Integration of Human-Computer Interaction Activities in Software Engineering for Usability Goals Achievement

Submitted in partial fulfilment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

by

Anirudha N. Joshi

(Roll No. 05429501)

Supervisor:

Prof. N. L. Sarda

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

(2011)

Approval Sheet

This thesis entitled *"Integration of Human-Computer Interaction Activities in Software Engineering for Usability Goals Achievement"* by Anirudha Joshi is approved for the degree of Doctor of Philosophy.

Examiners

[illegible] 17.3.11

[illegible]

____

Supervisor

[illegible]

Chairman

[illegible]

Date: 17 – March – 2011

Place: Mumbai

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea / data / fact / source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

[illegible]

(Signature)

Anirudha Joshi

(Roll No. 05429501)

Date: 17.3.11

Abstract

Software Engineering (SE) aims at having systematic approach to development, operation, and maintenance of software. SE process models, including waterfall, agile, and Rational Unified Process (RUP), guide software development projects. Human-computer interaction (HCI) is a multi-disciplinary field with a focus on the interaction between humans and computers. HCI also has a well-developed set of processes, activities, methods, and deliverables. Though HCI and SE have overlapping concerns, there are major gaps between HCI and SE in theory and practice. In this research, we focus on integration of HCI activities in SE process models.

We review current literature in design and HCI and propose a process framework. In this framework, we identify HCI activities, methods, deliverables, and skills that are essential and must be integrated in SE processes. We use this framework as a baseline for integrating HCI activities and propose extended process models for waterfall, agile, and RUP. In each case, our approach is to integrate all the HCI activities that we consider essential, to integrate them at a point in the SE process where the HCI deliverables will be most useful, and at the same time to keep the original intents of the SE processes intact.

To demonstrate the validity of our process framework and the integrated process models, we propose Usability Goals Achievement Metric (UGAM), a product metric that measures how well the usability goals of the product are achieved, and Index of Integration (IoI), a process metric that measures the extent to which HCI activities are integrated in a project as compared to a prescribed process model.

To help set goals systematically, we develop a Usability Goals setting Tool (UGT). UGT helps a design team to break down high-level goals into more granular goal parameters.

We evaluate and refine UGT through formative (qualitative) evaluations and validate it through summative (quantitative) evaluations. With data from 65 industry projects, we show that UGT is internally reliable and has a reasonable granularity and coverage. This data also established the need for such a tool. It was surprising to find that more than a third of the usability goals that were considered important in those project contexts were not achieved. We identify 8 goal parameters that are typically high-weighted but have insignificant weight-score correlations.

With the help of data from 61 industry projects using waterfall and agile processes, we statistically demonstrate that IoI and UGAM correlate significantly and the relationship between IoI and UGAM is linear. This implies that whenever projects integrate the HCI activities as per our extended models, they achieve their usability goals better. Correlation analyses reveal differences in usability goal achievement between agile and waterfall process models and between software product companies and software services companies.

Regression analysis of HCI activity scores on UGAM reveals that the individual HCI activities have a varying influence on UGAM. In particular, four HCI activities, viz. user studies, UI prototyping, usability evaluation after detailed UI design, and development support, have the greatest influence on usability goals achievement in projects using the waterfall model.

Keywords
Human-computer interaction, software engineering, processes, metrics, usability goals, goal achievement

Table of Contents

List of Tables

List of Figures

Abbreviations

BPO	Business process outsourcing
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
CRC	Class Responsibility Collaborator
CSUQ	Computer System Usability Questionnaire
DRE	Defect Removal Efficiency
HCI	Human-computer interaction
HTML	Hypertext Mark-up Language
IxD	Interaction design
IoI	Index of integration, described in chapter 11
IT	Information technology
ITES	Information technology enabled services
PUTQ	Purdue Usability Testing Questionnaire
QUIS	Questionnaire of User Interface Satisfaction
RUP	Rational Unified Process
SE	Software engineering
SUS	System Usability Scale
SUMI	Software Usability Measurement Inventory
SWEBOK	Software Engineering Body of Knowledge
UCD	User-centred design
UGAM	Usability Goals Achievement Metric, described in chapter 10
UGT	Usability Goals setting Tool, described in chapters 8 and 9
UI	User interface
UID	User interface design
UML	Unified Modelling Language
UX	User experience

1 Introduction

1.1 Software Engineering Processes

Software engineering (SE) is about producing a family of related software products using many, relatively unskilled programmers to create high quality, multi-version, multi-person software systems (Parnas, 1974). If someone needed to build a software system of a single version all alone that was never going to change, all he would need is *"programming"* (Clements, 1999). The term *"multi-person"* implies that the software would be non-trivial (i.e. beyond the scope of only one person developing the entire piece). Teams also imply that different people would have different roles, and there would be project management and inter-person communication involved. Such software products could be developed by distributed teams of people, many of whom may have never met each other. The term multi-version acknowledges the fact that usually most of the cost of a software product is incurred after it is deployed for the first time, in maintaining the software and enhancing it with features. The discipline SE emerged in the late 1960s with the objective of having systematic, disciplined, quantifiable approach to development, operation, and maintenance of software (IEEE, 1993).

According to Jalote, SE is a complex activity, which is *"full of various opportunities to introduce errors"* (Jalote, 1997 p. vii). In 1975, Brooks compared large-system programming to a tar pit in which many great beasts (i.e. large companies) have thrashed violently (Figure 1.1). He comments that though many have emerged with running systems, few have met goals, schedules, and budgets. No one thing seemed to be the problem; *"any particular paw could be easily pulled out, but the accumulation of simultaneous and interacting factors brings slower and slower motion"* (Brooks, 1995 p. 4).

SE process models provide methods to handle this complexity and enable project teams to produce reliable software systems with maximum productivity. SE process models represent accumulated experience from past projects. They tell the team what needs to be done when and minimise the risk of failure. SE processes form the basis of controlling

Figure 1.1: The "tar pit" of large systems programming (Brooks, 1995 p. 2).

software projects and establish the context in which methods are applied, work products are produced, milestones are established, quality is ensured, and change is managed (Pressman, 2005 p. 54). While several dozens of SE process models have been proposed, in our research, we focus on the three most popular ones.

The waterfall model (Royce, 1970) was perhaps the first attempt in organising the software development activities into a systematic process. Royce articulates that software development involves activities that go beyond the two *"genuinely creative"* steps of analysis and coding. The waterfall model became the de-facto standard software development process and the basis of other process models. Though the waterfall model has been extensively critiqued for its rigidity, it continues to be a popular model even today (Neill, et al., 2003).

Several alternatives to the waterfall process emerged, mainly emphasising iterative approaches to software development. Rational Unified Process (RUP) represents a combination of some of the formalised approaches that came together between years 1995 to 2000 (Kurchten, 2004 p. 33). RUP is an iterative, risk-driven, use-case driven, architecture centric software development approach (Kroll, et al., 2003 p. 3). It is very configurable and is supported by a process product and a range of modelling techniques that help formally specify and track the requirements and design of software components.

While RUP brought in iteration and formalised the software development process, many in the SE community thought that there was too much of formality in this approach and programmers were being bogged down with too much documentation. A third group of process models emerged in response, almost revolting against this formality. These process models, commonly referred to as agile process models, brought back the emphasis on delivering *"working software rather than documentation"* (Agile Manifesto, 2001). They countered the rigidity of the waterfall process by emphasising iterative development through informal interactions between individuals and by developing techniques to *"embrace change"* rather than being concerned about it.

1.2 Human Computer Interaction and Software Development

Human-computer interaction (HCI) is a multi-disciplinary field with a focus on the interaction between humans and computers. HCI emerged in the 1980s with a focus on usability of computer applications and productivity of users. With the spread of computing, HCI researchers expanded interests to include areas such as social computing, ubiquitous computing, creativity, accessibility, and entertainment (Carroll, 2009).

An interaction designer harmonizes form, content, and behaviour of interactive artefacts, both software and hardware, to deliver products that are useful, usable, and desirable. Interaction designer defines *"the structure and behaviours of interactive products and services and user interactions with those products and services"* (IXDA, 2009). Following the fundamental tenets of user-centred design, the practice of interaction design is grounded in an understanding of real users, their goals, tasks, experiences, needs, and wants.

One view of SE is that strives to develop high-quality software (Clements, 1999). Usability is an important quality attribute that is related to HCI. Are targeted users able to use the product? Do they need to be given a lot of training before they can start using it? Does it take too long for users to complete tasks? Do users make too many errors while doing tasks? The answers to these questions can determine the difference between the success of a product and its failure.

This is where the overlap of HCI with SE becomes critical. In the early days, HCI issues were limited as software was deployed in few domains, on limited platforms, clients' requirements were clear, and the software product was often used internally by a few operators. Usability issues did arise, but because the users were internal, in a pinch they could always be trained to work around the problems. Only severest of problems were escalated as enhancements or change requests.

With the rise of desktop computing in the 1980s, the Internet in the 1990s, and mobile telephony in the 2000s, software products reached beyond the safe group of trained users. Usability of products grew in importance as business success often depended on novice users being able to complete tasks.

