10 WHICH, WHEN AND HOW USABILITY TECHNIQUES AND ACTIVITIES SHOULD BE INTEGRATED

Xavier Ferre, Natalia Juristo, Ana M. Moreno

Universidad Politecnica de Madrid, Spain

Abstract

Software development organizations are paying more and more attention to the usability of their software products, as increasing importance is attached to usability as a critical software quality attribute. The HCI (Human-Computer Interaction) field offers techniques aimed at producing a software product with a good usability level, but their use is often not integrated into SE (software engineering) development processes. The integration of usability techniques into SE practice is not an easy endeavor, since both fields speak different languages and deal with software development from different perspectives. This chapter presents a framework for the integration of usability techniques and activities. This framework characterizes selected usability 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 most to the usability of the final software product. Software developers may then manage usability activities and techniques, include them in their software process, and understand in which activities usability and SE techniques have to be merged to achieve concurrent objectives. The proposed framework is aimed at software development organizations with a defined iterative development process that are looking to enhance their process with usability aspects.

A. Seffah (eds.), Human-Centered Software Engineering – Integrating Usability in the Development Process, 173–200.

10.1 INTRODUCTION

This chapter reviews some usability and SE methods looking at how they propose to integrate usability into the overall software development process, and builds an integration framework for incorporating usability activities and techniques into a defined software development process. The importance of this framework lies in the fact that there is now little or no guidance on the integration issue from a SE perspective. Software development organizations interested in improving the usability of their software products are willing to add usability activities and techniques to improve their software process, but usability textbooks do not offer support for this concern. This is a key question bearing in mind that usability techniques and the HCI approach to development are still relatively unknown and not well integrated in SE teams (Seffah and Andreevskaia, 2003). The work presented in this chapter is aimed at software development organizations with a strong SE background that are considering incorporating usability aspects into their practices, and cannot shift to a strictly usability-led development approach. For these organizations, usability is an important concern, but not the main focus, and even if there are some usability experts on their teams, software developers are expected to apply or be acquainted with some usability techniques.

According to the ACM SIGCHI Curricula for Human Computer Interaction, HCI is "a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them" (ACM, 1992). It is an established field, and one of its main concerns is the usability of computer systems. Usability techniques are applied in a variety of software development projects, where attaining an acceptable usability level is a very important, if not the main, goal. These projects are mostly developed following methods peculiar to the HCI field. Where this is not the case, that is, when usability practices are applied along with SE practices, their integration is tackled on a case-by-case basis (as in Anderson et al., 2001; Radle and Young, 2001). The main obstacle to HCI-SE cooperation is that the two fields speak different languages and deal with software development from different perspectives, as detailed in chapter 5. 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, 1990). 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.

In particular, the special emphasis it places on making software development systematic and disciplined has led the SE community to pay special 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. Every organization may have a differ-

ent software process, but some activities are common to all software processes. The software process can be described at different levels, and a feature common to 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 and Wastell, 1996; Derniame et al., 1999; Fuggetta, 2000). 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. Typical reasons why a software development organization may consider a defined software process valuable include: facilitating human understanding and communication, supporting process improvement, supporting process management, providing automated process guidance and providing automated execution support (Abran et al., 2004).

It should be noted that agile methods have recently appeared in response to all the importance software development practices attach to the software process. Agile methods 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 et al., 2001) sets out the main ideas behind this approach to software development. The agile philosophy is receiving a significant amount of attention in the SE field, and it looks like a promising approach from a usability point of view, as highlighted in chapter 12. Nevertheless, it is not yet considered part of the core practices of SE as defined software processes are. The SWEBOK (Software Engineering Body of Knowledge), which is a recent effort to gather what is considered commonly accepted knowledge in the SE field, does not include any reference to the agile approach in its Trial Version 1.00 (Abran et al., 2004), while defined software processes have their own chapter. Most software engineers put the accent on defining the software process in the belief that having and improving a defined process, as other production organizations do, is an approach that produces better quality software.

