Chapter #

IMPROVING THE SOFTWARE PROCESS WITH USABILITY ASPECTS

Xavier Ferre, Natalia Juristo, Ana M. Moreno

Universidad Politécnica de Madrid

Abstract: Software development organizations are paying more and more attention to the usability of their software products, due to the increasing importance attached to usability as a critical software quality attribute. The HCI (Human-Computer Interaction) field offers techniques aimed at producing a software product with the desired usability level, but their use is often not integrated into SE (software engineering) development processes. The integration of HCI techniques into SE practice is not an easy endeavor, since both fields speak different languages and they deal with software development from different perspectives. This chapter presents a characterization of selected HCI techniques and activities using SE terminology and concepts, according to what kind of activity they belong to and at what development stage their application contributes to a greater extent to the usability of the final software product. Software developers may then manage usability activities and techniques, include them in their schedule, and understand in which activities usability techniques have to be coordinated with SE techniques. The proposed characterization is aimed at software development organizations with a mature development process that are looking to enhance their process with usability aspects. The only requirement for the current software development process applied at the organization is that it be iterative, since this is a key characteristic for any effort directed at raising the usability level of the final software product.

Keywords: usability in the software process, user-centered process, software development process, HCI integration into the software process
1. INTRODUCTION

Usability is a quality attribute of software products that has been traditionally identified in SE as a graphical designer endeavor, because it is considered to be related to just the UI (user interface). However, usability is strongly related to the overall software structure and to the concept around which the system is built. The view of usability as just a UI issue has led to its relegation to a secondary role in the SE field. Usability concerns have been identified in SE practice with the development stage concerned with UI design. The main weakness of this approach is that the main decisions about the system have already been taken by the time this stage is reached. Therefore, the observed result of following this approach for years is that software systems have been developed that have a much lower usability level than would be desirable, or are not fit for use, or are too costly to improve.

On top of this, software development organizations have realized that the importance of usability in the perception of the overall quality of their software products is increasing, because the range of software systems users has widened tremendously over the last decade. Usability can provide an important competitive advantage in the tough software market. Therefore, software development organizations have identified the need to introduce the usability factor into their practices, just as other software quality factors are considered throughout the development of any software product. This shift towards usability integration throughout software construction is illustrated by the International Organization for Standardization’s (ISO) decision to include a new process, called usability process, in the standard for software processes, a very important issue in SE as described below. This change was introduced in the first amendment to ISO/IEC Standard 12207:1995, released in 2002 [ISO/IEC, 02]. The fact that an international standard stipulates that usability activities should be part of the software development process is an indication that HCI and usability are definitely on the SE agenda.

HCI is an established field and one of its main concerns is the usability of computer systems. HCI techniques are applied in a variety of software development projects, where attaining an acceptable usability level is a very important, if not the main, objective. These projects are developed mostly following methods particular to the HCI field. Where this is not the case, that is, when HCI practices are applied along with SE practices, their integration is tackled on a case-by-case basis. The main obstacle to HCI-SE cooperation is that both fields speak different languages and deal with software development from different perspectives [Constantine, 99]. HCI has a multidisciplinary essence, including topics related to fields like cognitive
psychology, ergonomics, and sociology. On the other hand, SE is defined in the IEEE Standard Glossary of Software Engineering Terminology as "the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software" [IEEE, 90]. Software engineers have traditionally focused on the internals of software, on its functionality, reliability, efficiency, and so on, and on the establishment of systematic software development practices. They have paid less attention to how the software product may better support the mental models of the user and the tasks he or she wants to perform. HCI processes are hard to apply in SE as they are, because of these differences in concept. Nevertheless, they cannot be applied independently from SE activities, because the two activity types overlap.

In particular, the aim of making software development systematic and disciplined has led the SE community to pay extra attention to the software process. Software process refers to the development roadmap followed by an organization to produce software systems, that is, the series of activities undertaken to develop and maintain software systems. Developers follow the software process established in their organization, which is enforced due to the underlying assumption that a good process leads to a good product. However, some activities are common to all software processes. The software process can be described at different levels, and the most common feature between different process descriptions is the description of the techniques applied in each process activity. Due to the emphasis placed on the software process in the SE community, a considerable amount of effort has gone into software process definition, evaluation, and improvement [Kawalek, 96] [Dermiane, 99] [Fuggetta, 00]. The goal of software process research is to improve software development practice by proposing: a) better ways of defining and modeling the development and therefore designing the developer organization processes, b) better ways of assessing the weaknesses of this organization, and c) better ways of improving this organization at the level of individual processes and the organization as a whole.

It should be noted that in response to the importance attached to the software process in software development practices, agile methods have appeared recently. They try to shift the focus to other issues like individuals and their interaction, and regard the software process as an important but secondary issue in software development. The Agile Manifesto [Beck, 02] sets out the main ideas behind this approach to software development. While the agile approach is receiving a significant amount of attention in the SE field, the vast majority of software developers still believe that having a defined process and improving it, as other production organizations do, is an approach that produces better quality software.
The goal of the work presented in this chapter is to offer a framework for the introduction of usability activities and techniques into any iterative software development process. This usability-enhanced process will be more capable of producing software products with the desired usability level.

The first obstacle to the introduction of usability techniques from HCI into the software process is the difference in process terminology between HCI and SE. Therefore, for our purpose of integration, we need to extract the essence, to look at the core ideas behind the terms to find the connections between both software development paradigms. We need to identify the motivations behind each activity to find their interrelationships.

Apart from the terminology breach, HCI does not regard the software process as formally as SE does, which results in HCI authors having different conceptions of the software process. Some authors do not structure usability efforts as activities (in the SE sense), so there are usability techniques that are not clearly assigned to activities in the HCI literature. The basic user-centered process (the HCI term for its process approach) is described in the ISO Standard 13407 [ISO, 99], but each author in the field has a particular vision of its translation into specific activities. For HCI-SE cooperation, usability techniques need to be mapped to the activities of a user-centered process.

