INFORMATION SOCIETIES TECHNOLOGY (IST) PROGRAMME

STATUS

"Software Architecture for Usability"

WORKPACKAGE 6:

DELIVERABLE D.6.6. Final results on the Integrated Software Process

Version: 1.0

Submission Date: November 29, 2004

Authors: Xavier Ferré, Natalia Juristo, Ana M. Moreno,

Partners: UPM

Stage:	**Confidentiality:**
[ ] Draft	**[X] Public - for public use**
[ ] To be reviewed by WP participants	**[ ] IST – for IST programme participants**
[X] Pending of approval by next consortium meeting	**[ ] Restricted – for STATUS consortium and PO**
[ ] Final / Released to CEC	

DOCUMENT CONTROL

Registration of Changes

Date	Version	Author of Changes	Comments
November 26, 2004	0.1	Ana M. Moreno, Natalia Juristo, Xavier Ferré	Initial draft
November 29, 2004	1.0	Ana M. Moreno, Natalia Juristo, Xavier Ferré	Final version
December 3, 2004	1.1	Ana M. Moreno, Natalia Juristo, Xavier Ferré	English remarks

List of Related STATUS Documents

Document Name	Version
Technical Annex	1.0
D.5.1.	1.0
D.5.2.	1.0

ACRONYMS AND ABBREVIATIONS

Acronyms and abbreviations	Meaning
WP	Workpackage
STATUS	Software Architecture for Usability
WL	Workpackages Leader
TL	Task Leader
IHG	Information Highway Group
UPM	Universidad Politécnica de Madrid
RuG	University of Groningen
ICTSM	Imperial College

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

The purpose of this deliverable is to update the results of WP 5 examining the relationship between usability techniques and the software process. These improvements have been derived from the technology transfer of the results of WP5 to the industrial partners and their application to real projects.

1.2 Document Structure

The document has been organised to present the different issues that have been updated from WP 5. In particular, section 2 gives a short overview of the work done in WP 5, as well as the motivation for the main improvements presented in this deliverable. Section 3 deals with the usability techniques studied and section 4 with the allocation of usability techniques to SE activities. Finally, section 5 focuses on the times at which the usability techniques can be applied during the software process.

2. WORK DONE IN WP 5

In WP 5 we worked on identifying what usability techniques were more relevant from a SE perspective. The candidate techniques were presented in D.5.1. We also packaged these techniques in what we called “deltas for usability” that were supposed to be applied during the different activities of the software process to improve the usability of the software products developed. The resulting deltas were presented in D.5.2.

These results were then formally presented to the industrial partners at special seminars and they applied the results in their applications. From this practical experience, we found that:

- The criteria for selecting the usability techniques that were used deliverable D.5.1. needed to be refined. These criteria defined which techniques better fitted the purpose of easing the integration of usability into the software development process from a software engineering viewpoint. The new set of criteria provides the reasoning behind the selection of the techniques, which can be transmitted to software developers. In this way, developers can decide if they are applicable to their particular case.
- We also found that it would be useful to label each selected technique as either "very useful" or just "useful" so that developers had two levels of useful techniques for integration into the process. This was designed to ease the process of deciding which techniques to include in the process and the process of adapting our selection to their particular conditions by making the reasoning explicit.
- Finally, after receiving some feedback from the industrial partners, we realised that packaging usability techniques as deltas may be counterproductive, since it could convey the wrong idea of a usability field with less variety than there actually is. Therefore, we have decided to not group usability techniques into deltas, but to just provide information about when best to apply the techniques in an iterative development, as this information adds to the to the above-mentioned criteria.

The resulting research results related to the integration of usability techniques into the software development process are discussed in the following.

3. ACTIVITIES AND TECHNIQUES IN A USER-CENTRED PROCESS

In this section we are going to detail a survey that we conducted of the HCI activities present in a user-centred process, and the techniques that can be applied in each activity. The purpose of this study is to establish the activities and techniques proper to the HCI field with a view to their possible integration into any sort of development process.

Whereas one of the goals of SE has been to formalise the process to make software development systematic and disciplined, HCI does not take an equally formalised view of the process. Each author in the HCI field has a particular view of the process in terms of what activities are part of a development project aiming to achieve a given usability level for the software product. The survey detailed in this chapter was conducted to get a set of standard activities from the activities proposed by several authors. These activities will be related in later sections to the standard SE development process activities. So, the usability activities survey is valuable for mapping established concepts in HCI around the development process, which can serve as a basis for communication between SE-trained developers and the usability team.

The usability literature usually allocates usability techniques to a (usability) activity. Therefore, the activity survey is a prior step to integrating techniques into the development process. HCI literature details a lot of usability techniques, as each author refers to the techniques in a particular way. To be able to work with a reasonable number of techniques, we have drawn up a taxonomy of techniques, for which purpose we have also studied and characterised the HCI literature to get a criterion for selecting the ones that we will propose for possible inclusion in the software development process.

First, section 2 details the survey of activities typical of a user-centred process. Then, section 3 reflects the taxonomy of techniques built. Section 4 characterises each technique according to the selected parameters and section 5 selects the most useful techniques for inclusion in the software development process. Finally, section 6 will detail how the basic references describing each technique were selected.

3.1 Usability Activities in the HCI Literature

To be able to present a representative schema of the development processes proposed by HCI, we have examined the development activities stated in the HCI literature. First, we will list the sources considered for this survey and then we will detail the results of the survey, including a tabulated overview and a description of each identified activity.

3.1.1 Sources

For this survey of usability activities in the HCI literature, we have considered only books and standards. Our goal is to examine what is considered to be commonly accepted knowledge in the HCI field and, therefore, books and standards are better suited for this purpose than research published in journals and at congresses.

From the vast array of books and standards published on HCI, we have chosen the most significant and most often cited volumes. A brief description of each selected work is given below:

- [Nielsen, 93] – "Usability Engineering". Jakob Nielsen.
  Usability engineering is a subdiscipline of HCI that focuses on the establishment of measurable usability goals and their evaluation through usability testing in each development cycle. This book has been the chief reference on this subject for a very long time. Nielsen

offers an engineering-like approach to constructing usable software, thereby bringing usability closer to SE.

- [ISO9241, 98] – "ISO 9241-11 Ergonomic Requirements for Office Work with Visual Display Terminals – Part 11: Guidance on Usability". International Organisation for Standardisation.
  This ISO standard gives the definition of usability that is used in the ergonomics standards. The ISO standards referring to ergonomic requirements have been widely adopted by industry.

- [ISO13407, 99] – "ISO 13407. Human-Centred Design Processes for Interactive Systems". International Organisation for Standardisation.
  This standard attempts to help hardware and software design process managers to effectively identify and plan the human-centred design activities at the right time. Like the above standard, being an ISO standard, it also is well accepted in industry

- [Shneiderman, 98] – "Designing the User Interface". Ben Shneiderman.
  Ben Shneiderman is one of the best respected authors in the field of HCI. He received the ACM-SIGCHI association’s "*CHI Lifetime Achievement Award*" in 2001. The three editions of this book have amounted to fundamental references in the field of UI design, because of the balance they strike between theoretical interaction issues and development-centred aspects.

- [Hix, 93] – "Developing User Interfaces: Ensuring Usability Through Product and Process" D. Hix and H. Hartson.
  This work presents a very practical approach applied to the subject of UI design. One of its goals is to serve as a textbook for UI development courses with a strong usability component.

- [Preece, 94] – "Human-Computer Interaction". J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, T. Carey.
  This work deals with a whole range of topics related to the HCI field. It is an encyclopaedic endeavour, and theoretical components carry quite a lot of weight. It was one of the main textbooks for general-purpose HCI courses at the time of writing.

- [Wixon, 97] – "The Usability Engineering Framework for Product Design and Evaluation". D. Wixon and C. Wilson. In *Handbook of Human-Computer Interaction, 2nd edition*.
  The work in which this article appears describes the diversity of the HCI field, in terms of both research and practice. Wixon and Wilson’s article offers a good overall description of the subdiscipline of usability engineering. The authors belong to the usability group at DEC who created this approach (as certified in [Gould, 88]).

- [Constantine, 99] – "Software for Use". Larry L. Constantine, Lucy A.D. Lockwood.
  Larry Constantine is a familiar name in SE. He has switched his interest from structured design in the 80s to usable software development in the last ten years. This work describes Constantine and Lockwood’s experience as usability consultants, distilled into an eminently practical method.

- [Mayhew, 99] - "The Usability Engineering Lifecycle: A Practitioners Handbook for User Interface Design".
  This text describes what is, viewed from SE, one the most comprehensive HCI processes, since it defines, for each activity, its precedence over other activities, roles that participate, techniques to be applied and products that are generated. It is the most recent reference focusing on usability engineering.

3.1.2 Results of the Survey

We have analysed these proposals to extract common activities or activities that are at the same level of abstraction and are common to several sources. Table 1 shows the result of this process. The goal is to be able to compare the different proposals. Therefore, we have grouped the activities that refer to the same concept in one row. Each role is labelled (in column 1 of the table) using either the most generic term or the term most commonly used by the surveyed authors. Table 1 includes a column for each source considered, containing the activities proposed by each source.

When an author lists several tasks within the same activity, the table includes the full name for the activity. For example, [Hix, 93] proposes Systems / Tasks / Functions / Users Analysis, which has been classified under the Context of Use Specification activity. When an author describes a generic activity that includes the activity that we are considering as a subtask, then the specific subtask is specified at the bottom of the table with the respective number between brackets. When an author proposes several activities that match the described activity, they are included preceded by an (*). Activities not mentioned by any author are entered in the table using a hyphen (-).

From the table, we find that the consulted sources tend to agree on the following activities: Usability Specifications, Prototyping and Usability Evaluation. Context of Use Specification, either as a single activity or as a combination of the Task and User Analysis activities, is also common to several sources.

Most discrepancy was found regarding the design activities (apart from Prototyping), such as Develop Product Concept and Interaction Design. Whereas some authors do not mention how to carry out the design apart from labelling it as user centred or advocating iterative design, other authors, like Constantine and Lockwood [Constantine, 99], do go further into design as an activity. Indeed, Constantine and Lockwood criticise the dominant trend in the field of usability engineering, which focuses almost exclusively on the evaluation of usability, disregarding the possibility of trying to do a good design from the start. This may be the reason why the other surveyed authors pay less attention to the subject of interaction design. A similar thing applies to the product concept, as only two sources ([Shneiderman, 98] and [Preece, 94]) clearly consider it to be a development activity and another two ([Hix, 93] and [Mayhew, 99]) deal with it tangentially when describing conceptual design preceding detailed design. However, both the creation of a good product concept and interaction design play an important role in the principal usability texts consulted. Therefore, failure to include them as an activity in the development process may be due to the fact that they are the result of more recent development work in the field of HCI, compared with, for example, usability evaluation, which has been more widely explored. The low HCI process formality leads to activities related to product concept creation and interaction design having been ignored or not taken seriously by many authors.

Table 1 – Usability Activities by Source

Activity	Nielsen93	ISO9241_98	ISO13407_99	Shneiderman98	Hix93	Preece94	Wixon97	Constantine99	Mayhew99
CONTEXT OF USE SPECIFICATION	Know the user	Specification of the planned context of use for a product	Understand and specify the context of use	Investigate and analyse needs	System / Task / Function / User Analysis	Functional / Task Analysis	* Specify and categorise users * Analyse tasks	Task modelling	* User profile * Contextual task analysis
USABILITY SPECIFICATIONS	Establish goals	Specify usability requirements for a product	Specify user and organisational requirements	Design concepts and a key screen prototype (1)	Requirements / Usability Specifications	Requirements Specification	* Define quantitative usability goals	-	Establishment of usability goals
DEVELOP PRODUCT CONCEPT	-	-	-	Develop the product concept	Conceptual Design	Conceptual Design / Formal Design	-	-	Conceptual model design
PROTOTYPING	Prototyping	-	Produce design solutions (2)	Design concepts and a key screens prototype	Rapid Prototyping	Prototyping	-	-	Prototyping of screen design standards
INTERACTION DESIGN	Iterative design	-	Produce design solutions	Iterative design and refinement	Design and design representation	Conceptual Design / Formal Design	-	Interface Content Modelling	Detailed user interface design
USABILITY EVALUATION	Interface evaluation	Usability measurement	Evaluate usability against requirements	Iterative design and refinement (3)	Usability Evaluation	Evaluation	Test the product against the usability goals	Usability Inspection	Iterative evaluation of user interface detailed design

Specific subtasks corresponding to the activity described within a broader activity:

(1): Create specific usability goals based on user needs

(2): Use simulations, models, mock-ups, etc., to make more definite design decisions

(3): Run a full-scale usability test

Therefore, we can confine ourselves to the Context of Use Specification, Usability Specifications, Prototyping and Usability Evaluation activities as being representative of a user-centred process, whereas we are going to accept the Develop Product Concept and Interaction Design activities with reservations more out of respect for them as important points mentioned in the usability literature than because of their inclusion as activities by most of the surveyed sources.

Figure 1 shows these usability activities, organised according to the traditional division of software development activities into analysis, design and evaluation activities. Note that the Context of Use Specification activity is decomposed into User Analysis and Task Analysis. Some authors ([Hix, 93], [Wixon, 97] and [Mayhew, 99]) make a distinction between these two activities, although they do admit that the two are closely related. However, we have decided to follow ISO 13407 terminology [ISO13407, 99] and list the Context of Use Specification, because this better reflects the close relationship there is between the two subactivities.

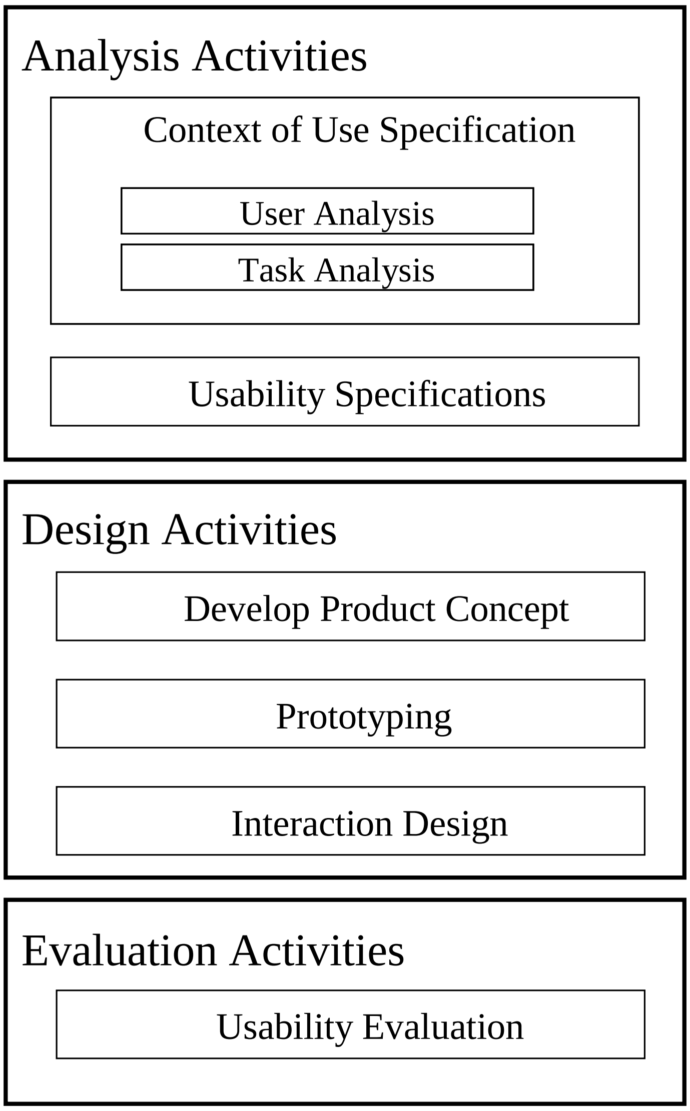

Figure 1 – Usability Activities Grouped by Activity Type

The following sections detail each of the specified activities.

3.1.3 Context of Use Specification

The goal of this activity is to understand and record the characteristics of the prospective context of use for consideration in design tasks, insofar as they might be relevant for the usability of the final software product.

The context of use is a very broad term composed of several interrelated aspects. As defined by [ISO13407, 99], it is made up of the following components:

- The characteristics of target users of the software. The identification of these characteristics is known as User Analysis.
- The tasks that the users are to perform. Task Analysis is concerned with this issue.
- The environment in which the users will use the system, including hardware, software and materials they are to use.

The first two subactivities, User Analysis and Task Analysis, are described in more detail below, whereas the third will be considered as part of the analyses carried out in User Analysis.

3.1.4 User Analysis

User Analysis considers and identifies the knowledge, needs and characteristics of the prospective users that are relevant for their interaction with the system. The characteristics to be identified include domain knowledge, skill, experience, training, physical traits, habits, preferences and aptitudes. For some system types, characteristics such as age, disabilities, colour blindness, etc., may also be relevant. All these characteristics are examined to be able to tailor the system under development to its future users.

Also significant with respect to user characteristics is what type of hardware and software they use. How experienced users are in using computer systems generally and especially systems that are similar to the one under development should be examined as part of the user analysis tasks. Indeed, even the type of system to which the user population is accustomed can be relevant for assuring that the system is designed according to what expectations users may have, which are usually based on their previous experiences with computer systems.

The physical environment is also important, although, strictly speaking, it is not a user characteristic. Factors like high noise levels, low luminosity, temperature and other similar characteristics regarding the user’s place of work should be taken into account. The software should, therefore, be designed to overcome as many limitations of the environment in which the system is to be used as possible, even if this would make the resulting system less efficient in other environments.

The social and/or industrial environment may also be a matter to be taken into account when organisational structure and work practices are relevant for the design to be undertaken. For example, it may be necessary to consider whether colleagues or customers will frequently interrupt the user. Another possibility to be taken into account in some projects is the fatigue factor, to which users may be exposed because of the special features of the type of work they do.

User Analysis may have to take into account different target user groups. The user population does not necessarily have to be a single and uniform group. In many projects, users need to be divided into a number of relevant groups. This complicates User Analysis somewhat, as a separate study will be needed to select what user groups to consider, and the User Analysis will then have to be run on each of the identified user groups.

3.1.4.1 Task Analysis

The purpose of Task Analysis is to “get descriptions of what people do, represent these descriptions, predict difficulties and evaluate systems against usability or functional requirements” [Preece, 94]. Briefly, it is concerned with what people do to take care of the matters for which they are responsible.