As they are, usability methods are hard to apply in SE, because of the conceptual differences between HCI and SE, and because overlaps with SE have not been settled. In particular, activities related to requirements engineering (an SE subfield) are tackled by both usability and SE methods. These overlapping areas are not clearly formulated from a SE viewpoint, forcing software development organizations to undertake costly research in order to plan the introduction of usability techniques and activities into such practices. The importance of the integration effort is sometimes mistakenly minimized, as a result of a perception of usability common in the SE field: it is considered to be related to just the UI (user interface). To average developers, the UI is the actual visual elements with which the user interacts and their response behavior (in visual terms), and it is therefore regarded as a graphic designer affair. Such misconceptions simplify the problem of integration in the software process modeler's mind: being a graphic designer issue, it only requires the addition of a usability activity in the process in which these issues are taken care of. In this case, there would be no or only slight

overlaps. Bearing in mind usability in the software development process, however, implies including usability activities throughout the entire process, with the challenge of integrating different development cultures into the same kind of activity.

For the present decade, Dumas & Redish (Dumas and Redish, 1999) predict continued growth of interest in usability from users to CEOs. Usability is becoming an important asset for a lot of software development organizations, and they demand guidelines for integrating usability activities and techniques into their software process. This trend 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. This change was introduced in the first amendment to ISO/IEC Standard 12207:1995, released in 2002 (ISO/IEC, 2002). The fact that an international SE standard stipulates that usability activities should be part of the software development process is an indication that usability is definitely on the SE agenda with respect to software process definition.

The goal of the work presented in this chapter is to offer a framework for introducing usability activities and techniques into any iterative software development process an organization may have in place. This framework does not define a particular software process, but sets out integration information in a way that it can be applied to a wide range of processes. The framework we propose details which kind of activities in a SE process are affected by usability techniques and when in development time each technique yields results that are most useful for the aim of raising the usability level of the final software product.

The first obstacle to the introduction of usability techniques 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 the two software development approaches. We need to identify the motivations behind each activity to find their interrelationships.

Apart from the terminology gap, HCI does not share the SE view of the software process. Some HCI authors, like Shneiderman, 1998, or Nielsen, 1993, do not structure usability efforts as activities (in the SE sense), so some usability techniques are not clearly assigned to activities in the HCI literature. The basic user-centered process (the HCI term for its process approach) is outlined in ISO Standard 13407 (ISO/IEC, 1999), but each author in the field has a particular vision of how this maps to specific activities. For effective HCI-SE cooperation, usability techniques need to be mapped to the most common activities present in user-centered processes.

The research presented in this work has been carried out as part of the STATUS project, financed by the European Commission (IST-2001-32298). The project goals include outputting methodological guidelines for integrating usability techniques into the software process, which we are presenting here. The two industrial partners in the project consortium have helped us to establish the framework's underlying premises. Pragmatically speaking, the industrial partners asked for a roadmap that could tell them which usability techniques and activities they should incorporate, and when in development time. They prefer this open solution that fits a wider range of processes to establishing the "perfect" software process integrating usability and SE practices.

The application of such a perfect process from a usability point of view would mean abandoning their current software process, which they do not wish to do.

Not all processes can be converted into proper user-centered processes by making just a few modifications. The transformation required for a process or an organizational culture based on a waterfall lifecycle approach to become user-centered would be far too drastic. This approach implies that detailed specifications are produced before any design and implementation is performed. The complexity of the human side in human-computer interaction makes it almost impossible to create a correct design at the first go. Cognitive, sociological, educational, physical and emotional issues may play an important role in any user-system interaction, and they cannot be completely predicted in advance. Therefore, the candidate for usability integration needs to be an iterative process. Of the characteristics of a user-centered process, iterativeness is the only one that is intrinsically inherent to the software process, as stated below in Section 10.7. Our framework then can help any organization with an iterative process to enhance this process with usability activities and techniques. This approach increases the practical applicability of the framework, since it does not require any specific original process as long as it is iterative. The appeal for a software development organization lies in the fact that it does not have to abandon the in-house process to adopt improvements, as it only has to modify the existing process.

The following three sections analyze how existing proposals deal with the integration problem: Section 10.2 details how integration is considered in usability methods, Section 10.3 details proposals for integrating usability techniques and activities into widely known SE methods, and Section 10.4 discusses the limitations and advantages of the proposals in the previous two sections. Section 10.5 details the mapping between usability and SE activities. Section 10.6 presents the assignment of the selected usability techniques to activities. Section 10.7 deals with the considerations on when to apply usability techniques and activities in an iterative development. The basic premises and context for the solution proposed are discussed in Section 10.8 and, finally, Section 10.9 presents the conclusions.