The last decade saw further expansion of the use of software beyond productivity and automation. Interactive products played a very important role in leisure, entertainment, and socialisation. In this context, the *"quality of experience"* afforded by a product to its users became a broader quality attribute beyond usability. What emotional responses does a product trigger in the users' minds before, during, and after use? Was using the product a nice experience? Do users trust the product? Do they feel confident while using it? Do they enjoy the experience? Do they feel anxious? Proud? Drained? Do they savour their time with the product and look forward to their next session with it, or do they dread the ordeal? In addition to usability, the answers to these questions determined the successes of products in the last decade and promise to do so in the next few years as well.

Like SE, HCI too gives a lot of emphasis on processes to achieve usability and user experience goals. HCI literature suggests many processes, activities, methods, skills, and deliverables. A typical HCI design process is made up one or more phases, each of which may consist of one or more HCI activities. Each activity may be associated with one or more methods. A method may require specific skills. An activity may result in a specific deliverable that may be an end in itself, or may be an input for another activity in the HCI design process or the software development process.

1.2.1 Nomenclature

The field of HCI has influences from several disciplines. Major influences come from cognitive psychology, computer science, library science, and design. Significant influences also come from ergonomics and human factors, writing and rhetoric, and entrepreneurship and management. Influences in understanding users and cultures come from the areas of ethnography and anthropology.

Further, the field itself is known by several overlapping disciplinary labels, often used interchangeably: human-computer interaction, usability, interaction design, information architecture, user-interface design, and user experience design. Yet, the various disciplines overlap extensively. Heller articulates the differences between interaction design and information architecture eloquently: *"Information Architecture is an arrow in an interaction designer's quiver. Sometimes that arrow is a whole other human being, who works beside an interaction designer, and that person is known as an information architect. But it works both ways. An information architect should also have interaction design theory as an arrow in their quiver, and sometimes that arrow is a person called an interaction designer (or similar)."* (Heller, 2002).

When defined in terms of the specific meaning, each disciplinary title identifies with a narrow sub-set of activities. For example, interaction design would mean the design of scenarios that define the impact of the product on the users' life and a high-level conceptual model of the product. However, when defined broadly in terms of what the practitioners does in real life (e.g. interaction design = things that an interaction designer does), the distinction between these terms disappears as all good practitioners do most of the activities.

In this report, we prefer to use the term human-computer interaction (HCI) to represent a wider umbrella that covers all disciplines mentioned above, unless we mean a specific activity (e.g. user interface design or usability evaluation). We use the term HCI practitioner to refer to a professional from any of these disciplines who is responsible for analysis, design, or evaluation of interactive products. We use the term design (and designer) in a broader sense, including design of interactive software and hardware products, but also including areas such as product design, furniture design, visual communication design etc.

1.3 Software Development in India

Offshore software development poses particularly interesting problems from an HCI design perspective. While ideas discussed here could be applicable in other contexts, our research cannot ignore situations where the team involved in software development may be distributed across national, cultural, and organizational boundaries, where major disparities in income and experience can exist among the team members, and where some team members would have limited, if any, access to users and their context. These situations are frequently prevalent in the Indian IT industry.

India has been an important destination of offshore software development for the last 20 years. India accounted for 65 per cent of the global industry in offshore IT in 2005 (NASSCOM, 2005). In financial year 2009, Indian IT industry had revenue of USD 47.3 billion (NASSCOM, 2009). Despite the ongoing slowdown in the global economy, the global offshoring market continues to grow rapidly as the addressable market is more than five times the current market size. NASSCOM expects Indian IT industry to go past USD 60 billion in revenues by year 2011(NASSCOM, 2009).

In the early years, contracted software development companies from India were perceived to be involved in coding and testing of the software *"as required"* by their clients. Over time though, many of these companies have evolved and achieved higher levels of SE process maturity. As of December 2006, over 440 Indian companies had acquired quality certifications of which 90 companies were certified at SEI CMM Level 5, higher than any other country in the world (NASSCOM, 2007). Currently, the Indian contracted software development companies offer full-lifecycle services in planning, modelling, construction, deployment, and maintenance of software.

As in other contexts, usability issues in contracted software development have also grown in importance. As software began to be developed for more domains and began to be used by a wider range of users, and as larger portions of the software development lifecycle began to be contracted out, many usability issues began to crop up, often late in the software development. There has also been an increasing awareness all around, and today clients often explicitly ask for usable software.

NASSCOM says that a lot of growth in the Indian IT industry has come from reducing costs, greater automation, code reuse, and tighter people management. But an important driver for the future growth needs to move beyond costs and drive operational excellence by focussing on areas including *"enhanced customer interaction and solution delivery"*, for example, *"by effectively influencing customer requirements and decisions"*(NASSCOM, 2005). Many Indian software companies responded by introducing HCI activities as one of their service offering towards the end of 1990s (Joshi, 2005). Over time, these offerings have grown.

Contracted software companies often promise a level of quality in the delivered user experience and a level of process compliance to clients. It is important for these companies to integrate HCI activities in software development and to track and improve them continuously.

1.4 Scope of This Research

Though HCI and SE have overlapping concerns, and have evolved side-by-side in the last three decades, the two disciplines did not interact until recently. The IFIP working group on User Interface Engineering remarks that there are major gaps between HCI and SE in academics, literature, and industrial practice, and the architectures, processes, methods, and vocabulary of one community are often foreign to the other (IFIP WG 2.7/13.4 on User Interface Engineering, 2004).

In this research, we focus on integration of HCI activities in the well-established SE process models. In doing so, we need to deal with two main problems.

Firstly, the HCI activities described in literature, even the ones that have been established for a long time, have not been integrated with the SE processes. Identification of user needs through user studies is a good example of an HCI activity for which several methods have been proposed. SE literature admits that communication with the customer during requirements specification is an unsolved problem. Yet, even recent editions of standard textbooks such as Kroll and Kruchten (Kroll, et al., 2003) and Pressman (Pressman, 2005) do not integrate established user study techniques like contextual inquiry (Beyer, et al., 1998).

Further, it is not clear which activities and methods should be integrated. Many HCI methods are defined in literature, but no particular list could be unanimously considered necessary and sufficient for integration with the SE process models. The Usability Body of Knowledge project, for example, attempts to create a *"living reference"* to represent the collective knowledge of the usability profession (Usability Professionals Association, 2004). The project was initiated in 2004, but as of 2010, it is still under development and not yet fully formalised, an indication of the complexity involved.

We review current literature in design and HCI and propose a process framework. In this framework, we identify phases, activities, methods, deliverables, and skills that must be integrated in SE process models. We use this framework as a baseline to integrate HCI activities in waterfall, agile, and RUP process models.

After establishing the integrated process model proposals, we focus on proving that the extended process models work. To demonstrate the validity of our process framework and the integrated process models, we propose Usability Goals Achievement Metric (UGAM), a product metric that measures how well the usability goals of the product are achieved, and Index of Integration (IoI), a process metric that measures the extent to which HCI activities are integrated.

To help development teams set goals systematically, we develop a Usability Goals setting Tool (UGT). UGT helps a design team to break down high-level goals into more granular goal parameters. We evaluate and refine UGT through formative (qualitative) evaluations and validate it through summative (quantitative) evaluations.

With the help of data from 61 industry projects, we statistically demonstrate that IoI and UGAM correlate significantly. Correlation analysis of subsets of data also reveals differences between agile and waterfall process models and between software product companies and software services companies. Regression analysis of HCI activity scores on UGAM reveals that the individual HCI activities have a varying influence on UGAM. We identify four HCI activities that make the highest impact in the waterfall model.

1.5 Organisation of This Report

This report is organised in three parts. In the first part, we identify the important HCI activities that must be integrated in SE processes and present our framework. In the second part, we identify issues and gaps between HCI and SE processes, review prior work related to HCI-SE integration, and present our proposals for integrating HCI activities with SE process models. In the third part, we present our Usability Goals Setting Tool, Usability Goals Achievement Metric, and Index of Integration and the studies we conducted with these tools and metrics.

Part I – Framework of HCI Activities
Chapter 2 reviews design and HCI literature and identifies the activities and techniques that must be integrated in SE processes. Chapter 3 presents a multi-disciplinary framework for the HCI design process and identifies key HCI activities and deliverables that a SE process must integrate.

Part II – Proposals for Integration
Chapter 4 reviews the process models of waterfall, agile and RUP and identifies problems with each SE process model from an HCI perspective. Chapter 5 summarises prior work that analyses the gaps in SE from an HCI perspective and the gaps in practices in 8 case studies from the Indian IT industry. Chapter 6 summarises prior work on approaches for integrating HCI with SE. Chapter 7 presents our proposals for integration of HCI activities with SE process models.

Part III – Correlating Integration with Goal Achievement
In chapter 8, we describe our proposal for the Usability Goal Setting Tool. In chapter 9, we present the formative evaluations that we did to refine UGT and the summative evaluations that we did to validate it. We also present analyses of data from 65 industrial projects that establish the internal consistency validity of the suggested usability goal parameters, identify latent goal parameters, and explore quantitative methods for recommending weights for usability goal parameters.

Chapter 10 describes our proposal for the Usability Goals Achievement Metric and a validation experiment. Chapter 11 describes our proposal for Index of Integration and the qualitative feedback that we got from the industry about the two metrics.

