INFORMATION SOCIETIES TECHNOLOGY (IST) PROGRAMME

STATUS

"Software Architecture for Usability"

WORKPACKAGE 5: Integrated development process with usability techniques

DELIVERABLE D.5.1. SELECTION OF THE SOFTWARE PROCESS AND THE USABILITY TECHNIQUES FOR CONSIDERATION

Version: 1.0

Submission Date: 4/2002

Authors: Xavier Ferré, Natalia Juristo, Ana Moreno

Partners: UPM

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

DOCUMENT CONTROL

Registration of Changes

Date	Version	Author of Changes	Comments
9/4/2002	0.2	UPM	Selection of software process added
16/4/2002	0.3	UPM	Section 7 reworked
21/4/2002	0.4	UPM	Refinement of sections 3 to 6
23/4/2002	0.5	UPM	Minor changes
23/4/2002	1.0	UPM	Final version to send to CEC

List of Related STATUS Documents

Document Name	Version
D.1.1 Periodic Progress Report	1.0
D.5.2 Specification of the software process with integrated usability techniques	
Technical Annex	

ACRONYMS AND ABBREVIATIONS

Acronyms and Abbreviations	Meaning
GOMS	Goals, Operations, Methods and Selection Rules
HCI	Human Computer Interaction
HTA	Hierarchical Task Analysis
JEM	Joint Essential Modelling
TAG	Task-Action Grammars
UAN	User Action Notation
WP	Work Package

TABLE OF CONTENTS

DOCUMENT CONTROL **2**

ACRONYMS AND ABBREVIATIONS **3**

1. INTRODUCTION **5**
1.1 PURPOSE 5
1.2 DOCUMENT STRUCTURE 5

2. REQUIREMENTS FOR A USER-CENTRED DEVELOPMENT PROCESS **6**

3. LITERATURE SURVEY ON USABILITY ACTIVITIES **8**
3.1 INTRODUCTION 8
3.2 SOURCES FOR THE SURVEY 8
3.3 [NIELSEN93] 9
3.4 [ISO9241_98] 10
3.5 [ISO13407_99] 10
3.6 [SHNEIDERMAN98] 10
3.7 [HIX93] AND [PREECE, 94] 11
3.8 [WIXON97] 12
3.9 [CONSTANTINE99] 13
3.10 [MAYHEW99] 13

4. SELECTION OF CANDIDATE USABILITY ACTIVITIES **15**
4.1 SUMMARY OF USABILITY ACTIVITIES 15
4.2 ANALYSIS ACTIVITIES 17
4.2.1 Specification of context of use *17*
4.2.2 Usability Specifications *19*
4.3 DESIGN ACTIVITIES 19
4.3.1 Develop Product Concept *19*
4.3.2 Prototyping *21*
4.3.3 Interaction Design *21*
4.4 EVALUATION ACTIVITIES 22

5. LITERATURE SURVEY ON USABILITY TECHNIQUES **23**
5.1 INTRODUCTION 23
5.2 SOURCES FOR THE SURVEY 23
5.3 [NIELSEN, 93] 23
5.4 [PREECE, 94] 25
5.5 [HIX, 93] 29
5.6 [SHNEIDERMAN, 98] 30
5.7 [CONSTANTINE, 99] 32

6. SELECTION OF CANDIDATE USABILITY TECHNIQUES **36**
6.1 SUMMARY OF REVIEWED TECHNIQUES 36
6.2 DESCRIPTION OF CANDIDATE TECHNIQUES 42
6.2.1 Analysis-Related Techniques *42*
6.2.2 Design-Related Techniques *43*
6.2.3 Evaluation-Related Techniques *45*
6.3 A PRIORI DISCARDED TECHNIQUES 45

7. CONCLUSION **48**

8. REFERENCES **49**

1. INTRODUCTION

1.1 Purpose

This document constitutes D.5.1 Selection of a general user-centred software development process. It is the first deliverable of Work Package 5.

As specified in D.1.1 Periodic Progress Report, D.5.1 will focus on two different issues:

1) This deliverable presents the requirements for a software development process that supports usability.
2) This deliverable presents the usability activities and techniques which are candidates to be included in a development process with the characteristics described in 1).

1.2 Document Structure

According to the purpose of this deliverable, section 2 presents what requirements a software development process should meet for it to be considered to support a high level of usability in the final product.

Section 3 details a literature survey on the usability activities proposed by several authors in the field. Based on this survey, section 4 describes the selection of candidate usability activities for integration into a development process with the characteristics described in section 2

Structured similarly, section 5 details a literature survey on usability techniques, while section 6 presents the usability techniques selected for inclusion in the already mentioned development process.

Finally, section 7 presents the conclusions of the document.

2. REQUIREMENTS FOR A USER-CENTRED DEVELOPMENT PROCESS

As it is specified in the Technical Annex, one of the objectives of WP 5 is the selection of a general user-centred software development process on which to integrate afterwards techniques that facilitate the development of a usable software system. Initially, some classical software processes were considered as candidates to this generic process, such as ISO Standard 12207 [ISO12207, 95] and IEEE Standard 1074 [IEEE1074, 91]. After some consideration, however, it was determined that linking WP 5 to a fixed software process could amount to a too demanding requirement for some organisations, leading to less extensive use of the results of this WP. Software development organisations with a consolidated development process are unlikely to switch it for a completely different process, even though it promises some improvements (for example, in the usability field).Therefore, instead of selecting a generic process that would force the organisation interested in using our research results to change its process, it was considered to be more useful to set out the conditions to be met by a software process that supports usability. Accordingly, each organisation can evaluate whether the type of process it has in place can be modified to incorporate usability-oriented techniques and activities or whether, on the contrary, the type and features of the process it is using are unsuitable for supporting usability. This new approach will increase the practical applicability of the resulting development process. The appeal for a software development organisation lies in the fact that it does not have to abandon the in-house process to adopt some improvements, as it is enough to just modify the existing process.

Therefore, this section identifies the characteristics that a software development process should have for it to be considered user-centred and, therefore, support the development of a final product with a high level of usability. These characteristics will be able to be used by any organisation to decide whether its software process can serve as a basis for the integration of usability techniques into software development or, on the contrary, it has to consider migrating to another type of process if it really intends to go for usability.

Albeit strictly in reference to user-centred *design*, [Preece, 94] gives a definition of user-centred that is potentially of interest for our process requirements search. It should:

a) be user-centred and involve users as much as possible so that they can influence the design,

b) integrate knowledge and expertise from the different disciplines that contribute to HCI design,

c) be highly iterative so that testing can be done to check that the design does indeed meet user requirements.

The ISO Standard on Human-Centred Design Processes for Interaction Systems [ISO13407, 99] defines that the incorporation of a human-centred approach is characterised by the following:

a) the active involvement of users and a clear understanding of user and task requirements;

b) an appropriate allocation of function between users and technology;

c) the iteration of design solutions;

d) multi-disciplinary design.

Looking for a different point of view, we find that [Constantine, 99] defines the elements of a usage-centred approach as follows:

a) Pragmatic design guidelines

b) Model-driven design process

c) Organised development activities

d) Iterative improvement

e) Measures of quality.

[Ferré, 01] also sticks with iterative refinement when dealing with the usability process: "It is crucial to evaluate all results during the product development process, which ultimately leads to an iterative development process. A pure waterfall approach to software development makes introducing usability techniques fairly impossible".

[Shneiderman, 99] states that a process that supports usability needs to be non-hierarchical in the sense that it is neither strictly top-down nor bottom-up; and it is radically transformational, implying the production of interim solutions that could ultimately play no role in the final design. This leads to an iterative development approach.

From the characteristics of a proper user-centred process detailed above, we can extract three main issues that need to be dealt with: user involvement, adequate understanding of user and task requirements and iterative process. However, it is the first two requirements that will be part of the research results of WP 5. That is, when and how the user should be incorporated and what usability knowledge should be applied and when. On the other hand, iterative development is an intrinsic development process requirement. Therefore, according to [Preece, 94], the organisation's design process should be highly iterative to support usability and, consequently, to be able to incorporate the modifications proposed in this WP.

Iterative refinement is a must. The usability level of the system cannot be predicted in advance. Some kind of usability evaluation is needed at the end of every iterative cycle. Therefore, the requirement of an iterative process is closely linked to the need to perform quality measures at the end of each cycle.

The complexity of the human side in human-computer interaction makes it almost impossible to create a correct design at the first go. Cognitive, sociological, educational, physical and emotional issues may play an important role in any user-system interaction. Interaction design must, therefore, be tested and refined all through the development process in order to obtain a satisfactory result from the point of view of usability.

The other two requirements: user involvement and adequate understanding of user and his or her tasks, are also changes in developers' way of doing things, although these changes can be located at definite development times and this, precisely, will be the final result of this WP.

For proper user-centredness, some degree of user involvement is necessary. On this subject, it should be mentioned that user involvement implies an important shift in developers' models and way of looking at problems. The system-centred view, present in numerous techniques and, therefore, in a lot of developers, must give way to user-centred techniques. Models and diagrams should not require an important technical background to be understood so that users can judge them and provide design alternatives for discussion. Some of the techniques that we select in section 6 deal with this issue.

Regarding user understanding, even though most well-defined development processes do happen to address requirements issues, the importance of user understanding (not only functional needs) for the usability of the final product makes it necessary to put particular emphasis on the issue. The point is to get the right requirements, requirements that actually address user needs, and task requirements that match users' way of doing things, their usual working environment and the organisational context. Once again, user understanding will be improved by the extra tasks and techniques that we plan to add to a regular development process.

3. LITERATURE SURVEY ON USABILITY ACTIVITIES

3.1 Introduction

The next sections of this deliverable will focus on the identification of a candidate list of usability activities and techniques to be used in a user-centred development process.

Sections 3 and 4 will focus on usability activities identified in literature and the candidate usability activities to be used in this project, respectively. Sections 5 and 6 will address usability techniques.

In that way, this section presents a survey of the activities/tasks that are proposed in the usability literature, which lead to the development of software systems with an acceptable usability level.

Although software engineering has made efforts towards software process formalisation, usability authors have not strived for formality. On the contrary, they propose tasks, activities, process heuristics and advice, which is not integrated into a process that can be used as a framework for development.

The sources vary as to the extent of formalisation. The set of usability-related activities proposed in the usability field are detailed in the sections below, where the sources are ordered from the least to the most formalised.

3.2 Sources for the Survey

For the literature survey, we will focus on just books and standards. We want to study what is commonly accepted knowledge in the usability field, and such kinds of manuscripts are better than research articles in journals and conferences for this purpose.

Of the numerous usability and human computer interaction books and standards, we have chosen the most relevant and most often cited volumes. A brief description of the nine chosen documents follows.

- Nielsen93 – "Usability Engineering". Jakob Nielsen.

  For a long time, this has been the main reference book for the usability engineering sub-discipline. Nielsen offers an engineering-like approach to building usable software systems, bringing usability issues closer to a software engineering view.

- 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 provides the definition of usability that is used in ergonomic standards. ISO standards on ergonomic requirements, e.g., VDT workstation, hardware & environment, have been widely adopted by industry.

- ISO13407_99. "ISO 13407. Human-Centred Design Processes for Interactive Systems". International Organisation for Standardisation.

  This standard aims to help those responsible for managing hardware and software design processes to identify and plan effective and timely human-centred design activities.

- Shneiderman98 – "Designing the User Interface". Ben Shneiderman.

  Shneiderman is one of the most respected authors in the HCI field (he received the ACM-SIGCHI CHI Lifetime Achievement Award last year). All three editions of this book have been fundamental reference works in user interface design, because of their balanced coverage of both theoretical and development-oriented aspects of interaction.

- Hix93 – "Developing User Interfaces: Ensuring Usability Through Product and Process" D. Hix and H. Hartson.

This book presents a very practical and hands-on approach to the issue of user interaction design. One of its objectives is to be a textbook for courses in user interface development with a strong usability focus.

- Preece94 – "Human-Computer Interaction". J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, T. Carey.

  This book presents the variety of topics addressed by the HCI field. It has an encyclopaedic aim, with some theoretical prevalence. It has been one of the main textbooks for general HCI courses to the time of writing.

- Wixon97 – "The Usability Engineering Framework for Product Design and Evaluation". D. Wixon and C. Wilson. In *Handbook of Human-Computer Interaction, 2nd edition*.

  The handbook contains articles that describe the diversity in HCI, both in research and practice. Wixon and Wilson's article gives a good overview of usability engineering. The authors belonged to the usability group at DEC that created the Usability Engineering method (as credited in [Gould, 88]).

- Constantine99 – "Software for Use". Larry L. Constantine, Lucy A.D. Lockwood. Larry Constantine is one of the gurus of the software engineering discipline. He has shifted in the last decade to the issue of usable software development; and his and Lockwood's experience as usability consultants is described in this very practical work. The book presents the authors' own method for developing usable software.

