Integrating usability engineering techniques into the software development process

12/8/2008
Carnegie Mellon University
Jaffer Haider
Executive Summary

Usability is an important quality attribute that has a great impact on the user experience of a software product. To the user, the interface is the product. Thus it is very important for software teams to ensure high levels of usability in their products for them to be accepted and liked by their end users.

There are a number of hindrances that prevent integration of usability tasks in the development lifecycle. The cultural gap between the fields of human computer interaction and software engineering is perhaps the underlying cause of many issues that arise when engineers are made to use usability techniques. Most teams are not aware the importance of usability, or are of the opinion that regular development staff can build perfectly usable interfaces. Management is sometimes of the view that usability tasks will be costly and time consuming. Education and training of engineers and management is required to overcome these issues, as well as customizing usability techniques so that they are cheap and lightweight, hence easy to adopt.

It is also important to define what the team understands by usability, since it encompasses a large amount of characteristics. Usability goals should be decided and usability characteristics of the system should be documented so that they are testable.

There are a wide variety of usability techniques and methods, each having its own strengths and weaknesses. It is important for teams to be able to make the decision as to which technique is required in a certain condition. This report looks at the different stages in the development lifecycle and which usability techniques are suitable therein. An overall approach to integrating usability techniques and practices into the different development stages of a typical iterative development lifecycle is presented. Planning activities and roles related to these techniques are also discussed. Usability integration with slightly different lifecycles, such as XP and ACDM are also examined.

For easier adoption of usability techniques at an organization level, it is important to introduce easy and cost effective techniques at the grass root level. This results in teams reaping the benefits of usability techniques without expending a lot of budgeted resources. In this regard, studies are looked at that highlight usability techniques that are widely used, cheap and easy.

Usability is an integral part of everything that is engineered, whether it be software or otherwise. With the increasing complexity of software systems, it is even more important to design interfaces that are easy to understand and use. Usability is also not something that can be slapped on at a certain point in time. Hence it is important to ensure that usability tasks are planned and carried out during all phases of the development lifecycle.
## Contents

1  Introduction ........................................................................................................................................ 4

   1.1  Target Organization .................................................................................................................. 4

2  Defining usability .................................................................................................................................... 5

3  Issues hindering adoption of usability techniques ............................................................................... 6

   3.1  Issues ........................................................................................................................................... 6

       3.1.1  “Usability?” ....................................................................................................................... 6

       3.1.2  “Regular development staff can surely design good interfaces” ........................................ 6

       3.1.3  “Usability tasks are costly, time consuming and complex” .................................................. 7

       3.1.4  “We don’t have the tools to support usability tasks” ............................................................ 7

   3.2  Evolution of usability engineering in organizations ....................................................................... 7

4  Usability engineering in software development lifecycles .................................................................... 8

   4.1  Usability engineering techniques ................................................................................................ 9

       4.1.1  Prioritizing usability techniques .......................................................................................... 11

   4.2  Integrating usability engineering ................................................................................................ 11

       4.2.1  Pre-design phase ............................................................................................................... 12

       4.2.2  Design phase ..................................................................................................................... 13

       4.2.3  Post-design phase ............................................................................................................. 13

       4.2.4  Planning usability activities ............................................................................................... 13

       4.2.5  Summary ............................................................................................................................ 14

   4.3  Usability engineering in specific development processes .......................................................... 15

       4.3.1  Agile processes ................................................................................................................... 15

       4.3.2  Architecture Centric Design Method (ACDM) .................................................................... 16

   4.4  Usability engineering roles ........................................................................................................ 17

5  Affordable integration of usability ........................................................................................................ 17

6  Conclusion .............................................................................................................................................. 18

7  Bibliography .......................................................................................................................................... 19
1 Introduction

Consumer software has long passed the point where competition is solely done based on the number of features. Most software products of today have far more features than typical users will ever need. Competition in such products is based more on quality attributes than features. Products that are faster, more reliable, and simpler and easier to use than the competition are generally more successful.

Usability is one such quality attribute that is particularly important when it comes to customer satisfaction. To the user, the interface is the product. Thus ease of use is vital for the success of a product that your customers will spend a substantial amount of time using. This report will not go into the details of explaining how important usability is. There are many published studies, articles, blogs and indeed, entire online communities dedicated to that topic.