In chapter 12, we establish the correlation between the two metrics and based on data from 61 industry projects, demonstrate the effectiveness of our proposals. We also present additional analyses that include validation of our metrics and identification of the four important HCI activities in the waterfall model. Chapter 13 concludes this research and identifies directions for future work.

PART I – FRAMEWORK OF HCI ACTIVITIES

2 Activities in Human Computer Interaction Design

2.1 Design Process

Archer defines design as a goal-driven problem-solving activity (Archer, 1965). Lawson argues that we may never find the true definition of design as it is *"too complex to be summarised in a book"*, but searching is probably much more important than the finding (Lawson, 1980 pp. 22-23). According to Jones, the effect of designing is to *"initiate a change in man-made things"* that in turn affect the manufacturers of those products, the distributors, the purchasers, the users, and ultimately, the society. An important job of the designer is to predict each of those behaviours and responses at each stage in the life of the product (Jones, 1970 pp. 4-9). Schön describes a designer simply as a person who *"makes things"*. Sometimes he makes the final product; more often, he makes a representation, a plan, program, or image, of an artefact to be constructed by others (Schön, 1983 p. 78).

One way to understand design is according to Lawson is to chart the design process. A systematic design process has been an essential part of design practice, teaching, and literature (Archer, 1965), (Jones, 1970), (Lawson, 1980), (Cross, 2000). At its bare bones, all authors agree that the design process comprises of three fundamental activities:

- Analyse user needs, problems and opportunities to identify goals and constraints
- Synthesise alternative solutions
- Evaluate them against goals and redesign the product.

These activities are iterative. Problems found with the proposed design at the time of evaluation are fixed in a new design solution iteratively until the most appropriate solution is found. As iterations progress, design moves from generic to detailed.

While the fundamental design activities look simple and straightforward, their implementation in industrial projects is challenging, particularly in designs that require innovative solutions. Most authors agree that design processes are complex. Lawson argues that analysis, synthesis, and evaluation can happen simultaneously, and sometimes synthesis can lead to analysis (Lawson, 1980 pp. 22-35). Schön says that because of the complexity of the process, the designer's moves tend to produce consequences other than those intended. When this happens, the designer may take account of unintended changes by making new moves (Schön, 1983 p. 79).

To deal with the complexity, designers have evolved detailed methods for each of these activities (Jones, 1970), (McKim, 1980), (Cross, 2000). The main effect of the design methods is to externalise what good designers do intuitively to allow design of complex and innovative systems that are beyond the experience of any one person.

Jones broadly divides design methods into three categories, which also represent the three approaches that a designer takes: divergence, transformation, and convergence (Jones, 1970 pp. 63-69). These three approaches and particularly the need for divergence before convergence in the design process were expressed by several authors, including (Laseau, 1980 p. 115) (Figure 2.1).

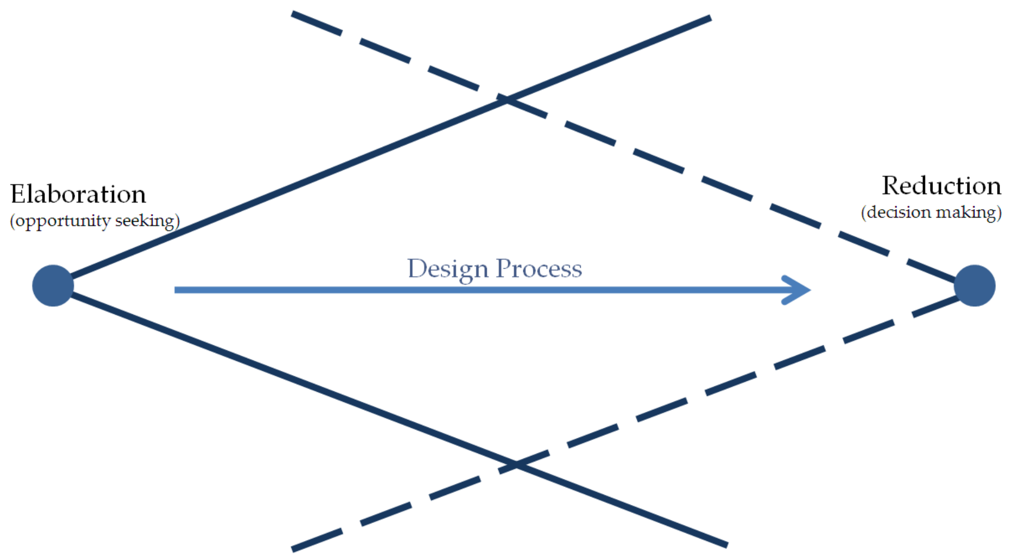


Figure 2.1: Elaboration (divergence) and reduction (convergence) working together in the design process (Laseau, 1980 p. 115).

Divergence (or elaboration) refers to the act of extending the boundary of the design situation to have a large search space in which to seek a solution. Key characteristics of the divergence stage are its tentativeness and instability. The objectives, the problem boundary, and the sponsor's brief are unstable, and evolve during this stage and evaluation is deliberately deferred so that nothing relevant is disregarded. As Schön says, "*In real-world practice, problems do not present themselves to the practitioners as givens. They must be constructed form the materials of problematic situations which are puzzling, troubling, and uncertain.*" (Schön, 1983 p. 40). Design methods related to this stage include benchmarking and evaluating current products, and understanding user behaviours, needs, and problems. These methods often require both rational and intuitive actions and many of them require *"legwork rather than armchair speculation"* (Jones, 1970 p. 65). The aim of divergent search is to restate the original brief while identifying the features of the design situation that will permit a valuable and feasible degree of change.

Jones calls transformation as the creative and the most interesting step of design when the objectives and the problem boundaries are fixed, critical variables are identified, constraints are recognised, and opportunities are taken. This could also be the stage where big blunders are made, and where experience and sound judgement are necessary. Design methods for searching for new ideas (such as brainstorming and synectics), and design methods to explore the problem structure (such as mind mapping, interaction matrix, and affinity) enable this transformation. Jones calls many of these methods as *"black-box methods"* as these are the ones that depend most on the chief designer's creativity and intuition (Jones, 1970 pp. 46-49).

Jones says that in convergence (or reduction), the designer's aim is to *"reduce the secondary uncertainties rapidly so that an optimum solution can be arrived at with minimal effort"* (Jones, 1970 p. 68). During this stage, the designer is working with the most details in the design and if it does not converge fast, the number of alternatives available can explode. Design methods related to convergence stage are related to evaluation, measurement, and analysis. Jones calls these as *"glass-box methods"* as these are very rational, analytical, and, in theory, can be automated (Jones, 1970 pp. 49-54).

2.2 Human Computer Interaction Design Processes

Several authors have articulated process models for design of interactive products from the 1980s until the current times. Many of these (particularly the early authors) came from backgrounds in psychology, and their process models reflect a stronger emphasis on analysis, usability evaluation, and convergent thinking. Nevertheless, there are many overlaps with traditional design processes, and particularly some of the later literature brings in methods supporting divergence and transformation. We summarise the significant work chronologically.

The basic ideas for design of interactive systems were already articulated by the 1970s, though these were not always followed. Gould and Lewis recommended three *"principles"* of design, which easily translate into the steps of a process: early focus on users and tasks, empirical measurement of user performance on prototypes, and iterative design to fix problems found during usability tests, *"as they will be"* (Gould, et al., 1985). They also acknowledged the importance of the process to ensure meeting goals such as making the system easy to use, user friendly etc. Several of the later authors elaborate on these principles to identify the activities or stages of a process.

Nielsen elaborated on these basic principles through an 11-stage usability engineering lifecycle (Nielsen, 1993):

- Know the user
- Perform competitive analysis
- Set usability goals
- Do parallel design (design alternative approaches)
- Participatory design (engage real users during design)
- Coordinate design of the total interface (ensure consistency)
- Apply guidelines and heuristic analysis
- Prototype
- Test empirically
- Iterate design
- Collect feedback from field use.

Nielsen also articulates 5 usability goals and 10 heuristics to guide the design process (Nielsen, 1993 pp. 24-37, 115-155).

Kreitzberg identified 6 stages of design methodology (Kreitzberg, 1996 pp. 65-88):

- Develop the product concept (high-level concept, business objectives, identify users and environmental issues)
- Perform research and needs analysis (break user population in homogenous segments, break activities down into tasks, analyse needs, identify major objects and structures)
- Design concepts and key-screen prototype (specify usability objectives, select navigational model and select a metaphor, develop a prototype of key screens, conduct initial reviews)
- Do iterative design and refinement (expand the key screen prototype, conduct evaluations)
- Implement software
- Provide rollout support.

Dix et al. look at software lifecycle as the basis of process (Dix, et al., 1998 pp. 179-221). This is perhaps the only book on HCI that directly looks at SE as *"a means of understanding the structure of the design process"*. They highlight the problem that an interactive system cannot be completely specified form the beginning. The only way to be sure of the requirements is to build a prototype, test it out on real users, and modify to fix the problems found. Dix et al. propose three main methods by which these iterations can be done: by building a throw-away prototype and evaluating it before requirements are finalised; by evolving the product through complete iterations of development, not throwing away the earlier versions, but using them as a basis for the next iteration; and by building the product incrementally as separate, smaller components. The first approach is perhaps useful in the context of the waterfall model as all the requirements are available before development starts. The other two approaches could be useful in the context of iterative process models such as agile processes and RUP.