- Mayhew99 - "The Usability Engineering Lifecycle: A Practitioners Handbook for User Interface Design". This is the most detailed piece of work on usability engineering, and it presents a complete life cycle focused on usability engineering principles. It is the most recent reference book for practitioners in the field of usability engineering.

3.3 [Nielsen93]

Nielsen proposes a variety of tasks that will help to develop more usable software. However, these tasks are not ordered with respect to overall development and can be considered rather as tips on the development process to improve final product usability. His proposal is:

- Know the user
- Competitive analysis
- Goal setting
- Parallel design
- Participatory design
- Coordinating the total interface
- Guidelines and heuristic evaluation
- Prototyping
- Interface Evaluation
- Iterative Design
- Follow-up studies of installed systems
- Meta-Methods (plan for usability activities)
- Prioritising usability activities.

3.4 [ISO9241_98]

This ISO standard specifies five usability activities:

- Specification of the intended context of use for a product
- Specification of usability requirements for a product
- Product development
- Specification or evaluation of product attributes
- Usability measurement.

These activities are not described in detail, but they are related to documents and other forms of output as shown in Figure 3.1.

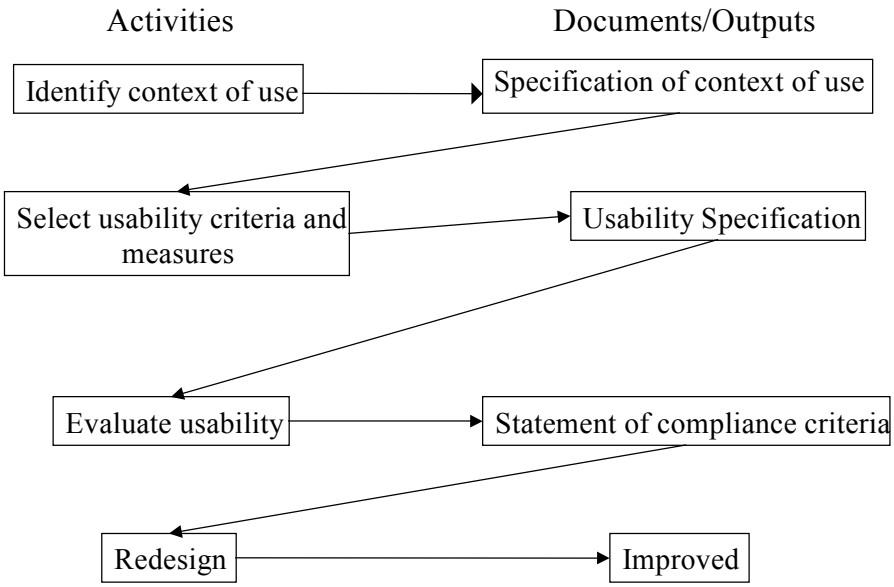

Figure 3.1 Usability activities and associated documents in [ISO9241, 98]

3.5 [ISO13407_99]

This standard describes human-centred design processes for interactive systems, and it proposes four kinds of human-centred design activities:

- Understand and specify the context of use
- Specify the user and organisational requirements
- Produce design solutions
- Evaluate designs against requirements.

3.6 [Shneiderman98]

Shneiderman describes the LUCID (Logical User-Centred Interactive Design) methodology, which is made up of the following six stages:

1. Develop product concept
2. Perform research and needs analysis

3. Design concepts and key-screen prototype
4. Do iterative design and refinement
5. Implement software
6. Provide rollout support.

3.7 [Hix93] and [Preece, 94]

Hix and Hartson try to cover both traditional development and usability activities. They propose the set of activities shown in Figure 3.2. The activities on a grey background are the ones that are more directly related to usability.

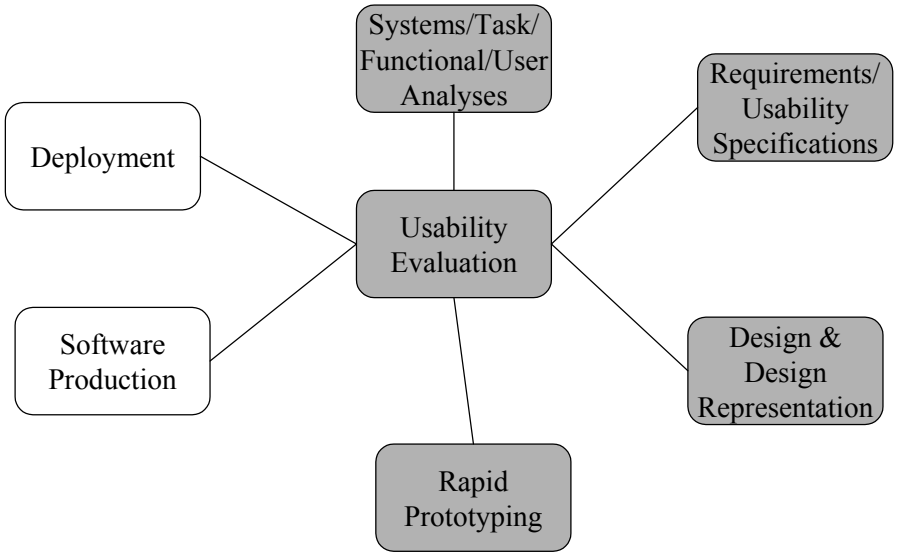

Figure 3.2 - Star life cycle

[Preece, 94] adapts Hix and Hartson's proposal, as shown in Figure 3.3. Preece et al. just give different names to some activities, and they compress Hix and Hartson's Deployment and Software Production into a unique Implementation activity, but they keep the same basic approach. Preece et al. state that the star life cycle does not prescribe a particular order for activities. The emphasis is on usability evaluation, which is central to both variants of the method. The aim is to complement a traditional top-down approach to software systems development, with a bottom-up or synthetic approach.

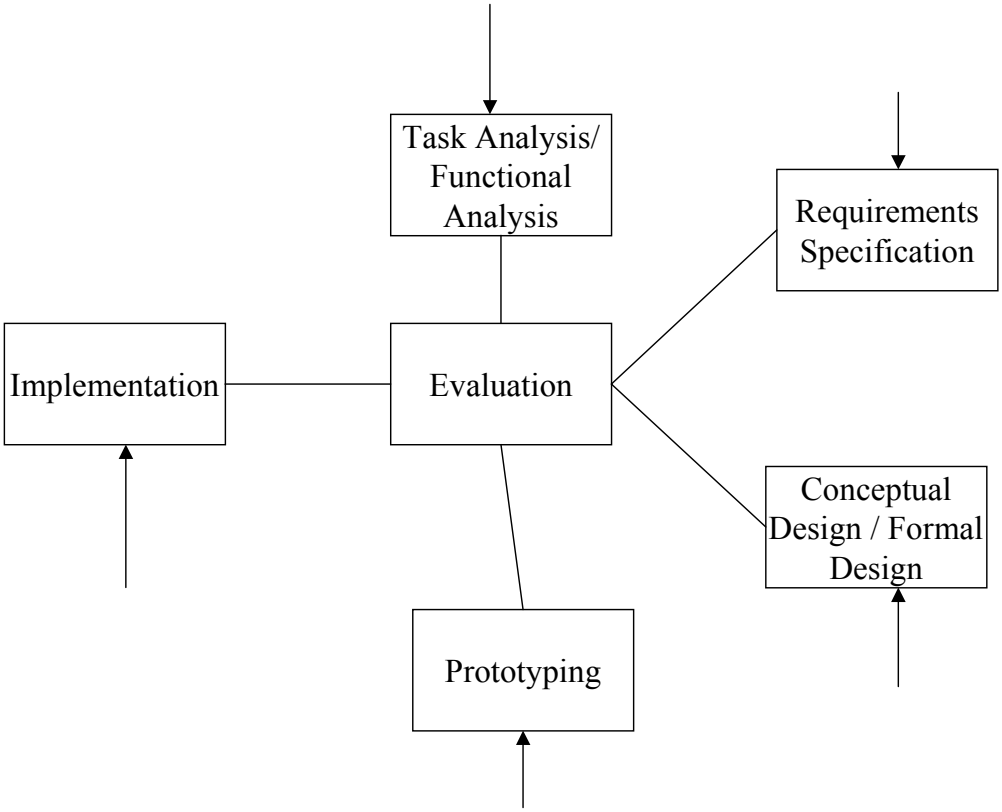

Figure 3.3 - Star life cycle, as adapted in [Preece, 94]

3.8 [Wixon97]

Wixon and Wilson propose some tasks to be done in a usability-centred process, which is based on the basic usability tasks proposed by Good et al. [Good, 86]. The tasks proposed by Wixon and Wilson are:

1. Define quantitative usability goals
2. Set levels of desired usability for each goal

   Steps 1. and 2. include:

   - Specify and categorise the users
   - Conduct a task analysis
   - Determine which usability attributes are relevant
   - Decide on a measuring instrument
   - Decide what measures will be taken from the measuring instrument
   - Set performance levels for each usability attribute
3. Test the product against the usability goals; if you meet your goals, no further design is needed
4. If further design work is needed, analyse the problems that emerge
5. Analyse the impact of possible design solutions
6. Incorporate user-derived feedback in product design
7. Return to Step 3 to repeat the test, analysis, and design cycle.

Note that the list of activities proposed is very poor, since it only deals with usability specifications and their evaluation. Its special interest is as the inspiration of all the later efforts in usability engineering, but it is the most primitive of the usability-oriented proposals considered.

3.9 [Constantine99]

Constantine and Lockwood propose a collection of coordinated activities, as part of their usage-centred design method. Their usage-centred design activity model is shown in Figure 3.4. The activities on a grey background are part of the larger software design and development process. As the figure suggests, the boundaries separating activities are not always sharply delineated; activities can overlap to some greater or lesser degree and often proceed in parallel. The development begins with the three activities at the top: Collaborative Requirements Dialog, Domain Modeling and Task Modeling, being this last activity at the core of the method. Afterwards Interface Content Modeling and Implementation are performed. Usability Inspection appears in two places, both preceding and following the implementation activities of Concentric Construction and Architectural Iteration. Running in parallel with all the modelling and design activities, there are two specialised activities: Operational Contextualization and Standards and Style Definition. The Help System and Documentation activity runs also in parallel with the development of the whole system.

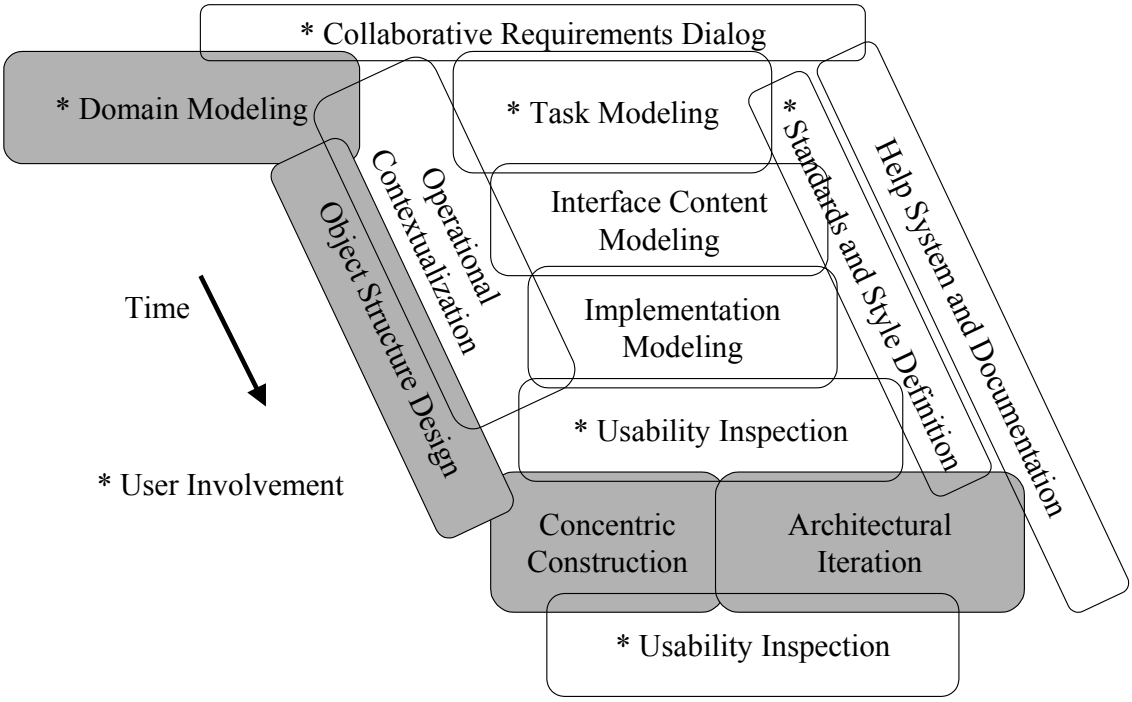

Figure 3.4 Usage-Centred Design Activity Model from [Constantine, 99]

3.10 [Mayhew99]

Mayhew proposes a set of usability tasks to be applied at different development stages:

1. Requirements Analysis:
    - User Profile
    - Contextual Task Analysis
    - Usability Goal Setting
    - Platform Capabilities/Constraints

- General Design Principles

2. Design/Testing/Development:
    - Work Reengineering
    - Conceptual Model Design
    - Conceptual Model Mock-ups
    - Iterative Conceptual Model Evaluation
    - Screen Design Standards
    - Screen Design Standards Prototyping
    - Iterative Screen Design Standards Evaluation
    - Style Guide Development
    - Detailed User Interface Design
    - Iterative Detailed User Interface Design Evaluation
3. Installation:
    - User Feedback

4. SELECTION OF CANDIDATE USABILITY ACTIVITIES

Next, we will summarise the activities analysed in the previous section and then go on to the actual activity selection.