In this report I will look at different ways in which usability techniques can be integrated into the development lifecycle of a software project, such that it ensures that the resulting software will be easy to use for its target user base. I will first look at the different aspects of usability; what exactly does it mean when one says ‘the software must be usable’. I will then present some reasons for why, in the industry in general, there is an absence of cases where such techniques are being used as part of the development effort on a software project. I will also talk about some recommendations for overcoming these problems, so that teams can lookout for telltale signs of such hurdles, avoid them and be able to apply usability techniques in their development processes. Then I will examine the different types of usability techniques, keeping in mind that certain types of techniques are more suitable for integration at specific points in the development lifecycle. In sections following that, I will discuss some different ways in which usability can be integrated into the development lifecycle. I will look at a number of specific development lifecycles and methodologies, such as XP and ACDM, and how usability tasks can be integrated with them, so that the reader gets a better idea of how to customize their own development processes for higher levels of usability. Finally, I will look at a subset of usability techniques that are more likely to be adopted by teams because of being cheap and easy to learn, so that readers have a starting point for introducing a low overhead usability process in their development cycle which has a high benefit to cost ratio.

It is important however to note that when it comes to customizing the development lifecycle, there is no one technique that will result in success for a project. Techniques that are selected, learnt, and customized by the team on their own result in higher output quality and easier adoption (1). Thus it is important to not follow a particular recommendation as is, but to identify techniques that will be useful and practical in a certain situation.

1.1 Target Organization

The material presented in this report will be in context to the needs of a small to medium sized software company that specializes in development of custom software solutions. It is assumed that such an organization will not have a dedicated user experience team that will help other teams in their usability tasks. It is also assumed that most engineers in the organization do not have a sound grasp of usability techniques and do not apply them in the development cycle in a systematic and planned way. Resource
and budget considerations are important to such an organization when evaluating the feasibility of planning usability tasks.

2 Defining usability

Usability can potentially be used to describe a wide range of software quality attributes. For practical purposes, usability must be defined in terms of the characteristics of a system, such that development teams are able to measure and test these characteristics.

There are a number of different definitions of usability. The ISO/IEC 9126-1 standard for software product evaluation emphasizes the importance of designing for quality (2). It defines a number of characteristics on which quality of software depends, usability being one of them. It is subdivided into 5 characteristics:

- Understandability, i.e. the capability of software to show to users is appropriateness to tasks to be accomplished and to the context of use.
- Learnability, i.e. the capability of software to help users to easily learn its features.
- Operability, i.e. the capability of software to make possible the execution and control of its functionality for its users.
- Attractiveness, i.e. the capability of the software to be pleasant for users.
- Compliance, i.e. the capability of the software to adhere to standards, conventions, and style guides for usability.

Another definition proposed along the same lines by Nielsen (3) also defines usability as a collection of the following characteristics:

- Learnability, i.e. the ease with which the user can learn the functionality of the software and understand its behavior.
- Efficiency, i.e. the level of attainable productivity, once the user has learnt the system.
- Memorability, i.e. the ease with which users can remember functionality of the software, so that infrequent users can return to the system after periods of non-use without needed to learn how to use it again.
- Low error rate, i.e. the level to which the software helps users in making less errors (and errors of less severity) and when an error does occur, how easily it allows users to recover.
- Subjective user satisfaction, i.e. the measure to which users like the system.

It is important to understand that depending on the nature of the project, some of these characteristics will have higher priority than others. It is up to the team to identify the higher priority ones, based on the analysis of users and their tasks. For example, efficiency will be important if the software is used repeatedly in a high pressure environment, such as software for a point of sale (POS) system in a busy supermarket. These characteristics must be used to define usability goals for the project, which will be discussed in the section 4.2.
3  Issues hindering adoption of usability techniques

Usability tasks in general are not a part of a typical software development lifecycle. Companies that systematically integrate usability tasks in the development lifecycle are definitely in the minority. I will now examine some issues that cause hindrance in the adoption of usability techniques, and present some ways to overcome them.

3.1  Issues

3.1.1  “Usability?”
One of the prime reasons for lack of adoption of usability techniques is the gap between the fields of software engineering and human computer interaction. HCI foundations come from the fields of psychology, sociology, industrial design, graphic design, and so forth whereas software engineering takes a typical engineering view (4). Because of this, there is a vast majority of engineers who are not fully aware of the importance of usability. It was common in the early years of the software industry to see engineers take great pride in the performance and quality of their code, but pay no heed to the user interface. This trend is still prevalent to some extent. Engineers find designing and writing code to be more ‘natural’, and thus they spend a great amount of effort in doing that. This is particularly true for most open source projects, which consist of volunteers whose interests lie in technical issues rather than usability (5). The user interface is something that is slapped on at the end.