A possible approach for integrating usability techniques into the software process is to take a specific SE process model and add usability techniques where applicable. This approach may be appealing to adept software process modelers. Nevertheless, linking the integration of usability into the process to a fixed software process could be far too demanding for some organizations, leading to less extensive use of the results of the work presented here. Software development organizations with a consolidated development process are unlikely to switch it for a completely different process, even if it promises some improvements (for example, in the usability field). Therefore, instead of selecting a generic process that would force the organization interested in using our research results to change its process, we think it would be more useful to set out the minimum conditions to be met by a software process that supports usability. The proposal for improving the process with usability aspects will then be described in a general enough way for its application to each process that meets the minimum requirements established. This approach will increase the practical applicability of the resulting development process. The appeal for a software development organization lies in the fact that it does not have to abandon the in-house process to adopt some improvements, as it is enough to just modify the existing process.

This work is directed at software development organizations with a strong SE background that are considering the incorporation of usability
aspects into their practices, and cannot shift to a strictly HCI-led development approach. For these organizations, usability is an important concern, but not the main focus. Even if there is a usability department at the organization, it is not especially big, and usability-aware personnel are needed on each development team to collaborate with the usability experts. This work may be useful for such developers addressing usability from the SE point of view in their effort to integrate usability into their existing software process.

The following section analyses the state of the art on the integration of usability techniques in the software process. Section 3 details the mapping between usability and SE activities. Section 4 presents the assignment of the selected usability techniques to activities. Section 5 deals with the considerations on when to apply usability techniques and activities in an iterative development. Other issues to be considered for the integration of usability into the software process are highlighted in section 6 and, finally, section 7 presents the conclusions.

2. STATE OF THE ART

In this section, we will present the main proposals related to the integration of HCI aspects into the SE process. We will consider first pure HCI processes and how they can be accommodated to a SE process. Then we will detail the proposals concerning an HCI process that provides links to SE. Finally, we will present the proposals based on the integration of HCI aspects into SE process models.

2.1 Pure HCI Processes and their Relationship to the Software Process

In this section, we will consider two pure HCI approaches to the software process and how they deal with integration with the overall software development process: The ISO Standard 13407 for Human-Centered Design Processes for Interactive Systems, and the Star Life Cycle proposed by Hix and Hartson.

2.1.1 ISO Standard 13407:1999

The ISO Standard 13407 [ISO, 99] provides guidance on human-centered design activities throughout the life cycle of computer-based interactive systems. Note that the term human-centered used in the standard is
equivalent to user-centered. We prefer the latter term, since it is the most widely used in the HCI literature.

The standard reasons why a user-centered focus should be adopted in interactive systems, and it includes the characteristics of such a focus: active involvement of users and clear understanding of user and task requirements; an appropriate allocation of function between users and technology; the iteration of design solutions; and multidisciplinary design. It also describes the essential activities in a human-centered process: understand and specify the context of use; specify the user and organizational requirements, produce design solutions and evaluate designs against requirements.

The standard also establishes that the human-centered process, including the procedures for integrating the usability activities with other system development activities, e.g. analysis, design, testing, has to be planned, although this is as far as it goes. This requirement calls for the development of usability roadmaps that are useful for fitting usability techniques into the overall software development process.

2.1.2 Star Life Cycle

The star life cycle by Hix & Hartson [Hix, 93] is a user-centered process that sets out the main HCI activities. It does not prescribe a particular order for activities, but it allocates a prominent role to usability evaluation, which is placed in the center of the star that represents the activities in the life cycle. The model includes two activities (Deployment and Software Production), which are considered by the authors as parts of the software process that are in the constructional domain and are not discussed in the book. Therefore, the part of development not related to usability (mainly implementation according to Hix and Hartson) is very small compared to what the authors consider to be usability-related activities.

Hix and Hartson describe the communication paths that should take place between usability activities (user interaction design) and software design. They strictly separate the development of the UI from the development of the rest of the software system, with two activities that connect them: systems analysis and testing/evaluation. The systems analysis group feeds requirements to both the problem domain design group and the user interaction design group. It is a simplistic approach to HCI-SE integration, but the authors acknowledge that “research is needed to better understand and support the real communication needs of this complex process” [Hix, 93].
2.2 HCI Processes that Provide Links to the Overall Software Process

While there is no strict separation between the two process proposals examined in this section (usage-centered design and the usability engineering lifecycle) and the ones discussed above, the following two proposals provide some more guidance on the issue of integration of their process proposals into the overall software development process.

2.2.1 Usage-Centered Design

Constantine and Lockwood [Constantine, 99] propose a collection of coordinated activities that contribute to usability, in the form of a method called usage-centered design. Their usage-centered design activity model includes some activities that correspond to the larger software development process (object structure design, concentric construction and architectural iteration), along with pure-usability activities, like task modeling or interface content modeling. The models that Constantine and Lockwood propose are appealing to software engineers, since they are closer than other HCI techniques to the kind of modeling used in SE. In particular, essential use cases, which are a cornerstone of the usage-centered approach, are a reinterpretation of the popular object-oriented technique of use cases. They can, therefore, serve the purpose of acting as a bridge between SE and HCI models. In fact, there are at least two popular SE reference books ([Larman, 01] and [Cockburn, 01]) that acknowledge Constantine and Lockwood’s work on essential use cases.