4.1 Summary of Usability Activities

We have analysed the activities proposed by the different authors in order to extract the common ones or, at least, the activities that are at the same abstraction level and are common to several sources. Table 4.1 shows the result of this process. Our aim is to be able to easily compare the different proposals, and, therefore, we have grouped activities that refer to the same concept in the same row. Each row has been labelled (first column in the table) with the most general term or the term more often used by the authors studied, and there is one column per author that contains the respective activity that they propose.

Where the author packs several tasks into the same activity, the complete name given for the activity (for example, Systems/ Tasks / Functional / User Analysis) has been included in the table. Some authors describe a generic activity that includes the activity we are considering as a subtask. In these cases, the specific subtask is detailed between brackets. On the other hand, where the author proposes several activities that match one of our activities, they are listed using an asterisk (*). For activities not mentioned in the source, the cell contains a dash (‘-‘).

There is a clear trend in most of the sources considered as regards the activities to be done: usability specifications, prototyping and usability evaluation. The specification of the context of use, either as a complete analysis of a variety of user and organisational issues or just with an aim of knowing the user, is also quite prevalent among the different authors.

We have found more discrepancies, and less information on other design activities, like the development of the product concept and the interaction design. While some authors give no clues as to the design activity, apart from labelling it as user-centred or advocating iterative design, Constantine and Lockwood [Constantine, 99] are more specific with respect design issues, criticising the trend in usability engineering that focuses almost exclusively on usability testing. They propose a set of models to place more emphasis on design. They have started a new trend in usability (or usage-centred design, as they call it). To date, however, they do not seem to have shifted the main focus in the field.

Therefore, we can stick to “specification of the context of use”, “usability specifications”, “prototyping” and “usability evaluation”, while tentatively considering the activity of “develop product concept” and opening up the possibility of an “interaction design” activity, even though it is not yet clearly defined.

In Figure 4.1, usability activities are shown according to the traditional classification of software development activities into analysis, design and evaluation. Note that Specification of the Context of Use is decomposed into User Analysis and Task Analysis. Some authors ([Hix, 93], [Wixon, 97] and [Mayhew, 99]) make a distinction between these two activities, even though they recognise that they are tightly related. We have chosen the ISO standard terminology [ISO13407, 99], because it better reflects this tight relationship between the two sub-activities.

Activity	Nielsen93	ISO9241_98	ISO13407_99	Shneiderman98	Hix93	Preece94	Wixon97	Constantine99	Mayhew99
SPECIFICATION OF THE CONTEXT OF USE	Know the user	Specification of the intended context of use for a product	Understand and specify the context of use	Perform research and needs analysis	Systems/ tasks / functional / user analysis	Task analysis / functional analysis	* Specify and categorise the users * Conduct a task analysis	Task modelling	* User profile * Contextual task analysis
USABILITY SPECIFICATIONS	Goal Settings	Specification of usability requirements for a product	Specify the user and organisational requirements	Design concepts and key-screen prototype (create specific usability objectives based on user needs)	Requirements / Usability Specifications	Requirements specification	* Define quantitative usability goals * Set levels of desired usability for each goal	-	Usability goal setting
DEVELOP PRODUCT CONCEPT	-	-	-	Develop product concept	Conceptual design	Conceptual design / formal design	-	-	Conceptual model design
PROTOTYPING	Prototyping	-	Produce design solutions (make design solutions more concrete using simulations, models, mock-ups, etc.)	Design concepts and key-screen prototype	Rapid prototyping	Prototyping	-	-	Screen design standards prototyping
INTERACTION DESIGN	Iterative Design	-	Produce design solutions	Do iterative design and refinement	Design & design representation	Conceptual design / formal design	-	Interface content modelling	Detailed user interface design
USABILITY EVALUATION	Interface Evaluation	Usability measurement	Evaluate design against requirements	Do iterative design and refinement (conduct full-scale usability tests)	Usability evaluation	Evaluation	Test the product against usability goals	Usability inspection	Iterative detailed user interface design evaluation

Table 4.1 - Usability Activities by Source

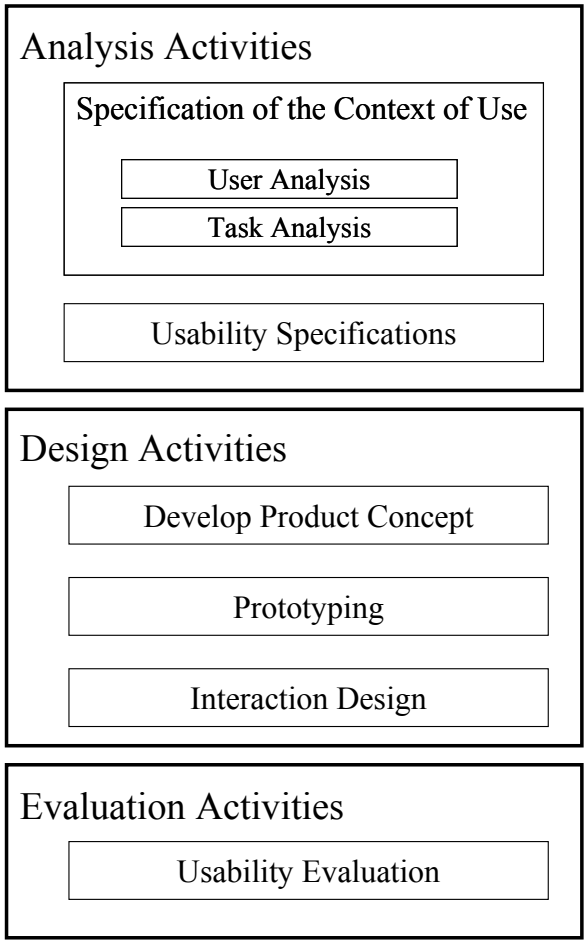

Figure 4.1 Usability Activities Grouped according to the Traditional Software Development Classification

In the following sections, details are given on every usability activity proposed. We will focus especially on details concerning the conditions of usage, implications and relationship with other activities. Where possible, information on their applicability by software engineers is given as well.

4.2 Analysis Activities

Considering user needs as part of usability-related issues corresponds to a broad view of usability. When the functions that the user will need are considered from the point of view of usability, there is an overlap with the realm of other quality attribute factors, like functionality. There is convergence between usability and requirements engineering on this issue, activities from both fields should be coordinated. Their common goal should be to find the right functionalities, the ones that the user actually needs. The aim is to provide functionalities that the user will actually use and to avoid features whose associated learning effort is not compensated by their user rating.

The following sub-sections consider usability activities that are related to analysis and, in particular, to requirements.

4.2.1 Specification of context of use

The aim of this activity is to understand and record the implications of the context of use so that they can be considered during system design.

Context of use is a broad term that comprises different interrelated aspects. As defined in [ISO13407, 99], they are:

- The characteristics of the intended users. The identification of these characteristics is known as user analysis.
- The tasks the users are to perform. Task analysis deals with this issue.
- The environment in which the users are to use the system, including the hardware, software and materials to be used.

We will describe the first two sub-activities, namely user analysis and task analysis, in more detail, while the third one will be included as part of the analysis performed during user analysis.

4.2.1.1 User Analysis

User analysis considers and identifies user knowledge, needs and characteristics that are relevant for their interaction with the system. The characteristics to be identified include knowledge, skill, experience, education, training, physical attributes, habits, preferences and capabilities. For some systems, age, disabilities, colour blindness, etc., should also be noted. These characteristics are studied in order to adapt the system to its users.

The kind of hardware and software equipment used by the target user population is also significant with regard to user characteristics. As regards this issue, user experience with computer systems generally and with similar systems is relevant for the study performed in user analysis. Indeed, even the kind of systems the user population is used to can be of interest to assure that the system is designed to meet user expectations (which are usually based on previous experience).

Physical environment is of interest as well, even though it is not, strictly speaking, a user characteristic. High noise levels, low luminosity and other physical characteristics of the users' place of work and similar data must be collected to design the software system in a way that overcomes, as far as possible, this kind of limitations, even though the resulting system would be less efficient in other environments.

Social environment can be an interesting matter, when organisational structure and work practices are relevant in the system design. For example, it may be necessary to consider that the user will be performing manual tasks while using the system, or that the user is likely to often be interrupted because of his or her job type or that fatigue may be a factor again due to the user's job type.

User analysis does not necessarily have to be performed considering the user population as a single and homogeneous group. Many projects must consider user stratification into a few relevant user groups. User analysis is more complicated in this case, as the selection of the relevant groups requires an analysis in itself, and the subsequent user analysis must be done for each group.

4.2.1.2 Task Analysis

Task analysis is aimed at eliciting descriptions of what people do, representing these descriptions, predicting difficulties and evaluating systems against usability or functional requirements [Preece, 94]. In short, it is concerned with what people do to get things done.

Task analysis is very close to functional requirements elicitation and definition, but it has the significant characteristic of focusing this activity on the overall goals of system use. The difference between a task and a function is that a task is considered to be meaningful to the user. The user believes it to be necessary and/or desirable to undertake tasks. Therefore, the term task embodies an intention or purpose that is absent in the concept of function.

The task description must include the role the user performs in the overall execution of the task, not just in terms of the functions or features provided by a product or system. Task analysis can be said to be function-oriented, but it supplements structured software engineering approaches with the consideration of user intentions when performing a task. Task analysis can be difficult to perform for software engineers because of its resemblance to functional decomposition. Whenever developers try to perform task analysis and forget to think in user terms, they may shift their mind to system

functions and end up with a task analysis that would be near nonsense for the people whose work it supposedly describes.

There is some confusion surrounding task analysis, because some authors, like [Preece, 94], use the term to refer to the activity of analysing the tasks now performed, while others, like [Hix, 93], refer to the design of the tasks that the systems will offer. As said above, for this research we will consider the former description (the current tasks performed).

The result of task analysis is very important as a starting point for the design of the functions the system will offer. When performed correctly, it is the foundation of a truly user-centred development process, because user goals should be very important and catered for throughout the whole development process.

4.2.2 Usability Specifications

Usability specifications are quantitative usability goals, which are used as a guide for ascertaining when a system has the proper usability level. They can be compared to non-functional requirements. Their concerns are user satisfaction and performance. Performance is interpreted as establishing the required performance of the new system formed by both the user and the software system, working together towards the achievement of certain goals. In this sense, usability specifications can also be called usability benchmarks.

The knowledge gathered in the specification of the context of use activity is the input for this activity. The usability specifications are defined according to the characteristics of the target user population and the goals and tasks identified in task analysis.

The set of usability specifications represent the system acceptance criteria from a usability point of view. Usability specifications are monitored at the end of each development cycle, establishing how much progress has been made towards the usability objective. They serve as criteria for determining when to stop iterating. While usability attributes are not directly measurable, usability specifications need to be. Usability attributes are then decomposed into sub-attributes and particularised for specific tasks, based on the result of the specification of the context of use. As a result, usability specifications are linked to a particular usability attribute, but refer to a particular aspect of the attribute in question.

Several authors prefer the term usability goal, as these specifications are established as a goal to be achieved in the system design. Usability goals drive design as information shared by the whole development team, which can provide decision criteria when different design alternatives are considered, not just as a test case that will be checked in the evaluation phase.

4.3 Design Activities

4.3.1 Develop Product Concept

The basis for this kind of activity lies in mental models. The concept of mental models comes from cognitive psychology, and it has manifested itself in psychological theorising and HCI research in a multitude of ways [Preece, 94].

The user always forms an image of the system in his or her head, which is usually an imperfect model of how the system works, of the logic that defines how functionalities and options are distributed in the user interface, and of the outcome of every user action when operating the system. The ideal thing for system designers would be for this user model of the system to completely match to the actual system image and for the user to be able to create this model in his or her head quickly and easily (see Figure 4.2 taken from [Preece, 94]).