One of the obvious solutions is to educate engineers on the importance and relevance of catering to the needs of users of their systems. This needs to be done both at the academic and professional level. Companies need to have processes in place to ensure education of their development staff on all important design aspects of software, including usability. Having dedicated usability experts (and evangelists) in the company, not necessarily on each team, can also help in spreading awareness amongst the development staff. Also, usability processes and techniques should be presented to the development staff using terminology and concepts which they are familiar with. Concepts common in software engineering should be used, instead of using notions from psychology, graphic design etc.

3.1.2  “Regular development staff can surely design good interfaces”
This is something that Nielsen (6) presents as part of his model of ‘evolution of usability engineering in organizations’. In these situations no attempts are made at acquiring staff with usability expertise or to perform user testing. If an effort is made by the development staff to address specific usability issues, discussions generally get reduced to religious debates because of an absence of techniques involving actual users (7). Decisions are solely based on the opinions of engineers. Software applications resulting from such teams generally suffer from a host of usability problems, but due to the team being oblivious to them, quality of the user experience is suboptimal.

Management education and buy in is essential for correcting such situations. Management needs to be educated on the financial benefits of developing highly usable products, so that inclusion of usability engineering tasks is facilitated in the development lifecycle.
3.1.3 “Usability tasks are costly, time consuming and complex”

Most managers conjure up scenes of budget and schedule overruns when usability tasks are mentioned. It is a common belief that usability tasks will involve hiring expensive consultants, require setting up usability labs with sophisticated equipment and having your engineers spend countless hours interviewing and running tests with users. In addition to this, most development teams tend to think of usability tasks as external interference and do not give them enough importance. Also because most team members are new to usability techniques, they do not trust the fact that these techniques will reduce the amount of rework on the user interface and result in reduction of overall development time (8). Additionally, most teams are overwhelmed by the complexity of some elaborate usability techniques and shy away from using them.

Usability engineering does not always have to be elaborate and time consuming. There are a number of studies and techniques (6) (7) that show that low tech usability techniques such as paper prototyping, shortened heuristic evaluations and usability testing with just 3 users can produce results with a very high benefit to cost ratio. These techniques also do not involve elaborate procedures that are unfamiliar to software engineers. Thus it is easier for regular development staff to use them. I will cover in detail a number of ‘budget usability techniques’ in a later section.

3.1.4 “We don’t have the tools to support usability tasks”

The processes, vocabulary and tools for HCI teams and software engineering teams are very different. This acts as a hurdle for teams that want to incorporate usability engineering tasks in their software development lifecycle. Perhaps the cultural issue, of the mental gap between the two types of teams, is even greater than this technical issue.

A case study (8) has shown that educational classes for each team (HCI and engineering teams) prepared by the other team have proved to be very successful. When learning a specific activity, focusing on the intent helps by allowing engineers to quickly grasp new techniques. Also, in this particular case, UML tools with which the engineering staff was thoroughly familiar with proved to be a suitable tool for documenting usability artifacts (e.g. workflow models, scenarios and user environment designs from contextual design techniques).

3.2 Evolution of usability engineering in organizations

Having looked at some common barriers to integration of usability engineering to the software development lifecycle, I believe it is important to look at the bigger picture at the organization level, to see how these different issues result in different stages of acceptance of usability. I will use the evolution model created by Nielsen (6), which is summarized in Table 1 to discuss this.

<table>
<thead>
<tr>
<th>Category</th>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skepticism</td>
<td>1.</td>
<td>“Usability does not matter”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Team focuses on the code only.</td>
</tr>
<tr>
<td></td>
<td>2.</td>
<td>“Good interfaces can surely be designed by regular developers”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Usability is considered ‘important’, but no specialized staff is hired, and no usability tasks are carried out.</td>
</tr>
</tbody>
</table>