Contextual Design is a process model developed by Beyer and Holtzblatt (Beyer, et al., 1998). Theirs is perhaps the first process model in HCI literature to encourage divergence and transformation in addition to convergence. Contextual design consists of contextual inquiry, consolidation, redesign of work, prototyping, and evaluation. Contextual inquiry is a key component of the contextual design process and perhaps its most important contribution. It is an interview technique, which draws upon ethnographic approaches. The interviews are documented through work models and findings are consolidated across users in order to identify critical problems, insights, and design ideas. The methods help the designers gain a deep understanding of users' tasks, roles, artefacts, environment, and culture and enable divergence in the designers' thinking. Before considering a technical solution, Beyer and Holtzblatt suggest explicit redesign of the structure of users' work. This vision brings about the transformation and drives changes to the organisational structure and procedures, and drives the definition of a new system to support the new work practice. Rough paper prototypes are then built, and iteratively improved upon and until the solution converges to a detailed user interface.

Mayhew brings the perspective of an external usability consultant to the product development process (Mayhew, 1999). She divides the process into three key phases: analysis, design / testing / development and installation (Figure 2.2). She further divides each phase into tasks. Mayhew suggests a variety of techniques to carry out each task, but her approach is open and flexible. One may substitute a quicker / cheaper technique to do a task, but each task must be done. In our proposals below (chapter 3), we use a similar distinction between activities and methods. Mayhew is also flexible on some of the steps in her process models (indicated by the dotted line in Figure 2.2). She suggests creating concrete deliverables for each task that feed into a continuously growing style guide. It is an effective process for consultants where one needs to adapt to several situations and needs of different development teams.

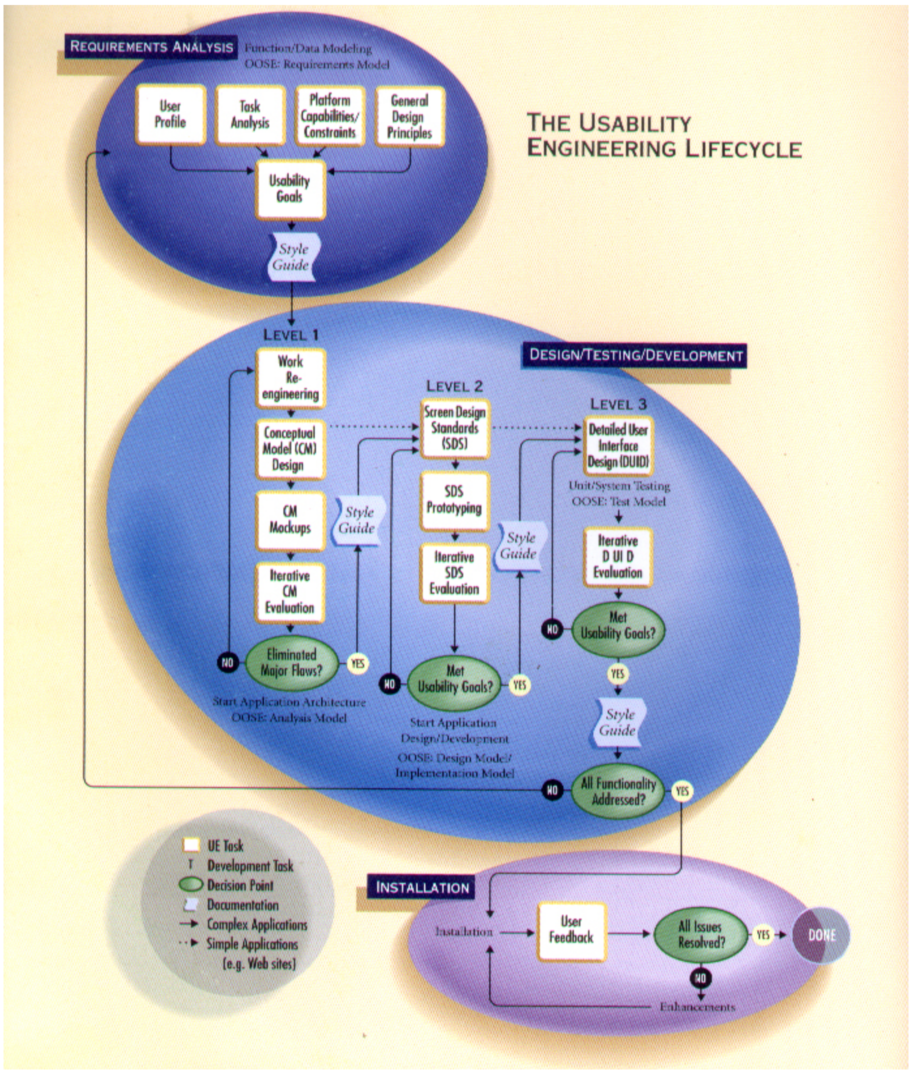


Figure 2.2: Usability Engineering Lifecycle (Mayhew, 1999)

Preece et al. describe the process of interaction design in terms of four basic activities (Preece, et al., 2002 pp. 168-170), (Sharp, et al., 2007 pp. 428-429): Identify needs and establish the requirements for the user experience; develop alternative designs that meet those requirements; build interactive versions of the designs so that they can be communicated and assessed; evaluate what is being built throughout the process and the user experience it offers. Like in case of other process models, these activities are supposed to inform one another and are meant to be repeated.

Cooper and Riemann articulate the design process used by Cooper design (Cooper, et al., 2003 pp. 14-20). Their goal-directed design process itself is rather simplistic and is driven by roles of disciplines (rather than being a multi-disciplinary process): Managers take preliminary inputs from users and initiate a project; designers in turn take additional inputs from users, design the product and hand over a specification to the programmers; programmers hammer out the code as per the specifications; software testers test the code to the specifications and hand back bug reports; usability testers take the product back to the users for feedback, and hand back bug reports to designers and / or programmers; once done, the product gets handed over to the marketing and sales for shipping. Their design step itself consists of sub-steps that reflect divergence, transformation, and convergence: Research users and the domain; model users and use contexts (in particular personas and their goals); define requirements of users, business and technology; create a framework to define the design structure and flow through scenarios to meet user and business goals; and refine the framework; design the interface details and validate them.

Within HCI literature, perhaps Buxton brings the most direct emphasis on divergence and transformation in design of interactive systems (Buxton, 2007). He insists that there are two common myths in the industry: that we know what we want at the start of the project, and that we know enough to start building it (Buxton, 2007 p. 77). He argues that the best way to explore the solution for an interactive product is to *"sketch it"*. Buxton documents many ways of sketching user experiences. His design process itself is somewhat sketchy, perhaps deliberately so, where he mainly insists on several iterations of divergent concept generation followed by controlled convergence. While following the process, the team starts with a large number of high-level ideas, and gradually converges to detailed refinements.

In addition to design processes, HCI literature also articulates the nature of HCI design deliverables and principles that should govern the process models of a design process.

Garrett suggests a conceptual framework to understand the users' experience of websites that can easily extend to other interactive products (Garrett, 2003 pp. 21-36). Garrett suggests that users experience websites in five layers: Surface (text, styling, images); skeleton (layouts, interface design, information design); structure (interaction design and information architecture); scope (functional and content specifications); and strategy (user needs and business objectives).

Garrett's model of user experience is not a process model in itself, but it has important implications for the process. Decisions at lower layers affect the choices available at the higher layers. A strategic decision will ripple through the scope, structure, skeleton, and surface layers. Similarly, changes in the scope will affect structure, skeleton, and surface. This implies that in a design process, we need to freeze on lowers layers of user experience (strategy, scope etc.) before we can freeze upon the higher layers (structure, skeleton, surface etc.). However, it does not imply that all decisions related to the user experience of each lower layer should be completed before any work on an upper layer can begin as it might seem in Figure 2.3a. In most situations, it is likely that decisions about the layers will overlap somewhat (Figure 2.3b), or even overlap a lot (Figure 2.3c). The implication of Garrett's model is that one should not *freeze upon* user experience decisions related to an upper layer of user experience, before one has frozen upon decisions on all lower layers. This still does not rule out an *"outside-in"* design methodology, where many upper layers are explored early and at some point in the process, all layers are explored simultaneously (Figure 2.3d).