Task Analysis is closely related to requirements elicitation and specification, but its key characteristic is that it focuses this type of activities on the ultimate goals of system use by the user. The difference between a task and a function is that a task actually makes sense to the user. The user thinks the task needs to or should be done. Therefore, the term task implies an intention or purpose that the concept of function offered by a system does not necessarily signify. For example, the functionality for generating intermediate files (needed for the sound operation of the system as it is conceived) is not a user purpose, it is a system constraint.

The task description should include the role that the user plays in global task performance and not only in terms of the functions or functionalities provided by a product or system. Task Analysis can be considered as function oriented, but it adds to SE function-oriented methods by considering what the user intends to do by performing a task. The main difficulty faced by a software engineer who wants to carry out Task Analysis is that it is similar to functional decomposition, which can lead to the use of the standard functional decomposition criteria rather than criteria proper to Task Analysis. Task Analysis should always be focused on user goals.

We find that there is confusion surrounding the activity of Task Analysis in the field of HCI, as some authors, like [Preece, 94] or [Mayhew, 99], use the term to refer to the activity of analysing tasks that are now performed, whereas others [Hix, 93] refer to the design of the tasks that the system is to offer. As specified earlier, this research takes the first view of Task Analysis, that is, based on the analysis of the tasks that are now performed. This decision is founded on the fact that it is the most common approach in the consulted sources and that it better matches the broader activity of Context of Use Specification, whereas the design of the tasks that the system is to offer fits in better with the interaction design activity, although both activities are closely related. The result of Task Analysis is very important as a starting point for the design of the functions that the system is to offer. If completed properly, it is at the heart of a true user-centred development process, as the goals of the user must be taken into account and given a prominent place throughout the entire development process.

3.1.4.2 Usability Specifications

Usability specifications are quantitative usability goals, which are used as a guide for determining when a system achieves the right usability level. They can be assimilated to non-functional requirements. They are based on two components: a) user efficiency and b) user satisfaction. Efficiency is interpreted as the desired efficiency level of the larger system composed of the user and the software system working together to achieve given user goals. In this respect, usability specifications can also be called usability benchmarks.

The knowledge gathered in the Context of Use Specification activities forms the basis for this activity. The usability specifications are defined according to the characteristics of the user population gathered in User Analysis and according to goals and tasks identified in Task Analysis.

The set of usability specifications represents the system acceptance criterion from the viewpoint of usability. The usability specifications are evaluated at the end of each development cycle, establishing the progress made towards the established usability goal. They serve as a criterion for determining when to stop iterating.

Although usability attributes are not directly measurable, the usability specifications should be. Therefore, usability attributes are decomposed into subattributes and are specified for definite tasks, based on the results of the Context of Use Specification. This way the values of the usability attributes can be measured indirectly. Each usability specification is linked to a specific usability attribute, but referred to a particular aspect of the attribute in question.

Some authors ([Nielsen, 93], [Wixon, 97] and [Mayhew, 99]) prefer the term “usability goals”, since these specifications are established as a goal to be met by system design. The usability goals drive design as they are part of the information shared by the whole development team, serving as a possible criterion for decision making on any alternative designs that are suggested. They do not only play the role of a test case that is evaluated to check whether or not it is met. Of the two terms we prefer “usability specifications”, because it is closer to the terminology used in SE to indicate what characteristics the system under implementation should have.

3.1.5 Develop Product Concept

This activity is based on mental models ([Norman, 90], [Preece, 94]). The concept of mental model belongs to the field of cognitive psychology and has been manifested in psychological HCI research theories in many different ways [Preece, 94].

Users always develop a model in their mind about how the system works, irrespective of whether an explicit Develop Product Concept activity has been completed during development. The user’s mental model is usually an imperfect model, that is, it does not exactly match the way the system operates, the logic behind the processes that it performs to respond to each user action and the way in which the options and functionalities of the UI are distributed. The ideal thing for system designers is for the user to be able to quickly and easily create a model in his or her mind that matches the picture of the system. The design model is what designers have in their mind, either implicitly or explicitly, and on which system development is based. If the system does not meet the user’s expectations, he or she will find it difficult to use and perplexing, and will make more mistakes. Also, if the product concept is ambiguous, inconsistent, obscure or not well conveyed to the user, he or she will build an incorrect mental model, which will lead to poor system use. Figure 2 (taken from [Preece, 94]) shows how the design model, the real picture of the system and the mental model formed in the user’s mind are related. The user develops a partial mental model (compared with the full design model that designers have) and can, therefore, only make use of part of the potential of the system as conceived by its designers. The product concept to which the name of this activity refers is the design model shown in Figure 2. The term is not as clearly defined in the consulted sources as other HCI concepts, like, for example, usability specifications. It is an elusive term, because it is interpreted differently by different authors. Nevertheless, the sources do agree on how important it is to help users to develop productive mental models in order to improve the learning curve. However, there are no precise instructions on how to achieve this goal, apart from establishing a consistent design model in line with user expectations.

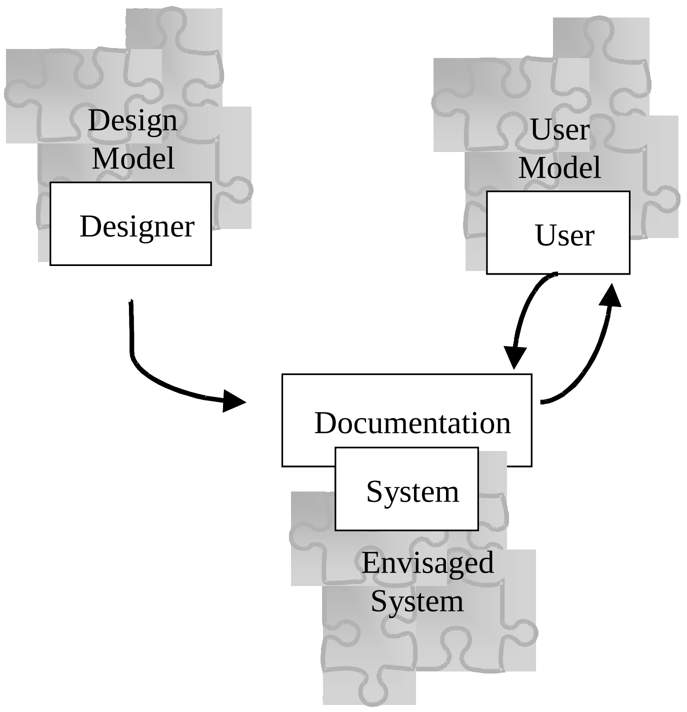

Figure 2 – Relationship between the Design Model, System Image and User Model

SE is not usually concerned with the product concept, as it normally starts from a well-formed idea of the type of software to be developed. Several HCI sources considered in this survey cite techniques that could be applied in the Develop Product Concept activity, as they are exploratory and aim to bring the design team round to a common view of the system under development. As these are tasks that are not routine practice in SE development processes and cover a series of techniques that are of interest for our purpose, we have decided to include this usability activity, even though it meets with less consensus than others listed in Figure 1.

This activity considers the general rules that will govern system operation, its main interaction spaces and how to work the system. If the decision is taken to use any type of metaphor (which relates the product concept to some real-world entity), it is considered in this activity. Metaphors are used to make the design model more easily understandable for the user.

When the product concept of a system is not clearly and explicitly defined, it is very unlikely to be consistent and the user will not be able to understand the system logic, because there will be no such logic. Good designers always have a product concept in mind, but this is not enough. They need to make the effort to specify this concept in line with user expectations. Highlighting its importance in the development process helps to shape the system and explicitly communicate the product concept to the user. One of the principal goals of this activity is to get the design team to take a common view of the product. This goal is close to one of the usual SE concerns, modelling, which aids communication among the team and rules out ambiguities. In the case of the HCI product concept, the user is one of the targets for this communication. The search for coherence and internal logic within a design is an innate goal of any branch of engineering, and HCI's input focuses this coherence on user expectations and knowledge.

3.1.6 Prototyping

ISO 13407 [ISO13407, 99] defines a prototype as “a representation of all or part of a product or system that, albeit somehow incomplete, can be used for the purposes of evaluation”. Prototypes allow designers to communicate more effectively with users and reduce the need and cost of reworking an implemented system when problems are identified later on in the development process. Prototypes need to be built because technical specifications and abstract models are not usually a good means of communication when users are to be involved in the development process.

Prototyping is closely linked to iterative development. For prototypes to be effective, the cost in terms of resources and time should be minimum. The difference between prototyping as proposed by HCI and the prototyping usually employed in SE developments is, again, the approach. Prototypes that are useful from the usability viewpoint primarily reflect user-system interaction to convey how the system will work from the user’s viewpoint. Accordingly, prototypes are used to test out design ideas with users and to gather their impressions [Preece, 94]. HCI prototypes are good for testing design ideas in the earlier development stages, which means that they are the least costly and furthest removed from the finished product.

At any level of abstraction whatsoever, the design should be communicated by means of a prototype. Prototypes increasingly true to the final product can be built from the product concept stage through to the full detailed design for use in usability evaluation activities. The prototyping approach in interactive systems development involves the construction of at least one initial version of the system to demonstrate the main features of the future system. When used early on in the development process, a prototype encourages the participation and involvement of the user, and developers can observe the behaviour of users and their reaction to the prototype [Hix, 93].

An important goal of requirements specification and task analysis is to assure that the proposed system has all the functionality required to carry out the tasks that the user wants to perform. Prototyping is a means of assuring that this goal is achieved, as it is useful for eliciting information from users on [Preece, 94]:

- Necessary system functionality
- Sequences of operations
- User support needs
- Necessary representations
- UI look and feel.

With the current upsurge of iterative approaches in SE, UI prototypes are increasingly more common in software development. HCI offers the possibility of rooting these prototypes in a user-centred approach by basing the prototypes on context of use specification considerations, the developed product concept and the usability specifications.

3.1.7 Interaction Design

Interaction Design is the least well-defined activity in any HCI process, because it varies considerably from one source to another. Hix and Hartson [Hix, 93] indicate, on the one hand, that design is a complex activity and that there are no generally applicable formulas to guarantee success and, on the other hand, assure that design as a process is one of the least well understood activities. One piece of advice common to several sources is that a user-centred focus should be maintained throughout the whole design process, and the work of the design team must, therefore, be based on the products of the earlier usability activities.

Interaction design and UI design are closely related and the differences of terminology between SE and HCI have led to the two concepts being confused. Most authors use the term *UI design* ([Nielsen, 93], [Hix, 93], [Mayhew, 99] and [Shneiderman, 98]), whereas [Preece, 94] uses the term *interaction design* and others use the term *design* generally ([Wixon, 97] and [ISO13407, 99]). Finally, [Constantine, 99] refers to *dialogue design* or *visual design*. We have opted for the term *interaction design* to underline in a SE context that this activity is not only concerned with the design of the visible elements of the UI.

Interaction design is responsible for defining the interaction environments and their behaviour. The inclusion of behaviour involves coordinating the interaction between the user and the system, which leads to design decisions that affect the internal structure of the system. It also includes the design of the visual elements making up the GUI (graphical user interface), when the interface is graphical.

3.1.8 Usability Evaluation

Usability is a complicated concept, because human nature itself is so complex. Without doing some sort of evaluation, it is impossible to find out whether the system is going to satisfy the needs of users and whether it will fit in properly with the physical, social and organisational context in which it is to be used [Preece, 94]. No matter how much emphasis is placed on performing usability activities in the development process, the usability level of the system cannot be exactly predicted beforehand. Therefore, usability evaluation activities need to be conducted throughout development, and especially at the end of each iterative cycle, to find out how usable the product is at any time and how much improvement is needed to achieve the previously established usability goals.

Usability evaluation needs to be conducted at all development stages, although the required formality varies depending on the design products available for evaluation at any time and on time and resource constraints.

According to ISO 13407 [ISO13407, 99], evaluation can be used to:

- provide feedback serving to improve design,
- evaluate whether the user and organisational goals have been achieved, and
- monitor the long-term use of the product or system.

One difference between usability evaluation and traditional validation in SE is that early usability evaluation is intended to drive design. This approach is known as formative evaluation, as it helps to give shape to design. The goal of formative evaluation is to find out which particular aspects of the system are good and which are not and how to improve the design. This approach is quite the opposite to summational evaluation, which aspires merely to produce a rating that indicates the quality of a system when it is finished or nearly finished.

Usability evaluation is a fundamental part of the iterative development approach, because evaluation activities can produce design solutions for application in the next development cycle or, at least, a better knowledge of what the detected interaction problem involves. Therefore, usability evaluation is not a test yielding a pass/fail result, it is an inherent part of the development process.

3.2 Literature Survey for HCI Usability Techniques

In this section we are going to proceed similarly to the way we did in section 3.1 regarding activities to survey the usability techniques discussed in the consulted sources. The goal of this survey is to get a set of techniques that are representative of HCI proposals. These techniques will be our candidates for inclusion in the software development process. Owing to the dispersion observed in the literature, we first ran a survey to get a taxonomy of techniques, organised according to the usability activities within which they are applied.

Because this survey targets an SE audience, we use the concepts of activity and technique as they are understood in SE software process terminology. Activity refers to a task to be carried out, whereas technique refers to the tool used to perform the task (the way in which it is done). That is, a distinction is made between what has to be done (activity) and how it is done (technique). However, these two concepts are not as clearly delimited in most of the HCI field sources, and the *what* and the *how* are mixed up on more than one occasion. Therefore, some concepts that appeared in the earlier section as activities also appear in this section as techniques. However, the approach differs in the two sections, depending on whether they are examined from the viewpoint of activity or technique. For example, the Usability Requirements/ Specifications activity mentioned in [Hix, 93] was included in the earlier section. From the viewpoint of activity, we have reported that efforts at developing usability specifications are made at some point during development. From the technique viewpoint, we analyse what particular format the authors proposed for developing these usability specifications, that is, what method or technique the author suggests for carrying out this activity.

We will first detail the consulted sources and then present the technique classification, to conclude by characterising these techniques for the purpose of selection.

3.2.1 Sources

The sources we consulted for this survey are the same as detailed in the preceding section, except for [Wixon, 97], [ISO9241, 98] and [ISO13407, 99]. These three sources have not be considered because they focus on development activities and include hardly any details on the individual techniques to be applied in each activity.

Additionally, the need for a basic reference for each source has led us to conduct a broader survey than in the last section. Some of the techniques selected for inclusion in the general-purpose development process are not well enough described in the original sources. Therefore, we have looked for a basic reference to which the user of the integration framework presented in this research can resort if he or she wants to know more about the technique.

Finally, one technique that is commonly used at present among HCI experts, namely Personas, is not referenced in the consulted sources. This is a very recent technique (first referenced in 1999 [Cooper, 99]), whose use is especially widespread in web development, although its author claims that it can be used to design software of any type. An example of the impact of this technique is its use by Microsoft’s MSN portal (called “MSN Personas”) in its marketing strategy for procuring advertisers, indicating that they are interested in who their users are [MSN, 03]. The technique is also listed by the following reputed web sites that deal with the usability issue as a technique that contributes to usable software development:

- UsabilityFirst http://www.usabilityfirst.com/ [DiamondBullet, 04]
  Compendium of the principal usability concepts offered by the Diamond Bullet Company. This firm offers usability consulting services. The site was created in 2002 and was updated in 2004. Its usability glossary contains over 1000 terms. Note that the technique appears in this glossary as “Persona” as opposed to “Personas”.

- Dey Alexander: Web usability and user-centred design consulting and resources http://deyalexander.com/ [Alexander, 04]
  Web site maintained by Dey Alexander, web usability and accessibility consultant. It contains an excellent collection of resources related to usability in the field of web development, composed of numerous links to other sources of information on the web for each subject, along with a brief explanation.

- SAP Design Guild http://www.sapdesignguild.org/ [SAP, 04]

SAP has recently adopted a policy of making usability a strategic goal of the company. The company offers this comprehensive web site as proof of its commitment to usability for the purpose of sharing its knowledge with the UI design community as a whole.

As regards the origin of the Personas technique, note that Cooper presents the technique as his in [Cooper, 03b], although he does admit that similar constructions have been used earlier by other interaction designers or marketing professionals. According to Cooper, the first time the technique was formulated as he proposes was in [Cooper, 99]. Both [Lombardi, 04] and [Mcmullin, 03] believe that the use of archetypal users goes back to before Cooper's proposal of calling it Personas. The term "Persona" dates back to classical Greek theatre, being the word by which the masks that were used to represent roles were known, and is a term that Carl Jung introduced into psychology as part of his personality theory (http://www.lessons4living.com/persona.htm). When we use the term "Personas" in this research, we are referring to the HCI technique stated by Alan Cooper.

3.2.2 Classifying Techniques

Having identified the technique proposed by each source, we have classified the techniques. This involved first assigning each technique to the user-centred development process activity in which it best fits. We had to do this because most proposals are not very formalised. Second, it involved grouping together the techniques proposed by different authors that refer to the same basic technique. Terminology varies in most cases from author to author, and we had, therefore, to choose, for each technique, the most representative name or the name that we considered most illustrative from the viewpoint of SE. The result of this classification is set out in several tables: three tables detail the techniques related to analysis activities (Table 2, Table 3 and Table 4), two tables list techniques related to design activities (Table 5 and Table 6), and another four tables specify techniques related to evaluation activities (Table 7, Table 8, Table 9 and Table 10). Each table has a column for each consulted source, and the two columns furthest to the left specify the activities within which the techniques fit and the names chosen for each technique. Each row is occupied by one technique. We have also accounted for possible variations on techniques, and both the generic technique and each variation has its own row (for example, Table 4 shows Card Sorting along with its four variants).

Table 2, Table 3 and Table 4 list the techniques related to analysis activities. As shown in Table 2, whereas some techniques are directly related to User or Task Analysis, other techniques related to Context of Use Specification are not only confined to one or another of the analysis types, but affect the context of use specification generally and are, therefore, listed in the table in first place.

Table 2 – Context of Use Specification-Related Techniques

Activity	Technique	Hix, 93	Nielsen, 93	Preece, 94	Shnei-derman, 98	Constan-tine, 99	Mayhew, 99
	Competitor Analysis		Competitor analysis				
	Financial Impact Analysis		Financial impact analysis				
	Contextual Inquiry	Contextual inquiry		Contextual inquiry			Contextual interviews
	Affinity Diagramming						Affinity diagrams
	Ethnographical Observation			Ethnography	Ethnographical observation		

Activity		Technique		Hix, 93	Nielsen, 93	Preece, 94	Shnei-derman, 98	Constan-tine, 99	Mayhew, 99
		JEM[1]						JEM	
	User Analysis	**User Profiling**		User profiles	Individual user characteristics		User profiles	Structured role model	User profile questionnaires
		User Role Mapping						User role map	
		Operational Modelling						Operational model	Platform constraints and capabilities
		Personas		*This recent technique is not mentioned in any of the sources, but is included for the reasons detailed above in section*					
	Task Analysis	**Essential Use Cases**						Essential use cases	
		Task Organisation Modelling			Hierarchical task decomposition				Task organisation model
		HTA[2]				HTA			
		GOMS Model Family[3]			GOMS	GOMS	GOMS		
			NGOMSL			NGOMSL	NGOMSL		
		Object-Action Interface Modelling					Object-action interface model		
		Task Scenarios							Task scenarios
		Task Sorting[4]							Task sorting

Table 3 lists the techniques designed for establishing the Usability Specifications. Note that these techniques are also closely related to usability evaluation activities, because the usability specifications describe the goals against which system usability is evaluated (except qualitative goals, which cannot be measured).