Table 1 Evolution model for usability engineering in an organization
<table>
<thead>
<tr>
<th><strong>Curiosity</strong></th>
<th>3. Having interfaces ‘blessed’ by the magic wand of usability experts</th>
<th>Usability specialists are brought in, often after the development phase, to evaluate the interface design of an application. At this time, it is too late to make any major changes, and the usability engineers have to work without end user involvement.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Acceptance</strong></td>
<td>4. GUI panic</td>
<td>Involving usability engineers in the project, but only at the beginning or at the end of the lifecycle. Again, no contextual information about users is gathered.</td>
</tr>
<tr>
<td><strong>Acceptance</strong></td>
<td>5. Discount usability engineering <em>sporadically used</em></td>
<td>Some simple and cheap usability methods are used in some projects, but mostly at an inappropriate time that does not result in maximum benefit. Such methods are only used because of persuasion from particular members on the team. There is no organization wide usability agenda.</td>
</tr>
<tr>
<td><strong>Acceptance</strong></td>
<td>6. Discount usability engineering <em>systematically used</em></td>
<td>Most projects start using simple usability methods. Methods are also used earlier in the lifecycle.</td>
</tr>
<tr>
<td><strong>Partnership</strong></td>
<td>7. Usability group and/or lab founded</td>
<td>After having seen the benefits of simpler usability methods, a companywide knowledge base is created to be used by projects so that more sophisticated usability methods can be used.</td>
</tr>
<tr>
<td><strong>Partnership</strong></td>
<td>8. Usability permeates lifecycle</td>
<td>This stage is seldom reached. It includes planning usability tasks in the lifecycle of projects from the beginning and performing these tasks throughout the lifecycle.</td>
</tr>
</tbody>
</table>

To be able to effectively advocate for change in an organization with respect to usability engineering, it is important to understand at which stage in this evolution model the organization is currently in. Although this is not a roadmap for the adoption of usability engineering, it can serve as a guide for organizations that are looking to incorporate usability in their development processes. Even though individuals who have experienced firsthand the benefits of usability testing and other techniques are easily convinced, it is far difficult to change the ways of an organization. It is evident from this evolution model that one effective way of bringing change is to spread the use of usability engineering from the bottom up. Starting with simpler and quicker techniques, more sophisticated ones can be employed once management buy-in has been achieved.

### 4 Usability engineering in software development lifecycles

In this section I will look at a number of different usability engineering techniques and examine which ones are suitable for application in which phases of a development lifecycle. Using this information on usability techniques, I will look at the overall picture of how these techniques can be integrated into a development lifecycle.
4.1 Usability engineering techniques

In most cases, usability engineering methods can be broadly categorized as inspection methods and test methods (9). Inspections methods typically do not require input from the user and are carried out solely by the engineering team. These methods concentrate on evaluating the user interaction design on the basis of certain usability heuristics and good design practices. Examples of inspection methods are heuristic evaluations and cognitive walkthroughs.

Test methods typically require input from users and concentrate on:

a) Gathering information on the needs of users and context in which the system will be used
b) Validating the design by having users interact with some manifestation of the software system (paper prototypes, story cards, etc.)

Examples of test methods are field observations and thinking aloud.

There are a vast number of different usability techniques, each having their own focus and strengths and weaknesses. One survey (4) identified up to 82 different techniques being used in organizations. Most of these techniques were similar in nature, and had minute differences between them.

If we examine the goals of usability techniques in the two categories mentioned previously, we can find three distinct goals; analysis, design and evaluation. This categorization of usability techniques is common across a number of usability lifecycle studies that have been looked at (4) (3) (2) (9).

The following table shows a number of important usability engineering techniques categorized based on their goals and the role they play in a development process. This table will serve as a reference when discussing integration of usability engineering methods in a software development lifecycle.

Table 2 Categorization of usability engineering techniques based on their goals

<table>
<thead>
<tr>
<th>Goals</th>
<th>Techniques/Activities</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td>• Contextual interviews</td>
<td>These techniques focus on understanding the problem domain, which includes factors such as the users themselves, the nature of their tasks, their work environment, and identity of the product.</td>
</tr>
<tr>
<td></td>
<td>• User and task analysis (GOMS)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Usability specifications and goals</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Ethnographic observations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Competitive analysis</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Essential use cases</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• User personas</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Product concept development</td>
<td></td>
</tr>
<tr>
<td>Design</td>
<td>• Prototyping and scenarios</td>
<td>These techniques help the engineering team in designing interactions between the system and users. The main focus of some of these activities is empirical testing, i.e. testing prototypes with users, and understanding the users' perception of the system.</td>
</tr>
<tr>
<td></td>
<td>• Interaction design (user interface specifications)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• User environment designs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Action analysis</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Thinking aloud</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Cognitive walkthroughs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Detailed use cases</td>
<td></td>
</tr>
</tbody>
</table>
(use of) Interface standards
Software architecture usability scenarios

**Evaluation**
- Heuristic evaluations
- Field observations
- Usability tests
- Attitude questionnaires
- Automated logging to user actions

These techniques allow the engineering team to evaluate their designs, whether they’re in the prototype stage or have been deployed in the users’ work environment. Evaluation is based on usability heuristics, as well as feedback from users.