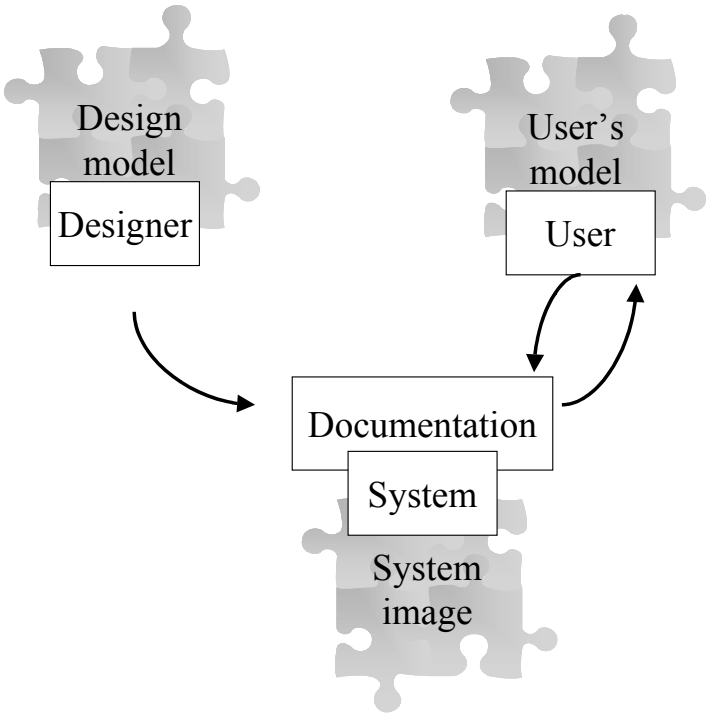

Figure 4.2 - User's model, design model and system image

The user usually develops a partial mental model, as compared to the designers' full design model, and can take advantage of only a part of the system capability as conceived by its designers. The product concept is this design model that developers have in mind. Anyway, it is an elusive and fuzzy term, because when it comes to the process of creating the product concept, different authors interpret it differently. [Preece, 94] states that there is a mismatch between the user's model and the design model when the design model is ambiguous, inconsistent or obscure. There is agreement in the field on the importance of helping the user to grow productive mental models. Nevertheless, there are very few suggestions on how to achieve this.

Metaphors in a product concept are considered at this stage. For systems whose interaction is not built upon a metaphor, the definition of the rules that will govern the user interaction ensures that the interaction design is performed properly.

When a system has no clear definition of its product concept, it will be lacking in consistency and the user will not be able to find the system logic, because there will be none. Good designers always bear in mind a certain product concept, but making it explicit and highlighting its importance in the software development process will help to shape the system in a way that explicitly communicates this product concept to the user.

Like the above-mentioned analysis activities, develop product concept is not foreign to software engineering, because it is connected with a general modelling aim: to ensure communication between development team members. It has the additional aim of including the user as a target of the communication effort. The search for coherence and logic in a product design is a genuine engineering endeavour, and some fine distinctions must be added in this activity in order to circumscribe this logic and coherence to user expectations and knowledge.

Failure to carry out this activity increases the risk of developing a system whose possibilities and way of usage are difficult to grasp, and where later efforts directed at increasing its usability are costly and have little impact.

4.3.2 Prototyping

[ISO13407, 99] defines prototype as "a representation of all or part of a product or system that, although limited in some way, can be used for evaluation". Prototypes allow designers to communicate more effectively with users and they reduce the need and cost of reworking that can occur when products need to be revised later in the life cycle. We need to build prototypes because abstract technical specifications and models are not a good way of communicating when we want to involve users in the design process.

Prototyping, and especially rapid prototyping, is closely related to iterative design. For prototypes to be effective, they should be built at a minimal cost in terms of resources and time. The difference from traditional software engineering system prototypes is again a difference of focus. Prototypes are useful for usability purposes when they depict mostly system-user interaction, so that they convey how the system will work from the user point of view. So, prototypes can be used to try out design ideas with users and to gather their feedback [Preece, 94].

Design at every level of abstraction should be communicated by means of a prototype. From the product concept to full-detail design, prototypes of varying fidelity to the final system can be produced for use in usability evaluation activities. Therefore, the prototyping approach to interactive system development involves the production of at least one early version of the system that illustrates essential features of the later operational system. When used early in the development process, a prototype encourages user participation and involvement and allows developers to observe user behaviour and reaction to the prototype [Hix, 93].

Ensuring that a proposed system has the necessary functionality for the tasks that users want to do is an important part of requirements definition and task analysis. Prototypes are a means of achieving this, they can serve to elicit information from users about [Preece, 94]:

- The necessary functionality of the system
- Operation sequences
- User support needs
- Required representations
- Look and feel of the user interface.

With the boom in iterative development approaches, user interface prototypes are becoming more and more common in software development. Usability offers the possibility of anchoring such prototypes to strong roots of user-centredness. This is possible by basing prototypes on the design output as a the result of the above-mentioned activities: specification of context of use, develop product concept and usability specifications.

4.3.3 Interaction Design

Interaction design is the least well-defined activity, which varies considerably from one author to another. According to Hix and Hartson, design is a complex activity, and there are no generally applicable formulas for success [Hix, 93]. Design as a process is one of the least understood development activities [Hix, 93]. The common advice on design issues is to keep a user-centred focus all through the design process, and, for this purpose, the design team will base their work on the analysis of previously developed products.

Interaction design and graphical user interface design are closely related, and some authors prefer to refer to user interface design [Nielsen, 93][Hix, 93][Mayhew, 99][Shneiderman, 98], while others prefer to refer to interaction design [Preece, 94] or to design generally [Wixon, 97][ISO13407, 99]. Constantine talks about dialogue design or visual design [Constantine, 99].

The result of interaction design should be a much more detailed design than the one produced in the develop product concept activity, so we could call this activity detailed interaction design instead.

4.4 Evaluation Activities

The only evaluation activity is usability evaluation. Usability is a very complex concept, due to the complex nature of humans. Without doing some form of evaluation, it is impossible to know whether or not the design or system fulfils the needs of the users and how well it fits the physical, social and organisational context in which it will be used [Preece, 94]. No matter how much we stress the performance of user-centred activities in the development process, we will not be able to exactly predict the usability level of the system in advance. For this reason, we need to perform usability evaluation at the end of every iterative design cycle to find out where the product is in usability terms and how much improvement is needed in order to reach the previously specified usability goals.

Usability evaluation must be a central activity in the usability process. Hix and Hartson [Hix, 93] stress the importance of usability evaluation by representing it at the centre of their star life cycle (see Figure 3.2 and Figure 3.3). [Preece, 94] takes it as the reference life cycle when considering design, and six of the thirty-four chapters of the book are given over to interaction design evaluation.

Usability evaluation is needed at every stage of the development process, although the degree of formality required varies. This variation in formality depends on the design products available for evaluation at any time, and the constraints on time and resources at every stage.

Evaluation can be used [ISO13407, 99]:

a) to provide feedback which can be used to improve design,

b) to assess whether user and organisational objectives have been achieved, and

c) to monitor long-term use of the product or system.

One difference between evaluation for usability purposes and validation in traditional software engineering is that early usability evaluation must be used to drive design. This approach is known as formative evaluation [Nielsen, 93][Hix, 93], a kind of evaluation whose goal is to learn which detailed aspects of the system we are designing are good and bad, and how the design can be improved. This is opposed to summative evaluation that aims at assessing the overall quality of a design after it is complete or nearly so.

We can say that usability evaluation is a core part of the iterative design process in the sense that evaluation activities can produce design solutions for application in the next design cycle or, at least, more insight into the nature of the interaction problem at hand. Therefore, evaluation is not seen as a mere fail/pass test but as a part of development.

5. LITERATURE SURVEY ON USABILITY TECHNIQUES

5.1 Introduction

In sections 5 and 6 we will follow an equivalent reasoning to the selection of usability activities above, in order to identify a list of candidate usability techniques to be considered in a development process. So, this section presents the different usability techniques commonly applied according to literature, while section 6 provides the selection of usability techniques considered appropriate for inclusion in a development process. The remaining work to perform in this WP 5 will consist on integrating usability activities and techniques with the traditional tasks in a software development process, resulting in D.5.2 Specification of the software process with integrated usability techniques.

As this literature survey has been conducted from a software engineering perspective, we have opted to retain software development concepts such as activity and technique. Activity means a job to be done and technique means the tool to perform this job (the way it is done). That is, we make a distinction between what is to be done (activity) and how it is to be done (technique). However, these two concepts are not so clearly differentiated in the usability field, and the what and how are often confused. This is why concepts that appeared in the activities survey in section 3 are sometimes repeated in the techniques survey below. However, the approach is different in each case depending on whether they are surveyed from a technique or an activity perspective. Remember, for example, about "Know the User", mentioned in section 3.3. From a usability viewpoint, we take note that there must some point in time in the process that is given over to “know the user”. If we analyse this concept from the technique perspective, we examine how the author suggests that the user should be known, that is, what method or technique he proposes for performing this activity and achieving the desired objective of getting to know the user. Note that in this section we are focusing on techniques, that is, on how to do certain tasks.

5.2 Sources for the Survey

For the literature survey we will focus on just books. The reason for this decision is that we want to consider usability techniques that are well tested and widely used, not techniques under research.

The same sources as described in section 3.2 were chosen for this survey, except for [Wixon, 97], [ISO9241, 98], [ISO13407, 99] and [Mayhew, 99]. We have not considered these four sources, because they focus on activities to be used in a usability-oriented process and give few details on the individual techniques to be used. The relevance of the sources considered was detailed in section 3.2 above.

5.3 [Nielsen, 93]

Nielsen tried to produce a readable and thought-provoking book, and did not make a special effort to present the different usability techniques mentioned in an ordered manner. Indeed, he did not even distil a proper name and definition for some. Nevertheless, for the purpose of comparison with other authors, we will try to extract the techniques mentioned in Nielsen’s text, even though some of them are not completely defined. The techniques proposed by Nielsen are as follows:

- **Know the User - Individual User Characteristics**: This technique deals with getting access to representative users in order to gather information about their work experience, educational level, age, previous computer experience, and so on. Their work environment and social context also need to be known.
- **Task Analysis**: It involves studying the users overall goals and their current approach to the tasks, their information needs and how they deal with exceptional circumstances or emergencies. Nielsen gives alternatives for performing task analysis, like systematic observation of users talking to their clients, or interviewing users.

- **Functional Analysis**: This technique tries to decide what it is that really needs to be done, and what are merely superficial procedures that can, and perhaps should, be changed.
- **Competitive Analysis**: It is desirable to analyse existing products heuristically according to established usability guidelines and to perform empirical user tests with these products.
- **Financial Impact Analysis**: It is an analysis of the financial impact of the usability of the system. It is easier to run for in-house development or for development under contract directly from the user organisation. It involves calculating how the degree of usability improvement translates into savings in user time (employee time that costs the user organisation money) due to the improvement in user performance. Time saved for increased ease-of-learning can be calculated as well.
- **Parallel Design**: Several different designers work out preliminary designs (working independently). A variant is called Diversified Parallel Design, where each designer is asked to concentrate on different aspects of the design problem.
- **Participatory Design**: Instead of guessing, designers should have access to a pool of representative users after the start of the design phase. Users should be involved in the design process through regular meetings between designers and users.
- **TAG (Task-Action Grammars)**: They can provide a consistency metric for small systems.
- **Heuristic Evaluation**: Heuristic evaluation is done by looking at an interface and trying to come up with an opinion about what is good and bad about the interface. It is better to have several evaluators to evaluate the same design independently, as they uncover far more errors than a single evaluator. The ideal thing is to have usability specialists perform the heuristic evaluation. A particular kind of heuristic evaluation is the pluralistic walkthrough:
    - **Pluralistic Walkthrough**: Heuristic evaluation is performed by representative users, product developers, and usability specialists.
- **Prototyping**: Prototypes are reduced versions of the full system, by either cutting down on the number of features in the prototype or reducing the level of functionality of the features such that they seem to work but do not actually do anything.
- **Follow-up Studies of Installed Systems**: Studies of the use of the product in the field assess how real users use the interface for naturally occurring tasks in their real-world working environment. It can include interviews, questionnaires, observational studies and logging data from running versions of the software.
- **Usability Testing**: A usability test involves testing the current version of the system with real users. A usability test usually has four stages: preparation, introduction, the test itself and debriefing.
- **Impact Analysis**: It involves first finding the usability problems and then going back to the videotapes to investigate exactly how many users had each usability problem and how much they were delayed by each problem. They can be used to prioritise the fixing of the usability problems in a redesign.
- **Thinking Aloud**: Nielsen distinguishes thinking aloud from other usability testing techniques by pointing out that it may be the single most valuable usability engineering method. A thinking-aloud test involves having a test subject use the system while continuously thinking out loud. Its strength is on qualitative data and not on performance measures. The idea is to get the user's impression while using the system to avoid later rationalisations. There are several approaches to the thinking aloud technique:
    - **Constructive Interaction**: It involves having two test users use a system together. It is also called Codiscovery Learning. It is based on the fact that people are used to verbalising when they are trying to solve a problem together.