Table 3 –Usability Specifications-Related Techniques

Activity	Technique		Hix, 93	Nielsen, 93	Preece, 94	Shnei-derman, 98	Constan-tine, 99	Mayhew, 99
Usability Specifications	**Usability Specifications**		Usability specifications	Usability goals	Usability specifications			Usability goals
		Performance goals	Objective measures					Performance goals
		Satisfaction goals	Subjective measures					Satisfaction goals
		Usability Goal Line		Usability goal line				

[1] JEM: Joint Essential Modeling
[2] HTA: Hierarchical Task Analysis
[3] GOMS: Goals, Operations, Methods and Selection Rules
[4] Task Sorting is a variation on the Card Sorting technique described in Table 4

Activity	Technique	Hix, 93	Nielsen, 93	Preece, 94	Shneiderman, 98	Constantine, 99	Mayhew, 99
	Preference Goals						Preference Goals
	Qualitative Goals						Qualitative Goals

Table 4 lists Card Sorting, alongside a series of variations of this technique. This technique is concerned with analysis, that is, with the elicitation of information from users, but is not directly related to either of the analysis activities. Note that Task Sorting appears both in this table and in Table 2. This is because it is a variant of Card Sorting that is directly related to the Task Analysis activity.

Table 4 – Analysis-Related Techniques not Specific to any Activity

Activity	Technique	Hix, 93	Nielsen, 93	Preece, 94	Shneiderman, 98	Constantine, 99	Mayhew, 99
Analysis generally	**Card Sorting**		Card Sorting			Card Sorting	
	Affinity Grouping					Affinity Grouping	
	Criteria Ordering					Criteria Ordering	
	Threshold Voting					Threshold Voting	
	Task Sorting						Task Sorting

Table 5 shows the techniques related to specific design activities. Note that the family of GOMS-type models is also included here, because it models the tasks performed by users at several levels. Whereas the higher levels (goals and subgoals) serve the purpose of Task Analysis, the more specific levels (operators, methods and selection rules) are the focus of the Interaction Design activity. Therefore, they are included in both Table 2 and Table 5.

Table 5 –Design-Related Techniques

Activity	Technique	Hix, 93	Nielsen, 93	Preece, 94	Shneiderman, 98	Constantine, 99	Mayhew, 99
Develop Product Concept	**Scenarios and Storyboarding**			Instantaneous scenarios and storyboards	Scenarios	Scenarios and storyboards	
	Visual Brainstorming			Visual Brainstorming			
	Conceptual Modelling						Conceptual Model
Prototyping	**Prototyping**	Prototyping	Prototyping				
	Scenario Prototyping		Scenarios				
	Active Prototyping			Requirements animation		Active prototypes	High-fidelity mock-ups
	Paper Prototyping			Low-fidelity prototyping		Passive prototypes	Low-fidelity mock-ups
	Scripted Prototyping			Scripted prototypes			

Activity	Technique	Hix, 93	Nielsen, 93	Preece, 94	Shneiderman, 98	Constantine, 99	Mayhew, 99
	Wizard of Oz Prototyping			Wizard of Oz prototypes			
Interaction Design	**Screen Snapshots**	Scenarios and screen snapshots					
	Product Style Guides						Product Style Guide
	Grammars				Grammars		
	GOMS-Family Models		GOMS	GOMS	GOMS		
	NGOMSL			NGOMSL	NGOMSL		
	UAN[5]	UAN			UAN		
	TAG[6]				TAG		
	Menu Trees				Menu Trees		
	Interface State Transition Diagrams	Interface State Transition Diagrams			Transition Diagrams		
	Harel State Diagrams				Harel State Diagrams		
	Interface Content Modelling					Interface Content Model	
	Navigation Map					Context Navigation Map	

Within the design-related techniques, we find a series of techniques that refer to how design decisions are made, to grasping the logic behind the design decisions or to the help subsystem design. These techniques, listed in Table 6, are not related in particular to any of the design activities identified in section 2.2.

Table 6 – Design-Related Techniques not Specific to any Activity

Activity	Technique	Hix, 93	Nielsen, 93	Preece, 94	Shneider-man, 98	Constantine, 99	Mayhew, 99
Design generally	**Integrational design**					Integrational design (*both-and design*)	
	Parallel Design		Parallel Design				
	Impact Analysis	Cost / Importance Analysis	Impact Analysis	Impact Analysis			
	Organisation of Help by Use Cases					Organisation of Help by Use Cases	
	IBIS[7] and PHL[8]			IBIS and PHL			

[5] UAN: User Action Notation
[6] TAG: Task-Action Grammars
[7] IBIS: Issue-Based Information Systems
[8] PHI: Procedural Hierarchy of Issues

Activity	Technique	Hix, 93	Nielsen, 93	Preece, 94	Shneider-man, 98	Constantine, 99	Mayhew, 99
	Design Spaces Analysis			Design Spaces Analysis			
	Statement Analysis			Statement Analysis			

The usability evaluation techniques have been divided into four tables. The first three list the techniques classed according to the three main usability evaluation types: Evaluation by Experts (Table 7), Usability Testing (Table 8), Installed Systems Follow-Up Studies (Table 9). Table 10 details other usability evaluation techniques that do not fall into any of these three categories.

Of the usability evaluation by experts techniques, listed in Table 7, note that for Constantine and Lockwood, the Conformity Inspections cover both Standard Conformity Inspections and Guideline Checklists.

Table 7 – Techniques Related to Usability Evaluation by Experts

Activity	Technique		Hix, 93	Nielsen, 93	Preece, 94	Shneider-man, 98	Constantine, 99	Mayhew, 99
Evaluation by Experts	**Heuristic Evaluation**		Heuristic Evaluation	Heuristic Evaluation	Heuristic Evaluation	Heuristic Evaluation	Heuristic Evaluation	Heuristic Evaluation
	Inspections	**Standard Conformity**			Standard Inspections		Conformity Inspections	Standard Inspections
		Guideline Checklists				Guideline Checklists		Guideline Checklists
		Consistency			Consistency Inspection	Consistency Inspection	Consistency Inspection	Consistency Inspection
		Collaborative					Collaborative Usability Inspections	
	Walkthroughs	**Cognitive**			Cognitive Walkthrough	Cognitive Walkthrough	Cognitive Walkthrough	Cognitive Walkthrough
		Pluralistic		Pluralistic Walkthrough	Pluralistic Walkthrough		Pluralistic Usability Walkthrough	Pluralistic Walkthrough

Table 8 lists the techniques that can be applied for usability testing.

Table 8 – Techniques Related to Usability Testing

Acti-vity	Technique		Hix, 93	Nielsen, 93	Preece, 94	Shnei-derman, 98	Constan-tine, 99	Mayhew, 99
Usabil-ity Tes-ting	**Thinking Aloud**		Concurrent Verbal Protocol Taking	Thinking Aloud	Thinking Aloud Protocol		Talk to me (thinking aloud)	Formal usability testing (in early stages)
		Constructive Interaction		Constructive Interaction				
		Retrospective Test	Retrospective Verbal Protocol Taking	Retrospective Testing	Post-Event Protocol		Deferred Reflection	

Acti-vity	Technique		Hix, 93	Nielsen, 93	Preece, 94	Shnei-derman, 98	Constan-tine, 99	Mayhew, 99
		Critical Incidence Taking	Criticial Incidence Taking					
		Training Method		Training Method				
	Performance Measurement				Reference Tasks		Performance Metrics	Formal usability tests (in later stages)
	Post-Test Information						Post-Test Information	
	Laboratory Usability Testing		Laboratory Test	Usability Laboratories		Laboratory and Usability Testing	Laboratory Testing	
	Field Testing		Field Testing				Field Testing	
	Video Recording		Video Recording	Video Recording	Video Recording			
	Audio Recording		Audio Recording		Verbal Protocol			
	Use Logging		Internal Interface Instrumentation	Use Logging	Software Logging	Continuous User Performance Logging		
		Time-Stamped Keypresses			Time-Stamped Keypresses			
		Interaction Logging			Interaction Logging			
	Remote Evaluation							Remote Control Logging
	Remote Videoconference Testing							Remote Videoconference Logging

Table 9 details the usability evaluation techniques that can be used to monitor installed systems. Note that a series of use logging techniques appear in both this table and Table 8, because they can be used for both traditional usability testing and to collect real interaction data on the use of an installed system.

Table 9 – Techniques Related to Installed Systems Follow-Up Studies

Activity	Technique		Hix, 93	Nielsen, 93	Preece, 94	Shnei-derman, 98	Constan-tine, 99	Mayhew, 99
Installed System Follow-Up Studies	**Direct Observation**			Observation	Direct Observation			
		Random Observation						Use Studies - Random Observation
	Questionnaires and Surveys			Questionnaires	Questionnaires and Surveys	Surveys		
	Interviews			Interviews	Interviews	Interviews		

Activity	Technique		Hix, 93	Nielsen, 93	Preece, 94	Shnei-derman, 98	Constan-tine, 99	Mayhew, 99
		Structured Interviews	Structured Interviews		Structured Interviews			
		Flexible Interviews			Flexible Interviews			
	Focus Groups			Focus Groups		Discussions Focus Groups		
	Use Logging		Internal Interface Instrumen-tation	Real Use Logging	Software Logging	Continuous User Performance Logging		Instrumented Remote Logging
		Time-Stamped Keypresses			Time-Stamped Keypresses			
		Interaction Logging			Interaction Logging			
		Random Activation Software Monitors						Use Studies – Software Monitors
	User Feedback			User Feedback		Suggestions Box or On-Line Error Reporting		
		On-Line User Help Services				On-Line or Telephone Operators		
		Forums				Newsgroups and Bulletin Boards		
		User Journals and Conferences				User Journals and Conferences		
		Semi-Instrumented Remote Evaluation						Semi-Instrumented Remote Evaluation

Finally, Table 10 lists other usability evaluation techniques that do not belong to any of the main categories of usability evaluation techniques.

Table 10 – Usability Evaluation Techniques Outside any of the Main Three Categories

Activity	Technique			Hix, 93	Nielsen, 93	Preece, 94	Shneiderman, 98	Constantine, 99	Mayhew, 99
Usability evaluation generally	**Experimental Testing**					Traditional Tests	Psychologically Driven Controlled Experiments		
	Predictive Metrics	**Procedural**						Procedural Predictive Metrics	
			Press Modelled			Press Level Model			

Activity	Technique		Hix, 93	Nielsen, 93	Preece, 94	Shneiderman, 98	Constantine, 99	Mayhew, 99
		Structural					Structural Predictive Metrics	
		Semantic					Semantic Predictive Metrics	
	Cooperative Evaluation				Cooperative Evaluation			

3.2.3 Technique Characterisation

The survey detailed in the preceding section came up with a total of 96 techniques (including the variants of each general technique). This is too many for them to be easily manageable for a software engineer, bearing in mind that they belong to a field outside SE. Such a wide range of techniques can lead to the problem of saturation, with the risk of software engineers giving up on the goal of integrating usability activities and techniques into the development process. To ease the work of the software engineer, we are going to detail how useful each technique analysed is for the overall goal of integration from the viewpoint of SE according to some specific criteria.

3.2.4 Criteria for Characterisation

The criteria we are going to use to assess usefulness for the set goal are described below. The value assigned to each criterion for each technique will be listed in a series of tables.

- **User Participation:** One of the basic points of a user-centred process is the active involvement of the future system users. Some usability techniques are specifically designed to encourage this involvement, and they are going to be highlighted in this study. The fact that a technique allows the active involvement of the user is specified in the tables by means of a cross (x) in the column labelled “UP”.

- **Training Needs:** This criterion refers to how much training an average software engineer would need to be able to apply the technique with any chance of success. A “very high” training need indicates that extensive usability experience is required to be able to apply the technique, that is, the personnel applying the technique need to be experts. The “high” value indicates that it would require at least a combined SE-usability profile, meaning that a software engineer that could apply the technique would need to have been extensively trained in usability. A “medium” value indicates that, although quite a lot of training is needed, it can be applied by average software engineers who have been given this training. The “low” value indicates that average software engineers only need basic training to be able to apply the technique.

- **General Applicability**: This criterion reflects the scope of the technique, that is, how applicable it is to wide range of software development projects. A high applicability score indicates that it can be of practical use in all sorts of projects. If it scores a medium value for this criterion, the technique is applicable in some project types but not in others. Finally, a low value for this criterion indicates that it is only suitable for application in specific project types that account for only a small percentage of all projects. The column for this criterion is labelled “Applicability” in the tables.

- **Proximity to SE**: This criterion reflects whether the principles on which the technique is based match the principles and approaches usually present in SE. A high value for this criterion indicates that software engineers can apply the technique, as it is based on standard SE skills and approaches. A medium value indicates that, while the technique is based on principles that do not belong to SE, it is not so far removed from it as to be considered foreign to the field. A low value for this criterion indicates that the technique requires a development approach and skills that have nothing to do with the ones that an SE-trained person usually takes or has.

- **Usability Improvement/Effort Ratio**: This criterion refers to how much the use of the technique can improve the usability of the final product compared to the effort involved in its application. As resources are usually scarce in many software development projects, we believe that cost/benefit type information can be very useful for selecting the techniques for application. The possible values for this criterion are high, medium and low, reflecting the size of the improvement in usability that the application of the technique can yield as compared to the effort that needs to be invested by the development team in its application. The column for this criterion is labelled "Improvement/Effort" in the tables.

- **Representativeness**: This criterion reflects how commonly the technique is applied in the field of HCI. As an indicator of this criterion, we are going to use the number of consulted authors who recommend technique application. The value is, therefore, numerical and falls within the range of 1 to 6. In the case of techniques for which variants have been defined, the aggregate value of all the authors who describe either the general technique or any of its variants is taken into account. For example, Table 9 shows that the Direct Observation technique is mentioned by [Nielsen, 93] and [Preece, 94], and its Random Observation variant also by [Mayhew, 99]. Therefore, its representativeness is 3.

It should be noted that the values detailed for each of these criteria are approximate. They have been assigned based on a SE perspective and aim to ease the task of integration by reducing the number of techniques for consideration. As the goal is to select techniques, we have summarised the combined value of each criterion in a single value called **Total Rating**, the possible values of which are very useful, useful and not very useful. The criteria for assigning each value are as follows:

- **Very useful:** Techniques that are especially useful for our purpose have the following features:
  - Their usability/effort ratio is very high, as we are especially interested in techniques that yield more usability with less effort.
  - They do not require a lot of training, because this will make them easier to introduce into organisations with little previous experience in HCI techniques.
  - They are within the medium to high applicability range, because we are concerned that they should have a fairly representative scope of application.
  - They are either very close to SE or are techniques that are commonly applied in the field of HCI (representativeness greater than or equal to three). In the first case, they will be easier to introduce into an organisation that has little HCI experience, whereas, in the second, they are techniques whose proven worth has led to their citation in several literature sources.

- **Useful**: Whereas the techniques that meet the above conditions are the best for the aim we pursue, other techniques may also be useful. This rating includes techniques that meet any of the following conditions:

  - The usability/effort ratio is high, because we consider that techniques with this value are useful irrespective of other criteria.
  - The usability/effort ratio is medium and also any of the following conditions apply:
    - Applicability is medium or high, as the solution to be presented to software engineers should be for general purposes and the total number of techniques should be small enough for it to be useful as a tool.
    - It is a technique that encourages user participation and meets the conditions detailed below. As it is a central feature of user-centred processes and is not common in SE, we have relaxed the conditions for this type of techniques to be considered useful. Therefore, even if the applicability of the technique is low, it is considered useful if its representativeness is greater than or equal to three, because it is then a standard technique in user-centred development processes.

- **Not very useful**: Techniques that do not meet the requirements for belonging to either of the above categories have been labelled as not very useful in the total rating. Although they may be useful for some development projects, we do not think that this is enough to merit their inclusion in the general integration schema, since software engineers could be deterred by the inclusion of the whole HCI field, which would make the proposal unnecessarily complicated.

The values of the criteria for each technique analysed is shown in several tables: the techniques related to analysis are considered in Table 11, design-related techniques in Table 12, and evaluation activity-related techniques in Table 13. The variants have not been detailed, unless the value of any criterion for any variant differed from the value of the parent technique.

3.2.5 Characterisation of Techniques Related to Analysis

Table 11 lists the values for analysis-related techniques, as detailed below:

- Contextual Inquiry and Ethnographical Observation are very similar insofar as they call for a lot of training, are not applicable to all project types (medium value for general applicability) because the customer organisation needs to be highly accessible, and score medium for proximity to SE. As regards the improvement/effort ratio, Contextual Inquiry sets up a dialogue with the users that can be very useful for the reason that it provides information that can be used in the design of a more usable system and takes less time to apply than Ethnographical Observation, where observation is passive. Therefore, Ethnographical Observation has a medium value for usability improvement, whereas Contextual Inquiry has a high value.

- Affinity Diagramming and Card Sorting are also similar techniques in that they are both easy to apply, which means that the training needs are low and they are useful in a wide variety of projects (high applicability). They have a medium rating for proximity to SE, because they are participatory techniques that are not usually applied by software engineers. They both also have a high improvement/effort ratio, as their cost of application is very low, whereas they can be applied to improve the usability of the final product.

- With respect to task modelling techniques (Essential Use Cases, Task Organisation Model and HTA), they all have medium training needs, because, although they are relatively simple notations, some training in how to think about the problem to be modelled is required for them to be applied with any prospects of success. The first two are applicable to any project type, because user tasks should be considered throughout the entire project (high applicability), since they are a basis of the user-centred approach. However, the HTA technique focuses on copying how tasks are now performed, which means that there already needs to be a system in place (medium applicability). Since modelling is common in SE, they have a high score for proximity to SE. As regards the improvement/effort ratio, Essential Use Cases are quite difficult to apply and refine, but the possible improvement to system usability is significant, which means that it scores high. The Task Organisation Model is easier to produce, and has a high score, too, because it also contributes a lot to usability improvement. Because it has a detailed application process, HTA, on the other hand, calls for a somewhat bigger application effort than the Task Organisation Model, and its contribution to improving the usability of the final product is not as big, which means that its rating for this criterion is medium.