If we look at these techniques more closely, we will notice that most of them can be applied at different stages of a development lifecycle. Assuming we are dividing our development lifecycle into the typical iterative stages: requirements engineering, design and implementation, and deployment/evaluation, it would be useful to apply many of the above mentioned activities in more than one stage.

The following table shows the relevance of the above mentioned usability techniques to each development stage. This table has been built on the information gathered from (9) & (4).

**Table 3 Relevant lifecycle stages for usability engineering techniques**

<table>
<thead>
<tr>
<th>Usability technique</th>
<th>Applicable in stages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contextual interviews</td>
<td>Requirements engineering</td>
</tr>
<tr>
<td>User and task analysis (GOMS)</td>
<td>Requirements engineering, design</td>
</tr>
<tr>
<td>Usability specifications and goals</td>
<td>Requirements engineering</td>
</tr>
<tr>
<td>Ethnographic observations</td>
<td>Requirements engineering</td>
</tr>
<tr>
<td>Competitive analysis</td>
<td>Requirements engineering, design</td>
</tr>
<tr>
<td>Essential use cases</td>
<td>Requirements engineering</td>
</tr>
<tr>
<td>User personas</td>
<td>Requirements engineering</td>
</tr>
<tr>
<td>Product concept development</td>
<td>Requirements engineering</td>
</tr>
<tr>
<td>Prototyping and scenarios</td>
<td>Requirements engineering, design</td>
</tr>
<tr>
<td>Interaction design (user interface specifications)</td>
<td>Design</td>
</tr>
<tr>
<td>User environment designs</td>
<td>Requirements engineering, design</td>
</tr>
<tr>
<td>Action analysis</td>
<td>Design</td>
</tr>
<tr>
<td>Thinking aloud</td>
<td>Requirements engineering, design</td>
</tr>
<tr>
<td>Cognitive walkthroughs</td>
<td>All</td>
</tr>
<tr>
<td>Detailed use cases</td>
<td>Requirements engineering, Design</td>
</tr>
<tr>
<td>Interface standards</td>
<td>Design</td>
</tr>
<tr>
<td>Software architecture usability scenarios</td>
<td>Design</td>
</tr>
<tr>
<td>Heuristic evaluations</td>
<td>All</td>
</tr>
<tr>
<td>Field observations</td>
<td>Deployment</td>
</tr>
<tr>
<td>Usability tests</td>
<td>All</td>
</tr>
<tr>
<td>Attitude questionnaires</td>
<td>All</td>
</tr>
<tr>
<td>Automated logging to user actions</td>
<td>Deployment</td>
</tr>
</tbody>
</table>
4.1.1 Prioritizing usability techniques

This list of 22 usability techniques is too big for any single software project to use. Typically projects will use a much smaller number of techniques, if any, to ensure suitable levels of usability in their software.

Nielsen has conducted a study (3) on 13 usability engineers across a number of teams to find out which usability techniques they had most recently used and what impact they felt the technique had on usability in general. Since only usability engineers were involved in this study, the results are not representative of a typical software development team, since most teams do not have dedicated usability engineers. The purpose of involving only usability engineers in the study was to collect views founded on actually using the techniques.

The top five and six (respectively) techniques according to frequency of use and perceived impact on usability, based on the data gathered from the study, are shown in the tables below.

<table>
<thead>
<tr>
<th>Table 4 Frequency of use in development process</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Frequency of use</strong></td>
</tr>
<tr>
<td>1. Visit to customer location before start of</td>
</tr>
<tr>
<td>design</td>
</tr>
<tr>
<td>2. Iterative design</td>
</tr>
<tr>
<td>3. Participatory design: involving users in the</td>
</tr>
<tr>
<td>design stage</td>
</tr>
<tr>
<td>4. Prototyping using computer based tools</td>
</tr>
<tr>
<td>5. Competitive analysis: looking at existing</td>
</tr>
<tr>
<td>competing products</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 5 Impact on usability of software</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Impact on usability</strong></td>
</tr>
<tr>
<td>1. Iterative design</td>
</tr>
<tr>
<td>2. Task analysis of user’s current task (similar to contextual design)</td>
</tr>
<tr>
<td>3. Empirical testing with real users as subjects</td>
</tr>
<tr>
<td>4. Participatory design: involving users in the design stage</td>
</tr>
<tr>
<td>5. Visit to customer location before start of design</td>
</tr>
<tr>
<td>6. Field observations at customer location once the product has been deployed</td>
</tr>
</tbody>
</table>

These shortlisted techniques can be used as candidate techniques for teams wanting to integrate new usability tasks in their development process. The data gives some sense of how useful these techniques are, and the fact that they are used frequently points to the fact that they have a benefit to cost ratio that is acceptable for most teams.