10.2 USABILITY METHODS APPROACH TO INTEGRATION

An organization wanting to include usability techniques and activities in the process may resort to HCI literature for help on the issue. In this section, we consider how the usability methods described in the HCI literature deal with the integration issue. We will consider just textbooks and international standards, since they are the sources more readily available to average developers with a SE background (who do not usually use conference proceedings and research journals as a source of information).

The **Star Life Cycle** by Hix & Hartson (Hix and Hartson, 1993) is a user-centered process that sets out the main usability activities. It does not prescribe a particular order for activities, but it does allocate a prominent role to usability evaluation, which is placed in the center of the star that represents the activities in the life cycle. 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 and Hartson, 1993).

ISO Standard 13407 (ISO/IEC, 1999) provides guidance on human-centered design activities throughout the life cycle of computer-based interactive systems. It is neither a method nor a software process, but it characterizes user-centered processes. Note that the standard authors use the term human-centered as equivalent to user-centered. We prefer the latter term, since it is more 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 on the integration issue. This requirement calls for the development of usability roadmaps that are useful for fitting usability techniques into the overall software development process.

Constantine & Lockwood (Constantine and Lockwood, 1999) propose the **Usage-Centered Design** method as a collection of coordinated activities that contribute to usability. Some HCI practitioners would not completely agree in considering usage-centered design an HCI method, but we have classed this method as such, since it is focused on the development of interactive systems for human use (and therefore fits the definition of HCI given in Section 10.1). The 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 usability 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, 2002, and Cockburn, 2001), that acknowledge Constantine and Lockwood's work on essential use cases.

Constantine and Lockwood offer some advice on integrating usability and UI design into the product development cycle, acknowledging that there is no one single way of approaching this question. 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 tai-

loring usage-centered design to the organization and its practices" (Constantine and Lockwood, 1999). 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.

Mayhew (Mayhew, 1999) 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 usability methods from the SE point of view. 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, 1992) and with rapid prototyping methods are identified, but Mayhew 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, 1999). Even though this is now the main trend in SE, it is not a valid definition of the discipline, since there are other development approaches that are valid from a SE viewpoint. A software engineer, taking up this work in search of help with the issue of usability integration into the software process may be put off by such misconceptions.

10.3 INTEGRATION PROPOSALS BASED ON SE METHODS

As we are trying to offer a solution for organizations that already have a process in place, this section will review integration proposals that are based on widely known SE methods. 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).

Costabile (Costabile, 2001) 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, 2002). Glass acknowledges that "requirements frequently changed as product development goes under way [...]. The experts knew that waterfall was an unachievable ideal" (Glass, 2003). SE literature has gradually come to accept that an iterative as opposed to a waterfall life cycle approach is the best for medium to high complexity problems when the development team does not have in-depth domain knowledge. Larman identifies the following problems with the waterfall life cycle: delayed risk mitigation, speculation and inflexibility of requirements and design, high complexity and low adaptability (Larman, 2002). Iterative development tackles most of these problems. Nevertheless, a waterfall mindset is still deeply rooted in day-to-day practice among software developers, mainly because it gives the complex activity of developing software systems an illusion of order and simplicity.

MUSE (Method for USability Engineering) (Lim and Long, 1994) is a method for designing the UI, and was one of the most well structured usability methods at the time of its publication (1994). It is divided into three phases: Information Elicitation and Analysis Phase, Design Synthesis Phase and Design Specification Phase. The method aims to ease integration with SE 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 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 current SE practices, in particular to object-oriented development.

The BIUSEM project (BIUSEM, 1995) 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, 1995).

The **Unified Process** (Jacobson et al., 1999; Kroll and Kruchten, 2003) 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, the Unified Process is the most widely used. The RUP© (Rational Unified Process) is a refinement of the Unified Process sold by IBM (previously by Rational Software Co.). The approach to usability integration presented in this section is not comparable to the above proposals in scope. It has been included, however, because of the current relevance of the Unified Process and RUP in SE.

The RUP 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 as 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, 2002) 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, 2003), 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, 2003). Despite this promising definition, Conallen's work focuses on modeling, and he describes the artefacts 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.