- The user modelling techniques (User Profiles, User Role Map and Operational Model) are close to SE, because they are specification and modelling tasks. As regards training needs, whereas they are low for the User Role Map, because this a very simple technique, the User Profiles and Operational Model call for a lot of knowledge of usability principles to be able to understand what information may or may not be relevant for final system usability. Therefore, the value for this criterion is high for both techniques. User Profiles are indispensable in any project, because the knowledge of future system users is one of the fundaments of the user-centred approach. Therefore, their general applicability is very high. User Role Maps are only applicable when there are a lot of different roles and relationships can be established between them, which means that their general applicability is medium. The Operational Model is only necessary as a separate technique when the characteristics of the environment in which the system under development is to be used are especially relevant for final system usability. This means that its use is confined to projects that have special features and, therefore, its general applicability value is low. As User Profiles are essential for system usability, their improvement / effort ratio is high, even though they call for quite a big effort. On the other hand, the User Role Map is not costly to apply, but has little usability benefit, which means that its score for this criterion is medium. Finally, the Operational Model calls for some application effort, but it improves usability appreciably when it is applicable. Its rating is, therefore, medium.

- The formal techniques for Task Analysis and related actions (GOMS and Object-Action Interface) have in common that they call for extensive training (the value is very high for GOMS and high for the Object-Action Interface, because, not including cognitive aspects, it is somewhat simpler). Their general applicability is low because they are not practicable for systems with a complex or very complicated UI, as it would take a tremendous effort to formalise all the possible actions. GOMS-type models are not close to SE, because they include cognitive aspects, which are not among the classical concerns of SE. Therefore, they have a low score for this criterion. Although the Object-Action Interface Model bear more resemblance to standard modelling used in SE, it is not that close, because it focuses on directly manipulable interfaces, which means its rating is medium. Like any formal technique, these three techniques have a high application effort, whereas the usability improvements in the final system are not so big. Therefore, the improvement/effort ratio is low.

- The JEM technique requires some training, as it is a detailed modelling process, and its training needs are, therefore, medium. It is applicable in specific, albeit fairly general cases, as it is appropriate for when you want to get all the stakeholders to feel that what they have to say counts. Accordingly, its general applicability is medium. It is close to SE (high value), because it is based on the JAD (Joint Application Design) technique, an SE technique related to the requirements process. Its application cost is considerable, as it involves multidisciplinary meetings, but its contribution to usability is sizeable, and its improvement / effort ratio is rated medium.

- The Task Scenarios and Personas techniques are applied when there is a wide range of users and real tasks, and they are useful for abstracting possible users (Personas) and/or their tasks (Task Scenarios), which, while they do not match up with any real user, are representative of what an average user and what a standard task are like. Accordingly, their general applicability is medium, because development will not necessarily by tackled in this way in all projects. Both are close to SE as regards their goal of abstraction, but the way in which Personas is applied bears more resemblance to what market researchers do than to how a SE process is enacted and Task Scenarios do not match up with the classical SE approach, which means that both techniques have a medium score for this criterion. Both techniques call for considerable training, and their needs in this respect are medium. The effort required to apply the Personas technique is considerable, but its contribution to usability can be very sizeable. Therefore, it has a high improvement / effort ratio, whereas the Task Scenarios technique does not offer as much return on usability, and its rating for this criterion is medium. As regards the Personas technique, its representativeness score would be zero, because it is not referenced by any of the considered sources, as mentioned above. We have rated the representativeness value for this technique as three, because its use is now widespread.

- Competitor Analysis calls for quite a lot of training, because engineers need to learn what to look for in a competitor product and how to get its usability-related strengths and weaknesses, which means that its training needs are medium. It is applicable to all sorts of projects, even to innovative systems, because the analysis of potentially similar products can be useful for imitating what works well. Accordingly, its general applicability is high. System evaluation is well known in SE, but the analysis of competitor products is not as common, which means that the value for proximity to SE is medium. Although the effort for applying this technique can be considerable in some cases, the benefits in terms of final system usability more than compensate for the costs, which means that the improvement / effort ratio is high.

- Financial Impact Analysis calls for considerable usability training, and, therefore, its value for this criterion is high. It is applicable whenever efficiency of use is a considerably important attribute, but it is not as useful in other cases, which means that its general applicability is medium. Return on investment studies are not foreign to SE, but this technique focuses on usability and uses variables that are not common in SE. Therefore, proximity to SE is medium. It makes little contribution to the usability of the final system, because the technique is primarily concerned with decision making on how big the project’s usability effort needs to be (low improvement / effort)

- Establishing Usability Specifications calls for considerable usability training and experience, including knowledge of what goals can be achieved at a reasonable cost. Therefore, this technique scores high for training needs. This technique is applicable to problems in which it is feasible to find out what tasks need to be supported at the start of development, whereas the technique is not valid if the system is not based on user tasks in an office environment. Therefore, the rating for the general applicability criterion is medium. Being non-functional requirements, usability specifications are close to SE, although they include the (human) user in the goals to be established. Accordingly, the knowledge required to establish the specifications is very unlike what a software engineer usually handles. Therefore, as it has two contradictory sides, we think that the value for proximity to SE is medium. Finally, the improvement / effort ratio is high insofar as they are a very useful tool for finding out the usability level to be achieved and being able to establish how far away we are from this level, although the effort for establishing the specifications can be high.

Table 11 – Rating of Analysis-Related Techniques

Technique	UP	Training Needs	Applic-ability	Proxim-ity to SE	Improve-ment/Effort	Representa-tiveness	Total Rating
Competitor Analysis		medium	high	medium	high	1	Useful
Financial Impact Analysis		high	medium	high	low	1	Not very useful
Contextual Inquiry	x	high	medium	medium	high	3	Useful
Affinity Diagramming	x	low	high	medium	high	1	Useful
Ethnographical Observation		high	medium	medium	medium	2	Useful
JEM	x	medium	medium	high	medium	1	Useful
User Profiles		high	high	high	high	5	Useful
User Role Map		low	medium	high	medium	1	Useful
Operational Modelling		high	low	high	medium	2	Not very useful
Personas		medium	medium	medium	high	3	Very useful
Essential Use Cases		medium	high	high	high	1	Very useful
Task Organisation Model		media	high	high	high	2	Useful
HTA		medium	medium	high	medium	1	Useful
Family of GOMS Models		very high	low	low	low	3	Not very useful
Object-Action Interface Model		high	low	medium	low	1	Not very useful
Task Scenarios	x	medium	medium	medium	high	1	Useful
Card Sorting	x	low	high	medium	high	3	Very useful
Usability Specifications		medium	medium	medium	high	4	Very useful

3.2.6 Characterisation of Design-Related Techniques

Of the design-related techniques, Prototyping generally and its variants of Scenario Prototypes and Active Prototypes have not been included, because, although they are described in the HCI literature, they are known as Mock-Ups or Executable Prototypes in SE. Neither have Driven Prototypes been included, because they are the standard SE demos. The following prototyping variations have been included: Paper Prototypes and Wizard of Oz Prototypes, as they are proper to HCI and are not usually applied in SE. As regards the techniques that capture design logic (IBIS, PHL, Design Spaces Analysis and Statement Analysis), their contribution to usability cannot be considered separately, as their goal is to capture any type of design decision. Therefore, they have not been included.

Table 12 lists the values for the techniques related to design, as detailed below:

- The Scenarios and Storyboards techniques are not close to SE (low value), as they call for the future system and its context to be imagined at a detailed level for a particular case, which is not a routine thing in SE. Also, this means that training needs are medium. Scenarios and

Storyboards are applicable when the system is innovative or when the stakeholders (principally the customer) do not know how to accurately express what it is they want. While a lot of projects will have these characteristics, a lot of others will not. Therefore, the general applicability level is medium. When they are applicable, they do lead to an improvement in the usability of the final product, which we can be qualified as high as compared to the effort invested.

- Visual Brainstorming is not a complex technique and, therefore, training needs are low. It is applicable to any type of problems in the early meetings that shape the project (high general applicability). It is not close to SE, because software engineers do not usually work like this. However, it is an inexpensive technique in terms of application effort, which makes a sizeable contribution to improving the usability of the final product (high improvement / effort ratio).

- The Conceptual Model and Product Style Guides techniques can be considered to have a medium rating for proximity to SE, because, on the one hand, they meet the need of specifying rules to drive design (a goal that is not foreign to SE), whereas, on the other, these rules are concerned with elements that are part of Interaction Design, which, according to the SWEBOK [SWEBOK, 01], is not a part of SE. Therefore, they have a medium score in terms of proximity to SE. The construction of a Conceptual Model or Product Style Guide calls for extensive experience in usability issues, and we, therefore, consider that training needs are high. The general applicability of the Conceptual Model is rated as high, as it is useful to capture even a simple conceptual model of any interaction design that is in any way complex. On the other hand, the Product Style Guide is only justified for somewhat complex systems, especially when a family of products is involved. Therefore, its general applicability is medium. The effort involved in constructing a Conceptual Model will depend on the complexity of the problem, and appreciable improvements in usability can be gained in all cases, which means that its improvement / effort ratio rating is high. However, the Product Style Guide is very costly in terms of effort, and the success in terms usability improvement is somewhat less. Its improvement / effort level is, therefore, medium.

- The two types of Prototype (Paper and Wizard of Oz) are simple techniques, which do not require too much training (low level). Paper Prototypes are applicable to any type of projects (high applicability level), whereas the Wizard of Oz Prototypes are useful only for projects in which the planned system behaviour can be easily acted out by a person at less cost than programming this behaviour, which means that its applicability is generally low. Paper Prototypes are close to SE (high score), as they are similar to some mock-ups that are made as demos for the customer. On the other hand, when these prototypes are used to test usability, as in the case of the Wizard of Oz Prototypes, they are not as close to SE (medium score). Paper Prototypes are easy to produce and are a technique that makes a very sizeable contribution to the usability of the final product, which means that their improvement / effort ratio is high. On the other hand, the effort involved in producing Wizard of Oz Prototypes is considerable, as the user must not suspect that he or she is not working with the real system. Therefore, the rating for this criterion (improvement / effort) is medium.

- The Screen Snapshots technique is a specification of the contents of the visible part of the UI, which is an activity type that is very close to standard SE activities (high score). Its training needs are medium, as UI design is not standard practice among software engineers, and the application of this technique would call for extensive training in usability. Its general applicability is low, because it is not worthwhile specifying the UI in most software development projects, as the specification effort is no less than for directly creating the final product. It is only recommended when the role of the UI design team needs to be spelled out, because a multidisciplinary and integrated team is out of the question (owing to geographical distance, for example). The specification effort using screen snapshots is high, and the improvement in usability is not especially noteworthy, which means that the improvement / effort ratio is low.

- The Grammars technique is only applicable for command line-based UIs, which account for a very small percentage of the UIs designed today, meaning that the general applicability value is low. The technique is relatively complex, and therefore calls for a medium training level. Being a highly formal type of technique, it is close to SE techniques (high score). Finally, its improvement / effort ratio is medium, as it has a high cost and the improvements in usability are appreciable, but not outstanding.

- The UAN and TAG techniques are both very complex, and their training needs are, therefore, very high. With respect to their proximity to SE, they can be rated as having a medium value, because, in spite of the fact that they are highly formal, they deal with mental concepts that are not commonly addressed by SE. Their general applicability is low, because as mentioned above for the GOMS family of models, these techniques will only be applied in projects where all behaviours need to be formally specified owing to their high cost. The application of the techniques leads to some usability improvements, but they take a substantial effort to apply, which means that the improvement / effort ratio is low.

- Menu Trees are a relatively simple specification technique (their training needs are low), whose general applicability is high, because menu-based UIs are very common. The technique is close to SE (high score), because it is similar to other modelling techniques from SE. Because of its simplicity, the technique is not costly to apply, and the usability benefits can be sizeable, which means that its improvement / effort ratio is high.

- Being relatively complex, the other interaction modelling techniques (Interface State Transition Diagrams, Harel State Diagrams, Interface Content Modelling and Navigation Map) all have medium training needs, except for Transition Diagrams that are simpler and have low training needs. Their general applicability is high, as they are based on describing the transition in a graphical UI, which is the most common UI at present. Only the Harel State Diagrams have a low applicability, as they are especially designed to reflect concurrency and synchronisation concerns, which are not big issues in most cases. These techniques are rated as being close to SE (high value), because, as discussed for Menu Trees, modelling is a standard SE activity. Interface Content Models are the exception, as they use resources that are not common in SE (such as Post-It notes), and this technique is rated medium for proximity to SE. The improvement versus effort ratio is low for Harel State Diagrams, as this is the most difficult technique to apply, and the usability improvement it can provide is not very high. The other techniques have a medium value, as they are somewhat costly, but do offer an appreciable improvement in final product usability.

- The Integrational Design and Parallel Design techniques are applied for projects requiring some level of creativity when several designs need to be tested to find the best solution. This is a common situation in many but not all projects. Therefore, general applicability is medium. Comparing different designs is a practice that often emerges in SE, therefore, proximity is high. As regards training needs, Parallel Design is not a sophisticated technique and scores low for this criterion, whereas Integrational Design does involve a special-purpose process that needs to be learned (medium level). Both design strategies consume quite a lot of resources, as there is more than one design team working separately towards the same aim, and, although they can contribute to final product usability, the improvement / effort ratio is low, because what they achieve are not so much usability improvements as a good design decisions.

- Impact Analysis is similar to using SE estimation techniques, which means that proximity to SE is high. The technique calls for some, albeit not excessive, training, which means that score for this criterion is medium. It takes a lot of effort to calculate the Impact Analysis but there are substantial usability benefits, and, therefore, the improvement / effort ratio is medium.

- The Organisation of Help by Use Cases technique calls for some training (medium training needs) and can be applied to systems in which a limited number of representative use cases can be extracted (medium general applicability). Being based on use cases, the technique is somewhat related to SE (high level) and, while it is not excessively costly to apply, the benefits are not outstanding either (medium improvement / effort ratio).

Table 12 – Rating of Design-related Techniques

Technique	UP	Training Needs	Applic-ability	Proxim-ity to SE	Improve-ment/Effort	Representa-tiveness	Total Rating
Scenarios & storyboards	x	medium	medium	low	high	3	Very useful
Visual Brainstorming	x	low	high	low	high	1	Useful
Conceptual Model		high	high	medium	high	1	Useful
Paper Prototypes	x	low	high	high	high	3	Very useful
Wizard of Oz Prototypes	x	low	low	medium	medium	1	Not very useful
Screen Snapshots		medium	low	high	low	1	Not very useful
Product Style Guides		high	medium	medium	medium	1	Useful
Grammars		medium	low	high	medium	1	Not very useful
UAN		very high	low	medium	low	2	Not very useful
TAG		very high	low	medium	low	1	Not very useful
Menu Trees		low	medium	high	high	1	Very useful
Interface State Transition Diagrams		low	high	high	medium	2	Useful
Harel State Diagrams		medium	low	high	low	1	Not very useful
Interface Content Modelling		medium	high	medium	medium	1	Useful
Navigation Map		medium	high	high	medium	1	Useful
Integrational Design		medium	medium	high	low	1	Not very useful
Parallel Design		low	medium	high	low	1	Not very useful
Impact Analysis		medium	medium	high	medium	3	Useful

Technique	UP	Training Needs	Applic-ability	Proxim-ity to SE	Improve-ment/Effort	Representa-tiveness	Total Rating
Organisation of Help by Use Cases		medium	medium	high	medium	1	Useful

3.2.7 Characterisation of Usability Evaluation-Related Techniques

With respect to the techniques related to usability evaluation, neither Evaluation by Remote Control nor Remote Testing by Videoconferencing have been included, because they are techniques that are very similar to Laboratory Testing in terms of the criteria examined. Likewise, Audio Recording and Video Recording have been grouped together for the same reason. Interviews are similar to Questionnaires and Surveys in terms of the surveyed characteristics and have, therefore, been lumped together with them.

The values for each evaluation-related technique are detailed in Table 13, as explained below:

- Heuristic Evaluation, being the least structured type of evaluation by experts, is the one that calls for most experience on the part of evaluators, which means that training needs are high. This also explains why proximity to SE has a low score, since unstructured techniques are not very commonly used in this discipline. On the other hand, this same feature makes the technique highly applicable, because it can be useful in all sorts of development projects. As regards usability improvement over effort, the score is high, because effort is not excessive and the improvements in usability are really quite sizeable.

- To correctly perform Inspections, skills in identifying inconsistencies and other usability errors are needed, which means that previous training is required (medium training needs). Inspections are applicable to all project types (high applicability), although Collaborative Inspections call for compliance by all stakeholders, which means that the applicability rating for this particular type of inspections is medium. Code inspections are a standard technique in SE and, therefore, the technique is close to SE as regards its mechanics but not in terms of the type of elements to be observed. This means that, as a whole, the proximity rating is medium. The usability improvement for Inspections is sizeable. In the case of Collaborative Inspections, effort is high because of how costly and slow inspection meetings are, which means that its improvement / effort ratio is medium. Effort is less for the other Inspections, making the score for the improvement / effort criterion high.

- Cognitive Walkthrough is a technique that calls for extensive knowledge of cognitive aspects and, therefore, its training needs are high. It focuses on the performance of an expert user in optimal conditions of use, which means that this is only applicable in situations where such users are relevant (medium applicability). Being focused on cognitive aspects, it is not especially close to SE, although walkthroughs are a common technique in SE. Therefore, the rating for the proximity criterion is medium. The usability improvement is sizeable, but the cost of application is high. Therefore, the usability improvement / effort ratio is medium.

- Like the above technique, a Pluralistic Walkthrough is costly (as it is a group technique). Nevertheless, it performs very well. Therefore, its improvement / effort ratio is medium. Likewise, being a walkthrough, it is close to SE, although it takes a different, participatory approach, which is not common in SE (medium proximity to SE). It is more generally applicable, but the stakeholders need to be open to this type of techniques, and, therefore, the rating for this criterion is medium. Finally, training needs are low, because it is designed to be applied by users, and it does not take too much knowledge to organise walkthrough sessions.

- The Thinking Aloud technique is not close to SE (low score), but training needs are not excessive (medium rating). It is highly applicable, as it is useful for a wide range of projects, and it makes quite a big contribution to improving usability, whereas application effort is low (high improvement / effort ratio).