Constantine and Lockwood offer some advice for integrating usability and UI design into the product development cycle, acknowledging that there is no one single way of approaching this introduction. Therefore, they leave the issue of integration to be solved on a case-by-case basis. They state that “good strategies for integrating usability into the life cycle fit new practices and old practices together, modifying present practices to incorporate usability into analysis and design processes, while also tailoring usage-centered design to the organization and its practices” [Constantine, 99]. Although some techniques that are closer to SE modeling are described, Constantine and Lockwood’s proposal is not formalized in process terms, and their work is more concerned with detailing the techniques than with specifying the process in terms of dependencies, products and roles.
2.2.2 Usability Engineering Lifecycle

Deborah Mayhew [Mayhew, 99] proposes the Usability Engineering Lifecycle for the development of usable UIs. The process structures the activities into three phases: Requirements Analysis, Design / Test / Development, and Installation. This approach to the process follows a waterfall lifecycle mindset: an initial Analysis phase, followed by a Design / Test / Development phase, and finally an Installation phase. The Analysis stage is only returned to if not all functionality is addressed, and this is, therefore, not a truly iterative approach to software development.

Nevertheless, it is one of the more complete HCI processes from the SE point of view. It focuses on HCI concepts and its approach to software development. Although Mayhew claims that the method is aimed at the development of the UI only, the activities included in this life cycle embrace an important part of requirements-related activities (like, for example, Contextual Task Analysis). Links with the OOSE (Object-Oriented Software Engineering) method [Jacobson, 93] and with rapid prototyping methods are identified, but the author acknowledges that the integration of usability engineering with SE must be tailored and that the overlap between usability and SE activities is not completely clear. The links with OOSE and rapid prototyping are very general, and Mayhew presents UI development as an activity that is quite independent from the development of the rest of the system.

Additionally, the author surprisingly defines software engineering as “an approach to software development that involves defining application requirements, setting goals, and designing and testing in iterative cycles until goals are met” [Mayhew, 99]. Even though this is the current trend in SE, it is not a valid definition of the discipline. A software engineer may be discouraged by such misconceptions, when approaching this work in search of help with the issue of usability integration into the software process.

2.3 Integration Proposals Based on SE Methods

Three proposals for integration will be considered in this section. They have a similar aim to ours, that is, to integrate usability techniques into an existing software development process. Costabile's proposal is based on the waterfall lifecycle. MUSE (Method for USability Engineering) is defined according to the characteristics of a structured method. Finally, we examine the User Experience addition to the RUP (Rational Unified Process®).
2.3.1 Costabile’s Approach

Costabile [Costabile, 01] offers a way of integrating user-centered practices into the software process to increase the usability of the software product. She condenses the user-centered approach into three main principles: analyze users and tasks, design and implement the system iteratively through prototypes of increasing complexity and evaluate design choices and prototypes with users. Costabile proposes a way of modifying the software life cycle to include usability. The basis she takes for such modifications is the waterfall lifecycle. The proposal adds two extra activities composed of pure usability activities –user and task analysis, on the one hand, and scenarios and UI specifications, on the other–, plus two intermediate activities which include the same tasks: prototyping and testing. It is possible to go back to a previous phase from any phase of the life cycle. According to the author, these backtracking paths, along with the two extra activities, emphasize the iterativeness of software development, which is necessary from a user-centered point of view.

Costabile’s proposal has an important drawback in the choice of the waterfall life cycle as a “standard” software life cycle. This model goes against the user-centered aim of evaluating usability from the very beginning and iterating to a satisfactory solution. Paths that go back in the waterfall life cycle are defined for error correction, not for completely changing the approach if it proves to be wrong, since it is based on frozen requirements [Larman, 01]. Glass [Glass, 03] acknowledges that “requirements frequently changed as product development goes under way [...]. The experts knew that waterfall was an unachievable ideal”. Glass also states that the waterfall life cycle has been dismissed in most parts of the software world. Larman identifies the following problems with the waterfall life cycle [Larman, 01]: delayed risk mitigation, speculation and inflexibility of requirements and design, high complexity and low adaptability. Iterative development tackles most of these problems.

2.3.2 MUSE (Method for USability Engineering)

MUSE [Lim, 94] is a method for designing the UI, and was one of the most structured usability method when published (1994). It is divided into three phases: Information Elicitation and Analysis Phase, Design Synthesis Phase and Design Specification Phase. The method aims to ease the integration with software engineering methods, and its integration with the JSD (Jackson System Development) method is described. The primary focus of the MUSE method is on design specification due to the identified lack of
integration in this stage, whereas, according to the method creators, later stages (usability evaluation) are well covered in the existing literature.

MUSE is based on the principle of delaying design commitment, ensuring that detailed design is preceded by appropriate design analysis and conceptual definition. Comparing MUSE with the rapid prototyping approach, Lim and Long [Lim, 94] state that MUSE, as a structured method, emphasizes a design analysis and documentation phase prior to the specification of a "first-best-guess" solution. Therefore, MUSE follows a waterfall life cycle, which is an obstacle to the application of a truly iterative approach.

As MUSE is a structured method, it is presented by its authors as easy to integrate into any structured SE method. Its integration with JSD is detailed as an example of this. JSD is presented as a method that is mainly used for the development of real-time systems. Real-time systems account for a very small part of interactive systems, so the integration of MUSE with JSD is not very useful from a generic point of view. Regarding the integration of MUSE with other SE methods, its usage of techniques like structured diagrams or semantic nets makes it difficult to adapt to other SE practices, in particular to object-oriented development.

The BIUSEM project [BIUSEM, 95] applied MUSE to three software development projects in different domains and with different SE methods to evaluate its applicability. Despite the positive outcome of the project (the application of MUSE improved the product quality, and the sharing of human factors insight with software engineers helped to elicit user-centered requirements), the project team acknowledged that "the body of published papers and the book describing MUSE are unnecessarily complicated and act as a deterrent to its wider use" [BIUSEM, 95].