With regard to usability integration into object-oriented development, the **WISDOM method** (Nunes, 2001) deserves a mention, even if it is not a commonly used

method. It includes an extension to UML (Unified Modeling Language) to allow user-centered models to be employed in conjunction with object-oriented models and, therefore, to facilitate usability integration in modeling efforts throughout development. The WISDOM method offers a comprehensive process for any organization interested in adopting a new process already integrating usability. The organization is then forced to adopt the process as a whole, including the underlying assumptions present in the SE part of the method. For example, the WISDOM method differentiates between an Analysis and a Design workflow, while this distinction (inherited from object-oriented methods prior to the Unified Process) is not retained in recent interpretations of the Unified Process: Kroll & Kruchten consider a single "analysis and design" discipline or workflow (Kroll and Kruchten, 2003), while Larman considers no analysis discipline and also states that the analysis model in the Unified Process is not necessary and seldom used (Larman, 2002). Nevertheless, the WISDOM method is still very interesting for software engineers, since it offers models for dealing with usability issues. Additionally, the way it deals with some process issues in a user-centered view could be mapped to processes other than the specific WISDOM method process.

10.4 SUMMARY OF INTEGRATION PROPOSALS

As presented in Section 10.2, the descriptions of the usability methods considered as to how they integrate with the overall software development process are not highly detailed. Actually, the textbooks describing these methods do not intend to detail this issue, as their main objective is to present the actual usability method. Consequently, a software engineer looking for an answer to the integration problem may find the information in these sources defined at a different level of detail than is usual in a defined SE software process. Some methods just present high-level activities, like the Star Lifecycle, the ISO Standard 13407, or the Usage-centered Design method. On the other hand, the Usability Engineering Lifecycle is more detailed, describing fine-grained activities and techniques that may be applied for each activity. But, as presented in (Mayhew, 1999), this method encompasses a not so iterative approach to software development. On top of these difficulties, the four HCI approaches considered use a terminology that is peculiar to the HCI field. Specifically, the Star and the Usability Engineering lifecycles are presented as methods for the design of highly usable UIs. As stated in Section 10.1, SE refers by UI design to just the design of the actual visual elements that form the UI and the UI response behavior in visual terms. It does not include any activity related to requirements engineering in the SE perspective. Nevertheless, both methods include standard requirements activities like task or user analysis. The terminology gap makes the task of integrating the usability methods into the overall software process especially difficult.

As for the integration proposals based on SE methods presented in Section 10.3, we have also identified some of the limitations observed in usability methods, like them not being truly iterative (in the case of Costabile's proposal and MUSE) or just addressing the design of the UI (like MUSE), and therefore confusing software engineers with regard to integration. On the other hand, the UX plug-in for RUP actually integrates some usability practices into a comprehensive process, but its main goal is

to incorporate some techniques that are often used in the web development field instead of integrating usability into the process. Therefore, the UX plug-in is limited to a few usability techniques and does not cover the whole range of activities in which usability techniques may apply.

Given that the information from the HCI literature is not detailed enough for the purpose of integration and is not formulated in SE concepts and terminology, and because SE proposals do not have a proper iterative focus or cover all activities, we propose a framework that addresses these concerns in the following sections. The framework is formulated according to a truly iterative approach (expressing time constraints in the form of iterative development stages), and it covers the whole range of SE activities for mapping usability activities and techniques to all relevant kinds of activities in a software process.

10.5 MAPPING BETWEEN USABILITY AND SE ACTIVITIES

To map usability terminology to SE terminology, the activities that form part of a user-centered process must be identified. 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, is a hurdle for this ambition. 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 (Ferre et al., 2002a) 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 10.1, grouped according to what type of development activity they belong to. Note that the activities are not listed in any time-related order.

There is a lot 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. It has been named following the ISO 13407 Standard terminology (ISO/IEC, 1999), and it is divided into User and Task Analysis because some authors make a distinction between the two activities (Mayhew, 1999; Hix and Hartson, 1993; Constantine and Lockwood, 1999). 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 well defined in the HCI literature that we consulted. The only activity cited by most authors is Prototyping. Prototypes are widely used in SE, particularly in iterative development. What HCI has to offer, however, is the particular usage of light prototyping to get more user involvement and for weighing up alternative designs. The most useful prototypes for this purpose are the less sophisticated ones, such as paper prototypes. Typical SE prototypes usually involve some degree of programming, while paper prototypes allow for faster iterations as they do not require any programming effort.