- The training needs for Performance Measurement, Laboratory Usability Testing and Field Testing are medium, as their mechanics need to be learned, which calls for some, albeit not too much training. Their applicability is medium in the case of Performance Measurement and Laboratory Usability Testing, because they focus on performance measurement, which may not be especially relevant in some projects, whereas there needs to be access to end users in the prospective operating environment for Field Testing and equipment has to be installed, which is not always possible (low score). Their proximity to SE is medium, because while they are not completely foreign techniques, they are somewhat removed from what is routine practice in SE. Effort is fairly high for Performance Measurement and Field Testing and quite considerable for Laboratory Testing because of the cost of setting up this sort of facility, whereas the usability improvement is sizeable, albeit not excessive, for all three techniques. Therefore, their improvement / effort ratio is medium.

- The Post-Test Information technique calls for some training to be able understand the purpose and the formats or forms to be used (medium rating). It is applicable to all project types, as it is valid for any type of usability test, be it thinking aloud or performance measurement (high applicability). It is not foreign to SE, but it is not close either, because usability tests with users are not common in SE (medium proximity to SE). As regards application effort, this is low, and usability improvement is considerably high. Therefore, the improvement / effort ratio is high.

- The main drawback of Audio/Video Recording is that the analysis of the recordings is time consuming. Accordingly, while it can provide details that can be used to improve usability, the improvement / effort is low because of the time factor. It is not similar to any SE technique, which means that it scores low on proximity to SE. It is highly applicable, as, in principle, it can be applied whenever usability is tested. Finally, it calls for some training (medium rating), mainly regarding the analysis of recordings.

- Use Logging ties in with the use of software monitors in SE to measure efficiency, which makes its rating for proximity to SE high. Nevertheless, it is a relatively complex technique, and training needs are high. It is not applicable to all cases, insofar as it is only useful where performance needs to be measured (medium applicability). The effort at establishing software monitors is high, although it yields valuable information on possible trouble spots concerning usability, because its shows up details of the real use of the different functionalities. Therefore, the improvement / effort ratio is medium.

- Training needs for Observation are high, because evaluators need to be very clear about what to look for, and it is not applicable to all project types (medium rating for general applicability), because it is only practicable if the organisation provides evaluators with the possibility of observing users. It is not a technique that is close to SE, because the installed product is not usually evaluated like this (low score for proximity to SE). Finally, the improvement / effort ratio is low, because it takes a long time to observe enough usability problems.

- Questionnaires, Interviews and Surveys are highly applicable generally, because they are useful for all sorts of projects. Training needs are sizeable but not excessive (medium rating).

Some questionnaires are used in SE (beta testing), so there is some proximity to SE (medium rating). The effort at putting together, distributing and analysing questionnaires is considerable, however, they can highlight a lot of usability problems and, therefore, the improvement / effort ratio is medium.

- Commercial software companies use Focus Groups, which means that this technique is not foreign to SE, although it is not a common technique either (medium rating for proximity to SE). Managing a Focus Groups-like meeting is not a straightforward matter, as the aim is to follow an agenda, whilst giving the participants the impression that the discussion is free flowing. So, a lot of training and experience is needed to apply this technique (high training needs). The technique can be applied only when there is a pool of interested users who can be gathered together, which is not usually the case (low general applicability). Finally, although the usability improvement is not negligible, the application effort is considerable, making the improvement / effort ratio medium.

- There are many types of User Feedback, but customer/user complaints are generally something with which software engineers are acquainted, so this technique can be considered to score high for proximity to SE. The complexity of the technique may vary, but the training needs are low for its more common forms. User feedback is available in all sorts of projects, which means that its general applicability is high. Finally, it is useful for discovering usability problems, and its cost is low, which means that the improvement / effort ratio is high.

- Both Experimental Tests and Predictive Metrics are techniques that require extensive training for application (very high training needs). Their applicability is low, as only in systems where usability is critical will it be worthwhile running psychological-like tests or will the effort of using Predictive Metrics be profitable (low general applicability). Both techniques are very costly to apply and the usability improvements are not as valuable, which means that the improvement / effort ratio is low in both cases. As regards proximity to SE, Experimental Tests are not familiar (low rating), as they are only used in research environments in the field. Predictive Metrics are, however, close to SE's formal methods, which means that their score for proximity to SE is high.

- Cooperative Evaluation is described as a technique that can be used by designers and users without HCI knowledge, which means that its training needs are low. General applicability is low, because there needs to be a pool of users who can play the decision-making role required by the technique, which is not usual practice. Likewise, in SE, users do not generally decide what technique to use and what to evaluate, which makes the rating for proximity to SE low. The usability improvement varies depending on the group of participants. Therefore, a medium value is assigned to the improvement / effort ratio.

Table 13 – Rating of Evaluation-Related Techniques

Technique	UP	Training Needs	Applic-ability	Proxim-ity to SE	Improve-ment/Effort	Representa-tiveness	Total Rating
Heuristic Evaluation		high	high	low	high	6	Useful
Inspections		medium	high	medium	high	4	Very useful
Cooperative Inspections	x	medium	medium	medium	medium	1	Useful
Cognitive Walkthrough		high	medium	medium	medium	4	Useful
Pluralistic Walkthrough	x	low	medium	medium	medium	4	Useful

Technique	UP	Training Needs	Applic-ability	Proxim-ity to SE	Improve-ment/Effort	Representa-tiveness	Total Rating
Thinking Aloud	x	medium	high	low	high	5	Very useful
Performance Measurement	x	medium	medium	medium	medium	3	Useful
Post-Test Information	x	medium	high	medium	high	1	Useful
Laboratory Usability Testing	x	medium	medium	medium	medium	4	Useful
Field Testing	x	medium	low	medium	medium	2	Not very useful
Audio / Video Recording	x	medium	high	low	low	3	Not very useful
User Logging		high	medium	high	medium	5	Useful
Observation		high	medium	low	low	3	Not very useful
Questionnaires, Interviews & Surveys	x	medium	high	medium	medium	3	Useful
Focus Groups	x	high	low	medium	medium	2	Not very useful
User Feedback	x	low	high	high	high	3	Very useful
Experimental Tests	x	very high	low	low	low	2	Not very useful
Predictive Metrics		very high	low	high	low	2	Not very useful
Cooperative Evaluation	x	low	low	low	medium	1	Not very useful

3.3 Technique Selection

After having characterised each usability technique identified in the literature, we are going to use the total usefulness rating to select the techniques. The selected techniques will be part of the integration framework that makes up the solution proposed in this research.

The integration framework is not designed as a solution that has to be adopted as a whole, because it will include a range of techniques that could be useful in some cases. Additionally, one goal mentioned earlier is to keep the complexity of the framework within reason to enhance comprehensibility and applicability from the viewpoint of SE. Accordingly, we have decided to select both techniques whose total rating is “very useful” and techniques for which the total rating is “useful”, as the resulting set of techniques (thirty-seven) achieves the above goals.

The selected techniques are:

- Competitor Analysis
- Contextual Inquiry
- Affinity Diagramming
- Ethnographical Observation
- JEM
- User Profiles
- User Role Map
- Personas

- Essential Use Cases
- Task Organisation Model
- HTA
- Task Scenarios
- Card Sorting
- Usability Specifications
- Scenarios and storyboards
- Visual Brainstorming
- Conceptual Model
- Paper Prototypes
- Product Style Guide
- Menu Trees
- Transition Diagrams
- Interface Content Model
- Navigation Map
- Impact Analysis
- Use Case Organised Help
- Heuristic Evaluation
- Inspections
- Cooperative Inspections
- Cognitive Walkthrough
- Pluralistic Walkthrough
- Reading Aloud
- Performance Measurement
- Post-Test Information
- Laboratory Usability Test
- Use Logging
- Questionnaires, Interviews and Surveys
- User Feedback

3.4 Basic Usability Techniques References

For the purpose of contributing to the applicability of the integration framework that condenses the result of this research, we are going to provide a basic reference for each usability technique. This reference will serve as a basis for developers who want to apply a technique as part of their development activities. It will serve as a starting point for learning the basic points about the technique and its application.

For the survey in previous sections we considered general-purpose HCI literature sources that gave an overview of the user-centred process and the techniques to be applied in each activity. To give developers as much information as possible about each technique, however, more specific texts are needed sometimes focusing on a small set of techniques or activities. Therefore, no constraint will actually be placed on the type of text to be given as a reference, except for it being an available source dealing with the technique in more detail and sufficiently referenced in the field.

The basic reference for each technique is detailed in Table 14.

Table 14 – Basic References for each Usability Technique

Activity		Technique	Basic Reference
Analysis	Requirements Elicitation, Analysis and Negotiation	Competitor Analysis	[Nielsen, 93]
		Contextual Inquiry	[Beyer, 98]
		Affinity Diagramming	[Beyer, 98]
		Ethnographical Observation	[Preece, 94]
		JEM (*Joint Essential Modelling*)	[Constantine, 99]
		User Profiles	
		User Role Map	[Constantine, 99]
		Personas	[Cooper, 03]
		Essential Use Cases	[Constantine, 99]
		Task Scenarios	[Mayhew, 99]
		Card Sorting	[Robertson, 01]
		Scenarios *storyboards*	[Carroll, 97a]
		Visual Brainstorming	[Preece, 94]
		Conceptual Model	[Mayhew, 99]
		Paper Prototypes	[Constantine, 99]
	Requirements Specification	Usability Specifications	[Hix, 93]
	Requirements Validation	Heuristic Evaluation	[Nielsen, 93]
		Inspections	[Nielsen, 94]
		Cooperative Inspections	[Constantine, 99]
		Cognitive Walkthrough	[Carroll, 97b]
		Pluralistic Walkthrough	[Bias, 94]
Design	Interaction Design	Product Style Guide	[Mayhew, 99]
		Menu Trees	[Shneiderman, 98]
		Transition Diagrams	
		Interface Content Model	[Constantine, 99]
		Navigation Map	
		Impact Analysis	[Hix, 93]
		Organisation of Help by Use Cases	[Constantine, 99]
V&V	Usability Evaluation	Heuristic Evaluation	[Nielsen, 93]
		Inspections	[Nielsen, 94]
		Cooperative Inspections	[Constantine, 99]
		Cognitive Walkthrough	[Carroll, 97b]
		Pluralistic Walkthrough	[Bias, 94]
		Thinking Aloud	[Nielsen, 93]
		Performance Measurement	[Rubin, 94]
		Post-Test Information	[Mayhew, 99]
		Laboratory Usability Test	[Rubin, 94]
		Use Logging	[Shneiderman, 98]
		Questionnaires, Interviews and Surveys	[Mayhew, 99]
		User Feedback	[Shneiderman, 98]

4. Assigning Usability Activities and Techniques to Development Activities

In previous section, we looked at the usability activities discussed in the HCI literature as a framework into which to embed the individual usability techniques, which we also surveyed. For their integration into a general-purpose development process, we need to use the terminology of SE activities, as this is what developers are familiar with. Accordingly, developers will be able to see how the techniques applied in the usability activities fit in with general-purpose development activities.

With this aim in mind, we are going to assign in this chapter each usability activity to the respective activity in a general-purpose development process. Then, based chiefly on this assignation, we are going to examine what activity each particular usability technique selected to form part of the usability integration framework that we are researching fits into.

4.1 Relationship between HCI Activities and Development Activities

As the types of activities carried out in HCI and SE are not independent, we need to identify how the HCI activities fit in with the activities that are performed as part of any traditional development process.

While some usability activities are integrated into existing activities, such as Usability Specifications that are incorporated into the Requirements Specification, additional activities that are not usually carried out in a non-user-centred development process, like, for example, Interaction Design, need to be added. Our basis, whenever possible, for the terminological definition of the activities in a traditional development process, is the SWEBOK (Software Engineering Body of Knowledge) [SWEBOK, 01]. We will use HCI terminology for the activities that are not common to SE development processes.

To describe the relationship between HCI activities and general-purpose development activities, they will be classed by activity types: Analysis, Design and Evaluation.

4.1.1 Analysis-Related HCI Activities

The two HCI activities related to analysis are Context of Use Specification and Usability Specifications. Context of Use Specification is divided into User Analysis and Task Analysis. From the description of these activities, we find that their goals are close to the goals of requirements engineering as it is usually performed in SE development processes. Indeed, this subdiscipline of SE (requirements engineering) has been performing key activities for final system usability, such as user and Task Analysis, observation of users in their usual environment, identification of user needs and conception of the software product that can best meet their needs. HCI can input its user-centred perspective to assure that these activities are performed in software development with a positive outcome in terms of final product usability, stressing this perspective throughout development.

As described in the SWEBOK, there are four activities in the requirements engineering process of the spiral requirements engineering process model: elicitation, analysis and negotiation, specification and validation. We believe that the first two, elicitation, and analysis and negotiation, are very difficult to separate clearly, as the information gathered in the elicitation activities is set out, organized and modelled in the requirements analysis activities. In a participatory model, elicitation and modelling take place at the same moment in time, as users and developers model together, setting out their domain knowledge in the process. For our purposes, it is pointless to delimit elicitation and analysis/negotiation, because more than one HCI technique merges the two activity types (for example, the JEM and Competitor Analysis techniques). On the other hand, activities related to requirements specification and validation do have a more clearly defined border, and these activities can be dealt with separately.

The Context of Use Specification activity includes both techniques for eliciting requirements and techniques that can set out observations in models that can be used as a reference in system interaction design. The User Analysis and Task Analysis activities fall within this category of requirements-related activities. Therefore, the Context of Use Specification activity matches the Requirements Elicitation and Analysis activity of a traditional development process. The specific User Analysis and Task Analysis activities should be considered within Elicitation, Analysis and Negotiation, as they are especially important for the usability of the software system under development.

With regard to Usability Specifications, as it is their job to establish usability goals to be achieved by the system under development, they match the SWEBOK's definition of the type of information that appears in the software requirements specification (SRS). According to the SWEBOK, this document "defines the high-level system requirements from the domain viewpoint" [SWEBOK, 01]. Usability specifications are always written from the viewpoint of the use to which the system is to be put and, therefore, belong to the problem domain. The HCI Usability Specifications activity matches the Requirements Specification activity.

4.1.2 Design-Related HCI Activities

Examining the activities described by the HCI literature as related to design, we found some that SE does not consider to be part of design, as SE uses a more restrictive definition of this term.

The general definition of design used in SE and expressly cited in the SWEBOK is the one listed in the IEEE Standard Glossary of Software Engineering Terminology [IEEE Glossary, 90]. According to this definition, design is "the process of defining the architecture, components, interfaces and other features of a system or component and the result of this process" [IEEE Glossary, 90]. The high level definition of what the system is, which we might term product conceptualisation, does not fit in with this definition of design.

Additionally, the Prototyping technique (routinely used in SE) is not confined to the design area, but is mentioned explicitly in the requirements area. In this regard, the Prototyping technique is mentioned as being applicable both for eliciting requirements and for their validation. Indeed, prototyping is given more attention in the SWEBOK chapter on requirements than in the design chapter (where it is only mentioned as a possible means of evaluating a design). As regards Prototyping as an activity in the field of HCI, it deals chiefly with the use of low fidelity prototypes, which are explicitly mentioned in the SWEBOK as possible tools for requirements elicitation. For these reasons, the HCI Prototyping activity has been related to the activity of Requirements Elicitation and Analysis of any SE process and we believe it should be considered within this activity to assure that the process is user centred.

The Develop Product Concept belongs to what the SWEBOK terms "invention design". The chapter on software design reads "I-design (invention design, which is usually carried out by systems analysts for the purpose of conceptualising and specifying a system to satisfy the discovered needs and requirements) will not be dealt with [in this chapter], because this subject should be considered as part of the requirements analysis and specification activity" [SWEBOK, 01]. The Develop Product Concept is a conceptualisation activity that we consider to be essential for the success of requirements-related tasks and also fits in well with the description of invention design. Therefore, we consider Develop Product Concept to be related to the Requirements Elicitation and Analysis activity.

Interaction Design, on the other hand, is not easily accommodated among the issues dealt with in the SWEBOK. This is because, as compared with the above two activities, the tasks related to interaction design are less dependent on the other development activities. This is a design activity that is quite separate from the other system design activities. The SWEBOK considers UI design not as part of SE, but has a related discipline. It does state, however, that UI design is related to the specification of the external appearance of the system and should be considered as part of requirements specification. The Interaction Design activity is not confined only to the design of the visual part of the UI, as it also specifies the behaviour of the interaction environments. Therefore, and because the tasks performed in this activity fit in with the definition of design given in the glossary of SE terminology listed above, we have considered that the new Interaction Design activity fits in with the design activities of any development process.

4.1.3 Evaluation-Related HCI Activities

Usability Evaluation is an activity that is performed separately from the other Validation and Verification (V&V) activities in software development. It is highly complex because of the variety of tasks that may be involved in its performance, as well as the many techniques available for the purpose. Being separate from the other V&V activities, we have opted to create a new activity, called Usability Evaluation, into which to group the activities related to usability evaluation. Within this activity, we make a distinction between three major groups of usability evaluation activities: Evaluation by Experts, Usability Testing and Follow-Up Studies of Installed Systems.

Within the Usability Evaluation activities, special mention should be made of the Evaluation by Expert group of activities. Whereas Usability Testing and Installed Systems Follow-Up Studies are chiefly based on an operational system, this does not necessarily apply to Evaluation by Experts. Accordingly, these activities can be useful for validating products on which work is being done in the requirements activities. Therefore, the Evaluation by Experts is related to the Requirements Validation activity, as well as to V&V activities.

4.1.4 Result

The relationships between HCI activities and general-purpose development activities, as described in the sections above, are shown in Figure 3. The user-centred process activities are listed on the left-hand side and the general-purpose development process activities on the right-hand side. An arrow links each usability activity to the respective development activity. The right-hand side only shows development activities affected by the application of usability techniques. Activities that are not common to SE processes and have been added because they are quite separate from the existing activities are highlighted in italics.