2.3.3 User Experience plug-in for the RUP (Rational Unified Process®)

The Unified Process [Jacobson, 99] is the process that is currently receiving the greatest attention in SE, since it is sponsored by the main object-oriented methodologists: James Rumbaugh, Ivar Jacobson and Grady Booch. It advocates a truly iterative approach. It denotes the activities that the process encompasses as "disciplines" to avoid the typical identification between activity types and process stages in the waterfall life cycle. Of the processes that actually have an iterative approach, RUP is the most widely used. The approach to usability integration presented in this section is not comparable to the above in scope. It has been included, however, because of the current relevance of the Unified Process in SE.
The Unified Process does not consider usability directly, but it is use-case driven, and use-case modeling has some similarities with HCI task modeling. Therefore, use cases could be used as a starting point for usability integration into the software process. However, the use-case model in the Unified Process plays a secondary role compared to system architecture. The use-case model is very important in cycle planning, but once the cycle starts, use cases are regarded as a preliminary version of elements of the internal functionality design. When design elements are labeled as use-case realizations, we are shifting use cases to the design world and, therefore, away from the user realm, losing most user-centered advantages with that shift.

The User Experience (UX) [Rational, 02] plug-in for RUP aims to integrate the work performed in the web development domain regarding the development of the web system concept, which usually drives the whole development, into RUP. It is based on Jim Conallen's work on web modeling [Conallen, 03], and there are big similarities between UX aims and classical HCI concerns. According to Conallen, the term User Experience “is used to describe the team and the activities of those specialists responsible for keeping the UI consistent with current paradigms and, most important, appropriate for the context into which the system is expected to run” [Conallen, 03]. Despite this promising definition, Conallen's work focuses on modeling, and he describes the artifacts for which the UX team is responsible as follows: screens and content descriptions, storyboard scenarios, and navigational paths through the screens.

Although it is an advance towards the aim of integrating usability into the software process, the UX addition to RUP does not cover the entire process and is limited to a few models. Nevertheless, it does indicate the growing interest in the web development domain for integrating usability expertise and techniques into the development process.

2.4 Summary of Proposals

As a summary of the state of the practice presented, Table 1 describes the main characteristics of each proposal in terms of the integration of usability into the software process. The Iterativeness column details whether the approach is truly iterative or, on the contrary, whether it encompasses a waterfall lifecycle approach to software development. The Formality column details whether the process just describes high-level activities or stages, along with their dependencies, or whether it describes fine-grain activities, their dependencies at this detailed level, along with the techniques for application in each activity and the products output. The Coverage column
Chapter 

refers to the aim of the method according to its authors: either it aims to cover the development of the whole interactive system, or it is presented as a method for the design of just the UI. The Integration column details how the proposal deals with integration with the overall software development process. Finally, the User-centeredness column shows which proposals can be considered as following the usability approach for software development advocated in the HCI field and are, therefore, user-centered. Note that we have not considered the issue of iterativeness for the rating in this last column.

Table #1. Comparison of User-Centered Software Process Proposals with regard to their Integration with the Overall Software Development Process

<table>
<thead>
<tr>
<th>Method</th>
<th>Iterativeness</th>
<th>Degree of Formality</th>
<th>Coverage</th>
<th>Integration</th>
<th>User-centeredness</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO 13407 [ISO, 99]</td>
<td>Iterative</td>
<td>High level activities</td>
<td>Whole system</td>
<td>Integration with other development activities must be planned</td>
<td>User-centered</td>
</tr>
<tr>
<td>[Hix, 93]</td>
<td>Iterative</td>
<td>High level activities</td>
<td>Design of the UI</td>
<td>General indications</td>
<td>User-centered</td>
</tr>
<tr>
<td>[Constantine, 99]</td>
<td>Iterative</td>
<td>High level activities</td>
<td>Whole system</td>
<td>General indications</td>
<td>User-centered</td>
</tr>
<tr>
<td>[Mayhew, 99]</td>
<td>Nominally iterative but waterfall</td>
<td>Fine grain activities</td>
<td>Design of the UI</td>
<td>Dependencies with OOSE and phases of a rapid prototyping method are described</td>
<td>User-centered</td>
</tr>
<tr>
<td>[Costabile, 01]</td>
<td>Nominally iterative but waterfall</td>
<td>High level activities</td>
<td>Whole system</td>
<td>Integrated on the waterfall life cycle (on which it is based), but without details</td>
<td>User-centered</td>
</tr>
<tr>
<td>MUSE [Lim. 94]</td>
<td>Nominally iterative but waterfall</td>
<td>Fine grain activities</td>
<td>Design of the UI</td>
<td>Detailed with JSD</td>
<td>User-centered</td>
</tr>
<tr>
<td>RUP with UX plug-in [Rational, 02]</td>
<td>Iterative</td>
<td>Fine grain activities</td>
<td>Whole system</td>
<td>Detailed (it is a SE process with a minor usability addition)</td>
<td>Partially user-centered</td>
</tr>
</tbody>
</table>

3. MAPPING BETWEEN USABILITY AND SE ACTIVITIES

To map usability terminology to the SE terminology, the activities that form part of a user-centered process must be identified. An obstacle to this aim is the heterogeneous landscape of methods and philosophies offered by the HCI field, like, for instance, usability engineering, usage-centered design, contextual inquiry, and participatory design. Each author attaches importance to a few techniques, and the terminology may vary from one author to another. For this reason, we have surveyed the HCI literature [Ferré, 02] to identify the most agreed upon usability activities that should be part of the software development process. We have listed these usability
activities in Figure 1, grouped according to what type of development activity they belong to. Note that there is no assumption of precedence in time between the different activities.

