the term *Computer Aided Usability Engineering (CAUSE)*. At the moment we see no basis for such a comprehensive demand.

ⁱⁱ At the moment determined to Microsoft Systems; beneath the *ObSys-Tool* we have a VNC-based solution that works on all platforms that be connect to the internet

indicated by the thickness of the lines (Gellner and Forbrig 2003, Gellner 2003b).

ZEITPUNKT	MESSAGETYPE	PARAML	PARAMH
00:00.43.23.716	512	229	392
00:00.43.23.795	512	229	392
00:00.43.23.795	512	229	392
00:00.43.23.873	512	229	393
00:00.43.24.029	512	229	393
00:00.43.24.029	512	229	393
00:00.43.24.420	512	229	393
00:00.43.24.420	512	229	393
00:00.43.25.091	256	283	1
00:00.43.25.107	512	229	393
00:00.43.25.201	257	283	1
00:00.43.25.310	512	230	393
00:00.43.25.326	512	230	392
00:00.43.25.341	512	230	390
00:00.43.25.373	512	232	386
00:00.43.25.388	512	236	366
00:00.43.25.810	512	930	759

Figure 9: List with recorded events

MouseMaps summarize to an image what as a video sequence has to be watched as long as it took to record it, see Figure 10. On the other hand selecting to much events for a MouseMap are to complex to be analyzed qualitatively .

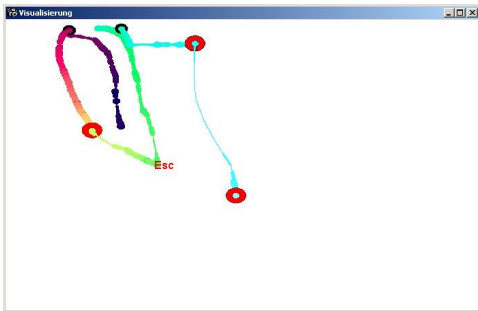


Figure 10: MouseMap

At the moment the error detection is not automated. An easy way to find error »candidates« is given with the time series of the events. If the time differences are visualized peaks can appear. Assumed that an user acts with a personal workflow a peak can indicate a tool based problem (but also someone asking something are else).

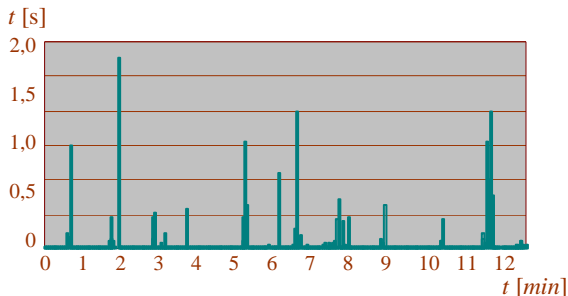


Figure 11: Differences between events

In an usability evaluation session all interruptions (reading scenarios, getting explanations etc.) are documented. So it is easy to distinguish between deflections and other sources for peaks.

The shown scenario in Figure 11 was observed by editing paragraphs in WinWord with built-in problems. Around the half of the peaks were caused by reading the scenario items. The other peaks indicate indeed problems. Analyzing six (± 3) exact positions in a 10 minute scenario is much more easier than watching the scenario three times on video to find fairly the same positions.

5 View and Further Work

For finding the described tools (and others) the Eight Phase Pattern was a helpful and effective approach. Further tools can be deduced and have to be realized. The most important step is the implementation of the observed error patterns for automating the detection. As a result the evaluator would get video snippets with relevant scenes and MouseMaps with the happenings around the error situation. Based on that material the evaluator must decide what steps has to be done next. A high amount of the time consuming summarizing and finding of the relevant positions were eliminated. With the shown approaches even software developers would be able to conduct usability tests and to evaluate the results. Other focuses are on the first phases. In the next month a tool for selecting well suited methods based on the algorithm in (Gellner 2002; Gellner 2003a) will be completed.

With the MouseMap visualization we found something like a body language users show by working with mice and keyboards. We assume that there are much aspects beyond simple causalities like Fitt's Law or the Steering Rule that can be investigated more in detail.

References

Beck, K. (1999), Extreme programming explained: embrace change. Addison-Wesley.

Constantine, L. L. and Lockwood L. A. D (1999), Software for use, a practical guide to the models and methods of usage-centered design. Addison Wesley Longman, Inc., Reading, Massachusetts.

Dumas, J. S. and Redish, J. C. (1999), *A Practical Guide to Usability Testing*. Revised Edition. Intellect Books Limited, Exeter, England.

Closing the Gap: Software Engineering and Human-Computer Interaction

- Eckstein, J. (2002), XP – eXtreme Programming: Ein leichtgewichtiger Software-Entwicklungsprozess. In: *basicpro*, Vol. 35, 3, pp. 6-11.
- Gellner, M. (2000), Modellierung des Usability Testing Prozesses im Hinblick auf den Entwurf eines Computer Aided Usability Engineering (CAUE) Systems. In: *Rostocker Informatik-Berichte*, Vol. 24, pp. 5-21. Rostock, 2000.
- Gellner, M. (2002), A Pattern Based Procedure for an Automated Finding of the Right Testing Methods in Usability Evaluations. In: Forbrig, P., Limbourg, Q., Urban, B. und Vanderdonckt, J., *Bricks & Blocks: Towards Effective User Interface Patterns and Components*, Proceedings of the 9th International Workshop Design, Rostock, 2002, pp. 423-427.
- Gellner, M. (2003a), Automated Determination of Patterns for Usability Evaluations. In: Hruby, P. und Sørensen, K. E.: *Proceedings of the VikingPLOP Conference 2002*, Micorsoft Business Solutions, ApS, 2003, pp. 65-80.
- Gellner, M. (2003b), Mousemaps – ein Ansatz für eine Technik zur Visualisierung der Nutzung von Software und zur Automation der Entdeckung von Bedienungsfehlern. (submitted and accepted at *Mensch & Computer 2003*)
- Gellner, M. und Forbrig, P. (2003), *ObSys – a Tool for Visualizing Usability Evaluation Patterns with Mousemaps*. (submitted, accepted and presented at *HCI International 2003*)
- Mayhew, D. J. (1999), *The Usability Engineering Lifecycle*. Morgan Kaufmann Publishers, Inc., Kalifornien, San Francisco.
- Nielsen, J. (1993), *Usability Engineering*. AP Proffessional, New Jersey.
- Preece, J., (1994), *Human-Computer-Interaction*. Addison-Wesley, Harlow,.
- Rubin, J., (1994) *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*, John Wiley & Sons, Inc..