  - **Retrospective Testing**: The usability testing session is recorded on a videotape and the user is requested to review the recording. User comments while reviewing the tape are sometimes more extensive than comments while performing the task in the test. The reviewer can stop the tape and ask the user questions at any time, without fear of interfering with the test, which has essentially already been completed.
  - **Coaching Method**: The experimenter (or "coach") steers the user in the right direction while using the system. The user can ask the experimenter questions, and the questions may show up usability problems that would remain uncovered otherwise. The experimenter will answer to the best of his or her ability.

- **Usability Assessment through Observation**: Observation involves visiting one or more users and taking notes and maybe videotaping the user activities. But the observer should not interfere with the user's work, all the observer task must be done unobtrusively.
- **Questionnaires and Interviews**: They are indirect methods of study of the user interface, because they provide the development team with the users' opinions, but no direct information on the user interface. They are especially suited for getting the users' subjective satisfaction. Questionnaires can be administered by mail, e-mail or with the software itself. Interviews may be conducted personally or over the phone.
- **Focus Groups**: In a focus group, about six to nine users are brought together to discuss new concepts and identify issues over a period of about two hours. Each group is run by a moderator who is responsible for maintaining the focus of the group on whatever issues are of interest.
- **Logging Actual Use**: Logging involves having the computer automatically collect statistics about the detailed use of the system. It is usually a way of getting information about the field use of a system after release, but it can be used as a supplementary method in usability tests.
- **User Feedback**: Feedback can be collected by giving users access to special electronic mail addresses, network newsgroups, or bulletin boards. Users can send their complaints and requests for change or improvement.

5.4 [Preece, 94]

Preece et al. offer a very good overview of the HCI field, giving details on a great number of techniques, thirty-eight, of which thirty are different from those proposed by the previous author. These are their proposed techniques:

- **Participative Design**: Users participate by analysing organisational requirements and planning appropriate social and technical structures to support both individual and organisational needs.
- **Sociotechnical Approach**: This is a form of cooperative design that focuses on developing complete and coherent human-machine systems. The emphasis of this approach is on considering social and technical alternatives to problems, for example, the OSTA (Open Systems Task Analysis) method.
- **HTA (Hierarchical Task Analysis)**: The basic construct of HTA is a graphical representation of a decomposition of a high level task into constituent subtasks and operations, or actions. It is based on the structured chart notation and is concerned with establishing an accurate description of the steps that are required in order to complete a task.
- **Cognitive Task Analysis**: The aim of cognitive task analysis is to capture some representation of the knowledge that people have or need to have in order to complete a task. Cognitive theory is applied for this purpose. Rather than a technique, it is a set of techniques: GOMS, TAG, ETIT, YSS, CLG and KAT/TKS.

  - **GOMS (Goals, Operations, Methods and Selection Rules)**: GOMS can be used to analyse human-computer interaction at different levels. Goals are decomposed into operations, which are decomposed into methods, and, at the maximum of the degree of detail, there are the selection rules. It concentrates on the user is "how-to-do-it" knowledge.

- **Visual Brainstorming**: This is a sketching technique employed for exploring alternative designs. After producing initial sketches, the best ideas can be further developed by constructing cardboard representations of the design, which can be evaluated with users. This can then be followed by developing scenarios, software or video prototypes.
- **Design Rationale Techniques**: There are several techniques for capturing the design decisions so that they can be reviewed at future meetings:
  - **IBIS (Issue-Based Information Systems)**: IBIS aims to capture design decisions as the design progresses. The central activity of IBIS is deliberation, that is, considering the pros and cons of alternative answers to questions. The notation used is a graphical diagram called an issue map, which illustrates the issues and their relationships.
  - **PHI (Procedural Hierarchy of Issues)**: PHI is a hypertext version of IBIS that addresses the main problems identified in IBIS. The notions of issue and issue resolution are extended in PHI, while it restricts the order in which issues are identified.
  - **Design Space Analysis**: Design is viewed in this technique as an exploration of a space of alternatives. The designer is actively encouraged to explore alternative designs. Therefore, the final result is not just a detailed description of the "whys" and "wherefores" of a design, but also better quality designs, since the designer will have explored many more alternatives.
  - **Claims Analysis**: It is based in the psychological claims that the designer is making or has made with regard to the use of a system, the user of the system, the environment of use and so on. A claim relates some aspect of a system design to an important consequence for the user. Claims analysis is done by creating scenarios of system use and analysing them for claims. Core tasks that the system is intended to support and key errors that the system must be able to handle should be included. The primary purpose is to identify how the system positively supports the user, but trade-offs can be also identified.
- **Prototyping Techniques**: There are several aspects of the system on which the prototype may focus:
  - **Requirements Animation**: Possible requirements are demonstrated in a prototype, which can then be assessed by users.
  - **Rapid Prototyping**: It is analogous to throwaway prototypes.
  - **Incremental Prototyping**: Large systems are installed in phases to avoid delays between specification and delivery.
  - **Evolutionary Prototyping**: The initial prototype is constructed, evaluated and evolved continually until if forms the final system.
  - **Chauffeured Prototyping**: The user watches while another person, usually a member of the development team, "drives" the system.
  - **Wizard of Oz**: The user interacts with a screen, but instead of a piece of software responding to the user actions, a developer is sitting at another screen answering the user requests. The user is unaware of the fact that the answers are being given by a person instead of a software system.

- **Direct Observation**: Individual users may be directly observed doing specially devised tasks or doing their normal work, with the observer making notes about interesting behaviour or recording their performance in some way, such as by timing sequences of actions.
- **Indirect Observation - Video Recording**: Video logging provides an alternative to direct observation, which is much preferred because it provides a permanent record to which you can return as often as necessary later. Analysing video data can be very time consuming. A ratio of 5:1 is often cited: one hour of videotape could take five hours or even a day or more to analyse.
- **Verbal Protocol**: Verbal protocol means some form of audio record in a observation session. It adds an extra dimension to the information gathered by addressing the cognitive activity underlying the user's physical behaviour. A particular technique of verbal protocol is the think aloud protocol:
    - **Think Aloud Protocol**: As mentioned in Nielsen's proposed techniques, it is a special case of verbal protocol in which the user says out loud what he or she is thinking while he or she is carrying out a task or doing some problem solving.
- **Post-Event Protocol**: With this technique, users view videos of their actions and provide a commentary on what they were trying to do. It should be noted that post-event protocols can contain recalled information that was not used during the tasks sequence: hindsight can produce a rationalisation of the user's own actions.
- **Software Logging**: It has two main advantages: it does not require the researcher to be present, and it is unobtrusive. There are two kinds of logging, which can be combined with each other and also with video recording. They are as follows:
    - **Time-Stamped Keypresses**: A record of each key pressed is kept, along with the exact time of the event.
    - **Interaction Logging**: The whole interaction is recorded, so it can be reproduced completely in real-time.
- **Interviews**: There are two main kinds of interviews:
    - **Structured Interviews**: The interview has a fixed structure, and there is no exploration of individual attitudes.
    - **Flexible Interviews**: They generally have some set topics, but no set sequence, and the interviewer is free to follow the interviewees' replies and to find out about personal attitudes.
- **Questionnaires and Surveys**: There are two types of questions: closed (the respondent is requested to select an answer from a choice of alternative replies) and open (the respondent is free to provide his or her own answer). Closed questions usually have some form of rating scale. Three of these scales are: a multi-point rating scale, the Likert scale and the semantic differential. A questionnaire is sometimes used before and after studies of user performance. These are known as pre- and post-questionnaires.
- **Traditional Experiments**: This kind of experimental studies are traditional psychological experiments performed to address specific aspects of human-computer interface design. The investigator can manipulate a number of factors associated with the design and study the effect on various aspects of user performance. Well-designed experimental studies usually have a clear hypothesis and they conclude with a statistical analysis of the data collected.
- **Usability Engineering**: The authors present usability engineering as yet another "method" in human-computer interaction, defined as a process whereby the usability of a product is specified quantitatively and in advance. Then, as the product is built, it can be demonstrated

that it does or does not reach the required levels of usability. It comprises the following techniques:

  - **Benchmark Tasks**: Tests in a usability laboratory with video recording and usually with logging of keystrokes as well.
  - **Impact Analysis**: This technique involves listing the usability attributes alongside the proposed design decisions. Next, the percentage impact of each design solution is estimated for each of the attributes and entered into the appropriate part of the table. It is slightly different from the impact analysis proposed by Nielsen.

- **Contextual Inquiry**: It is a form of elicitation that can be used in evaluation. Users and researchers participate to identify and understand usability problems within the normal working environment of the user.
- **Cooperative Evaluation**: Users are involved in deciding both what the evaluation issues and techniques for collecting and analysing video protocols are. Cooperative evaluation is a technique designed to be low cost, and it can be used by designers and users without specialist HCI knowledge.
- **Participative Evaluation**: It is like cooperative evaluation but more open and subject to greater control by users. It is an extension of participative design.
- **Ethnography**: It is a traditional method used in anthropology. Ethnographic researchers strive to immerse themselves in the situation that they want to learn about.
- **Inspection Methods**: Specialists who have knowledge of both the technology and the intended users inspect the system. Usually, the emphasis is on inspecting the user-system interaction dialogue. Depending on the focus given to the inspection there are several variants of this technique:
  - **Standards Inspection**: Inspections for compliance with standards.
  - **Consistency Inspections**: Teams of designers, at least one from each project, who meet to inspect a set of interfaces for a family of products.
- **Structured Expert Reviewing**: The authors make a distinction between inspection methods and this kind of reviewing, which is performed with predefined tasks and following a plan. The techniques belonging to this category are as follows:
  - **Heuristic Evaluation**: The authors refer to the same technique as described by Nielsen.
  - **Walkthroughs**: The goal of a walkthrough in HCI is to detect problems very early on so that they can be removed. Walkthroughs involve constructing carefully defined tasks from a system specification or screen mock-up. There are two specific variants or walkthroughs:
    - **Cognitive Walkthrough**: It is a manual simulation of the cognitive activities of the user to identify potential usability problems.
    - **Pluralistic Walkthroughs**: This is the same technique as described by Nielsen.
- **Analytic Evaluation Methods**: These methods can be used to calculate task performance times that can be achieved by experienced users, by extracting information from the specification of the system functionality to be examined and a task analysis where tasks are broken down into components. It can give an idea of the minimum performance time for specified commands.

5.5 [Hix, 93]

Hix and Hartson focus on two main issues in their work: the UAN (User Action Notation) representation technique, and the establishment of usability specifications to be evaluated in usability tests. The other techniques mentioned in this source are not explained in the same amount of detail.

The techniques that Hix and Hartson propose are as follows:

- **Needs analysis**: It establishes that a new system is actually needed, based on the goals of the organisation and demands of the marketplace.
- **User Profiles**: It involves defining representative classes of users in terms of the tasks to be performed and the skills and knowledge those users bring to the tasks.
- **Functional Analysis**: It requires description of the internal system functions that must be designed and implemented in the non-interface software to support the tasks identified for the user.
- **Conceptual design**: User interaction design is divided into two kinds: conceptual design and detailed design. Conceptual design is at a higher level and has to do with synthesising objects and operations. It describes how the operations will be invoked, and how the user will gain access to the objects.
- **Screen Pictures**: This technique involves sketching some initial pictures of the screen, including the application/interaction objects, menus, buttons, and icons. The functions are labelled, and notes can be added about the behaviour of objects where appropriate.
- **UAN (User Action Notation)**: UAN is a technique for representing user interaction designs from the behavioural view of the user. The UAN addresses the creative mental act of problem solving (i.e., creating new designs) and the physical act of documenting a representation of the design. The primary abstraction in UAN is the user task. An interface is represented as a quasi-hierarchical structure of asynchronous tasks, the sequencing within each task being independent of that in the others. User actions, corresponding interface feedback, and state change information are represented at the lowest level.
- **Interface State Transition Diagrams**: These diagrams complement the UAN and they indicate sequencing and state or mode information.
- **Usability Specifications**: They are quantitative usability goals, which are used as a guide for knowing when an interface is good enough. They can be based on objective or subjective measures. Objective measures are commonly associated with a specific **benchmark task**, while subjective measures are commonly associated with a **user questionnaire**.
- **Rapid Prototyping**: The prototyping approach to interactive system development involves production of at least one early version of the system that illustrates essential features of the later, operational system. With rapid prototyping, the process of constructing prototypes is accelerated, so that the time from beginning a prototype to evaluating user interaction with it is short enough to leave time for substantial changes, if needed, to the product.
- **Heuristic Evaluation**: These authors refer to the same technique described above in previous sections.
- **Qualitative Data Generation Techniques**: These techniques are to be performed in usability tests, and they are extremely important when the aim is to perform formative evaluation. They are as follows:
    - **Concurrent Verbal Protocol Taking**: This is the same thinking aloud technique as defined by Nielsen and Preece.
    - **Retrospective Verbal Protocol Taking**: The evaluator lets participants work relatively uninterruptedly during a taped session. Then, immediately after the session,

the evaluator and each participant review the videotape together, and the evaluator asks the participant to analyse what was occurring during the session.