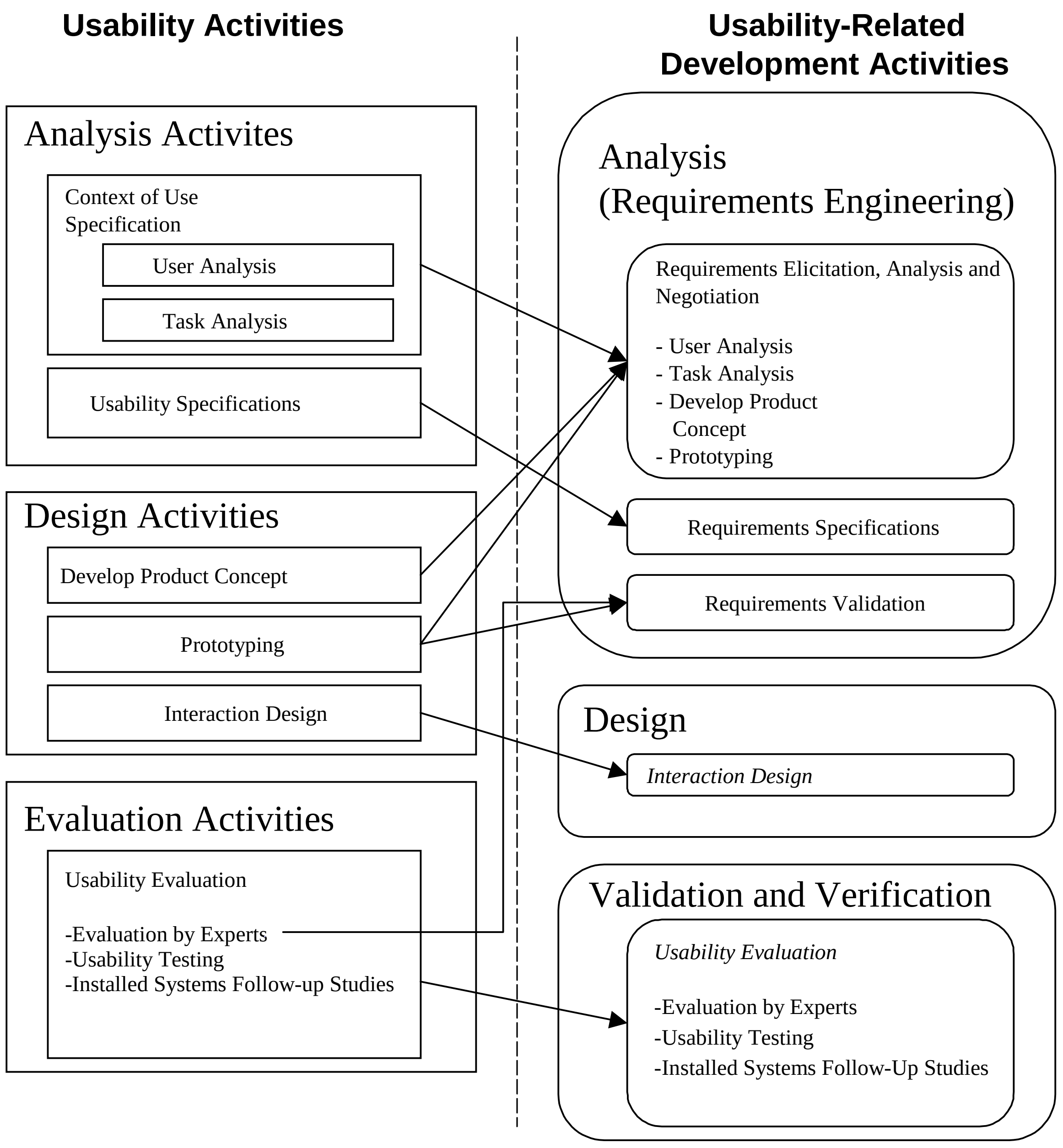

Figure 3 – Relationship between HCI Activities and Development Activities Affected by Usability

The relationships described are a sort of general guideline for linking the HCI process view to the one that is commonly used in SE. However, the development activities in which the application of particular techniques can be most beneficial in terms of improving the final usability product need to be considered one by one. Therefore, this allocation of activities will serve as a basis for assigning individual techniques to development activities. This is neither an inflexible nor a permanent assignation, it is merely a means of easing the process of allocation.

4.2 Assigning Usability Techniques to Development Activities

Based on the description of activities in section 2, we are now going to present the allocation of usability techniques to development activities. This is quite a difficult task, because although usability techniques are usually described in the HCI field as part of an activity, this is not always the case.

Even when they are assigned to an activity, we need to check each individual technique to find out whether there is any trait that affects its allocation to a development activity. Therefore, we are going to consider the allocation of techniques to activities one by one, grouped according to the development activities that appear on the right-hand side of Figure 3, and highlighting for each technique what it inputs to this activity.

4.2.1 Techniques Assigned to Analysis Activities

The three major activities that we have considered for linking the activities of a HCI process to the usability-related development activities are Requirements Elicitation, Analysis and Negotiation, Requirements Specification and Requirements Validation.

4.2.1.1 Requirements Elicitation, Analysis and Negotiation

Requirements elicitation, analysis and negotiation is recognised in SE as one of the key tasks for the success of the entire development project. In elicitation efforts, usability techniques can provide additional sources of information and resources for extracting knowledge to what are traditionally used in requirements engineering. Also for analysing the information, the task and user models usually employed in the field of HCI can focus the tasks more effectively on the actual users of the system under development and their real needs. As regards requirements negotiation, participatory techniques are especially good for adding a participatory component to the requirements negotiation techniques.

Most techniques proposed by the HCI field for this purpose are embedded in a specific activity in a user-centred development process, and we are going to deal with these techniques grouped according to the activity in the user-centred process in which they would be applied: User Analysis, Task Analysis, Develop Product Concept, Prototyping and other techniques that do not directly fit in with any of the above.

4.2.1.1.1 User Analysis

User Analysis is a crucial activity in user-centred processes, and the inclusion of this type of activity in any development project is the pillar upon which the other usability activities applied during development can be built. The selected techniques that are related to this type of analysis are as follows:

- **User Profiles**: The use of User Profiles converts gathering information about the prospective system users into a systematic process. The definition of each profile sets out what different types of characteristics could be relevant in a user survey. Accordingly, elicitation efforts are focused on what is likely to be most useful for the design of a system with a satisfactory usability level. By structuring the information on users, the use of this technique can help developers not acquainted with User Analysis as it is performed in HCI, because it details what relevant information about users should be gathered during requirements elicitation and analysis.

- **User Role Map**: When there are a number of potential system user types, it may be helpful to use a User Role Map to structure the relationships between users and get an overview of system users. These models are very useful in requirements negotiation tasks for checking with all stakeholders whether the right goals are being addressed as far as the prospective system users are concerned.

- **Personas**: This technique helps to synthesise all the data available about the prospective system users as archetypal users. These data can be used to reach agreement within the development team and to focus design discussions. It also helps to determine what the product should do, related to the needs to be satisfied, which means that it plays a role throughout the whole requirements analysis and negotiation process. By providing a common language for referring to specific users of the system (specific as opposed to the generic and often mistaken term "system users"), it helps to reach consensus within the development team. In particular, it eludes a common problem suffered by a number of developers who tend to equate the capabilities and wishes of future system users with their own. This attitude leads to the production of software systems that only users with a marked technological profile can use.

4.2.1.1.2 Task Analysis

A software engineer is acquainted with the concept of system functions or processes, and one of the activities that he or she is used to performing is functional decomposition. Task analysis as performed in user-centred processes pursues a similar aim, albeit with a different focus, targeting the ultimate goals of the user when using the system. The techniques included in this section can fulfil the function of focusing a functional system analysis on the user, on the tasks that he or she wants to do and for which he or she uses the software system. Thus, the functions that the system should offer will be analysed from a user-centred perspective, assuring that the developed system meets the needs of and matches the procedures followed by the user. The techniques for this purpose are as follows:

- **Essential Use Cases**: This technique supplements the use of the use cases technique in SE, which is very commonly used as part of object-oriented development analysis activities. From the user-centred viewpoint, the use cases technique is usually applied incorrectly. This is because the production of use cases includes a series of decisions that affect usability, which are taken without following any criterion concerning user goals. Essential use cases are a way of focusing user and system interaction decisions on what the user intends to do in each step he or she takes and how the system supports user decision making in each step through the responses it gives.

- **Task Modelling**: This technique is useful for structuring the information observed about how the user organises the tasks that he or she usually performs as part of his or her job. Therefore, the use of this technique can supplement the requirements elicitation and analysis efforts when the system is designed to support the user in the performance of his or her routine work.

- **HTA**: This technique pursues the same aim as the above and has the additional feature of detailing a process by means of which the user tasks are decomposed hierarchically. Therefore, it is equally useful for requirements elicitation and analysis tasks.

- **Task Scenarios**: This technique fulfils a similar function in Task Analysis to the Personas technique in User Analysis. It gathers the tasks that are most representative of each user type and thus helps the development team to reach agreement on what are the principal tasks that the system should support. It plays an important role in requirements analysis and negotiation.

The variant of Card Sorting, known as **Task Sorting**, adds to functional decomposition, a possible requirements analysis technique, the user's view of what tasks he or she performs. It has the user participate in the requirements analysis tasks that involve ordering the data collected from elicitation. Accordingly, it adds a participatory component to task decomposition that considerably eases the tasks related to requirements negotiation with all the stakeholders. The Card Sorting technique has been classed as not fitting into any particular activity in a user-centred process, as detailed below.

4.2.1.1.3 Develop Product Concept

Of the activities related to requirements elicitation and analysis, the establishment of an overview of the product under development plays an important role and is all the more important, the more innovative the project is (as opposed to projects for mechanizing manual tasks). Likewise, it is essential for requirements negotiation that all the stakeholders should take one and the same view of the product under development, that is, have the same product concept. The following techniques contribute to these goals:

- **Scenarios and storyboards**: When you are trying to convey to all the stakeholders what type of system is to be built, scenarios and storyboards help to focus the narration on what the system will be like on definite users that perform specific tasks. This technique can be combined with the Personas technique to achieve a better definition of the type of system to be built, who it targets and what needs it aims to meet. The details that are added to an overview serve to focus requirements negotiation discussions, making them less abstract so as not to rule out the participation of customers and users.

- **Visual Brainstorming**: Visual Brainstorming is a group technique for creating ideas that, owing to its traits, is especially suited for trying to define the features of an innovative software product. The visual variant of the technique, which involves sketching the UI to drive the discussions, allows users, who, if the dynamics were to focus on technical digressions, would find it more difficult to participate, to contribute to the production of ideas. Insofar as this technique aims to output the idea of the product under development, it can be considered as part of the requirements elicitation efforts.

- **Conceptual Model**: The decisions taken about the product concept on which the development will be based can be set out in a model for later use. This type of modelling would be part of the requirements analysis and negotiation activities, as it aims to shape the product concept to satisfy all stakeholders.

4.2.1.1.4 Prototyping

In the section on requirements elicitation, the SWEBOK [SWEBOK, 01] lists prototypes as a useful tool for clarifying requirements. One major risk to system development is developers pursuing a mistaken goal and not developing the required system. Such errors often stem from misunderstandings between the team writing the requirements and the other stakeholders. To minimise the risk of this problem occurring, the following technique can be useful for working closely with customers and users:

- **Paper Prototypes**: Hand-drawn sketches or computer-aided drawings produced using a graphics program can be used as part of the early requirements meetings. These prototypes can convey to the customer what idea the team has of the system to be built, and the team can check that they are working in the right direction. As compared to prototypes that run on a computer, they have the benefit that the customer does not think that the system has already been built, because their rough appearance conveys the idea that there is still a lot of work to be done before the final product is built. Even though they are only drawings on paper, these prototypes can also be used to show the interactive dimension of the system, as a member of the team can explain to the customer and/or user how the interaction has been conceived to work, switching sheets to show what will be displayed depending on the actions taken by the user.

4.2.1.1.5 Other Techniques

Of the techniques that can be applied for requirements elicitation, analysis and negotiation, there are some that do not exactly match any of the HCI activities discussed above, although they are closely related to one or more of them. The input of these requirements elicitation, analysis and negotiation techniques is detailed below:

- **Competitor Analysis** As it is the internal part of the software systems that has received most attention traditionally in SE, it is not surprising that the comparison with other software products has not been standard practice in most software development projects. Unlike the internal part of the system, the UI and interaction design generally can actually be observed in a competitor product. The application of this technique is a source of inspiration for the development team on what interaction mechanisms the system development should include and on problems to be eluded. If a software project addresses the problem of mechanization of an administrative task that has been performed manually to date, the advantages of competitor analysis are not very sizeable, but this technique can be more beneficial when the product to be developed is more innovative. Therefore, this technique is related to both the Task Analysis and Develop Product Concept and Prototyping activities.

- **Contextual Inquiry**: This technique is closely related to the requirements elicitation tasks concerning user and routine Task Analysis. It is suited for use in development projects where there is a possibility of directly dealing with representative users willing to cooperate on the common task of figuring out how they perform their routine tasks. Accordingly, it achieves a deeper understanding of user needs than traditional interviews, because users are often not aware of details of why they do what they do.

- **Affinity Diagramming**: This technique can organise information, extracted either from requirements elicitation or brainstorming sessions, in the development of the product concept. The cooperative and integrational component can bring all participants round to a common view. In particular, it provides for end users to take part in the requirements analysis sessions.

- **Ethnographical Observation**: Like Contextual Inquiry mentioned earlier, this technique is closely related to the requirements elicitation tasks concerning user and routine Task Analysis. It is a technique that supplements a traditional interview, because it yields first-hand data on the behaviour of the user rather than as a viewpoint possibly distorted by several factors (shyness, desire to make a good impression, etc.). Therefore, it is especially useful when the aim is to match the software system under development to the user organisation’s culture.

- **JEM**: This technique describes a cooperative, driven and structured process for getting users and developers involved together in modelling activities carried out during requirements analysis. It is an extension of the JAD technique (Joint Application Design), known in SE. It introduces the user-focused approach into this technique. This technique is especially useful when the aim is to give the whole requirements process a participatory flavour, whilst assuring that it is organised and, therefore, well controlled.

- **Card Sorting**: This technique gives an understanding of the representation of information used by users, and can, therefore, serve as a tool for eliciting and analysing requirements. It is useful when there is already information about the domain in which the users are working, but this information needs to be organised according to the user’s mental structure. This is a relatively simple participatory technique that yields better results for this particular purpose than interviews. It can also contribute to prototyping, as the menu structure can be established on the basis of the information gathered by applying this technique.

4.2.1.2 Requirements Specification

Requirements Specification is related to the production of a document stating what requirements the system must meet and is particularly concerned with the structure, quality and verifiability of this document. As far as usability is concerned, the techniques detailed in the above point can contribute to the process of extracting and putting together information on requirements, but the structure of the document is influenced exclusively by usability specifications. The description of how this technique contributes to the specification of requirements is detailed below:

- **Usability Specifications**: Usability specifications are usability goals that are established for the software product under development. To be useful as a guide for finding out whether the project is advancing in the originally stated direction, they need to be able to be verified in an iterative development process. The inclusion of these specifications in the software requirements specification document introduces usability as another aspect that can be established quantitatively and in advance. The usability specifications also serve as inspiration for developers throughout the entire development process, and whenever a choice needs to be made between design alternatives.

4.2.1.3 Requirements Validation

According to the SWEBOK [SWEBOK, 01], there are four important questions in requirements validation: requirements inspections, prototyping, model validation and acceptance testing. Usability techniques can be useful for validating requirements that affect one or more of these questions, except as regards acceptance testing. The only usability technique involved in these tests is Usability Specifications, which should to be put together with a view to acceptance testing and, therefore, have to be able to be verified, as indicated above. The usability techniques applicable for validating requirements are as follows:

- **Heuristic Evaluation**: Provision can be made for heuristic usability evaluation to be run alongside the traditional requirements inspections for the purpose of assessing any possible usability deficiencies in the products built in the requirements activities (use case specification, prototypes, etc.).

- **Inspections**: Although also related to requirements inspection activities, they focus chiefly on prototype validation. They are similar to heuristic evaluation, but focus on a particular issue (standard or design guideline compliance, interaction design consistency). In view of how important prototyping is as part of requirements activities for achieving highly usable software products, these techniques are especially relevant as part of requirements validation.

    - **Collaborative Inspections**: This variant contributes to the integration of users in the team that is to run the inspection in a structured manner. The SWEBOK [SWEBOK, 01] states that there should be at least one customer representative on the inspection team. In a user-centred process, user representatives should be members of this team, and this technique provides a structured way of undertaking the inspection process.

- **Cognitive Walkthrough**: This technique can validate a prototype from the viewpoint of the cognitive activities that the user is obliged to perform. Therefore, as regards requirements validation, it is related both to prototyping and model validation, because it evaluates the quality of a prototype understood as a model of user-system interaction.

- **Pluralistic Walkthrough**: This technique serves the purpose of validation insofar as it organises requirements inspection sessions (in the prototyping part) in which all the stakeholders participate. Not only does it aim to assess the validity of the developed prototype, but it can also play an important role in requirements negotiation, as it contributes to bringing all the development stakeholders around to one mind. It is especially well suited to making the user feel that the issues he or she finds most important are properly looked after and that the development team understands his or her view of the problem.

4.2.2 Techniques Assigned to Design Activities

The only major activity added to the design activities that are usually performed in any development process is Interaction Design, but there are other HCI techniques related to design activities that do not directly fit in with any activity.

4.2.2.1 Interaction Design

Being an extra activity apart from the activities usually performed in any development process, it includes just the HCI techniques that provide additional tools for working on interaction design tasks and for representing the results generated by performing these tasks. These techniques are as follows:

- **Product Style Guide**: In any development process, models and other design documentation serve to assure that the development team takes a common view of the design decisions and as a basis for future modifications and/or extensions. The style guide serves the same purposes, but refers to all aspects related to interaction design. It is especially useful when teams are to be multidisciplinary and some developers have little previous experience in HCI issues, as the style guide sets out the rules to be followed as regards interaction design.

- **Menu Trees**: Generally, we find that the structure of UI elements is not modelled in a lot of development projects, because developers do not have techniques to do this sort of modelling. Menu Trees are a technique for modelling a particular type of UI that is based on a menu structure. The technique is relatively easy to use. It provides a visual representation of the menu structure and can be used for decision making on the design of this structure.

- **Interface State Transition Diagrams**: Like Menu Trees, this technique provides a way of modelling a particular type of UI, based either on different modes or on a system of modal windows (users can only interact with the window active at the time). This technique can also be useful for putting together user manuals, as it can convey the logic of system interaction to the user.

- **Interface Content Modelling**: This technique can be used to perform graphics-based interaction design tasks and encourages the discussion of alternatives. It is well suited for window-based UIs with different interaction spaces. Like the other two techniques above, it fills the gap that there is in terms of UI modelling outside user-centred development processes.

- **Navigation Mapping**: This technique can represent the possibilities for navigation between different interaction contexts, which means that it is useful for the same type of systems as above. The consistency between the different elements in the UI is one of the key features concerning usability, and this technique offers a graphical representation that provides an overview of navigation between the different interaction spaces showing up any deviations from the general rule in terms of consistency.