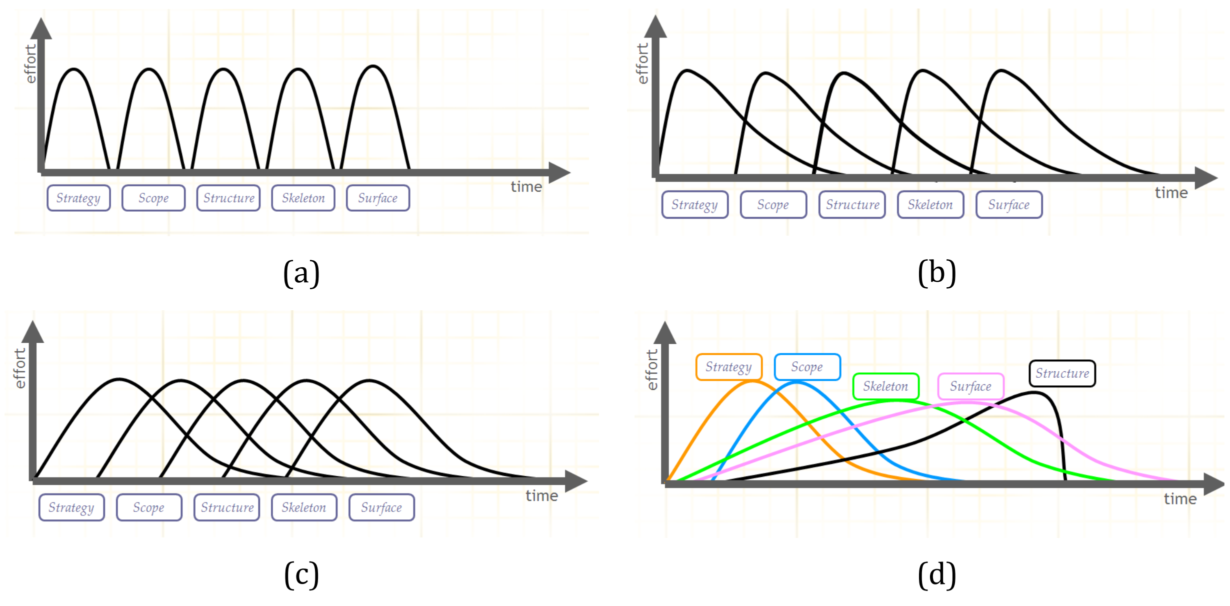


Figure 2.3: Process implications of Garrett's model of user experience. (a) and (b) are

Gulliksen et al. review HCI literature by several authors and list key principles of user centred design (Gulliksen, et al., 2003). These are summarised below:

- Users should be in focus from the beginning. The goals of the activity, the work domain or context of use, the users' goals, tasks, and needs should guide the development. Representative users should be actively involved early and continuously throughout the development.

- The system development should be iterative and incremental. Each iteration should consist of user and context analysis, design, and documented evaluation.
- User interface and interaction design should be explicit, conscious activities, not implied in the design or construction of software.
- Usability goals should be specified, and evaluation of the design should be against these goals. Early and continuous prototypes should be used to evaluate ideas in cooperation with end users.
- The design and its implications must be represented in such ways that it can be easily understood by users and other stakeholders, not just by the software development teams.
- The design should be holistic, considering all aspects including impact of design users' work, on the organisation, roles, etc. All parts of the product (task organisation, user interface, online help, user training, health and safety aspects etc.) should be influenced by common design thinking.
- There is no one-size-fits-all HCI process. The techniques and order of activities could be customised to fit the needs of specific projects and the needs of the specific software development process being used.

2.3 Summary

Prior work on design establishes the three steps of the design process as analysis of user needs, synthesis of a solution, and evaluation of the solution against goals. Design literature emphasises on the need for divergence, where the boundary of the design situation is extended, before transformation and convergence to a solution. Literature identifies several design methods, some of which are called *glass-box* methods as these are analytical and rational, while others are called *black-box* methods as these depend on creativity and intuition of the individual.

Prior work in HCI adapts design processes with the help of specific methods (contextual inquiry, usability evaluation), deliverables (affinity model, personas), models (Garrett's model of user experience), principles (*"simple design representations"*), and heuristics (*"minimise user memory load"*). Some of the later HCI authors stress on divergence and transformation in addition to convergence.

Several authors have described their preferred methods and activities in detail. However, there is no consensus on what constitutes a complete list of HCI activities, methods, deliverables, and skills and its relationship with the three stages of design thinking. None has talked about the relative importance of HCI activities, or the extent to which each activity affects the usability and the user experience of the delivered product. Other than a few exceptions, most HCI authors do not suggest how to integrate their activities, methods etc. with the popular SE process models such as waterfall, agile and RUP.

In the next chapter, we present our proposal for a multi-disciplinary framework, and identify the key HCI activities and deliverables that a SE process model must integrate.

3 A Multi-disciplinary Framework of Human Computer Interaction Activities

3.1 Human Computer Interaction Process Framework

By combining the essential characteristics of the various processes discussed in design and HCI literature discussed in chapter 2, we propose a multi-disciplinary process framework for HCI design. Figure 3.1 captures the phases, activities, methods, skills, and deliverables in a visual form. Table 3.1 on page 27 summarises the same in a tabular view.

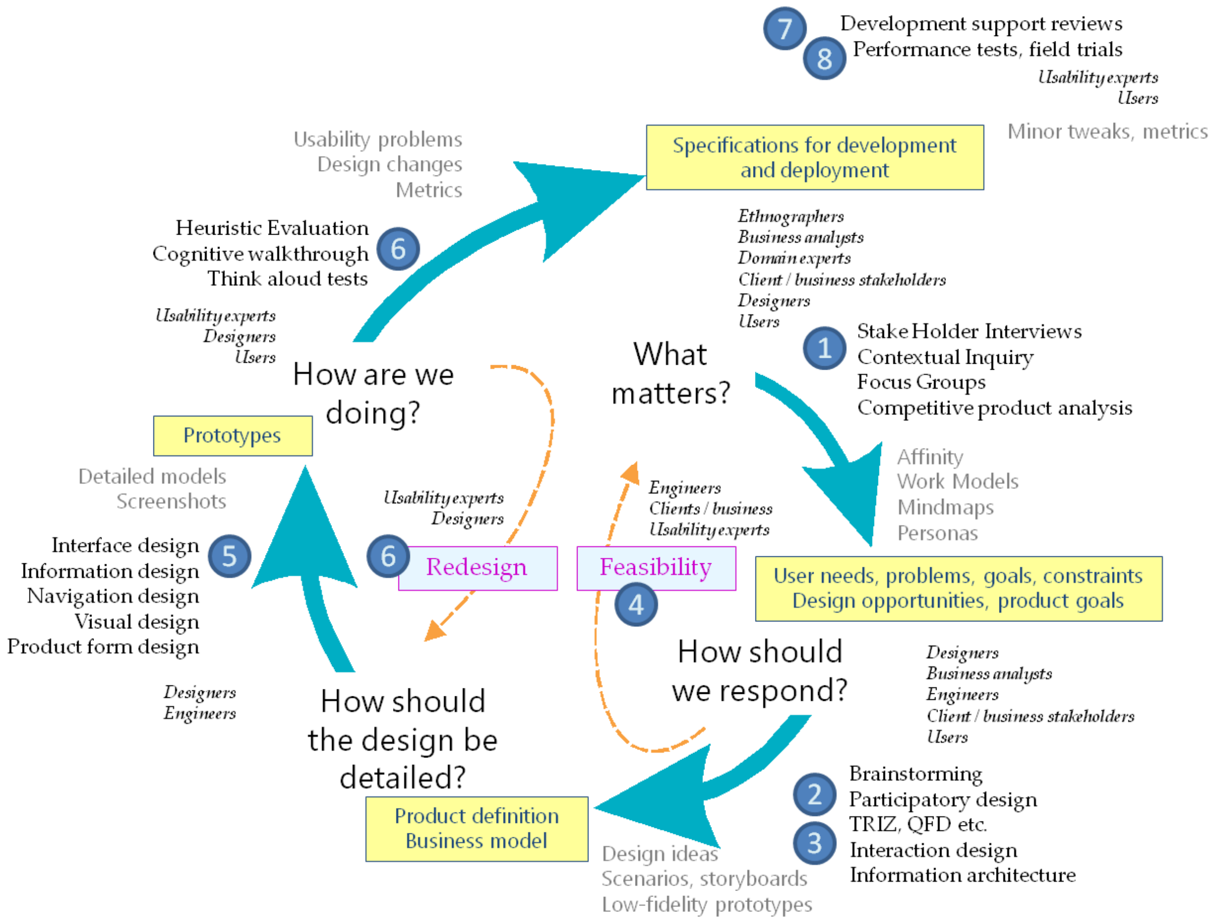


Figure 3.1: A framework for HCI design process.

Our framework is proposed to be a flexible way of understanding and communicating the work of HCI practitioners in different contexts. Our objective is not to come up with another prescriptive *"one-size-fits-all"* HCI design process model, but rather to articulate the typical HCI activities within which several methods and deliverables can be assimilated. Not all activities or methods may be essential in each instance of the process.

We divide our framework into phases. Each phase consists of one or more activities. Each activity is associated with one or more methods. Each method requires specific skills and could be associated with a particular discipline. Each activity results in specific deliverables. A deliverable may be an end in itself, or may be an input for another activity in the HCI design process or the software development process. For example, usability evaluation is an HCI activity that is a part of almost every process. Usability evaluation could be performed by several methods such as a think-aloud test, a performance test, a heuristic evaluation, a cognitive walkthrough, or an expert review. Performing each method requires a specific set of skills. E.g. the think-aloud test requires skills in prototyping, qualitative test design, user recruitment, interviewing users, and analysing data. The activity results in deliverables such as usability problems with the design, potential ideas to improve the design, and possibly a decision about the future course of development.

In our framework, we identify eight HCI activities that, we propose, are essential for integration with SE process models. We organise these activities in four phases, which we describe in terms of four questions derived from (Beyer, et al., 1998 pp. 10-11):

- What matters?
- How should we respond?
- How should the design be detailed?
- How are we doing?