Develop the Product Concept is based on mental models (Norman, 1990; Preece et al., 1994): 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. The importance of helping the user to grow

productive mental models for the usability of the system is especially stressed. 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 from one author to another, but we have identified the definition of the interaction that will take place between the user and the system as a common aim in the design process. It includes designing the user-system dialogue, that is, the sequence of actions needed to operate the system, and the user-system information exchange, in detail. By interaction design we mean the design of the coordination of information exchange between the user and the system. 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 most profusely detailed in HCI literature. Usability is very difficult to strive for, due to the complex human nature. 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 et al., 1994). 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 10.1: 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 (Abran et al., 2004) has been used as a basis for defining the activities in a traditional software development process. HCI terminology has been used for other activities that are new to SE and do not fit any existing activity.

The mapping of usability activities to development activities considered in this chapter is shown in Figure 10.1. Each usability activity on the left-hand side of Figure 10.1 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, and the other activities in a software process are not included.

Regarding the analysis-related activities, 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, Analysis and Negotiation; 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, Analysis, and Negotiation 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 it is not addressed in the chapter of the SWEBOK devoted to software design (Abran et al., 2004). 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, Analysis and Negotiation in our framework.

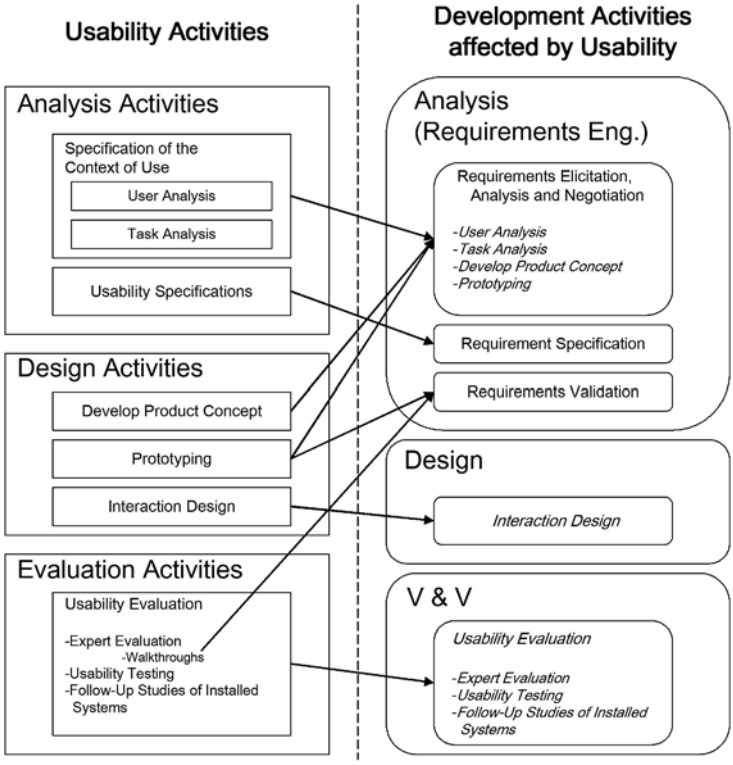

Figure 10.1 Mapping between usability and SE Activities

Usability-related design activities are quite separate 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" (IEEE, 1990). 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 unconnected with 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 10.1) 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.

10.6 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 HCI literature to their respective SE activities.

Note that the integration framework presented in this chapter is aimed at software development organizations that do not have a big usability department (if they have one at all!) and, therefore, need usability concerns to be shared with the developers throughout development. Nevertheless, for organizations where usability expertise is widely available, communication problems inside multidisciplinary teams are an important concern, and the proposed 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 usability, we have made a selection where there was more than one usability technique with the same objective available 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 (Ferre et al., 2002a), the resulting set of techniques has been reduced to just 36 techniques. They appear in the column furthest to the right in Table 10.1, which is explained in the next paragraph.

We have used the definition of each usability technique in the 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 10.1 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.

10.7 WHEN TO APPLY USABILITY ACTIVITIES AND TECHNIQUES

We examined the HCI literature to identify what characteristics a software development process should have for it to be considered user-centered. Shneiderman, 1998, Nielsen, 1993, ISO/IEC, 1999, Hix and Hartson, 1993, Constantine and Lockwood, 1999, Preece et al., 1994, all 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 usability 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 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.