There is a high degree of consensus in HCI regarding analysis activities. Specification of the Context of Use is an activity whose aim is to understand and record the implications of the context of use so that they can be considered during system design. We have named it following the ISO 13407 Standard terminology [ISO, 99], and it is divided into User and Task Analysis because some authors differentiate between both activities ([Mayhew, 99][Hix, 93][Constantine, 99]). Usability specifications are quantitative usability goals, which are used as a guide for ascertaining when a system has the proper usability level. They can be considered non-functional requirements.

Design activities are less defined in the HCI literature consulted. The only activity cited by most authors is Prototyping. Prototypes are widely used in SE, in particular related to iterative development, but what HCI may offer is the particular usage of prototyping in order to get greater degrees of user involvement, and to consider alternative designs. The most useful prototypes for this purpose are the less elaborate ones, such as paper prototypes.

Develop the Product Concept is based on mental models ([Norman, 90], [Preece, 94]): When the product concept is vague, ambiguous, inconsistent or obscure, there will be a divergence between the user mental model of the system and the design model that developers work with. It is stressed the importance of helping the user to grow productive mental models for the usability of the system. Good designers always bear in mind a certain product concept, but making it explicit and highlighting its importance in the software development process will help to shape the system in a way that explicitly communicates this product concept to the user.

Interaction Design varies considerably between authors, but we have identified as a common aim in the design process the definition of the interaction that will take place between the user and the system. It includes designing in detail the user-system dialogue, that is, the sequence of actions needed to operate the system, and the user-system information exchange. Apart from tackling UI design (the design of the elements of the UI that will make the interaction possible), it also includes decisions that affect the internal logic of the system, to the extent that this internal logic is reflected in the user-system interaction.

Usability evaluation is the activity that is more profusely detailed in HCI literature. Usability is a very complex concept, due to the complex nature of humans. Without doing some form of evaluation, it is impossible to know whether or not the design or system fulfils the needs of the users and how
well it fits the physical, social and organizational context in which it will be used [Preece, 94]. Usability evaluation is a core part of iterative development, in the sense that evaluation activities can produce design solutions for application in the next design cycle or, at least, more insight into the nature of the interaction problem at hand. Therefore, evaluation is not seen in HCI as a mere fail/pass test but as a part of development. Three big families have been highlighted within the Usability Evaluation activity in Figure 1: Expert Evaluation, Usability Testing and Follow-Up Studies of Installed Systems.

The set of activities is based on HCI terminology, with which most software developers are not familiar. Therefore, the terms must be translated to a generally accepted SE terminology, so that developers know where to plug in the usability additions to the software process. Wherever possible, the SWEBOK (Software Engineering Body of Knowledge) [IEEE, 01] has been used as a basis for defining the activities in a traditional software development process. For other activities that are new to SE and do not fit any existing activity, HCI terminology has been used.

The mapping of the usability activities to the development activities considered in this chapter is shown in Figure 1. Each usability activity on the left-hand side of Figure 1 is mapped to a development activity on the right by means of an arrow. Some activities have been added to the usual SE activities, because they do not match an existing SE activity. They are highlighted in italics (for example, Interaction Design). Only activities that are affected by usability are represented on the right, being not present the rest of activities in a software process.

Regarding the type of activities related to analysis, note that usability activities are intertwined with standard analysis activities. Therefore, they can be directly mapped to the different types of SE analysis efforts. Following the SWEBOK definitions, we have selected the requirements activities that are likely to be enhanced by the introduction of usability techniques: Requirements Elicitation and Analysis, Requirements Specification and Requirements Validation. Four activities, presented in the HCI literature as being necessary for understanding users, their context and their needs, have been highlighted within the Requirements Elicitation and Analysis activity: User Analysis, Task Analysis, Develop Product Concept and Prototyping.

The activities of Develop Product Concept and Prototyping are considered differently in HCI and in SE. According to the SWEBOK, Prototyping is considered in SE as a technique that can be used in Requirements Elicitation and Validation. As for Develop the Product Concept, it is a design activity, but the kind of design that is known as invention design. According to the SWEBOK, invention design is usually
performed by systems analysts with the objective of conceptualizing and specifying a system to satisfy the discovered needs and requirements, and they are not addressed in the chapter of the SWEBOK devoted to software design [IEEE, 01]. This conceptualization activity is usually undertaken as part of requirements elicitation activities, and is fundamental for the success of requirements engineering efforts. Because of its close connection with requirements activities and because the SWEBOK considers invention design as part of the requirements analysis activity, we have considered Develop the Product Concept as part of Requirements Elicitation and Analysis in our framework.

---

**Usability Activities**

**Analysis Activities**
- Specification of the Context of Use
  - User Analysis
  - Task Analysis
  - Usability Specifications

**Design Activities**
- Develop Product Concept
- Prototyping
- Interaction Design

**Evaluation Activities**
- Usability Evaluation
  - Expert Evaluation
  - Walkthroughs
  - Usability Testing
  - Follow-Up Studies of Installed Systems

---

**Development Activities affected by Usability**

**Analysis (Requirements Eng.)**
- Requirements Elicitation and Analysis
  - User Analysis
  - Task Analysis
  - Develop Product Concept
  - Prototyping
- Requirement Specification
- Requirements Validation

**Design**
- Interaction Design

**V & V**
- Usability Evaluation
  - Expert Evaluation
  - Usability Testing
  - Follow-Up Studies of Installed Systems

---

*Figure #1. Mapping between Usability and SE Activities*
Usability-related design activities are quite independent from general design activities. Therefore, a new activity, called Interaction Design, has been included under the Design activities. The SWEBOK considers UI Design not as part of SE but as a related discipline. However, it also states that UI design deals with specifying the external view of the system and that it should be considered as part of requirements specification. Nevertheless, the chapter devoted to requirements in the same source (the SWEBOK) does not include UI. On the other hand, Interaction Design fits the definition provided by the IEEE Standard Glossary of Software Engineering Terminology for design: "the process of defining the architecture, components, interfaces, and other characteristics of a system or component and the result of that process" [IEEE, 90]. Therefore, we have considered Interaction Design as a design activity, because it is not clearly located in the SWEBOK and because it fits the general definition of design.