While the whole process is iterative, there are two inner loops as shown in Figure 3.1: feasibility of the product definition, and redesign of the prototype to fix problems found. Sometimes, it may be necessary to go through these inner loops more than once, but going through them once will be required for all projects.

*Table 3.1: A multi-disciplinary framework for the HCI design process. Deliverables marked with * could be considered to be essential.*

Phases	Disciplines involved	HCI Activities	Methods	Deliverables / outcomes
What matters?	Ethnographers, business analysts, domain experts, client / business stakeholders, designers, users	1. User studies, user modelling, market analysis	Stakeholder interviews Contextual inquiry Focus groups Competitive product analysis	Analysis of individual interviews User models such as affinity, work models, mind-maps, personas User needs, problems, goals and constraints* Opportunities for design interventions Product goals (including usability goals)*
How should we respond?	Designers, business analysts, engineers, client / business stakeholders, ethnographers, users	2. Ideation	Brainstorming Participatory design TRIZ QFD	Design ideas
		3. Product definition	Interaction design Information architecture	High-level use scenarios, storyboards Low fidelity prototypes, wireframes of software, foam models of hardware Business model Strategy, scope and structure of Garrett's model
Feasibility	Engineers, client / business stakeholders, usability experts	4. Formative usability evaluation 1 and refinement	Heuristic evaluation	Refined and approved product definition and product goals* Technology feasibility approval* Business feasibility approval*
How should the design be detailed?	Designers, engineers	5. Design detailing	Interface design Information design Navigation design Visual design Product form design	Medium to high fidelity UI prototypes through iterations Structure, skeleton and surface of Garrett's model
How are we doing?	Usability experts, designers, users	6. Formative usability evaluation 2 and...	Heuristic evaluation Cognitive walkthrough Think aloud test Card sorting	Usability problems Metrics
		... refinement	Same as in design detailing	Refined, detailed UI prototypes* UI specification*
Software design and development	Designers, usability experts	7. Development support	Reviews during development	Minor tweaks
Early version of the software	Usability experts, users	8. Summative usability evaluation 3	Usability performance test Field trials	Usability approval* Metrics

The following sections describe the details of these activities, including the disciplines and the skills involved, methods that could be used to perform the HCI activities, and expected deliverables or outcomes.

3.1.1 What matters?

It is not only about what is *"required"* by someone. *"What matters"* is a broader question and asks the design team to look at the problem at hand as holistically as possible. This question is answered through divergent thinking, looking beyond what had been specified in the design brief, and trying to set the problem before solving it.

The HCI activity associated with this phase is **user studies, user modelling, and market analysis** (activity 1 in Table 3.1). To understand the key concerns of the users, the stakeholders, the domain, and the context deeply, the team uses methods such as stakeholder interviews, contextual user interviews, focus groups, field observations, log analyses etc. (Beyer, et al., 1998), (Kuniavsky, 2003), (Hackos, et al., 1998). The team may also study related issues such as the environmental, social, or cultural aspects. Most teams would do a benchmark analysis of competitive products.

This is a very multi-disciplinary phase where ethnographers, business analysts, domain experts, client / business stakeholders, HCI practitioners, and potential users are involved.

At the end of this phase, the team gets a good understanding of users' needs, problems, goals, and constraints. They also have a good understanding of the design opportunities. The phase ends with identifying product goals, including usability goals.

3.1.2 How should we respond?

This is a holistic question as well. Now the design team is not just describing the situation but also transforming the problem space so that one or a few solutions become evident.

The team begins with **ideation** (activity 2 in Table 3.1). With their creativity, but also using a range of ideation techniques such as brainstorming, synectics, participatory design, quality function deployment (QFD), and theory of inventor's problem solving (TRIZ) the team comes up with a range of design ideas that solve the problems and realise the opportunities (Jones, 1970), (McKim, 1980), (Lawson, 1980), (Buxton, 2007). The ideas are typically wild and divergent to begin with, and early on, the focus is on generating more ideas rather than evaluating them.

Eventually the team reaches a coherent understanding and articulation of the context and creates a meaningful, holistic response, a high-level **product definition** (activity 3 in Table 3.1). It is not a matter of merely designing a particular product. The product definition is an organisational response to the big picture and describes how the product fits in the life of the targeted users (Beyer, et al., 1998), (Mayhew, 1999), (Cooper, et al., 2003), (Buxton, 2007).

In interaction design and information architecture, these responses are typically represented in the form of scenarios and presented as storyboards, videos, or theatrical enactments. Low-fidelity prototypes are created to support the scenarios. If design involves a new hardware, the form factor of hardware is modelled. If it involves a piece of software, wireframes of the screens are created. At this stage, HCI practitioners are primarily interested in the high-level descriptions comprising, typically of the primary persona and his/her high-level product usage scenarios, rough mock-ups or wireframes. Buxton calls these methods *“sketching user experiences”* (Buxton, 2007 pp. 105-125). The product goals are constantly guiding the team in this phase, but the product goals also are reviewed and tweaked.

The first loop in the framework is the feasibility loop that occurs just after the product definition. Here, the engineers evaluate the product definition for technical feasibility and the client or the business stakeholders evaluate it for the business feasibility. If competing product definitions are still in contention, a choice is made. If none of the proposed product definitions are found to be feasible, the team will go back and think of more alternatives or may go all the way back and do more user studies to identify other opportunities or problems.

At this stage, the design team may do a **formative evaluation of the product definition** (activity 4 in Table 3.1), possibly by lightweight methods such as a heuristic evaluation or a cognitive walkthrough (Nielsen, 1993). The product definition would be refined to fix any problems found. At the end of this activity, product goals are finalised and technically and financially feasible product definition is agreed.

In this phase again, a multi-disciplinary team is involved, as prescribed by each method. This includes HCI practitioners, business analysts, engineers, and client / business stakeholders. If the method of participatory design is used, users are involved. If a formative usability evaluation is done, usability experts are involved.

As the phase progresses the team reaches a coherent understanding and articulation of the context and converges to one or a few meaningful responses at a high level. At the end of this phase, product goals are finalised and a technically and financially feasible product definition is agreed.

3.1.3 How should the design be detailed?

Once a feasible product definition is agreed upon, the **detailed user interface is designed** (activity 5 in Table 3.1). This activity completes the transformation and initiates the convergence.

Designers explore the details of the user interfaces such as labels, icons, and behaviour of widgets. The text is written. Information is visualised. Visual elements such as typography, colours, fonts, and layouts are designed. Product form of hardware products is finalised. Several authors describe methods for this activity, including (Mayhew, 1999), (Preece, et al., 2002), (Cooper, et al., 2003), (Snyder, 2003).

The purpose of creating a prototype is to evaluate it. Design decisions that seem particularly risky are prototyped first so that feedback on these can be sought early. The fidelity of the prototypes keeps increasing through the iterations as more details are added.

This activity is primarily the designers' responsibility, though truly innovative designs may require collaboration between design, technology, and business. The output of this phase is one or more prototypes capturing and representing the design decisions.

3.1.4 How are we doing?

In this phase, the team seeks to converge to a usable solution quickly. As it may happen, the initial design decisions do not fit all the users' needs. When a prototype is ready, a **formative usability evaluation** is done against the usability goals to identify potential problems with the design (activity 6 in Table 3.1). Card sorts and think-aloud tests are the most preferred methods of evaluation at this stage, but other methods may also be used (Nielsen, 1993), (Dumas, et al., 1999), (Mayhew, 1999), (Kuniavsky, 2003), (Tullis, et al., 2008). Usability evaluators do the evaluations, though designers may also participate to get a first-hand feedback. The evaluation generates a list of problems and design ideas to fix them.

The second loop in the framework is that of *redesign*. As Gould and Lewis noted, interactive systems are particularly prone to having problems in the early designs that need to be fixed (Gould, et al., 1985). After a round of evaluation, problems are fed back and products are redesigned until an acceptable solution is found. The fidelity of the prototypes keeps increasing through the iterations as more details are added. This cycle of design, prototype, evaluate, redesign needs to be tight, quick and consume as few resources as possible.

This is the reason why we suggest that the evaluation should be formative, which is often more flexible and quicker than summative evaluations. We also suggest that designers should be involved during this evaluation because the feedback is immediate and perhaps the design could be changed on-the-fly (Nielsen, 1993 pp. 98, 170, 180-181).

The outcome of the redesign phase is a detailed UI prototype that has been refined after evaluations, and a specification of the user interface. The specification should be written in a form that can be directly consumed by the software engineering process, preferably in the form of use cases accompanying the prototype.

Changes to the design inevitably happen during software development. Some of these changes are inadvertent slip-ups (e.g. change of typeface, colour, or layout). It may be possible to integrate the decisions from the prototype directly into code (e.g. by reusing HTML code) and to set up tools by which the slip-ups can be traced and controlled automatically by the design team without bothering the development team. Other changes have to be made because the original design was not feasible. Yet other set of changes happen because there is a change in requirements or change in technology platform. In all cases, ongoing collaboration between the design and engineering teams is important during software development. We call this **development support** (activity 7 in Table 3.1). In practice, this may be not so easy because *"having delivered the project"*, the design team is usually disbanded, and the HCI practitioners move on, sometimes to a project in another city.