- **Critical Incident Taking**: This technique implies recording both negative incidents (signs of frustration, either with remarks or actions), and positive incidents (satisfaction or closure expressions). Negative incidents help to identify the more important usability problems, while positive incidents help to identify metaphors or details to be used more thoroughly in the user interface because of their success.
- **Structured Interviews**: In the form of a post-experiment interview, the evaluator asks each participant a series of pre-planned questions

- **Data Collection Techniques**: Techniques for collecting data in usability tests or user observation of any kind. They include the following techniques:
  - **Real-Time Note Taking**: The most basic technique is when the evaluator is prepared to take copious notes as activities proceed during a session. It can barely be considered a technique.
  - **Videotaping**.
  - **Audiotaping**.
  - **Internal Instrumentation of the Interface**: This technique is the same as software logging described by previous authors.
- **Cost/Importance Analysis**: This technique is very similar to Preece's impact analysis, as it involves considering the relative importance of the usability problems identified in usability evaluation, along with the development cost of the possible solutions.

5.6 [Shneiderman, 98]

Shneiderman presents an overview of all usability and user-interface related issues. However, his approach differs from the one taken by Preece et al. in that he does not intend to be extensive. Shneiderman does not detail each technique in the field, but mostly describes a selection of techniques he finds useful. For most techniques, he just provides a general description and some references. The only techniques presented in more detail are user interaction specification techniques and for the ones for evaluation during active use.

The techniques proposed by Shneiderman are as follows:

- **Usage Profiles**: These profiles characterise the users in several combinations of knowledge and usage patterns.
- **TAG (Task-Action Grammars)**: They are presented by the author as an example of a notational structure to evaluate and promote consistency in the user interface. TAG's can help the design team to think about consistency, and it can sharpen their thinking on the issue.
- **The Object-Action Interface Model**: This is Shneiderman's own model. It is an explanatory model that focuses on task objects and actions and on interface objects and actions. Both task and interface objects and actions are decomposed in a hierarchical manner. The underlying idea is to anchor interface objects and actions to task objects and actions, which are already known by the user. This kind of model is especially applicable for directing manipulation interfaces, which have replaced older command languages and form-filling interfaces.
- **Ethnographic Observation**: Since interface users form a unique culture, ethnographic methods for observing them in the workplace are likely to become increasingly important. As ethnographers, user-interface designers gain insight into individual behaviour and the organisational context.

- **Scenario Development**: For systems that will suffer substantial changes (as in business-process re-engineering) or when a novel application is planned, there are usually no reliable data on the range and distribution of task frequencies and sequences. An early and easy way to describe a novel system is to write scenarios of usage and then, if possible, to act them out as a form of theatre.
- **Expert Reviews**: Reviewers are experts in the application or user-interface domains. Depending on the review focus, there are different kinds of reviews:
    - **Heuristic Evaluation**: As detailed above for previous authors, the expert reviewers criticise an interface to determine conformance with a short list of design heuristics.
    - **Guidelines Review**: The interface is checked for conformance with the organisational or other guidelines document.
    - **Consistency Inspection**: The experts verify consistency across a family of interfaces, checking for consistency of terminology, colour, layout, input and output formats, and so on.
    - **Cognitive Walkthrough**: The experts hold a courtroom-style meeting, with a moderator or judge, to present the interface and to discuss its merits and weaknesses.
- **Usability Testing and Laboratories**: Usability tests have fewer subjects than controlled experiments, and the outcome is a report with recommended changes, as opposed to validation or rejection of hypotheses.
- **Surveys**: Written user surveys are a familiar, inexpensive and generally acceptable companion for usability tests and expert reviews.
- **Acceptance Tests**: For large implementation projects, the customer or manager usually sets objective and measurable goals for hardware and software performance. These notions can be neatly extended to the user interface. Explicit acceptance criteria should be established when the requirements document is written or when a contract is offered.
- **Evaluation during Active Use**: Systems refinements are based on the results of the following usability evaluation techniques during the system active use:
    - **Interviews and Focus-Group Discussions**: Interviews are usually with individual users. After a series of individual discussions, focus-group discussions are valuable to ascertain the universality of comments.
    - **Continuous User-Performance Data Logging**: The software architecture should make it easy for system managers to collect data about the patterns of system usage, speed of user performance, rate of errors or frequency of requests for online assistance.
    - **Online or Telephone Consultants**: Consultants are an excellent source of information about problems users are having and can suggest improvements and potential extensions.
    - **Online Suggestion Box or Trouble Reporting**: Electronic mail can be employed to allow users to send messages to the maintainers or designers. Such an "online suggestion box" encourages some users to make productive comments, since writing a letter may be seen as requiring too much effort.
    - **Online Bulletin Board or Newsgroup**: Users may have questions on the usage or applicability of a software package, and they cannot address anyone in particular. Then bulletin boards and newsgroups can be helpful.
    - **User Newsletters and Conferences**: In systems with a substantial number of users who are geographically dispersed, managers have to work harder to create a sense of

community. Printed newsletters have an appealing air of respectability. Conferences allow workers to exchange experiences with colleagues and face-to-face meetings increase the sense of community among users.

- **Controlled Psychologically Oriented Experiments**: Academic and industrial researchers are discovering that the power of the traditional scientific method can be fruitfully employed in the study of interfaces.
- **Specification Methods**: The author presents several specification methods, according to the characteristics of the user interface that they specify:
    - **Grammars**: Command languages are well specified by grammars.
    - **Menu-selection and dialog box trees**: For many applications a menu-selection tree is an excellent selection style because of the simple structure that guides designers and users alike.
    - **Transition diagrams and statecharts**: Menu trees are incomplete because they do not show the entire structure of possible user actions. Transition diagrams allow for a more precise specification of every possible transition. Statecharts are an extension of state transition diagrams that allow sub-graphs to be represented inside a node. They represent concepts like concurrency and synchronisation better than transition diagrams.
    - **UAN (User Action Notation)**: Shneiderman agrees with [Hix, 93] on using this technique.

5.7 [Constantine, 99]

Of the sources considered, [Constantine, 99] is the one that differs most in the sense that their proposal does not follow the most commonly accepted framework in the field of human-computer interaction and, in particular, usability. Constantine and Lockwood propose a method of their own. Even though their method employs some basic usability techniques, most of the models proposed are adaptations of models used in the field. Models are the line around which the book is presented, so techniques are somehow spread throughout the whole work and are difficult to extract from the context of the method and the particular models used.

The techniques proposed by Constantine and Lockwood are as follows:

- **Structured User Role Model**: A role model is a list of the user roles to be supported by a system, which describes each role in terms of the needs, interests, expectations, behaviours and responsibilities that characterise and distinguish that role. Some roles are highlighted as focal roles, and are the ones that are judged to be the most common or typical or that are deemed particularly important from a business perspective or from the standpoint of risk or some other technical content. User roles and their relationships are represented in a user role map. These models can be difficult to define, especially for the inexperienced analyst and for complex systems. The structured user role model offers a methodical way for capturing as much relevant information as possible about the relationships of users to the planned system.
- **Task Modelling - Use Cases**: The authors employ use cases as their preferred way of performing task modelling. A use case is a case of use or one kind of use to which a system can be put. Other explanations of a use case include: supplied functionality, an external, “black box” view, a narrative description, interaction between a user -in some user role- and a system and a use of a system that is complete and meaningful to the user. Use cases are described in a structured manner, where the narrative is divided into two parts: the user action model and the system response model. There is an abstract way of expressing use cases as follows:

  - **Essential Use Cases**: At a higher level of abstraction, use cases are defined in terms of user intentions and system responsibilities, keeping a technology-free and implementation-independent focus. They can be used to work with use cases at the beginning of the development process, without having to take too many decisions on the details of the user interface. The use case map partitions the total functionality of the system into a collection of interrelated essential use cases. Please note that essential refers to the abstract focus used for the use case description, not to specific user interface details, and it is applicable to all use cases, it does not refer to a particular set of especially important use cases.

- **Post-It Notes and Context Navigation Map**: The typical process of content modelling begins with examining the validated use case narratives line by line to determine, for each use case, what tools and materials will have to be supplied within a given interaction space in order for the user to enact the use case. As each required tool or material is identified, a Post-It placeholder is added to the interaction space. There can be several of these interaction spaces or interaction contexts. Their relationships and the way of navigating between them are represented in a context navigation map.
- **Both-And Design**: This is a technique for creative interface engineering. Both-and thinking looks for a creative synthesis of apparently opposing ideas or seemingly exclusive alternatives. Rather than compromise, it seeks to incorporate the best of both worlds and to satisfy conflicting goals simultaneously. The authors describe a process to help to find both-and design solutions.
- **Prototyping**: There are several variants of prototypes, depending on: action (passive vs. active), fidelity (high vs. low-fidelity) and orientation (horizontal vs. vertical). Another characterisation of prototypes focuses on their active or passive nature:
  - **Passive Prototypes**: They include paper prototypes, computer drawings prepared with graphics software, and non-functioning mock-ups created using programming tools.
  - **Active Prototypes**: They are also called functional prototypes or working prototypes. They can be software mock-ups, simulations or limited implementations. The term mock-up usually refers to a prototype of limited fidelity and performance constructed in a software medium other than a full-blown programming language.
- **Help Design Techniques**: A technique based on use cases is described for designing the help subsystem:
  - **Organising Help by Use Cases**: If essential use cases have been well constructed, then they will reflect how users think and conduct their work. Each use case becomes an entry in the help file.
- **Operational Modelling**: The operational model is a collection of various operational and contextual influences that can play a role in usability. These collections are referred to as profiles. Operational factors that can affect user interface architecture and detail design include: characteristics of users and user roles, aspects of the physical work environment, features and limitations of operating equipment and interface devices, and general and specific operational risk factors.
- **Expert Evaluation**: This is a subjective assessment of the usability of a product based on the experience and judgment of an expert or experts in usability.
- **Inspections and Walk-Throughs**: Inspections refer to any of several forms of more or less formal, systematic processes for locating usability problems. Usability inspections employ developers and/or usability specialists, sometimes in conjunction with users, to identify usability defects. The authors present the following variants of these techniques:
  - **Heuristic Evaluation**: Again Constantine proposes this technique for inspections.

  - **Cognitive Walk-Throughs**: In a cognitive walk-through, the group steps slowly through a task scenario, conducting a detailed analysis of the user's intent, knowledge, thought processes, and interpretations for every action. The focus is on one single usability attribute: learnability.
  - **Pluralistic Walk-Throughs**: This is a collaborative process involving end users, developers and usability experts, with all participants expected to play the role of users. Its goal is "coordinated empathies".
  - **Collaborative Usability Inspections**: It is a systematic examination of a finished product, a design, or a prototype from the standpoint of its ultimate usability by intended users. The review process is a team effort that includes software developers, end users, application or domain experts and usability specialists, collaborating to perform a thorough and efficient inspection. Some variations of a collaborative usability inspection are called focused inspections. They can be of two kinds:
    - **Consistency Inspections**: In consistency inspections, the goal is to identify inconsistencies across interaction contexts and their contents.
    - **Conformance Inspections**: In conformance inspections, the goal is to identify departures from the governing user interface standards or style guidelines. All participants must be familiar with the applicable standards and style guidelines, and users are not normally included.

- **Usability Assessment Based on Predictive Metrics**: By distilling information from design artefacts, such as visual designs for screen layouts, and applying predictive metrics, some estimators are obtained. These estimators are predictors of one or more aspects of the actual performance that can be expected once a system has been implemented and put to use.
- **Laboratory Usability Testing**: Laboratory testing involves tests conducted in a fixed setting specifically configured for usability testing. The main advantage of this kind of usability testing is that it provides a controlled and consistent environment in which to evaluate software. Comparing the results of different tests, different users or different systems is easier and more defensible under these conditions. Although several of the techniques described below are for usability testing generally, either laboratory testing or in the field, we will describe them at this point:
  - **Talk to me (Think Out Loud)**: The usage of these technique is also considered by the authors.
  - **Post-Test Feedback**: Most testing plans include a debriefing or post-test interview with each subject. Subjects are typically thanked for their participation and reassured about their performance.
  - **Measured Performance**: They quantify and summarise important aspects of actual usage either under controlled laboratory conditions or within an ordinary work environment.
- **Field Usability Testing**: Field testing takes the usability tests into the workplace. It is performed at the user's organisation, although not necessarily at the user's workstation and office. There is a particular variant of this technique:
  - **Beta-Testing**: Some companies consider beta-testing as a form of field evaluation of software usability. Alpha releases are typically early versions of software intended for internal release. Although it may be less robust and efficient than the final shipping version, a beta release is assumed to be essentially complete and fully functional.
- **Concentric Construction**: This approach to software construction is based on essential use cases. They become the appropriate unit of product delivery because each use case represents one useful piece of work, one meaningful task to some users. Phasing the construction and

delivery of systems based on use cases and collections of related use cases assures that users receive the most useful collection of features and capabilities with each release. This approach is the same as incremental software development processes.