Regarding evaluation, a new activity, Usability Evaluation is created, since it groups usability techniques that are independent from other general evaluation activities. However, walkthroughs can be used during requirements validation, so they have been highlighted within Expert Evaluation (on the left of Figure 1) to show this link with analysis-related activities. Evaluation activities are termed V&V (Verification and Validation) in SE, so this is the label used for evaluation activities.

After having matched usability activities to their respective SE activities, we need to address the individual techniques to be employed in each activity.

4. ASSIGNMENT OF USABILITY TECHNIQUES TO ACTIVITIES

For developers to be able to apply usability techniques, they need to know in which activities they are applied. The previous section matched the activities in the usability literature to their respective SE activities.

Note that the usability framework presented in this chapter is aimed at software development organizations that do not have a big HCI department, and, therefore, need usability concerns to be shared with the developers throughout development, even if developers do not have extensive HCI knowledge. Nevertheless, for organizations where usability expertise is widely available, communication problems between the HCI and the SE team are an important concern, and this framework would also be of interest in such cases.

Bearing in mind that, in our approach, the usability techniques could be applied by non-experts in HCI, we have made a selection where there was more than one usability technique available with the same objective and
have included usability techniques that are less alien to a SE mindset in our framework. From more than 80 techniques described in the HCI literature [Ferré, 02], the resulting set of techniques has been reduced to just 36 techniques. They appear in the column furthest to the right in Table 2, which is explained in the next paragraph.

We have used the definition of each technique in the HCI literature as regards its application in a particular activity to allocate usability techniques to activities, and this definition has again been compared with the definition of activities in the SWEBOK. Table 2 shows the classification of usability techniques according to activities. The techniques are grouped according to the activities in a generic software development process that are listed in the central column.

**Table 2. Allocation of Usability Techniques to Activities**

<table>
<thead>
<tr>
<th>HCI Activities</th>
<th>Activities in Software Process</th>
<th>Usability Techniques</th>
</tr>
</thead>
</table>
| Analysis - Specification of the Context of Use - User Analysis | Analysis - Requirements Analysis and Elicitation | Ethnographic Observation [Preece, 94]  
Contextual Inquiry [Beyer, 98]  
Structured User Role Model [Constantine, 99]  
Operational Modeling [Constantine, 99]  
JEM (Joint Essential Modeling) [Constantine, 99] |
| Analysis - Spec. Context of Use - Task Analysis | | Essential Use Cases [Constantine, 99]  
GOMS* [Kieras, 97]  
Affinity Diagrams [Beyer, 98]  
Visual Brainstorming [Preece, 94]  
Competitive Analysis [Nielsen, 93]  
Scenarios [Carroll, 97a]  
Prototypes (paper and chauffeured [Constantine, 99]; and wizard of Oz [Preece, 94]) |
| Design - Develop Product Concept | | |
| Design - Prototyping | Requirement Specification | Usability Specifications [His, 93] |
| Analysis - Usability Specifications | Requirements Validation | Cognitive Walkthrough [Carroll, 97b]  
Pluralistic Walkthrough [Bias, 94] |
| Usability Evaluation - Expert Evaluation | | |
GOMS* [Kieras, 97]  
Screen Pictures [His, 93]  
Card Sorting [Robertson, 01]  
Menu-selection Trees [Shneiderman, 98]  
Navigational Paths [Conallen, 03]  
Product Style Guide [Mayhew, 99]  
Impact Analysis [His, 93]  
Help Design by Use Cases [Constantine, 99] |
| Design - Interaction Design | | |

* GOMS can be applied as part of Requirements Analysis (the part concerning goals and operators), but its full application would be as part of Interaction Design.
5. CONSIDERATIONS ON STAGES IN DEVELOPMENT

We examined the HCI literature to identify what characteristics a software development process should have for it to be considered user-centered. [Preece, 94], [ISO, 99], [Constantine, 99], [Hix, 93], [Shneiderman, 98] and [Nielsen, 93] agree on considering iterative development as a must for a user-centered development process. The other two characteristics that are mentioned by several sources are: active user involvement and a proper understanding of user and task requirements. These two conditions can be met by introducing usability techniques that can help software developers to integrate users into the design process and to enhance requirements activities with specific usability aspects. On the other hand, the first condition (that is, iterativeness) is an intrinsic characteristic of a software process, and needs to be stated as a requirement for an existing development process to be a candidate for the introduction of usability techniques and activities.

The HCI practices described in the literature are deeply rooted in this process characteristic and, for the application of usability techniques, there are indications on when in development time each technique yields the most useful results for improving the usability of the final product. These indications on the best time to apply usability techniques have to be
transmitted to developers. Hence, it is not enough just to assign usability techniques to development activities, extra guidance also needs to be provided on what usability techniques are to be applied concerning exactly when in development time. Consequently, the activities and their techniques need to be interrelated with development stages. For this purpose, we will now present a generic description for the stages of any process based on iterative development and then actually interrelate activities / techniques and stages.

5.1 Stages in an Iterative Development Process

Different times or stages can be defined in an iterative process, where one and the same activity may be more or less important or have a different meaning. For instance, the discovery role is more predominant in the analysis activities that are performed during the early stages than in analysis activities carried out during subsequent stages, where a specific part of functionality is being analyzed and an understanding role takes over.

Even though each iterative process has its particular approach and terminology in terms of development stages, they usually follow a similar pattern in this respect. This pattern is represented in Figure 2. Each stage is represented by a cloud, because it is not a development phase as in the waterfall life cycle, but a set of iterations organized according to the moment in time represented by the x-axis.