4.2.2.2 Other Design-Related Techniques

Of the techniques selected for the purpose of this research, two are related to design activities. Their application is, however, not confined to the Interaction Design activity. They are:

- **Impact Analysis**: This technique offers a development team concerned about usability a structured mechanism for making decisions about the prioritisation of observed usability problems in terms of the order in which the identified problems should be addressed. The design modifications to be assessed can have an impact on the interaction design only or on internal system functionality, and, therefore, this technique is not confined to just the Interaction Design activity.

- **Organisation of Help by Use Cases**: When developers plan the design of a help subsystem, they often do not have any structured technique that they can use for this purpose. This technique can fill this gap, by linking the help subsystem structure to use cases, thus serving as guidance for the development of this subsystem. When they are built without any guidance (such as, for example, is suggested by this technique), the help facilities run the risk of becoming an information repository that users end up not using, because it is hard to find the information required at any time.

4.2.3 Techniques Assigned to V&V Activities

As indicated earlier, the usability evaluation activities are carried out separately from the other V&V activities. Usability evaluation activities are the topic that has traditionally attracted most attention in HCI, and is, therefore, the most mature area. Therefore, as opposed to other activities where we had difficulties in establishing a taxonomy of techniques, usability evaluation does have a widely accepted structure in the field of HCI. As the new Usability Evaluation activity within the V&V activities is separate from other activities, we are not going to justify the assignation of techniques to each activity type. We will, however, highlight their contribution to development taken as a whole.

As detailed earlier, the three major groups of usability evaluation are Evaluation by Experts, Usability Testing and Installed Systems Follow-Up Studies. We are going to detail the Usability Evaluation techniques classed according these three groups.

4.2.3.1 Evaluation by Experts

The selected techniques that belong to the Evaluation by Experts group were dealt with before as regards their possible contribution to requirements validation. In this section we are going to detail what they can contribute to V&V activities outside requirements validation.

- **Heuristic Evaluation**: This is the least structured technique in this section. Unlike the other V&V techniques, this technique focuses on the usability of the software product under evaluation.

- **Inspections**: Inspections are similar to the software inspections usually performed in software development, except that they switch the focus of the inspection to a UI or a prototype UI. Nevertheless, there are inspections, known as consistency inspections that are specific to HCI, because they focus on an especially important feature with a view to software product usability.

    - **Collaborative Inspections**: This variation on inspection provides a participatory component that can integrate the user into the development team.

- **Cognitive Walkthrough**: Like inspections, walkthroughs are a known technique for searching for software defects. This type of walkthroughs focuses on an especially important concern from the viewpoint of usability: the cognitive load to which the user is exposed during software application use.

- **Pluralistic Walkthrough**: as indicated before, the biggest contribution of this participatory technique is that it allows the user to appreciate that the concerns that are most important to him or her are properly looked after and that the development team understands his or her view of the problem.

4.2.3.2 Usability Testing

Usability tests are the most common type of technique used in user-centred development processes, as this approach is based on the premise that it is impossible to assure how usable a prototype or software product is without testing it beforehand with representative users performing the tasks that the system supports. What usability tests contribute to the development process is a way of evaluating usability objectively, as they attempt to reproduce system use as faithfully as possible by using representative users performing their most routine tasks. It offers more objective decision-making criteria on system quality in terms of usability than can be gathered from a system (or system prototype) demonstration for the customer or even for users. A demonstration in which a member of the development team uses the system shows up only the strengths of the system and also follows the development team’s way of thinking. The customer and user may approve a system that has been demonstrated in this manner only to discover, when it is installed at their organisation, that there are important usability problems, because it does not fit in with the peculiarities of everyday work or with how the users conceive their jobs.

Although usability testing is founded on a very basic central premise (having representative users test the system by performing the tasks for which the system was conceived), there are variants on the basic technique, whose contribution to development is detailed below:

- **Thinking Aloud**: What this variant contributes to usability testing is the expression of the user’s internal reasoning, the knowledge of which can be a key factor for analysing the identified usability problems.

- **Performance Measurement**: With a view to making the result of usability testing as objective as possible, this variant measures the efficiency of use of each participant using the system to yield quantitative values reflecting the usability of the system as regards this particular attribute. Therefore, this variant is applied when there is a prototype that is complete enough to allow the participant to use it and when efficiency of use is one of the relevant usability attributes. Its use is incompatible with the above variant (Thinking Aloud), as the participant’s effort at verbalizing what he or she is doing slows down system use, invalidating any performance measurements taken.

- **Post-Test Information**: To separate experience in system use and how much attention the user needs to pay to system use from the verbalisation of his or her actions, this variant enables the user to analyse his or her usability testing session after he or she has finished. This variant can be used to combine the benefits of Performance Measurement and the Thinking Aloud variant.

- **Laboratory Usability Testing**: This technique stands on the performance of usability tests in facilities especially conceived for this purpose. The reason for setting up such facilities lies in the complexity of some of the equipment that is employed in usability testing to record audio and/or video and for observation of the development process by all the stakeholders (developers, customer, project sponsor, etc.). A usability laboratory provides a controlled and consistent environment in which to run the usability tests. Therefore, results for different users or different systems can be more easily compared. These facilities are costly, and their construction and use is only recommended for medium- to large-sized organisations whose usability budget can accommodate this cost.

The **Use Logging** technique can provide additional information on the actions taken by the user participating in the test during usability testing. As it plays a bigger role in the group of techniques that belong to the Installed Systems Follow-Up Studies, it has been placed in that group rather than being described under this point.

4.2.3.3 Installed Systems Follow-Up Studies

Tasks related to requirements and to situation evaluation and problem solving for the maintenance and extension of systems already deployed at the user organisation differ, as there is much more information than when developing a system from scratch. As regards usability, we have selected three techniques that can evaluate the usability of installed systems, whose contribution to maintenance and/or extension is described below:

- **Use Logging**: The use of this technique is closely related to the Task Analysis conducted as part of the requirements activities. As the use to which the system is put has been modelled as it is expected to be, use logging mechanisms allow the development team to observe the deviations between the expected and actual use. This type of information generates objective and detailed information on possible usability problems, functionalities that are used less than expected or repeated interaction patterns that lead the user to make mistakes.

- **Questionnaires, Interviews and Surveys**: These techniques provide subjective information concerning user satisfaction. Apart from the value of this usability attribute, flexible interviews also provide information about the system areas with the biggest usability deficiencies or work to which a higher priority should be given owing to its importance for the user.

- **User Feedback**: In all the techniques discussed above, the user plays a role in which he or she is driven as he or she does not decide to apply the technique. In this set of techniques, however, it is the user who, owing to some problem or deficiency in the software system that he or she is using, takes the initiative of going to the software developer organization to report the matter. When an on-line customer or help service is available, the information gathered about the user queries is very useful for identifying and prioritising the usability problems to be dealt with quickest.

4.3 Summary

As a summary of the assignation of usability techniques to development activities, Table 15 lists these techniques grouped by the activity with which they match up. Note that the Evaluation by Experts technique appears under the activity of the same name within the V&V activities and as a Requirements Validation activity within Requirements Engineering

Table 15 – Usability Techniques Grouped according to the Development Activity in which their Application is most Useful

Activity	Technique

Requirements Engineering	Requirements Elicitation, Analysis and Negotiation	User Analysis	User Profiles
			User Role Map
			Personas
		Task Analysis	Essential Use Cases
			Task Organisation Model
			HTA
			Task Scenarios
		Develop Product Concept	Scenarios and Storyboards
			Visual Brainstorming
			Conceptual Model
		Prototyping	Paper Prototyping
			Competitor Analysis
			Contextual Inquiry
			Affinity Diagramming
			Ethnographical Observation
			JEM
			Card Sorting
	Requirements Specification		Usability Specifications
	Requirements Validation		Heuristic Evaluation
			Inspections
			Collaborative Inspections
			Cognitive Walkthrough
			Pluralistic Walkthrough
Design	Interaction Design		Product Style Guide
			Menu Trees
			Interface State Transition Diagrams
			Interface Content Model
			Navigation Map
			Impact Analysis
			Organisation of Help by Use Cases
V&V	Evaluation by Experts		Heuristic Evaluation
			Inspections
			Collaborative Inspections
			Cognitive Walkthrough
			Pluralistic Walkthrough
	Usability Testing		Thinking Aloud
			Performance Measurement
			Post-Test Information
			Laboratory Usability Testing
	Installed Systems Follow-Up Studies		Use Logging
			Questionnaires, Interviews and Surveys

		User Feedback

5. APPLICATION TIMES IN AN ITERATIVE DEVELOPMENT PROCESS

The approach adopted for integrating usability into the software process has been to provide an integration framework as a basis for decision making about what techniques to add to what activities in the development process. To be able to adopt the integration framework produced in this research, we first need to establish what minimum requirements the development process in place at the organisation should meet. As we will see in the following, the only such requirements are that the process be based on an iterative development approach.

An iterative process is carried out in cycles, but not all the cycles are the same. For example, more effort goes into work related to delimiting the problem and to establishing the baselines for the solution than to other development activities in the early cycles. Apart from assigning techniques to activities, as we did in the last chapter, we believe that developers need to be given guidance about when the application of each usability technique is likely to be most useful. This chapter details this guidance about when to apply usability techniques in an iterative development process.

First, we will detail our survey of the minimum requirements to be met by the original development process. We will then outline the types of iterative process cycles, upon the basis of which the guidance will be developed. The following section details the reasons why each technique was classed as important at each time, organised according to the three cycle types. Finally, we summarise the guidance on when to apply techniques in a table.

5.1 Development Process Requirements

As discussed in D.5.2, rather than selecting a development process model upon which to base our integration framework, we opted not to link the solution to any particular process model so as not to limit the applicability of the framework to organisations that follow the selected process model. Therefore, the goal is to provide a solution that is flexible enough to account for a whole range of development processes in place at software development organisations.

To assure that the process generated after integration actually meets the goal of achieving a satisfactory usability level for the developed software product, we need to examine what features the original development process should have. We are looking to get an integrated process whose primary quality is user-centredness. Therefore, we are going to examine what characteristics of a user-centred process can be achieved by adding activities and techniques (and are, therefore, provided by the integration framework) and which are intrinsic to the process (and, therefore, are requirements for the original process). Having identified these process-related characteristics, they can be used by any organisation interested in integrating usability to decide whether usability can be incorporated into the development process in place or whether it should consider migrating to another type of process if it is really concerned about achieving usability goals.

5.1.1 Characteristics of a User-Centred Process

Apart from the terminological difference from SE (some authors speak of user-centred design rather than user-centred process), the HCI field offers different views of what a user-centred process is. Nevertheless, there is a common core, which is what we present in the survey detailed in this section.

In reference to user-centred design, Preece et al. [Preece, 94] indicate that it should have the following characteristics:

a) be user-centred and get users as involved as possible so that they can influence design,

b) integrate knowledge and experience from all the disciplines that contribute to HCI design,

c) and be highly iterative so that tests can be run to check that the design really meets user requirements.

ISO 13407 on Human Centred Development Processes for Interactive Systems [ISO13407, 99] specifies that the inclusion of a human-centred approach is characterised by the following:

a) the active involvement of users and a clear understanding of the requirements of users and their tasks,
b) a satisfactory allocation of functions between users and technology
c) the iteration of design solutions, and
d) multidisciplinary design.

In a definition that takes a different viewpoint, Constantine and Lockwood [Constantine, 99] define the elements of a use-centred approach as follows:

a) Pragmatic design guidelines

b) Model-driven design process

c) Organised development activities

d) Iterative improvement

e) Quality measures

Shneiderman [Shneiderman, 98] upholds that a process concerned with usability should not be hierarchical, meaning that, strictly speaking, it should be neither top-down nor bottom-up, whereas it should be radically transformational, involving the production of interim solutions that can end up by not playing any role in the final design. This definition points to a strongly iterative approach.

5.1.2 Process Requirements

From all the characteristics outlined above, we can extract four key elements that constitute the essence of a user-centred process: user involvement, satisfactory understanding of the user and the tasks that he or she performs, multidisciplinary knowledge and an iterative process. This latter characteristic is the only one mentioned by all the consulted sources and is also the only characteristic that is intrinsic to the development process. Adding a number of activities and/or techniques (related to usability in this case) to a process that is not based on an iterative approach will not make it iterative.

The other characteristics can be achieved by incorporating the techniques that are part of the integration framework proposed in this research into the process, as justified below:

a) **User involvement**: Some involvement by real users is needed to achieve a user-centred process. It is noteworthy that user participation in development amounts to an important change of developer mentality as regards how they address problems. The models and diagrams that they use should not call for in-depth technical knowledge if users are to be able to make contributions to design, and it is essential to assure that the voice of users is not drowned out by the opinions of the engineers. There are some usability techniques that play this role, as noted in section 3.2 ,and, therefore, meet the need for user involvement in the process

b) **Satisfactory understanding of users and the tasks that they perform**: Although well-defined development processes are usually very much concerned with the requirements question, the importance of understanding the user seen as part of a broader context is the key for final product usability. It is a matter of addressing problem understanding by taking a wider view than just functional requirements to deal with the ultimate needs of the user. Apart from contributing to the general goal of understanding the user, the understanding of the tasks performed by the user also provides a basis upon which to design tasks that match the routine way in which the user does things. The addition of usability techniques to requirements activities contributes to the satisfactory understanding of the user and his or her tasks.

c) **Multidisciplinary knowledge**: To properly understand the user within the context of his or her job and organisation and to provide an interaction suited to his or her characteristics and limitations, issues from more than one discipline need to be taken into account. The usability techniques that are part of the integration framework proposed in this research are steeped in this multidisciplinary knowledge because they come from the field of HCI, and, therefore, their application meets this requirement.

Consequently, the only requirement identified as intrinsic to the process and, therefore, needed in any development process into which usability is to be integrated using the proposed framework is that it be based on an iterative approach. The complexity of the human side of human-computer interaction makes it almost impossible to create a satisfactory design first time round. Iterative processes are common practice in HCI. In SE, it is the most prominent development approach in current SE literature, and, therefore, its identification as a requirement for the development process in place is not foreign to the best SE practices today.

5.2 Types of Cycles in an Iterative Process

A framework needs to be established according to which instructions as to the time of application for each technique can be entered. We are going to make a distinction between three stages in the cycles of an iterative process: initial cycles, central cycles and evolution cycles. This particular division is made for the following reasons:

- In HCI development processes, work at the start of development focuses on the construction of a satisfactory product concept. In any iterative process, the early cycles are given over to the points identified as being the riskiest, and one of the biggest risks as far as usability is concerned is not developing a product with the right concept. Some usability techniques focus on the production and/or evaluation of early tentative designs and, therefore, they are especially well suited for application in the initial cycles. Having established the product concept, these techniques do not usually need to be applied in the remainder of the development process. Therefore, a distinction is made between the initial and central cycles, which start once development has focused on the concept of the actual product.

- The central cycles are defined as the ones that are carried out between the initial cycles and the evolution cycles. Unlike the techniques for use in the initial cycles, other techniques call for analysis and specification work to have been done beforehand, if they are to be applied with any chance of success. These techniques are better suited for the central cycles than for the initial cycles.

- Finally, some techniques (chiefly usability evaluation techniques) cannot be applied unless a version of the software system under development has been installed at the end user's workplace. Therefore, to be able to indicate when these techniques are applicable, we define the evolution cycles as the cycles that are carried out once part of the software has been installed in the end user's environment. These cycles match the transition phase in unified process terminology [Jacobson, 99].

Figure 1 illustrates the three stages into which the cycles of an iterative process have been divided.

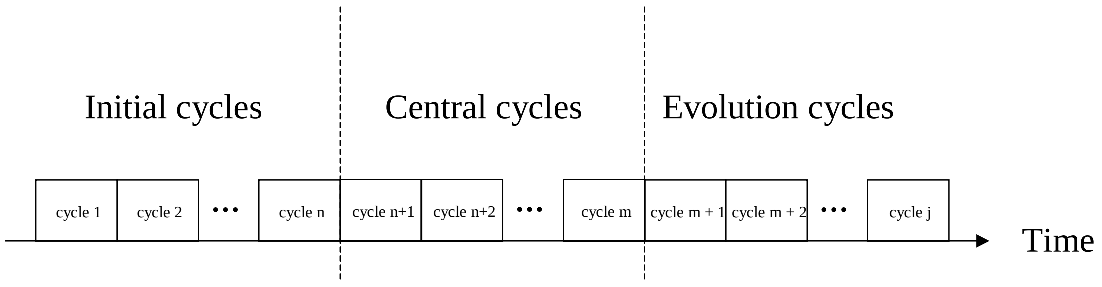

Figure 1 Division of Iterative Cycles into Stages

Each usability technique selected in section 3 will have a level of applicability in each of these three cycle types or stages in an iterative process. The suitability of the technique for one particular stage will be rated as one of the following three:

- Especially well-matched for the development stage in question: this means that the technique will be of utmost usefulness when applied in the development stage in question.
- Can be applied: the technique is applicable in this stage, although this stage does not stand out as being any better matched than other development times.
- Not often applied: this means that the technique is not usually applied at this stage of development.

5.3 Techniques for Application in each Development Stage

Below we detail for each stage the techniques organised according to how suitable their application is at the development time in question.

5.3.1 Techniques for Application in the Initial Cycles

In HCI methods, we find that more emphasis is laid on activities related to the observation and analysis of users in their usual environment at the start of development in order to identify the main characteristics of users and the features of the tasks that they perform. Indeed, a satisfactory analysis of users and their tasks is one of the fundamental issues in a user-centred approach. Another type of activity associated in the HCI field with initial development efforts is related to the development of the product concept. In projects that are in any way innovative (as opposed to directly mechanising tasks that are performed manually), one point that is identified as important is getting the whole development team to take a common view of the basic principles according to which the software system is to operate. It is important for this common view or general outline of operation to be established early on in development, as it props up the remainder of the development process. This is a similar (albeit not identical) concept to the system architecture baseline, as identified in the Unified Process, where one of the principal goals of the Elaboration stage is to establish a solid architectural foundation to drive the work of the later development phases [Jacobson, 99].

5.3.1.1 Especially Well-Matched Techniques

The techniques that are especially well-matched for the initial cycles are, firstly, the observation techniques, by means of which contact is first made with users and their tasks, and, secondly, techniques that help to establish the product concept. Specifically, the techniques to be taken into account in this stage are as follows:

- Ethnographical Observation: The observation of users to learn how they work and reason is suited for the initial development stages, as it helps to elicit the relevant characteristics of users that will have to be taken into account throughout the entire development process.