- **Architectural Iteration**: The basic internal architecture of the system, the core assumptions around which the program is organised, may change as the application grows in each iterative refinement cycle. The architecture then needs to be repeatedly refined, along with the code that is based upon it. That is, in each round of refinements, the basic architecture of the software is reviewed to ascertain its continued viability.
- **User Participation in Usage-Centred Design**: Participation of users in the activities of the Usage-Centred Design methodology are limited to where they can act as experts, experts on their work and on their own system usage. Users can be intensively involved in the collaborative requirements dialog, task modelling, domain modelling, standards and style definition and usability inspection. A particular technique for user participation is JEM:
    - **JEM (Joint Essential Modelling)**: It is a structured process for collaborating with users to develop usage-centred requirements specifications through concurrent modelling. It resembles somewhat its ancestor JAD (Joint Application Design). The activities performed in JEM are as follows:
        1. Pre-modelling and consolidation
        2. Role modelling
        3. Task modelling
        4. Model auditing
        5. Feature allocation

6. SELECTION OF CANDIDATE USABILITY TECHNIQUES

As a basis for technique selection, we will classify and summarise the techniques analysed in the previous section before going on to actual technique selection.

6.1 Summary of Reviewed Techniques

After merging the techniques suggested by different authors that refer to the same basic technique, we still have eighty-two techniques. For the purpose of selection, the techniques have been divided into groups, where the most abstract level of classification is the principal task performed by the technique, taken from the three basic types of tasks in software development: analysis, design and evaluation (note that a similar approach was followed for the classification of usability activities in section 4 above).

We have represented the techniques in several tables: two tables, respectively, for analysis-related techniques (Table 6.1 and Table 6.2) and design-related techniques (Table 6.4 and Table 6.3) and four tables for evaluation-related techniques (Table 6.5, Table 6.6, Table 6.7 and Table 6.8). Each table contains a column for each author, and the left-hand columns specify the technique category and chosen name. For each technique, we have chosen the name we consider to be the most representative. Techniques that are in the same row refer to the same basic technique. The possibility of variants of some techniques is also considered; and then the general technique and each of the variants have their corresponding rows (for example, Table 6.6 shows the Thinking Aloud technique along with four of its variants).

The tables also show the selected as candidate techniques. Techniques on a white background are selected as candidates for inclusion in the software development process. Techniques that have not been selected appear on a grey background. The lighter grey background is for techniques that have been selected, albeit for optional application when the project meets certain characteristics.

Although the reasons for discarding a priori the corresponding techniques are discussed in detail in section 6.3, a technique can be discarded due to one or more of the following reasons:

- It is a special technique for projects with specific characteristics that are not generally applicable.
- It is alien to software engineering, so the developer will find it very difficult to learn to use it.
- Its application will require the use of extra resources from outside the project team.
- It is made redundant by another selected technique. That is, the expected benefits provided by the application of the technique are already covered by a selected technique, and the selected technique offers some additional advantages
- It is not a usability technique. It is a software engineering technique, and it does not make sense to define it as a usability addition to the development process. These techniques rows are highlighted in the tables by means of italics.
- It deals with development process issues, and there are other reasons apart from usability to be taken into account. It must be dealt with in the context of the whole development process.
- The technique is directed at gaining support for usability activities in the development process. This kind of support is a pre-requisite for our process.
- It is presented by just one author, and we consider that it is not generally accepted as a usability technique in the field. This reason will be considered only in conjunction with other reasons, never by itself.

As shown in Table 6.1, analysis techniques have been classified into two main groups: elicitation and analysis techniques, and modelling techniques. Analysis and elicitation techniques refer to how the

information required is obtained and analysed, while modelling techniques serve to represent the knowledge gathered by the former techniques.

Table 6.2 shows usability specification techniques, which are applied at the same time as analysis activities are performed. Even though they are more related to evaluation activities, we have decided to present them along with analysis techniques, as they are usually integrated with analysis tasks.

Analysis-Related Techniques			**Nielsen93**	**Preece94**	**Hix93**	**Shneiderman98**	**Constantine99**
Elicitation and Analysis	Cognitive Task Analysis			Cognitive Task Analysis			
	Functional Analysis		*Functional Analysis*		*Functional Analysis*		
	Needs Analysis				*Needs Analysis*		
	Competitive Analysis		Competitive Analysis				
	Financial Impact Analysis		Financial Impact Analysis				
	Scenarios		Scenarios			Scenario Development	
	Contextual Inquiry			Contextual Inquiry	Contextual Inquiry		
	Ethnographic Observation			Ethnography		Ethnographic Observation	
	Sociotechnical Approach			Sociotechnical Approach			
Modelling	User and User Environment Modelling	Structured User Role Model					Structured User Role Model
		User Profiles	Individual User Characteristics		User Profiles	Usage Profiles	
		Operational Modelling					Operational Modelling
	Task Modelling	Essential Use Cases					Essential Use Cases
		HTA		HTA			
	Cognitive Modelling	GOMS	GOMS	GOMS		GOMS	
		TAG	TAG			TAG	
		Object-action Interface Model				Object-action Interface Model	

Table 6.1 Analysis-Related Techniques

Usability Specification Techniques		Nielsen93	Preece94	Hix93	Shneiderman98	Constantine99
Usability Specifications	Based on Benchmark Tasks		Benchmark Tasks	Benchmark Tasks		
	Based on Preference Questionnaires			User Questionnaires		

Table 6.2 Usability Specifications to be established in Parallel with Analysis Activities

Design-related techniques are summarised in two tables. Prototyping and its different techniques are presented in Table 6.3. Prototyping strategies in the development cycle are first summarised, and the different kinds of prototypes that can be built are presented underneath. The other design techniques are shown in Table 6.4, where they are classified according to the following categories: interaction modelling, design alternatives management, specification techniques, techniques for user involvement in design, help design and design rationale.

Prototyping Techniques			Nielsen93	Preece94	Hix93	Shneiderman98	Constantine99
Prototyping Strategies	Rapid Prototyping			Rapid Prototyping	Rapid Prototyping		
	Incremental Prototyping			Incremental Prototyping			
	Evolutionary Prototyping			Evolutionary Prototyping			
Kinds of Prototypes	Requirements Animation	Mock-ups (Limited Implementation)					Active Prototypes
	Non-functioning Prototypes	Chauffeured Prototypes		Chauffeured Prototyping			
		Paper Prototypes					Passive Prototypes
		Wizard of Oz		Wizard of Oz			

Table 6.3 Prototyping Techniques

Design-Related Techniques			Nielsen93	Preece94	Hix93	Shneiderman98	Constantine99
Interaction Modelling	Early Design	Conceptual Design			Conceptual Design		
		Post-It Notes					Post-It Notes
	Detailed Design	Screen Pictures			Screen Pictures		
		Use Cases					Use Cases
Design Alternatives Management	Both-And Design						Both-And Design
	Parallel Design		Parallel Design				
	Impact Analysis		Impact Analysis	Impact Analysis	Cost/Importance Analysis		
Specification Techniques	Grammars					Grammars	
	Menu-Selection and Dialog Box Trees					Menu-Selection and Dialog Box Trees	
	Transition Diagrams				Interface State Transition Diagrams	Transition Diagrams and Statecharts	Context Navigation Map
	UAN				UAN	UAN	
User Involvement	JEM						JEM
	Visual Brainstorming			Visual Brainstorming			
Help Design	Organising Help by Use Cases						Organising Help by Use Cases
Design Rationale	IBIS			IBIS			
	PHI			PHI			
	Design Space Analysis			Design Space Analysis			
	Claims Analysis			Claims Analysis			
Concentric Construction							*Concentric Construction*
Architectural Iteration							*Architectural Iteration*

Table 6.4 Design-Related Techniques

Evaluation techniques are summarised in four tables: Table 6.5 presents techniques for expert reviews, Table 6.6 for usability testing, Table 6.7 for follow-up studies of installed systems and, finally, Table 6.8 summarises the other usability evaluation techniques.

Expert Review Techniques			Nielsen93	Preece94	Hix93	Shneiderman98	Constantine99
Expert Reviews	Heuristic Evaluation		Heuristic Evaluation	Heuristic Evaluation	Heuristic Evaluation	Heuristic Evaluation	Heuristic Evaluation
	Inspections	Conformance Inspections		Standards Inspection		Guidelines Review	Conformance Inspections
		Consistency Inspection		Consistency Inspection		Consistency Inspection	Consistency Inspection
		Collaborative Usability Inspections					Collaborative Usability Inspections
	Walkthroughs	Pluralistic Walkthrough	Pluralistic Walkthrough	Pluralistic Walkthrough			Pluralistic Usability Walkthrough
		Cognitive Walkthrough		Cognitive Walkthrough		Cognitive Walkthrough	Cognitive Walkthrough

Table 6.5 Usability Evaluation Techniques for Expert Reviews

Usability Testing Techniques				Nielsen93	Preece94	Hix93	Shneiderman98	Constantine99
Usability Tests	Thinking Aloud			Thinking Aloud	Think Aloud Protocol	Concurrent Verbal Protocol Taking		Talk to Me (think out loud)
		Constructive Interaction		Constructive Interaction				
		Retrospective Testing		Retrospective Testing	Post-Event Protocol	Retrospective Verbal Protocol Taking		
		Critical Incident Taking				Critical Incident Taking		
		Coaching Method		Coaching Method				
	Measured Performance							Measured Performance
	Post-Test Feedback							Post-Test Feedback
	Laboratory Usability Testing						Usability Testing and Laboratories	Laboratory Usability Testing
	Field Usability Testing	Direct Observation		Usability Assessment through Observation	Direct Observation			
		Beta-Testing						Beta-Testing
		Indirect Observation	Video Recording		Video Recording	Videotaping		
			Verbal Protocol		Verbal Protocol	Audiotaping		

Table 6.6 Usability Evaluation Techniques for Usability Testing

Techniques for Follow-Up Studies			Nielsen93	Preece94	Hix93	Shneiderman98	Constantine99
Follow-up Studies of Installed Systems	Questionnaires		Questionnaires and Interviews	Questionnaires and Surveys			
	Interviews		Questionnaires and Interviews	Interviews		Interviews and Focus Group Discussions	
		Structured Interviews		Structured Interviews	Structured Interviews		
		Flexible Interviews		Flexible Interviews			
	Focus Groups		Focus Groups			Interviews and Focus Group Discussions	
	Logging Actual Use		Logging Actual Use	Software Logging	Internal Instrumentation of the Interface	Continuous User-Performance Data Logging	
		Time-Stamped Keypresses		Time-Stamped Keypresses			
		Interaction Logging		Interaction Logging			
	User Feedback		User Feedback			Online Suggestion Box or Trouble Reporting	
		Online or Telephone Consultants				Online or Telephone Consultants	
		Online Bulletin Board or Newsgroups				Online Bulletin Board or Newsgroups	
		User Newsletters and Conferences				User Newsletters and Conferences	
	Surveys			Questionnaires and Surveys		Surveys	

Table 6.7 Usability Evaluation Techniques for Follow-up Studies of Installed Systems

Other Usability Evaluation Techniques	Nielsen93	Preece94	Hix93	Shneiderman98	Constantine99
Experimental Tests		Traditional Experiments		Controlled Psychologically Oriented Experiments	
Predictive Metrics		Analytic Evaluation Methods			Usability Assessment Based on Predictive Metrics
Acceptance Tests				*Acceptance Tests*	
Cooperative Evaluation		Cooperative Evaluation			
Participative Evaluation		Participative Evaluation			

Table 6.8 Other Usability Evaluation Techniques

6.2 Description of Candidate Techniques

Of the host of techniques presented in section 6, we will detail, in this section, which ones are adequate for inclusion a priori in a software development process and which ones are not. As the techniques are classified, we will sometimes refer to a whole category of techniques when the difference between the individual techniques is not relevant for our purpose.

For our description of selected techniques, we will follow the classification schema described in section 6.1 above.

6.2.1 Analysis-Related Techniques

6.2.1.1 Elicitation and Analysis Techniques

There is a wealth of elicitation and analysis techniques in the usability field. We will select the techniques we consider to be most valuable to complement the elicitation and analysis techniques that are currently applied in software development.

6.2.1.1.1 Cognitive Task Analysis

We select this technique because it adds the study of the cognitive operations that the user performs to the traditional analysis activities. It will need the usage of a cognitive modelling technique, like GOMS, in order to represent the mental or cognitive operations performed by the user, along with the physical actions performed on the user interface.

6.2.1.1.2 Competitive Analysis

Even though it is not a usability-specific technique, its particular utility as regards the consideration of what characteristics the system to be built should have makes it a good candidate for inclusion in the development process. It is a relatively cheap way of focusing on usability during task elicitation and analysis.