![Figure 2. Stages in an Iterative Software Development Process](image)

An explanation of the stages in Figure 2 follows:

- Elaboration cycles: This stage represents the early efforts in the software development process, where the problem is delimited
and the basic information is gathered for later development in the iterative cycles.

- Iterative cycles (i): These are the iterations found in any iterative approach. For usability techniques to be applied in the cycles, a distinction will be made between two moments:
  - Central moments: The main part of each cycle.
  - Final moments: The last part of each cycle, where certain activities are performed, typically V&V activities.
- Evolution cycles: These iterations represent the cycles that are undertaken after the system has been installed and is operational at the customer’s site.

5.2 Time Constraints for Usability Technique Application

Apart from the activity of which they are part, the description of usability techniques in the HCI literature includes indications on the moments in development time when it is to be applied with more or less emphasis. This section details this information, organized according to the stages in a generic iterative process presented in the previous section.

5.2.1 Techniques to be Applied at the Elaboration Stage

Before the actual iterative cycles begin, there must be an initial effort where the needs are identified and the general scheme that the system will follow is established. A general aim is that the products of this stage are quite stable, even though they are open to changes in the iterative development cycles.

The following techniques are clearly to be applied at elaboration time, because they are good for approaching the problem for the first time or handling a solution that is not well defined: Ethnographic Observation, Contextual Inquiry, Affinity Diagrams, Scenarios, Visual Brainstorming, and Paper and Chauffeured Prototypes.

Competitive Analysis can be applied later on, but it can help at elaboration time because it is good for coming up with design ideas on the product concept.

Modeling the user and his or her environment, and the basic dialogue between the system and the user is a prerequisite for any development that aims to care about the user and about the usability of the resulting product. For this reason, the following techniques should be applied at elaboration time, even though they may be applied later for refining the products:
- Essential Use Cases: If there are a lot of use cases, they do not all need to be described. It suffices to detail the main ones to assure that elaboration is not too time consuming.
- Structured User Role Model
- Operational Modeling
- JEM (Joint Essential Modeling)
- Cognitive and Pluralistic Walkthrough: Walkthroughs evaluate an interaction dialogue. So, as soon as these dialogues are defined in the essential use cases, walkthroughs can be applied as an evaluation technique.
- Heuristic Evaluation: Low fidelity prototypes and early designs of the UI may be evaluated heuristically.

The specifications document should include Usability Specifications. So, this technique will be applied at elaboration time if such a document is created at this stage, but it can be completed as development advances.

Techniques related to UI design can be applied at the Elaboration stage, because the UI is the part of the implementation that the user can understand better. Its design may be undertaken at the early stages of the development in order to get feedback from the user. Thus, even though these techniques will have more weight in the iterative cycles, they are also present at Elaboration stage. These techniques are Detailed Use Cases, Screen Pictures, Card Sorting, Menu-selection Trees. Only Navigational Paths has a predominant role at the Elaboration stage, since it is good for describing the high-level view of the navigation.

### 5.2.2 Techniques to be Applied in the Iterative Cycles (i)

There are other techniques that, even though they could fit in well in elaboration cycles, require a greater effort than the ones detailed above in the Elaboration stage section. They are predominantly to be applied in iterative cycles, in order to avoid bulky elaboration cycles. These techniques are GOMS, which requires a detailed description of means for performing an operation, and the Product Style Guide, which requires a considerable amount of effort to complete.

Prototyping techniques that demand some implementation, such as Wizard of Oz Prototypes, may be employed at Elaboration time, but they fit better in iterative cycles. Help Design by Use Cases is also most useful in iterative cycles.

Some other UI design techniques named in the previous section have some more weight during iterative cycles, but they are fit for both stages (Elaboration and Iterative Cycles): Detailed Use Cases, Screen Pictures, Card Sorting, and Menu-selection Trees.
Impact Analysis may be employed at the beginning of any cycle in either the iterative or evolution cycles.

Some techniques are adequate for application at the end of a development cycle, that is, in the final moments. They are the ones proposed in the literature for usability evaluation purposes:
- Heuristic Evaluation
- Usability Inspections: Consistency, conformance and collaborative usability inspections.
- Thinking Aloud: Constructive interaction, retrospective testing, critical incident taking, and coaching method.
- Performance Measurement
- Laboratory Usability Testing
- Post-Test Feedback / User Questionnaires

5.2.3 Techniques to be Applied at the Evolution Stage

This moment in development time groups the activities performed after the system has reached initial operational capability in the customer organization. The usability techniques to be applied at this time are techniques to evaluate the usability of an installed system. They are as follows:
- Questionnaires / Surveys (questionnaires may be used in previous stages as well)
- Structured and Flexible Interviews
- Direct Observation
- Video / Audio Recording (it can be used in previous stages as well)
- Focus Groups
- Logging Actual Use: Time-stamped keypresses and interaction logging.
- Online User Feedback Facilities: Online or telephone consultants, online suggestion box or trouble reporting, online bulletin board or newsgroup, user newsletters and conferences.