- Contextual Inquiry: This technique is an alterative to the traditional interviewing technique applied in requirements engineering. As it also helps to understand how the user reasons and what his or her real goals and motivations are, it is especially suited at the initial stages of development.

- User Profiles and User Role Map: Modelling standard users, their environment and their main tasks is an activity that is especially well-suited for performance at the start of development, although it can be refined and added to in later stages.

- Personas: Although this technique calls for an in-depth study of potential users, the purpose of its output is to later drive the whole design process. Therefore, it is reasonable for it to be applied primarily in the early development cycles.

- Task Organisation Model and HTA: As these techniques serve to model how users organise and what resources they use to perform their activities, they are especially well-matched for application in the initial cycles. This is because this is when most of the user observation is done to try to get a better understanding of user activity-related mental schemas.

- Affinity Diagramming: This technique is useful for two goals. On the one hand, it is used to organise the freehand notes taken during contextual inquiry or ethnographical observation and, as such, should be applied alongside these techniques. On the other, it can be effective for organising ideas from brainstorming meetings, which are often also held in the early development stages to define the product concept.

- Visual Brainstorming: As mentioned above, brainstorming meetings are especially well-matched for the start of a project to evaluate possible solutions without any sort of previous condition.

- Competitor Analysis: The analysis of similar systems plays an important role in the search for what characteristics a software product under development should have. Whereas the reuse of solutions developed by others has played a secondary role in SE[9], HCI does attach importance to the analysis of solutions adopted by competitor products, probably because the system-user interaction features are directly observable and, therefore, can be assimilated into one's own

[9] Traditional SE has attached a lot of importance to the development process, stressing that the application of a good process generates a good product. A host of SE methods have been published, each one of which suggests that its application produced better software products. Taking this claim to the extreme, we would not need to look at existing solutions, because the application of the method itself assures the creation of a quality product.

design. As competitor analysis may be useful in the process of generating design ideas for the product concept, this technique can be identified as especially well-matched for the initial cycles.

- Scenarios and Storyboards: The aim of these two technique is to make the development team think about the context of use, about tangential issues, apart from the pure functionality that the user will have to use, that go beyond classical requirements. Therefore, they are especially well-suited for use in the early stages, because they assure that the context of use is kept in sight throughout later design work.

- Paper Prototypes: As opposed to the prototypes traditionally used in SE, which are relatively expensive, paper prototypes are very low cost and are, therefore, especially well-suited at very early development times, when there are rough design ideas that need to be examined with the user.

- Usability Specifications: According to Hix and Hartson [Hix, 93], usability specifications should be established as early on as possible in the development process. They are used in later cycles as an indication of whether progress is being made in the right direction towards the goal of outputting a system with the satisfactory usability level. Therefore, this technique is especially well-matched for the initial development cycles.

- Conceptual Model: As this model can specify the view of how the system is to be operated at the most abstract level, it is a technique that is especially appropriate for reflecting in the initial cycles what the product concept is from the viewpoint of interaction design.

- Transition Diagrams and Navigation Map: These techniques can be used to roughly describe the transition between the main windows of the system under development. This type of representation can visualize the high-level structure of the interaction, which is a way of conveying the product concept. As this technique can be used in discussions on the product concept at the start of development, we believe that it is especially recommendable in the early cycles. Later use, when specific aspects of the interaction are designed, is not ruled out, however.

5.3.1.2 Applicable Techniques

Some techniques are applicable in the initial cycles, although they cannot be said to be more relevant at this development stage than at other later stages. Most of the modelling techniques fall within this category, as the models are refined and extended with new elements dealt with in each cycle as the iterative development process advances. Most of the techniques that are applied in design activities can also be applied in the initial cycles, alongside some usability evaluation techniques that can be run without the system having to be operational.

The techniques applicable in the initial cycles are as follows:

- Essential Use Cases and Task Scenarios: The tasks that the user is to perform using the system are modelled throughout the whole development process and are not especially well-matched to any stage.

- JEM (Joint Essential Modelling): This technique focuses on cooperative user and (especially) task modelling, including users, developers and any other stakeholders. As modelling is carried out throughout development, this technique cannot be singled out as being especially well-matched to the initial cycles, but can be applied throughout the entire development process.

- Interface Content Modelling: This technique calls for more effort than paper prototypes, as it specifies what the UI should be like. It can be applied in the initial cycles, but also in later cycles, where it would serve the purpose of conveying to UI programmers what elements the UI should contain.

- Menu Trees and Card Sorting: These two techniques are used to model and design the application menus, respectively. The application menus can be considered in both the initial cycles and later development stages. Therefore, these techniques are especially outstanding, although they can be applied in the initial cycles.

- Organisation of Help by Use Cases: The design of the help subsystem can be addressed at any time during development. Therefore, this technique, which offers a way of organising the structure of the help subsystem, can be applied in any development stage.

- Cognitive and Pluralistic Walkthroughs: Both usability evaluation techniques require nothing more than a representation of the screens under evaluation. As they only require paper prototypes, they can be applied in the initial cycles, even if none of the software system has been implemented yet.

- Heuristic Evaluation and Inspections: These usability evaluation techniques can also be applied in the initial cycles, because all they need are representations of the UI screens to be run.

5.3.1.3 Techniques Not Often Applied

The usability techniques not mentioned in the above two sections are not usually applied in the initial cycles for the following reasons:

- Product Style Guide: When a product style guide is established, it is often so detailed that it is not usually developed until the overall concept of the system has been established, which means that it is not commonly applied at the initial stage of development. One reason for its application in the initial cycles would be if there were already an organisational style guide that required just small adaptations for the specific project being undertaken.

- Impact Analysis: This technique is used to decide which usability problems are addressed first, to establish their priorities with a view to the start of a cycle. For its application, it requires a list of usability problems identified in the usability evaluation activities. As work in the early phases revolves mainly around the definition of the product concept, specific usability problems are not usually dealt with until the central cycles.

- Usability techniques that call for an operational system: These techniques are not very common in the initial development cycles, because a software prototype is not usually built until the central cycles. The techniques within this category are as follows:
  - Techniques related to usability testing:
    - Thinking Aloud
    - Performance Measurement
    - Post-Test Information
    - Laboratory Usability Testing

- Installed Systems Follow-Up Studies:
  - Use Logging
  - Questionnaires, Interviews and Surveys
  - User Feedback

5.3.2 Techniques for Application in the Central Cycles

Some usability techniques are characterised as involving detail and can be considered as especially well-matched to the central cycles that start once the product concept has been decided on and the interaction issues that had been dealt with at a higher level in the initial cycles can be addressed in detail. Moreover, most techniques are suited for these central cycles, as we will see in the following.

5.3.2.1 Especially Well-Matched Techniques

The techniques that are especially well-matched for the central cycles are the ones that call for more detail and are not usually useful before a specific product concept has been decided. These techniques are related to the detailed interaction design, that is, its concrete visual part. Specifically, they are the following techniques:

- Product Style Guide: The style guide is closely related to the visual elements of the UI, which are built in the detailed interaction design carried out in the central cycles. Therefore, we consider that this technique is especially well-matched for application in the central cycles, although the style guide can include additional elements established in the initial cycles.

- Interface Content Model: As the chief purpose of this technique is to serve as a tool for communication between high-level interaction designers and the UI programming team, its application is especially well-matched for the central cycles, when more structured work is being done than in the initial cycles, where the formal specification carries must less weight.

5.3.2.2 Applicable Techniques

Below we detail the techniques that are applicable in the central cycles, although they cannot be said to be more relevant at this development stage than at other stages:

- User Profiles and User Role Map: Some data on users and/or user types may need to be refined or added to in the central cycles.

- Essential Use Cases, Task Scenarios and JEM: As indicated above, the system task modelling activities are carried out throughout development, there is no stage to which they are particularly well matched.

- Competitor Analysis: This technique, apart from helping to develop the product concept, can be a source of inspiration for the detailed interaction design that is carried out in the central development cycles.

- Paper Prototypes: All prototyping is suited for application in the central cycles and, particularly, also low-fidelity prototypes, like paper prototypes.

- Cognitive and Pluralistic Walkthroughs: As mentioned above, these techniques are applicable throughout development, as well as in the central cycles.

- Menu Trees, Card Sorting, Conceptual Modelling, Transition Diagrams and Navigation Map: Being useful for modelling UI elements, these techniques can be applied in the central cycles, which necessarily include detailed interaction design activities.

- Impact Analysis: This technique is useful at the start of a cycle to decide which of the identified usability problems are going to be addressed first. Therefore, it is fully applicable in the central cycles.

- Organisation of Help by Use Cases: As mentioned above, the design of the help subsystem can be undertaken at any time during development. Therefore, this technique can also be applied in the central cycles of development.

- Usability evaluation techniques: All the techniques are applicable, except the ones that call for a product that has already been installed in the context of use for which it was developed. The applicable techniques are as follows:
  - Heuristic Evaluation
  - Inspections (including Collaborative Inspections)
  - Cognitive and Pluralistic Walkthroughs
  - Thinking Aloud
  - Performance Measurement
  - Post-Test Information
  - Laboratory Usability Testing
  - Questionnaires, Interviews and Surveys

5.3.2.3 Techniques that are Not Often Applied

The techniques not often applied in the central cycles fall into two groups: techniques that are designed to gather initial information about users and for building the product concept and techniques that can only be applied if the system has been installed in the environment in which it is to be used. These techniques are as follows:

- Ethnographical Observation, Contextual Inquiry, Task Organisation Model and HTA: Used to gather information about users and model how they work, which means that they are most often applied in the initial cycles.

- Personas: Aimed at condensing the main archetypal user characteristics, which would be used to drive the entire development. For this reason, it is applied chiefly in the initial cycles.

- Affinity Diagramming, Visual Brainstorming, Scenarios and Storyboards: These techniques are applied to build the product concept. As this concept has been built by the time the central cycles start, these techniques are not usually applied.

- Usability Specifications: The purpose of the specifications is to drive all development. They are, therefore, expected to be stable by the end of the initial cycles. Consequently, they are not very often applied in the central cycles, because they are established beforehand.

- Use Logging and User Feedback: As indicated above, they require a system installed in the environment in which it is to be used and are, therefore, applied chiefly in the evolution rather than the central cycles.

5.3.3 Techniques for Application in the Evolution Cycles

The only special characteristic of the evolution cycles is that part of the system should be ready for installation in the end user environment. Therefore, they are not very different from the central cycles.

5.3.3.1 Especially Well-Matched Techniques

Some usability evaluation techniques are based on the system or part of the system being in use. Therefore, these techniques are especially well-matched for application in the evolution cycles. They are as follows:

- Real Use Logging
- Questionnaires, Interviews and Surveys
- User Feedback

5.3.3.2 Applicable Techniques

The techniques applicable in the evolution cycles are the same as the techniques applicable in the central cycles. The only difference lies in the techniques that are especially well-matched for the central cycles (Product Style Guide and Interface Content Modelling), which are just applicable in the evolution cycles, that is, are not particularly outstanding. They are as follows:

- User Profiles and User Role Map
- Essential Use Cases, Task Scenarios and JEM
- Competitor Analysis
- Paper Prototypes
- Cognitive and Pluralistic Walkthroughs
- Menu Trees, Card Sorting, Conceptual Model, Transition Diagrams and Navigation Map
- Interface Content Model
- Product Style Guide
- Impact Analysis
- Organisation of Help by Use Cases
- Heuristic Evaluation
- Inspections (including Collaborative Inspections)
- Thinking Aloud
- Performance Measurement

- Post-Test Information
- Laboratory Usability Testing

5.3.3.3 Techniques that are Not Often Applied

The techniques that are not often applied in the evolution cycles are the same as the techniques for the central cycles as regards techniques that are applied in analysis activities and differ only as to the techniques related to evaluation activities, as some are especially well-matched for the evolution cycles, whereas they are not very often applied in the central cycles. The techniques not often applied in the evolution cycles are as follows:

- Ethnographical Observation, Contextual Inquiry, Task Organisation Model and HTA
- Personas
- Affinity Diagramming, Visual Brainstorming, Scenarios and Storyboards
- Usability Specifications

5.4 Summary

The fitness of each technique for each development time, as detailed in the above section, is listed in Table 17 and Table 18. Three shades of grey have been used to indicate the fitness of each technique for each development time, as illustrated in Table 16. The darkest shade indicates that the technique is especially well-matched for the stage in question, the mid-grey that it is simply applicable and the lightest shade that the technique is not often applied.

Table 16 – Fitness of a Technique for a Development Time

Content	Key
⬛	Especially well-matched for the development stage
▩	Can be applied
⬜	Not often applied

Table 17 – Summary of the Application Times for each Usability Technique (Techniques related to Analysis Activities)

Activities		**Techniques**	**Initial Cycles**	**Central Cycles**	**Evolution Cycles**
Analysis	Requirements Elicitation, Analysis and Negotiation	Competitor Analysis [Nielsen, 93]	▓	▒	▒
		Contextual Inquiry [Beyer, 98]	▓	░	░
		Affinity Diagramming [Beyer, 98]	▓	░	░
		Ethnographical Observation [Preece, 94]	▓	░	░
		JEM (*Joint Essential Modelling*) [Constantine, 99]	▒	▒	▒
		User Profiles [?]	▓	▒	▒
		User Role Map [Constantine, 99]	▓	▒	▒
		Personas [Cooper, 03]	▓	░	░
		Essential Use Cases [Constantine, 99]	▒	▒	▒
		Task Organisation Model [?]	▓	░	░
		HTA [?]	▓	░	░
		Task Scenarios [Mayhew, 99]	▒	▒	▒
		Card Sorting [Robertson, 01]	▒	▒	▒
		Scenarios and storyboards [Carroll, 97a]	▓	░	░
		Visual Brainstorming [Preece, 94]	▓	░	░
		Conceptual Modelling [Mayhew, 99]	▓	▒	▒
		Paper Prototypes [Constantine, 99]	▓	▒	▒
	Requirements Specification	Usability Specifications [Hix, 93]	▓	░	░
	Requirements Validation	Heuristic Evaluation [Nielsen, 93]	▒	▒	▒
		Inspections [Nielsen, 94]	▒	▒	▒
		Cooperative Inspections [Constantine, 99]	▒	▒	▒
		Cognitive Walkthrough [Carroll, 97b]	▒	▒	▒
		Pluralistic Walkthrough [Bias, 94]	▒	▒	▒

Table 18 – Summary of the application Times of each Usability Technique (Techniques related to Design and V&V Activities)

Activities		Techniques	Initial Cycles	Central Cycles	Evolution Cycles
Design	Interaction Design	Product Style Guide [Mayhew, 99]	░	▓	▒
		Menu Trees [Shneiderman, 98]	▒	▒	▒
		Transition Diagrams [?]	▓	▒	▒
		Interface Content Model [Constantine, 99]	▒	▓	▒
		Navigation Map [?]	▓	▒	▒
		Impact Analysis [Hix, 93]	░	▒	▒
		Organisation of Help by Use Cases [Constantine, 99]	▒	▒	▒
V&V	Usability Evaluation	Heuristic Evaluation [Nielsen, 93]	▒	▒	▒
		Inspections [Nielsen, 94]	▒	▒	▒
		Collaborative Inspections [Constantine, 99]	▒	▒	▒
		Cognitive Walkthrough [Carroll, 97b]	▒	▒	▒
		Pluralistic Walkthrough [Bias, 94]	▒	▒	▒
		Thinking Aloud [Nielsen, 93]	░	▒	▒
		Performance Measurement [Rubin, 94]	░	▒	▒
		Post-Test Information [Mayhew, 99]	░	▒	▒
		Laboratory Usability Testing [Rubin, 94]	░	▒	▒
		Use Logging [Shneiderman, 98]	░	░	▓
		Questionnaires, Interviews and Surveys [Mayhew, 99]	░	▒	▓
		User Feedback [Shneiderman, 98]	░	░	▓

6. CONCLUSIONS

As a result of the STATUS project, we have come up with a framework for integrating a range of usability techniques throughout the entire development process. The ultimate goal is to provide developers with guidance as to how, when and where to apply HCI techniques in their development processes, as well as which are the best suited techniques. This tailoring process depends on several characteristics:

- the type of organisation in which the developer is working,
- the type of practitioners who are going to apply the techniques,
- the type of development process followed,
- the type of projects developed,
- etc.

So, the proposed integration framework is flexible enough to be applicable in different settings, and needs to be tailored to each case.

Note that these research results are complementary to those developed in other project WPs that target the inclusion and evaluation of particular usability mechanisms in a software system. Both types of results should be applied to contribute to usable software development.

7. References

[Constantine, 99]	L. L. Constantine, L. A. D. Lockwood. *Software for Use: A Practical Guide to the Models and Methods of Usage-Centred Design*. Addison-Wesley, New York, NY, 1999.
[Good, 86]	M. Good, T. Spine, J. Whiteside, P. George. “User-derived impact analysis as a tool for usability engineering” in *Proc. of the ACM CHI’86 Conference*. pp. 241-246. ACM, 1986.
[Gould, 88]	J.D. Gould. “How to Design Usable Systems” in *Handbook of Human Computer Interaction*, edited by M. Helander. Elsevier, 1988.
[Hix, 93]	D. Hix, H.R. Hartson. *Developing User Interfaces: Ensuring Usability Through Product and Process*. John Wiley and Sons, 1993.
[IEEE1074, 91]	IEEE. *IEEE Standard for Developing Software Life Cycle Processes, IEEE Standard 1074-1991*. IEEE, 1991.
[ISO9241, 98]	ISO. *ISO 9241-11. Ergonomic Requirements for Office Work with Visual Display Terminals. Part 11: Guidance on Usability*. ISO, 1998.
[ISO12207, 95]	ISO/IEC. *ISO/IEC International Standard: Information Technology. Software Life Cycle Processes, ISO/IEC Standard 12207-1995*. ISO, 1995.
[ISO13407, 99]	ISO. *ISO 13407.Human-Centred Design Processes for Interactive Systems*. ISO, 1999.1
[Mayhew, 99]	D. J. Mayhew. *The Usability Engineering Lifecycle*. Morgan Kaufmann, 1999.
[Nielsen, 93]	J. Nielsen. *Usability Engineering*. AP Professional, 1993.
[Preece, 94]	J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, T. Carey. *Human-Computer Interaction*. Addison Wesley, 1994.
[Shneiderman, 98]	B. Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley, 1998.
[Wixon, 97]	D. Wixon, C. Wilson. “The Usability Engineering Framework for Product Design and Evaluation”. In *Handbook of Human-Computer Interaction*. pp. 653-688. Ed. by M. G. Helander et al. Elsevier North-Holland, 1997.