Table 10.1: Allocation of usability techniques to activities

HCI Activities	Activities in Software Process		Usability Techniques
Analysis - Specification of the Context of Use - User Analysis	Analysis	Requirements Elicitation, Analysis and Negotiation	Ethnographic Observation (Preece et al., 1994)
			Contextual Inquiry (Beyer and Holtzblatt, 1998)
			Structured User Role Model (Constantine and Lockwood, 1999)
			Operational Modeling (Constantine and Lockwood, 1999)
			JEM (Joint Essential Modeling) (Constantine and Lockwood, 1999)
Analysis - Spec. Context of Use - Task Analysis			Essential Use Cases (Constantine and Lockwood, 1999)
Design - Develop Product Concept			Affinity Diagrams (Beyer and Holtzblatt, 1998)
			Visual Brainstorming (Preece et al., 1994)
			Competitive Analysis (Nielsen, 1993)
			Scenarios (Carroll, 1997)
Design - Prototyping			Prototypes (paper and chauffeured (Constantine and Lockwood, 1999); and wizard of Oz (Preece et al., 1994)
Analysis - Usability Specifications		Requirement Specification	Usability Specifications (Hix and Hartson, 1993)
Usability Evaluation - Expert Evaluation		Requirements Validation	Cognitive Walkthrough (Lewis and Wharton, 1997)
			Pluralistic Walkthrough (Bias, 1994)

Continued on next page

Table 10.1: Allocation of usability techniques to activities

HCI Activities	Activities in Software Process		Usability Techniques
Analysis - Spec. Context of Use - Task Analysis	Design	Interaction Design	Detailed Use Cases (Constantine and Lockwood, 1999)
Design - Interaction Design			Screen Pictures (Hix and Hartson, 1993)
			Card Sorting (Robertson, 2001)
			Menu-selection Trees (Shneiderman, 1998)
			Navigational Paths (Conallen, 2003)
			Product Style Guide (Mayhew, 1999)
			Impact Analysis (Hix and Hartson, 1993)
			Help Design by Use Cases (Constantine and Lockwood, 1999)
Evaluation - Expert Evaluation	V & V	Usability Evaluation	Heuristic Evaluation (Nielsen, 1993)
			Usability Inspections (Nielsen and Mack, 1994)
			Cognitive Walkthrough (Lewis and Wharton, 1997)
			Pluralistic Walkthrough (Bias, 1994)
Evaluation - Usability Testing			Thinking aloud (Nielsen, 1993)
			Performance Measurement (Dumas and Redish, 1999)
			Laboratory Usability Testing (Dumas and Redish, 1999)
			Post-Test Feedback / User Questionnaires (Mayhew, 1999)
Evaluation - Follow-up Studies of Installed Systems			Questionnaires / Surveys (Mayhew, 1999)

Continued on next page

Table 10.1: Allocation of usability techniques to activities

HCI Activities	Activities in Software Process		Usability Techniques
			Structured and Flexible Interviews (Preece et al., 1994)
			Direct Observation (Hix and Hartson, 1993)
			Video / Audio recording (Hix and Hartson, 1993)
			Focus Groups (Mayhew, 1999)
			Logging Actual Use (Shneiderman, 1998)
			Online User Feedback Facilities (Shneiderman, 1998)

10.7.1 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, most requirements discovery and refinement is usually undertaken during the early iterations (Larman, 2004). These early iterations are packed in a stage: Elaboration. A stage comprises a sequence of iterations in development with similar basic objectives.

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 10.2. 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.

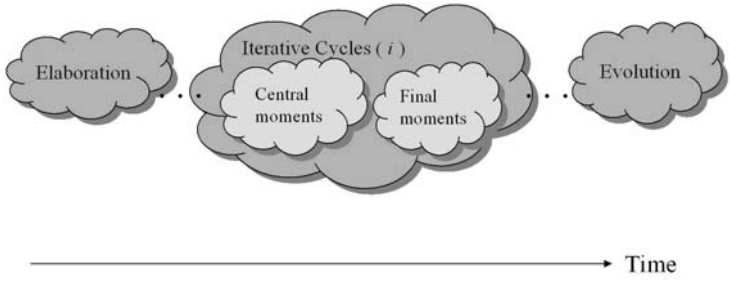

Figure 10.2 Stages in an iterative software development process

An explanation of the stages in an iterative development 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.