4.2 Integrating usability engineering

User interface design is a matter of compromise and tradeoff. User interfaces must be powerful yet simple and clear. They must be easy to use yet easy to learn. They must be consistent yet allow for optimization of individual tasks. User interface designers are always faced with these conflicting goals. That is why it is very important to have a usability process in place that will allow a team to make informed decisions on all goals. (10)

Nielsen presents (3) a general usability engineering model that can be used as a baseline for augmenting a wide variety of software development lifecycles. The fundamental rules of this model are:

1. Early focus on users and their tasks
2. User participation in the design
3. Coordination of different parts of the user interface
4. Empirical user testing
5. Iterative revisions of designs based on test results

It is evident that these rules are applied in sequence. Concepts that complement these rules are:

- **Iterative development**: it is nearly impossible to get the user interface right the first time. Iterations with users that involve testing and evaluating the user interfaces must be performed to refine the interface based on the users’ needs.
- **Get started now**: It is easy for development teams to get overwhelmed by usability techniques and to put them off till other projects. Teams should start with simpler usability techniques, because even simple techniques can result in large payoffs.

The important phases of the software development lifecycle in which these usability activities will be integrated are before, during and after product design and implementation. Nielsen recommends these activities be applied in an iterative fashion similar to Boehm’s spiral model (11). Details of each of these phases are provided below.

### 4.2.1 Pre-design phase

The purpose of activities in this phase would be to understand target users, their tasks and their work environment. The key is to not rush into the design phase, until clear usability goals have been set. Individual user differences and variability in tasks are two factors that have a large impact on usability, thus users must be observed in their natural work environment. By knowing the target demography of the user population, personas can be developed that will allow the engineering team to anticipate learning difficulties and other factors. This will allow the team to design the UI such that it caters to these factors.

With regards to the tasks that users perform, the overall goals of users must be examined. Information flow and the sequence of events of normal tasks and exceptional tasks should be studied. Apart from this, functional analysis must be performed to find out what is it that users are really trying to do. Contextual design techniques and its artifacts can be used to find and model all this information.

Competition or existing products that the software will replace must be taken into account. These will influence factors such as the UI of the new software must not conflict with skills users have already learnt. It is a good idea to perform heuristic analysis of existing competition, and use that information to create prototypes, which can be used to further understand and refine requirements.

The end result of usability activities in this phase should be to come up with a concrete set of usability goals. Usability goals should be based on the different defining parameters of usability (understandability, efficiency etc.). These different parameters should be prioritized, i.e. how important is efficiency compared to understandability. Furthermore, these requirements should be documented in a concrete way, for example as quality attribute scenarios. This will result in these requirements being measurable and testable. Nielsen also recommends defining different sets of levels for these quality
attributes, such as a **worst acceptable level, planned usability level and best possible level**. This will allow teams to better gauge the scope and required quality of the software.

### 4.2.2 Design phase

The main goal of the design phase would be to define an interaction specification that is usable and implementable. This should be achieved by creating a prototype based on the gathered requirements and usability principles and performing test iterations with target users.

Experimental prototyping and heuristic analysis should be done early on in the design phase so that the design can be refined quickly. In early stages, prototypes can be low fidelity paper based ones. As they are tested out with users and refined, they could be implemented using software. Also, the advantage of heuristic analysis is that it can be applied without having a running system. This allows the design team to test their interfaces when they cannot be tested with users.

Additionally, it is also important to have the technical tools in place to ensure a consistent and high quality development of the interface. A centralized authority can be setup that will coordinate different aspects of the interface design to ensure consistency. Software frameworks, libraries and standards should be used to ensure a consistent look of the interface.

A number of different empirical testing techniques can be applied in this phase to verify the designs, such as thinking aloud, attitude questionnaires, automatic logging of user actions and usability testing. It is important to prioritize the problems found during testing, because it might not be possible to solve all of them. It is important to perform tests in an iterative fashion, and be ready to change and retest the interface when problems are uncovered. It is recommended that at this stage, elaborate tests of single design ideas should be avoided. Instead, different design ideas should be tested in small tests, so as not to wear out test users. It is also important to realize that if your test users become too accustomed to the prototype being tested repeatedly, they stop being the representative novices you should be performing tests with.