When an early version of the production code becomes available, it is a good idea to do a **summative usability evaluation** against the usability goals (activity 8 in Table 3.1) (Nielsen, 1993 pp. 170-171), (Mayhew, 1999 p. 342). It is particularly important to do so for highly innovative, one-of-a-kind products as the HCI practitioners had no experience on such platforms and because prototypes of such products (on which the formative evaluations were done) tend to be quite crude and non-representative of the final product performance. It is also important to do summative evaluations as a part of acceptance testing on mission critical projects such as life-critical applications for healthcare and defence, but also on mass-use products such as electronic voting machines or ATMs.

Often, summative evaluation is done in a lab-based quantitative performance test (Nielsen, 1993), (Dumas, et al., 1999), (Kuniavsky, 2003), (Tullis, et al., 2008). In some cases, it may be done by deploying the product in the field. Preferably, a summative evaluation is done by an external evaluator. The main outcome of a summative evaluation is (hopefully) a usability approval. Though summative evaluation is not supposed to affect the design, if serious usability problems are found, these ought to be fixed before release. Minor problems could be archived and dealt with in the next release. A set of relevant metrics could also emerge from summative evaluations, which will help refine a profile for the product and set benchmarks for the next version, if applicable.

3.2 Discussion

In the framework described above, not all elements are mandatory. While all activities are essential, the specific methods mentioned above need not be used to do those activities. These methods are only suggestive, and alternatives could be found. This distinction is similar to the difference in tasks and techniques mentioned by (Mayhew, 1999 p. 19).

Further, the deliverables are of two types: essential and optional. Some deliverables are related to a particular method, and will emerge only if that method is used (e.g. work models, affinity). Other deliverables are internal work products often preferred by HCI practitioners (such as personas and storyboards), but not mandatory. However, some deliverables are essential for completing the HCI design process. For example, user needs, problems, goals, and constraints must be found and product goals must be defined at the end of user studies activity, as these are essential to carry out design and evaluation activities. The ideation, product definition, and refinement activity should result in a high-level product definition that is found to be feasible from a technical and business perspective before diving into detailed design. The detailing, evaluation and refinement should result in a detailed UI prototype and a specification. The project should end with an approval for usability, however informal. The deliverables that could be considered essential in this framework from an HCI perspective are marked with an asterisk in Table 3.1 on page 27 above.

The process framework described above maps on to the divergence, transformation, convergence design stages as described by Jones (Jones, 1970 pp. 64-69). Many of the methods or techniques that help answer the question *"what matters"* and some that help answer the question *"how should we respond"* are the ones that help the team in divergence and problem setting. Many of the methods and techniques that help answer the questions *"how should we respond"* and some of *"how should the design be detailed"* help in the transforming the problem space so that a solution become evident. Finally, the methods and techniques that help answer the questions *"how should the design be detailed"* and *"how are we doing"* help in converging to a particular solution. These ideas are summarised in Figure 3.2.

Our proposed framework describes the activities done by a multi-disciplinary team, including the HCI disciplines, technology, business analysis, domain experts, and marketing. It should be seen as a framework *to design interactive products* rather than a process framework *to be followed by designers*.

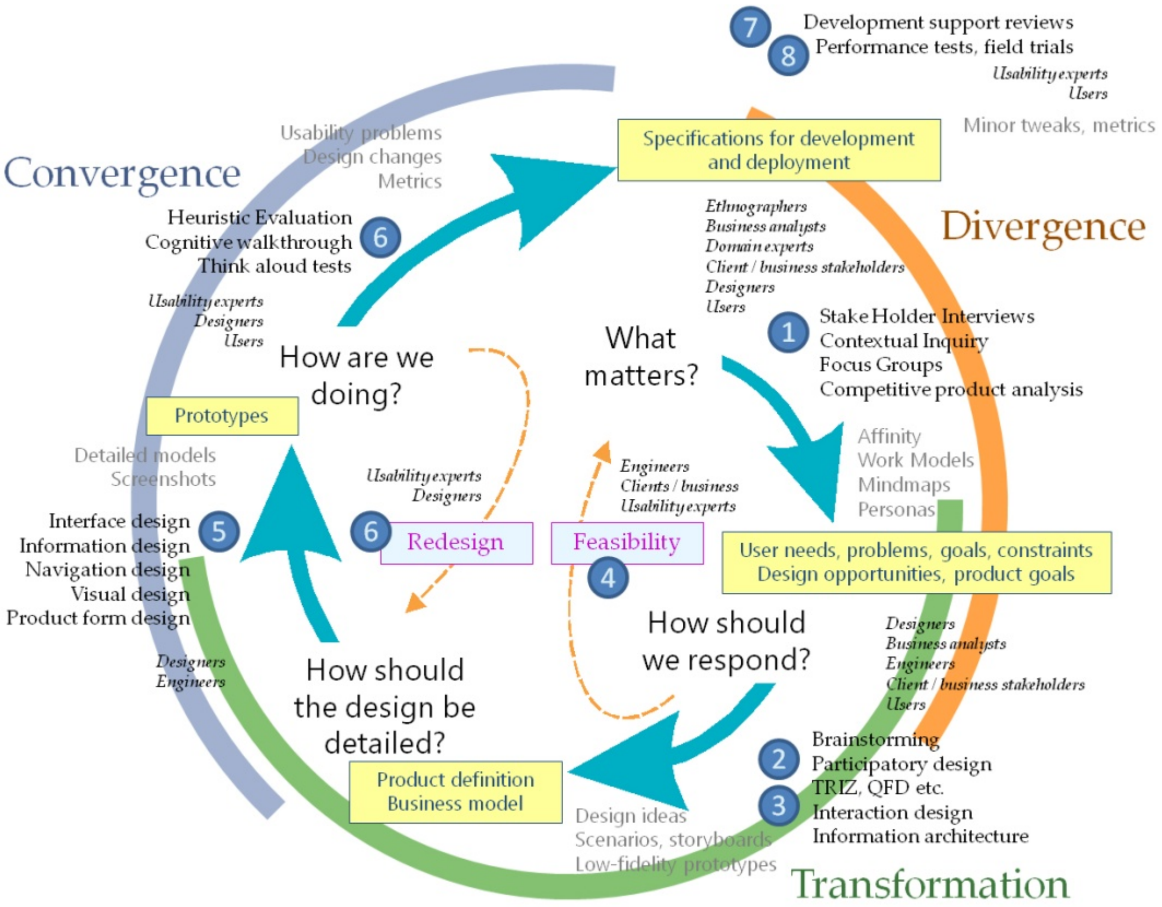


Figure 3.2: Divergence, transformation and convergence and the HCI design techniques and methods.

The names of these professions mentioned above, viz. ethnographers, business analysts, domain experts, designers, engineers, usability experts, and business stakeholders, have been used in a general sense. We do not imply that a qualified professional must be involved to do each of those activities (though it is imaginable that some projects may do so). It is customary, and perhaps even desirable in some contexts, that one person plays several roles. We have come across HCI practitioners who do ethnography, usability evaluation and write a bit of code to make prototypes, in addition to taking design decisions. Several designers prefer to do the formative evaluation themselves to save on time and iterations. Often during such evaluations, they modify the prototype on-the-fly, blending design and evaluation into a single activity. We have seen usability experts and occasionally engineers do ethnography and design with equal vigour.

In another sense, we have used the term 'designer' quite broadly to include interaction designers, information architects, visual designers and product designers. Often these roles are separated, or at least people prefer one title to the other, usually to reflect their educational background or the nature of their interests. We have also seen cases where people super-specialise in narrower areas: typography, information visualisation, layouts, colour, or animation. We have ignored these differences for brevity. If specialisation or generalisation of skills is meaningful in a project context, the activities presented here will need to be reorganised accordingly.

3.3 Summary

In this chapter, we defined a framework in which HCI activities, outcomes and disciplines fit together. We identified HCI activities that could be considered essential for integration with SE process models and some of the important methods and deliverables of these HCI activities. In chapter 7, we use this framework as the basis to integrate the HCI activities in SE process models of waterfall, agile and RUP. In chapter 12, we demonstrate the relative contributions of the various HCI activities discussed in this chapter with the help of data from 61 industry projects.

In the next chapter, we review the SE process models of waterfall, agile, and RUP from an HCI perspective and identify issues with each of them.

PART II – PROPOSALS FOR INTEGRATION

4 Human Computer Interaction Issues with Software Engineering Process Models

The importance of a systematic process in software development was realised quite early. Several SE process models have been proposed and many of these have matured in use. In this chapter, we review three popular software engineering process models, viz. waterfall, agile, and RUP, and analyze them from a HCI perspective.

4.1 Waterfall Model

The waterfall model is the classic process model for software development, and is one of the oldest paradigms for SE. Several variations of the waterfall model exist with minor differences in nomenclature and grouping. Winston Royce was the first to propose a version of the waterfall model (Royce, 1970). Royce proposed the following activities for software development of a large software system that is meant to be used not only by its creators, but by others as well:

- System requirements
- Software requirements
- Preliminary program design
- Analysis
- Program design
- Coding
- Testing
- Operations

In more recent times, Pressman organised the activities of the classical waterfall model in terms of five phases (Pressman, 2005 pp. 79-80) (Figure 4.1):

- Communication (project initiation, requirements gathering)
- Planning (estimating, scheduling, tracking)
- Modelling (analysis, design)

- Construction (code, test)
- Deployment (delivery, support, feedback).