Any organization wanting to apply the proposed framework will have to translate these generic stages to the ones they have in place, using their specific terminology. Even if our representation of development stages is a common one in iterative processes, not all iterative development approaches will necessarily match our stage representation. For example, some projects may not have an elaboration stage, in which case the techniques to be applied in this stage would be applied in the iterative cycles instead, since this is the next stage in our representation.

10.7.2 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 they are to be applied. This section details this information, organized according to the stages in a generic iterative process presented in the previous section. For a comprehensive study of the time constraints for usability technique application, (see Ferre et al., 2002b).

The **Elaboration** stage corresponds to the initial cycles where the needs are identified and the general system outline is established. A general aim is for the products of this stage to be 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 the first examination of the problem for handling an ill-defined solution: 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.

Analyzing the user and his or her environment, and the basic dialogue between the system and the user is a prerequisite for any development that intends to cater for the user and 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 completing the models produced:

- Essential Use Cases
- 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 development in order to get feedback from the user. Thus, even though these techniques will carry more weight in the iterative cycles, they are also present at the Elaboration stage. These techniques are Detailed Use Cases, Screen Pictures, Card Sorting, and 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.

Iterative cycles may include the application of techniques which require a greater effort than the techniques detailed above, like Product Style Guide or certain prototyping techniques that demand some implementation, such as Wizard of Oz Prototypes. Both techniques might fit in well in Elaboration cycles, but they are predominantly to be applied in iterative cycles in order to avoid bulky elaboration cycles.

Some already mentioned UI design techniques carry 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

The **Evolution** stage 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 (they 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.

When the development project involves replacing a system that is already in operation, all of these techniques can also be used as data gathering techniques in the Elaboration stage of the project.

10.7.3 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 10.2. Techniques highlighted in bold face within a stage carry more weight in this stage, that is, this is the stage in which they are best suited, even though they can be applied at other stages.

Figure 10.3 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, analysis and negotiation 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. 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, analysis and negotiation followed by specification and then validation. Note that the amount of work on each activity represented in Figure 10.3 is approximate, it should not be taken literally.

Table 10.2: Usability techniques to be applied at each stage and their significance

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

Continued on next page

Table 10.2: Usability techniques to be applied at each stage and their significance

Activities		Stages			
		Elaboration Stage (cycles 1 to i)	Iterative Cycles (cycles 1 to j)		Evolution Stage (cycles 1 to k)
			central moments	final moments	
Design	Interaction Design	- Detailed Use Cases - Screen Pictures - Card Sorting - Menu-selection Trees - **Navigational Paths**	- Product Style Guide - Help Design by Use Cases - Impact Analysis - Detailed Use Cases - Screen Pictures - Card Sorting - Menu-selection Trees - Navigational Paths		- Impact Analysis

Continued on next page

Table 10.2: Usability techniques to be applied at each stage and their significance

Activities		Stages			
		Elaboration Stage (cycles 1 to i)	Iterative Cycles (cycles 1 to j)		Evolution Stage (cycles 1 to k)
			central moments	final moments	
V & V	Usability Evaluation	- Cognitive Walkthrough - Pluralistic Walkthrough - Heuristic Evaluation - Usability Inspections		- Cognitive Walkthrough - Pluralistic Walkthrough - Heuristic Evaluation - **Usability Inspections** - **Thinking aloud** - **Performance Measurement** - **Laboratory Usability Testing** - **Post-Test Feedback / User Questionnaires** - Video/audio recording	- Pluralistic Walkthrough - Thinking aloud - Performance Measurement - Laboratory Usability Testing - Post-Test Feedback / User Questionnaires - Questionnaires / Surveys - **Structured and Flexible Interviews** - **Direct Observation** - **Video/audio recording** - **Focus Groups** - **Logging Actual Use** - **Online User Feedback Facilities**