### 4.2.3 Post-design phase

The main goal of this phase is to gather data for future versions of the software. Follow-up field observations should be performed to gather data on how the system is performing. Features built into the system, such as automated collection of usage information and allowing users to easily send feedback to vendors can be considered. Dillon (12) emphasizes the importance of these activities because typical usability tests with prototypes fail to capture how the relationship between the user and system evolves over time, since it is not a holistic evaluation approach. Field observations allow teams to overcome this shortcoming, and be prepared to anticipate changes in usability requirements.

### 4.2.4 Planning usability activities

Apart from performing these activities, Nielsen also recommends a set of planning activities that will help in ensuring that the team is able to effectively apply usability techniques in the development lifecycle.
For teams new to usability techniques, he recommends writing out *explicit plans* on what needs to be done in each technique. These plans should include enough details to allow novice team members to perform these techniques. Additionally, *pilot activities*, with no more than 10 percent of resources, can be planned to identify any issues that the team may encounter in performing a certain technique. The results can be used to change the activity for the 90 percent of remaining resources.

Furthermore, an *overall usability plan* should be defined which specifies which usability techniques will be performed at what times in the lifecycle. This will ensure proper planning of tasks and help in communicating status to management, since management buy in is vital when trying out new techniques.

### 4.2.5 Summary

Using the information in Table 3 and the usability engineering model presented by Nielsen, we can summarize the integration of techniques in Figure 1.

![Diagram of mappings of usability techniques to development phases]

*Figure 1 Summary of mappings of usability techniques to development phases*

More detailed mapping diagrams can be found in (4) & (2).
4.3 Usability engineering in specific development processes

Using the model from the previous section as a base, we can now examine how usability engineering can be integrated in a couple of software development processes that are slightly different from the typical iterative model used in this section.

4.3.1 Agile processes

Typical agile processes use very minimalistic techniques when it comes to usability. These techniques involve user stories and paper prototyping. User reactions to paper prototypes cannot be substituted for more detailed prototyping techniques. Additionally, these prototyping sessions are not done in an iterative fashion, which prevents the engineering team for identifying usability problems, fixing them and validating changes with users.

Constantine (13) recommends integrating a lightweight version of usage-centered design with agile processes. A card based modeling and decision making approach is recommended, which uses index cards to document task cases that concentrate on user intentions and system responsibilities. Brainstorming sessions are recommended to identify and prioritize roles that users will play when interacting with the system and task cases (essentially minimalistic use cases) to support identification of the roles.

After roles and tasks have been identified, task cases are described in more details, using the standard use case notation, starting with the success cases and moving on to exception cases. All this documentation is done on index cards, in classic agile style. After details have been added, tasks are organized into groups based on the likelihood that they will be performed together under a common scenario or sequence of events. Each group of task cases is treated as a single interaction context (screen, dialog etc.) on the user interface. Paper prototypes are drawn and tested with users with the help of tasks defined in the task cases. Based on inspection results, the prototypes are revised, and using the latest prototypes and task cases, construction of the interface is begun.

This technique assumes the iterative nature of agile techniques, in that in every iteration, roles, task cases and paper prototypes will be refined. Also, along with these techniques, it is emphasized that a minimal up-front user interface design be performed, because it will benefit revision of the user interface in subsequent iterations. Things to keep in mind when creating the initial design are:

- A versatile navigation scheme that is common to different parts of the UI
- A consistent look-and-feel scheme across the interface
- Organization of parts of the user interface that fits the structure of user tasks

A visual representation of this process is shown in Figure 2.
4.3.2 Architecture Centric Design Method (ACDM)

So far I have not discussed the relationship between usability of a software product and its underlying software architecture. The common belief is that separation of the UI from the functionality is sufficient to support modifiability of the UI. But there are many manifestations of usability in a software product that are not supported by this separation of UI and functionality. In fact, there are many instances of usable features that have very little to do with the implementation of the interface (14), for example giving the user the option to cancel an operation, or showing them real time progress status of an operation.

The referenced study has identified 27 such common usability scenarios that have an impact on the architecture of software. Each scenario is mapped onto a number of software architectural patterns that need to be applied to the architecture in order for the scenario to be implemented. The study shows that there is a large overlap between patterns for different scenarios, thus implementing a solution for a particular scenario promises more than one usability benefit.

ACDM (15) is an architectural centric software design methodology that treats the architecture and quality attributes of the software as the centerpiece for all design activities. This methodology is perfectly suited for ensuring usability at the architecture level. ACDM has 8 stages, sequential pairs of
which are part of the requirements, design/refinement, experimentation, and production phases. Standard usability techniques can be applied in the requirements stages (1 and 2), such that usability requirements are documented. ACDM prescribes documenting all requirements in an Architectural Drivers Specification document, which as far as usability is concerned, requires the team to document concrete quality attribute scenarios. Thus quality attribute scenarios will ensure usability requirements are measurable, testable and that the team can identify which architectural patterns to use when creating the architecture is stages 3 and 4.