6.2.1.1.3 Scenarios

Scenarios can be used to get information from users and/or domain experts or to approach system usage for the first time. This technique will be used with the special flavour with which usability authors describe it, even though it is a traditional analysis technique.

6.2.1.1.4 Contextual Inquiry and Ethnographic Observation

We select both techniques for the help they can provide concerning the difficult task of getting information about the user tasks and environment. The user observation techniques employed in software engineering are not very formalised, while contextual inquiry and ethnographic observation provide an explicit and carefully planned observation process. This planning makes observation tasks yield appropriate and useful information to feed the other analysis techniques.

6.2.1.2 Modelling

We will select at least one technique for each kind of model to assure that the sort of information it can provide for the development process is not lost.

6.2.1.2.1 User and User Environment Modelling

Most authors are somewhat vague when describing user profiles for modelling the user and user environment, where all kinds of information about the user are gathered. We select the user structured role model technique to model the user, because, like user profiles, it provides the kind of information modelled, albeit with a more defined structure that can help non-usability experts to work with user profiles.

Operational modelling gathers the kind of information that describes the user's environment in a broad sense. Other authors include this kind of information in user profiles, but we appreciate the structured manner in which the information is separated by the user structured role model, on one hand, and operational modelling, on the other. This is why we also select the operational modelling technique for inclusion in the development process. We consider that, as stated above for the user structured role model, this technique will help non-usability experts in the process of gathering and representing information about the different user categories.

6.2.1.2.2 Task Modelling

Use cases, when focused on user intentions and system responses at an abstract level, serve the purpose of task modelling from an analysis point of view very well. Essential use cases have this property. They are selected, because they can evolve to use cases for interaction modelling. which will be selected as a design technique.

6.2.1.2.3 Cognitive Modelling

Cognitive modelling offers a representation of a kind of knowledge that is not usually considered in software development, even though it is quite important from a usability point of view. We, therefore, consider that a cognitive modelling technique should be selected, and GOMS is the chosen one, because three authors reference it, and there are predictive metrics based on it for calculating user productivity.

6.2.1.3 Usability Specifications

Usability specifications are a cornerstone in usability evaluation. We consider the definition of usability specifications based on benchmark tasks and also usability specifications based on preference questionnaires. Both kinds of usability specifications define complementary requirements on the usability of the system to be developed.

6.2.2 Design-Related Techniques

6.2.2.1 Interaction Modelling

Interaction is at the heart of the usability of an interactive system. Due to its importance from a usability point of view, we will select all the available techniques for interaction modelling and design.

They are conceptual design and post-it notes for early design modelling, and screen pictures and use cases for detailed design. The four techniques are complementary, so they will all be selected, and we will delimit their usage in later WP tasks.

6.2.2.2 Design Alternatives Management

We consider both-and design and parallel design to be useful techniques for application in projects with particular needs. Neither are usability-specific techniques, but their usage can be very fruitful from a usability point of view, especially when they provide design solutions that overcome apparently unavoidable usability trade-offs. Therefore, we select both techniques, albeit for optional application in the right circumstances.

Impact analysis, on the other hand, is applied to decide which usability problems must be tackled first, and, therefore, it is an important technique for selecting design alternatives. The alternative that solves the highest-ranking usability problems should be chosen. Therefore, we select impact analysis because it is useful for guiding design decisions.

6.2.2.3 Specification Techniques

From the available specification techniques, we have chosen the ones that better adapt to interactive systems: menu-selection and dialog box trees for menu-based interaction, and transition diagrams for systems that provide different views or contexts to the user. We consider that both kinds of systems need an adequate specification technique, and, therefore, at least one technique for each kind of system has to be included in the development process.

6.2.2.4 User Involvement

User involvement is one of the most important contributions of usability to the development process, and it means that techniques other than the traditional design techniques, which are aimed at professional developers, need to be used. The first selected technique, JEM, details when user involvement is appropriate and more useful, and how to get this involvement from a practical point of view. Visual brainstorming is a specific technique for developing the product concept and is suitable for use with non-experts. This is why it is selected.

6.2.2.5 Help Design

The help subsystem has special characteristics, and its design must consider the human abilities involved in a learning process. Use cases are selected as a technique for interaction modelling, and, therefore, the system will reflect use cases in its interaction model. This makes the organising help by use cases technique useful for our purpose, which is, therefore, selected for inclusion in the development process.

6.2.2.6 Prototyping

Even though prototyping is not a usability-specific technique, it needs to be selected for inclusion in the development process, due to its importance from a usability point of view. The particular kind of prototype to be used is highly dependent on the characteristics of the project and the stage of development, but we will highlight non-functioning prototypes, as they are the techniques with which software developers are least familiar. The three kinds of non-functioning prototypes are complementary and, for this reason, we select all of them for inclusion in the development process. Requirements animation is more common in traditional software development, but we also select it for inclusion because it can provide a usability focus for prototyping.

6.2.3 Evaluation-Related Techniques

6.2.3.1 Expert Reviews

Most authors advocate one form or another of expert review. This is a type of evaluation that complements traditional usability testing, as it uncovers different kinds of errors.

We select all the specific expert review techniques because they are mentioned in most sources. They include heuristic evaluation, and the different kinds of inspections and walkthroughs.

6.2.3.2 Usability Tests

Usability testing is an essential part of development with a usability focus. It is not possible to predict a software system's usability without testing it with real users. It is, therefore, necessary to select a set of usability testing techniques.

We select thinking aloud, because it, along with its variants, is mentioned in most sources, and we consider it to provide complementary information to other kinds of usability testing activities.

Measured performance is necessary in order to verify the attainment of usability specifications based on benchmark tasks. Therefore, we select it for inclusion in the development process.

We also consider post-test feedback to be an important source of information from the test subjects, even though only one author has highlighted it as a technique. We select it as well.

Aditionally, we select direct observation from the field usability testing techniques, because we consider it to be necessary to perform this kind of usability testing. We select the other techniques that are particularisations of field usability testing (beta-testing and the two kinds of indirect observation) as optional techniques, as they can be useful for the development of systems with specific characteristics.

As for Laboratory Usability Testing, we consider that the deployment of a full-scale usability laboratory is very expensive, and a certain number of organizations can find it to have a low investment return. Therefore, we consider the technique of performing usability tests in a laboratory especially prepared for it (with a one-way mirror, several video cameras, etc.), as optional. These kind of premises are appropriate when the size of the organization is substantial, or when the budget for usability investment is high enough.

6.2.3.3 Follow-Up Studies of Installed Systems

Installed systems can provide the most faithful information of all the evaluation-related usability techniques. Follow-up studies apply to given software projects, in which the usability of an existing system needs to be improved. We select these techniques for this kind of projects, leaving the decision on which particular technique or techniques to employ to the development team, depending on the specific project characteristics. Therefore, we consider all the techniques described and their variants (questionnaires, interviews, focus groups, logging actual use, user feedback and surveys) for selection.

6.2.3.4 Predictive Metrics

Of the other evaluation-related techniques, we will consider only predictive metrics. They can appeal to software engineers for their conceptual similarity to some formal evaluation techniques used in software development. For this reason, we select predictive metrics for inclusion in the development process.

6.3 A Priori Discarded Techniques

The techniques on a darker grey background in the tables in section 6.1 have not been selected a priori for inclusion in the development process.

As said above, the criteria used for not selecting a technique is one or more of the following ones:

- It is a special technique for projects with specific characteristics that are not generally applicable.
- It is alien to software engineering, so the developer will find it very difficult to learn to use it.
- Its application will require the use of extra resources from outside the project team.
- It is made redundant by another selected technique. That is, the expected benefits provided by the application of the technique are already covered by a selected technique, and the selected technique offers some additional advantages
- It is not a usability technique. It is a software engineering technique, and it does not make sense to define it as a usability addition to the development process. These techniques rows are highlighted in the tables by means of italics.
- It deals with development process issues, and there are other reasons apart from usability to be taken into account. It must be dealt with in the context of the whole development process.
- The technique is directed at gaining support for usability activities in the development process. This kind of support is a pre-requisite for our process.
- It is presented by just one author, and we consider that it is not generally accepted as a usability technique in the field. This reason will be considered only in conjunction with other reasons, never by itself.

Table 6.9 presents the specific reasons for not considering the grey background techniques as candidate techniques.

Discarded Technique	Reason
Functional Analysis	The reason for not selecting this technique is that it is a software engineering technique, with no particular usability flavour. It is mentioned in just one source.
Needs Analysis	The reason for not selecting this technique is that it is also part of a traditional software engineering process. And it is mentioned in just one source as well.
Financial Impact Analysis	This technique would only be useful for getting management support for usability investment in a project. In our process, we consider that usability investment is a prerequisite for its application. Therefore, we do not select this technique.
Sociotechnical Approach	We do not select this technique because it is appropriate for a very particular kind of systems, where sociotechnical aspects are relevant. Most software products do not fall into this category. Furthermore, it is mentioned by just one author.
User Profiles	We do not select this technique, because it is redundant with the structured user role model and operational modelling. These two techniques model the same kind of information present in user profiles in a more structured manner. We consider that informal techniques for user profiles are more alien to software engineering than structured techniques.
HTA	The reason for discarding this technique is that it is redundant with essential use cases, which have been selected as a task modelling technique.
TAG	The reason for discarding this technique is that it is redundant with GOMS, which has been selected as a cognitive modelling technique.
Object-action	The reason for discarding this technique is that it is redundant with GOMS, which

Discarded Technique	Reason
Interface Model	has been selected as a cognitive modelling technique.
Grammars	This specification technique is only useful for interfaces that comprise command languages. We do not select it because it is for projects with very specific characteristics.
UAN	The reason for not selecting this technique is that its authors state that it has similarities with GOMS. As this kind of techniques are very difficult to implement for non-usability experts, and there are no predictive metrics defined for UAN (as there are for GOMS), we do not select this technique because GOMS is better for our usage purpose all through the development process.
Design Rationale Techniques	We do not select any of the design rationale techniques because they are not usability techniques, as the design rationale can be captured for any kind of design.
Concentric Construction	The reason for not selecting this technique is that it is a software engineering technique, which falls outside the usability field.
Architectural Iteration	The reason for discarding this technique is also that it is a software engineering, not a usability technique.
Prototyping Strategies	The reason for not selecting any of the prototyping strategies described is that their selection is a development process issue, to be decided in each project considering other issues apart from usability.
Experimental Tests	The reason for discarding this technique is that we consider that psychological experiments are too alien to software engineering to be performed as a usual technique in a software development process.
Acceptance Tests	The reason for not selecting this technique is that it is a software engineering technique. It can include usability specifications as acceptance conditions, but we do not consider it to be a different technique from the Usability Specification techniques described above.
Cooperative Evaluation	The reason for discarding this technique is that it is too alien to software engineering and it is mentioned by just one source. We consider that a non-usability expert may find very difficult to design usability evaluations if they must be agreed with users as well.
Participative Evaluation	This technique is not selected because it is for projects with special characteristics. It is for projects where users have a lot of decision-making power, it is considered that they have the right to decide on all development issues, evaluation included.

Table 6.9 Summary of Discarded Techniques

7. CONCLUSION

The results of deliverable D.5.1. “Selection of the Software Process and the Usability Techniques for Consideration” are a first step towards a proposal for applying usability activities and techniques and traditional software development process tasks in an integrated manner. The end objective is to provide developers with the knowledge they need to incorporate, in a systematic and reasoned manner, mechanisms into their development process to improve the usability of the resulting software products.

These preliminary results are founded on several aspects. Firstly, instead of proposing a particular software process into which to incorporate usability mechanisms, we considered it to be more useful to study what characteristics a software development process should have for usability tasks to be incorporated, yielding satisfactory results that compensate the effort invested. From this study, we have determined that a software process that supports usability should, primarily, be iterative. This iterativeness is what will make it possible to check that the usability techniques employed provide satisfactory results and, if not, recommend their reapplication.

We then presented a review of usability activities and techniques referenced in the literature. As specified throughout the document, there is no agreement among the different authors as regards either terminology, since, as mentioned in section 5, one and the same concept is dealt with by different authors as a technique and as an activity, or the number and type of techniques and activities to be used. This lack of agreement is especially evident with respect to usability techniques, where there is considerable variability among authors.

As a result of this review, we have selected a preliminary set of candidate activities and techniques depending on their applicability to the software development process. During the next reporting period, both the relationship between the proposed activities and techniques and their relationship with classical development process tasks will be studied in detail. The end objective, which will be materialised in D.5.2. “Specification of the software process with integrated usability techniques”, is, as mentioned above, to propose a set of usability tasks and techniques to be applied in each software development phase. This has been referred to in D.1.1. Periodic Progress Report as an increment (Δ) that can be incorporated into a variety of individual software processes with the characteristics identified in D.5.1.

8. 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.

[Ferré, 01] X. Ferré, N. Juristo, H. Windl, L. Constantine. "Usability Basics for Software Developers". *IEEE Software*, vol.18, no.1, pp. 22-29. January/February 2001.

[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.