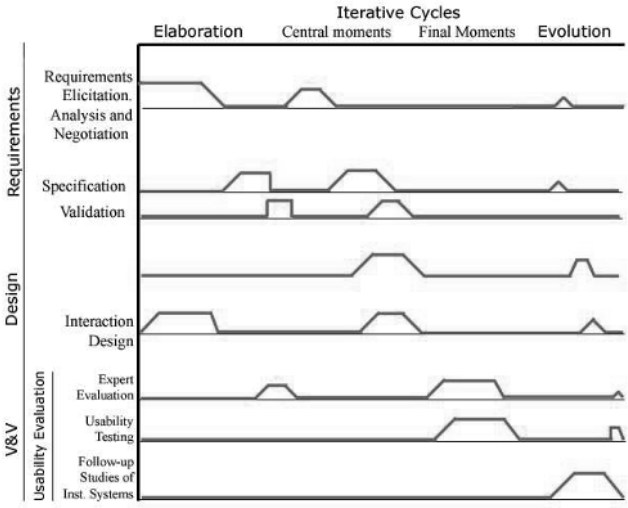

Figure 10.3 Amount of work on each activity type at the different development stages

10.8 DISCUSSION

HCI and SE take different but complementary views of software development. 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. The approach taken in this chapter for usability integration into the software process tackles the integration issue from a SE point of view.

Our framework targets organizations that are considering introducing usability into their practices, but not at any cost. They want to keep the software process they have in place, because it is a valuable asset, although they aim to continuously improve this process (by adding usability activities and techniques, for example). For organizations looking for a more radical shift towards a user-centered approach with an even higher degree of user participation in design efforts, the principles enumerated in chapter 2 may apply. The proposed framework can be valuable to organizations where such principles are in conflict with organizational objectives and concerns, like limited availability of representative users or geographically distributed teams. Another alternative approach to be considered is presented in chapter 13 for organizations wanting to keep two separate processes (the usability process and the SE process), so that software development needs do not take over usability concerns. Our framework may be

valuable to organizations that have not identified the need to place usability in such a prominent place in development, that is, organizations that consider usability an important quality attribute but not to the extent of considering it as valuable as all the other quality attributes together.

For the purpose of integrating usability into the process, the user needs to be placed at the center of the whole development effort. The framework includes several techniques (mainly from the Usage-Centered Design method—Constantine and Lockwood, 1999) for modeling the user and his/her tasks and the interaction between users and the software system. If the development team considers the user as the final measure of software development project success, it has already taken a big step towards the adoption of a user-centered perspective in development, and models may support this objective. Other techniques in the framework favor a higher degree of user participation, facilitating communication in multidisciplinary teams. The application of some techniques calls for a reformulation of activities that were already in place before the usability integration, but the biggest part of the development process does not usually need to be profoundly altered.

The proposed framework serves the purpose of identifying which usability activities and techniques may be useful for an organization to enrich its software process, and where they have to be incorporated in the process. But some additional issues, like how to modify existing practices in order to incorporate the new ones, must be resolved for an effective integration. The work by Gulliksen and Göransson, 2001, complements the information expressed in the framework by providing a recipe for action for evaluating a process for its user-centeredness and modifying it where necessary.

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 these techniques and activities and adopting a user-centered focus in development. 'Caring about usability' is a change to the philosophy and viewpoint with which developers are accustomed. The framework for usability integration presented in this chapter needs to be supplemented by good training for developers. For the industrial partners of the STATUS project to apply the framework in practice, their developers needed to take a 24-hour course on usability principles and techniques. The course was designed to raise their usability awareness, clearing up common misconceptions about the issue. Chapter 8 presents some practices that may be helpful for educational purposes and to get buy-in between developers for the user-centered approach.

10.9 CONCLUSIONS

In this chapter we presented a framework that may allow a more successful introduction of usability techniques and activities into the software process. Usability activities and techniques from the HCI field have been positioned in the framework 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 resulting framework 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. 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 the industrial partners of the STATUS project has contributed to refinement of the present proposal, but, as changing as software development practice is, it is open to further refinement and specification as software development evolves and, hopefully, incorporates more and more usability aspects. In particular, information may be added to the framework on the products of each usability technique and their possible integration with SE models and documentation.

Acknowledgements

We would like to thank the partners in the STATUS project for their input and their cooperation, and we would like to acknowledge the support of the European Union under grant STATUS (IST-2001-32298).

We would also like to thank the anonymous reviewers of the chapter and the editors of the book for the thorough job they have done, and for the valuable insights their comments have provided.