Furthermore, stage 6 of ACDM recommends creating technical prototypes (ACDM uses the term experiments instead of prototypes) so that the team is able to evaluate and refine the architecture. With regards to usability, experiments can be created that test high priority and/or high difficulty usability scenarios, such that technical implementations of such important scenarios can be evaluated.

4.4 Usability engineering roles
Along with defining a plan for usability activities in a development process, it is also important to define roles related to usability activities. Anderson (8) recommends defining 3 roles; usability engineer, user interface designer and usability evaluator. It is not necessary to have a different person take each role; one person can perform more than one role.

The usability engineer has the primary responsibility of gathering and analyzing user requirements, and documenting them using use cases or other notations. These activities differ from regular requirements gathering in that the usability engineer should pay special attention to characteristics and needs of target users. The role’s goal is to capture the end users’ mental model of the tasks they perform.

The user interface designer creates the user interface based on the requirements and usability principles. The role also evaluates the usability of design prototypes. The role’s goal is to express the users’ mental model as closely as possible within the constraints of the information and technical architecture.

The usability evaluator is primarily responsible for testing the usability of designs, analyzing the results and communicating the results and recommendations back to the team. These tests should be performed in the early stages of UI design. This role will be responsible for arranging usability tests and other forms of evaluation.

5 Affordable integration of usability
Having looked at a substantial number of usability techniques that can be integrated in the development process of software, it would be useful to look at a subset of tasks that studies have shown to be very cost effective in their simplified versions. A number of references (7) & (6)) recommend using prototyping, heuristic analysis and simple forms of usability testing for achieving substantial improvements in usability while keeping costs to a minimum.

Krug recommends that as long as you start prototyping and testing early in the lifecycle, even testing with 1 user is 100% better than not testing at all. Nielsen’s studies also show 3 users as being an optimal
number for cheap usability tests, in that they give the highest return on investment. It is also recommended to not spend too much time and effort in trying to recruit a representative set of users for usability tests. Depending on the nature of the application, in most cases, general users can give provide good results in usability tests, especially for commercial products. Nielsen highly recommends using heuristic analysis, even in the absence of usability experts, because it does not require customer input, high fidelity design specifications or large amounts of time. Thus it is a very cheap and quick technique to have in a team’s toolset, which can help in ensuring the design is created keeping accepted usability principles in mind.

One of the most costly usability techniques is usability testing. Both Krug and Nielsen emphasize the fact that good results can be captured from usability testing by using a minimal of equipment. Expensive usability labs are not required. In most cases a computer setup to a video camera can suffice for recording of the test, and quick prototypes can be drawn on paper. Nielsen even recommends getting rid of recording all together and relying on notes taken by a team of evaluators. This can drastically reduce to the cost of purchasing and maintaining the recording equipment and the time that is spent analyzing the recordings.

It is important to realize that to get the most out of usability techniques; they should be done early and iteratively. This ensures that cost of rework is minimal throughout the different stages in the lifecycle. If we look at the data in tables 4 and 5, we can see that iterations and prototyping are two of the most commonly used and most fruitful techniques related to usability. Nielsen’s studies also show a 20% improvement in the usability of designs just by using these simple and cheap usability techniques. Thus companies looking for starting off with simple and cheap usability techniques should look into using these simple forms of prototyping, usability testing and heuristic analysis.

6 Conclusion
This report touches upon a number of different topics that are related to the integration of usability engineering techniques in a software development lifecycle. Defining how the team interprets usability is very important, since it encompasses a large number of different characteristics. Teams and individuals also need to be aware of potential problems they will face when introducing usability techniques in a development process. Management buy in and involving the engineers in defining usability goals are vital for ensuring usability is ensured in the development lifecycle, because it is not something that can be slapped on sometime during development.

Success teams should not only be familiar with a number of usability techniques, but they must also be able to quickly make decisions on which techniques are suitable for a given situation or stage in the development lifecycle. Additionally, all usability efforts and roles need to be planned from the start, so that usability of the software is ensured during all stages of the development lifecycle. For usability to be successfully integrated into the development process people need to realize that usability cannot be slapped onto the product at the end; it needs to be ensured throughout the lifecycle.
7 Bibliography