5.3 Mapping of Usability Activities / Usability Techniques / Development Stages

The description of the techniques to be applied at each stage in the previous section is summarized in Table 3. Techniques highlighted in bold face within a stage carry more weight in this stage, that is, even though they can be applied at other stages, this is the stage where they fit better.
Table 3. Usability Techniques to Be Applied at Each Stage and their Significance

<table>
<thead>
<tr>
<th>Activities</th>
<th>Stages (cycles 1 to i)</th>
<th>Iterative Cycles (cycles 1 to j)</th>
<th>Evolution Stage (cycles 1 to k)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Elaboration Stage</td>
<td>central moments</td>
<td>final moments</td>
</tr>
<tr>
<td></td>
<td>(cycles 1 to i)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Iterative Cycles</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(cycles 1 to j)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Evolution Stage</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(cycles 1 to k)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reqs. Eng.</td>
<td>Requirements Elicitation and Analysis</td>
<td>- Ethnographic Observation</td>
<td>- Competitive Analysis</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Contextual Inquiry</td>
<td>- Essential Use Cases</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Affinity Diagrams</td>
<td>- Structured User Role Model</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Visual Brainstorming</td>
<td>- Operational Modeling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Competitive Analysis</td>
<td>- JEM (Joint Essential Modeling)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Scenarios</td>
<td>- GOMS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Essential Use Cases</td>
<td>- Wizard of Oz Prototypes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Paper and Chauffeured Prototypes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Wizard of Oz Prototypes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Structured User Role Model</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Operational Modeling</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- JEM (Joint Essential Modeling)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Cognitive Walkthrough</td>
<td>- Usability Specifications</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Pluralistic Walkthrough</td>
<td>- Usability Specifications</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Pluralistic Walkthrough</td>
</tr>
<tr>
<td>Requirement Specification</td>
<td>- Usability Specifications</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements Validation</td>
<td>- Cognitive Walkthrough</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Pluralistic Walkthrough</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Activities</td>
<td>Stages</td>
<td>Elaboration Stage (cycles 1 to i)</td>
<td>Iterative Cycles (cycles 1 to j)</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>---------------------------------</td>
<td>--------------------------------</td>
</tr>
</tbody>
</table>

Figure 3 shows another way of looking at the relationship between cycles and activities. It is a distribution of work across the different activity types, related to the time in the development process when each effort is performed. Each horizontal line represents an activity type, and the height of
the red line indicates the amount of work of this kind to be done at that particular development stage. For example, requirements elicitation and analysis activities are mostly performed in Elaboration cycles (with more emphasis on the early stages), while some elicitation and analysis activities are performed at the beginning of the central moments within the Iterative Cycles, and a small amount of work may be done in Evolution cycles. The x-axis represents time. Therefore, slopes in different lines denote some precedence between the different activity types, like, for example, between the different Requirements activities within Iterative cycles: first, there is some elicitation and analysis, followed by specification and then validation. Note that the amount of work on each activity represented in Figure 3 is approximate, it should not be taken literally. It represents a specific software development process that we have taken to illustrate general issues regarding development time.

*Figure #3. Amount of Work on each Activity Type at the Different Development Stages*
6. OTHER ELEMENTS THAT AFFECT USABILITY INTEGRATION INTO THE SOFTWARE PROCESS

Knowing where to plug usability techniques and activities into the existing software development process is a necessary starting point, but it does not automatically make software engineers capable of applying them and adopting a user-centered focus in development. Technology transfer is not a simple matter, and it is not usually enough to give developers a document for them to change the way they do things. ‘Care about usability’ is a change to the philosophy and viewpoint to which developers are accustomed. The framework for usability integration presented in this chapter needs to be supplemented by a carefully planned strategy for usability integration into development practices. Training and the integration of usability experts in the development teams at an organizational level are the two main issues to be considered for the success of the integration endeavor.

The software developers who are to apply the usability framework presented in this chapter need to be trained. Training should include both an introduction to the user-centered philosophy in general and to the particular usability techniques selected to be applied by the development team. Ideally, all software developers should attend training courses with this content, but, if this is not possible, some developers may be trained to act as 'usability champions' in their respective teams, so they can liaise with the central usability department (if any).

How to integrate usability experts into development teams must be given careful consideration as well. The members of the usability team are sometimes regarded by SE teams as a sort of ‘usability police’ who have to give their approval to the work produced by the SE team. Such approaches do not usually work well, and there should be more convergence between the objectives of the usability team and the SE team. Project teams (and their managers) should be rewarded for the attainment of usability levels, and not just for complying with deadlines and for producing defect-free software products. Nevertheless, the issue of project team composition is out of the scope of this chapter.

Another issue to be tackled, which is related to the above, is the cost-justification of usability efforts to evaluate the need for them to be undertaken. A prerequisite of the proposal presented in this chapter is acceptance by senior management of the need for usability practices to be incorporated into the development of their software products. Nevertheless, cost-justifying issues may arise when deciding which usability techniques are to be employed in a specific project.
7. CONCLUSIONS

HCI and SE take different but complementary views of software development. Both have been applied separately in most projects to the date, but, recently, demand for the integration of both into a common software development process has grown. SE as a discipline is pervasive in software development organizations all over the world. Its concepts are the ones with which the majority of developers are familiar, and this is especially true of senior management at software development organizations. HCI, on the other hand, has been traditionally considered as a specialist field, and its view of development is not as present in software development organizations as the SE perspective. Thus, the approach taken in this chapter is to tackle the integration of usability aspects from a SE point of view. Usability activities and techniques from the HCI field have been located with regard to standard SE activities. Time constraints for the application of usability techniques and activities with respect to the stages in a generic iterative process have been detailed as well.

The usability framework presented in this chapter targets software development organizations that have already decided to incorporate usability activities and techniques into their current development practices. The only prerequisite for its application is that the software development process currently in place must be based on iterative development. This is necessary, because iterative development is one of the essential principles of the user-centered approach from the HCI field. This requirement is not especially restrictive from a SE point of view, because it is in line with the current trends in SE.

The proposal does not have to be adopted as an all-or-nothing issue. It aims to provide a framework that allows decisions to be made on the inclusion of particular usability techniques and activities in any iterative software development process. It responds to the demands of software practitioners who are asking for pragmatic approaches instead of theoretical constructs that remain on the shelves unused.

Feedback from software development organizations has contributed to refinement of the present proposal but, as changing as the software development practice is, it is open to further refinement and specification as software development evolves and, hopefully, incorporates more and more usability aspects.

REFERENCES