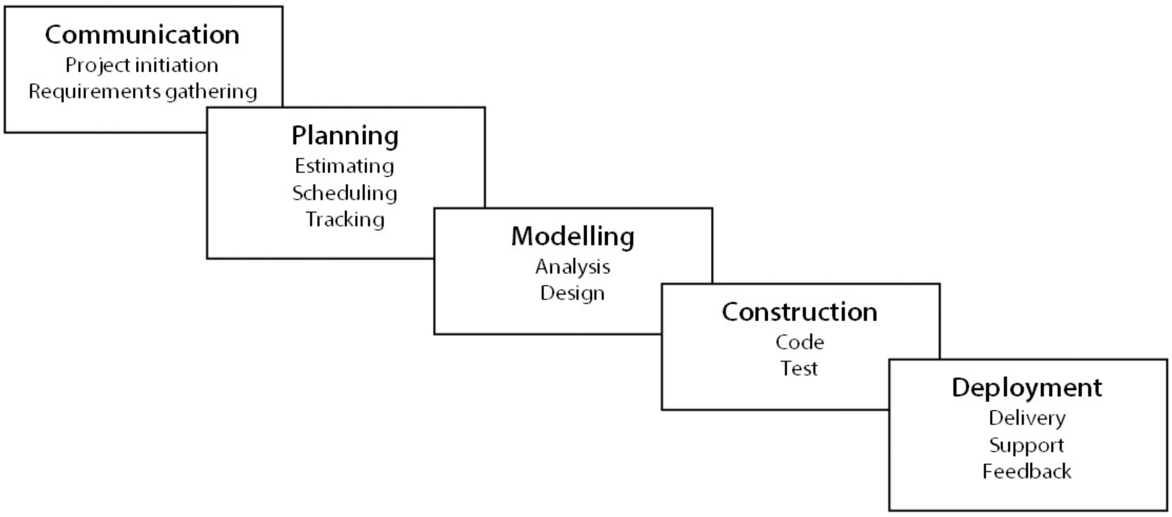


Figure 4.1: Waterfall model (Pressman, 2005).

The key idea behind the waterfall model is that artefacts flow directly from the previous phase to the next. The purpose of the sequence of its activities was to tackle high-impact changes sooner to control the cost of software development.

However, the linearity of the *"waterfall"* was never meant to be taken literally even in its original proposal. Already in 1970, Royce comments that the testing phase is too late in the process when timing, storage, input / output are experienced for the first time. Major errors in earlier stages, if undetected until such time, will be disastrous. Therefore, the waterfall model by Royce made provision for feedback loops. Royce hoped and believed that such iterative interaction is limited to the successive steps but feared that iterations are never confined to successive steps. This is because the software development phenomena are not precisely analysable, and at times fail to satisfy external constraints. In that case, a major redesign effort is required and requirements must be modified or substantial change in design is required. Royce suggested two development iterations, *"so that the version finally delivered to the customer for operational deployment is actually the second version"* (Royce, 1970).

The waterfall model has been critiqued frequently. Hanna says that the product takes too long to develop with the waterfall model (Hanna, 1995). Moreover, all the requirements are often not know in advance. Even if these could be specified, requirements could change during the duration of the project. Pressman comments that real projects rarely follow the sequential flow that the model proposes, even with the iterations (Pressman, 2005 p. 79). On the other hand, a majority of organizations that apply the waterfall model treat as if it were strictly linear while planning a project. Iterations are generally regarded as wasteful, though inevitable. Kroll and Kruchten say that the waterfall model with feedback loops leave a lot of team members idle for extended periods, that it still defers integration of code and testing until it is very late and when problems are harder to resolve, and hence is poor at managing risks (Kroll, et al., 2003 pp. 6, 50).

Despite criticisms, the waterfall model continues to be popular. In a survey of 200 practitioners, Neill and Leplante reported that the waterfall model was the most dominant and 35% of the practitioners claim using it (Neill, et al., 2003). The proportion was higher for projects with less than 2-year duration (40%) and among technical respondents (42%) over management respondents (28%). Waterfall is also a convenient process model to learn for the beginner. A variant of the waterfall model is routinely taught in SE courses in universities as the first process model. In a brief study conducted by the author in 2006 (reported in chapter 5), 7 out of 8 software development projects in the IT industry in India used a variant of the waterfall model. In a more recent study in 2009 (reported in chapter 12), 50 out of 61 projects used the waterfall model.

The waterfall model has momentum on its side. People have been using it for a while and they know how it works. Particularly when the software development activity needs to be outsourced to an external organisation, the waterfall model is perceived to have several advantages. It seems to afford clarity to all concerned. Everyone knows what to expect: the customer, the developers, and the managers. Customers can specify expectations early and get to perform acceptance tests at the end to ensure that they are getting what they paid for (or at least so they believe). The developers are told clearly what to build, so their job becomes easier. Freezing the software requirements before development starts seem to make the software projects predictable and allows the managers to estimate projects precisely and deliver within estimates. The practitioners of the waterfall model seem to work around the problems and, to them, the advantages seem to outweigh the disadvantages.

From a HCI design perspective, the waterfall model closely follows the traditional design activities: analysis, design, develop, test, deploy. However, there are several concerns from the HCI design perspective that have not been addressed in the waterfall model, which we will analyse in the next section.

4.2 Human Computer Interaction Issues with the Waterfall Model

4.2.1 Communication and Requirements Specification Issues

The most important issue with the waterfall model is of communication with the customer, and this has been known for a long time. The Standish Group reported that 31% projects are cancelled before completion and 52% run over budgets (The Standish Group Report, 1995). Only 16% projects are completed on time, and only 9% in larger companies. Among factors believed to have contributed to the successful projects, the top 3 were user involvement, executive management support, and clear statement of requirements. Among the factors believed to have contributed to the challenged projects, the top 3 were lack of user input, incomplete requirement specifications, and changing requirement specifications. Among the factors believed to have contributed to the cancelled projects, the top 2 were incomplete requirements and lack of user inputs while changing requirement specifications was the number 6 cause.

In the waterfall model, there is much emphasis on creating unambiguously stated requirements during the Communication phase, but this is easier said than done. SE literature has been readily admitting that *"the hardest single part of building a software system is deciding precisely what to build"*, (Brooks, 1995 p. 199) or more recently, *"effective communication is among the most challenging activities that confront a software engineer"* (Pressman, 2005 p. 133) and *"understanding requirements of a problem is among the most difficult tasks that face a software engineer"* (Pressman, 2005 p. 174). In a review of SE methods, Sutcliffe concluded that the otherwise rich process-methodological research in SE is relatively immature in the requirements area (Suitcliffe, 2005).

One issue that hampers communication with users and stakeholders is the **language** for such communication. As Gulliksen et al. note, use cases, UML diagrams, or requirements specifications are not sufficient to give an idea to users and stakeholders about what to expect in a future system; prototypes, sketches, mock-ups, simulations, and scenarios are better (Gulliksen, et al., 2003).

The other issue in communication is that of **techniques**. SE literature talks about very basic techniques related to interpersonal communication between people: *"listen"*, *"prepare before you communicate"*, or *"negotiation is not a contest or a game, it works best when both parties win"*, but little beyond this. The phrase *"requirements elicitation"* (Pressman, 2005 p. 184) betrays an assumption that the users know what they want (and that must be the right design decision), and we have to just ask them and help them write it down. The oft-used phrase *"requirements gathering"* conjures an image in the mind's eye that requirements are like flowers fallen overnight under a *parijatak* tree, and one needs to only gather them for the morning *pooja*.

It is well known in the HCI community that one cannot simply ask users what they want and base our designs on that. Nielsen says that users often do not know what they want or what is good for them. They have a hard time in predicting how they will interact with a potential future system with which they have no experience. Further, users may have divergent opinions (Nielsen, 1993 pp. 11-12). Beyer et al. say that sometimes users may know the general area of the problem, e.g. *"installation is the number one problem"*. Nevertheless, they may not be able to articulate what is wrong there with installation in a way that a suitable solution can be designed (Beyer, et al., 1998 pp. 30-33). At other times, users ask for features that they believe will work, but in reality, there may be a much better way of solving the problem if the problem is considered in the original.

Even if users and stakeholders were able to articulate all their needs precisely, this would still gather only the expressed needs. This will not unearth the latent needs. Identifying users' problems, goals, wants, and needs has to be a process that is a lot more active than merely asking users what they require from the software. Beyer and Holtzblatt give this example: when someone checks his answering machine as soon as he walks in to the office, this action tells a keen observer that the user wants to know at once who has tried to reach him when he was away, and might suggest a design idea of a visible / audible notification of a missed communication attempt. However, nothing in the user's environment *"declares the requirement of a blinking light"*. HCI practitioners have to make this leap (Beyer, et al., 1998 p. 17).

Instead of asking users what they want and putting it into the design blindly, it is much better to directly watch what users do, identify design problems and opportunities and then design a system that solves the problems and realises the opportunities. The HCI community has borrowed, developed, and articulated a variety of techniques that are suitable for such observation and analysis (Beyer, et al., 1998), (Kuniavsky, 2003), (Hackos, et al., 1998). SE literature refers to these techniques only obliquely, if at all, and does not suggest how to integrate these in the waterfall model.

4.2.2 Lack of HCI Design Process for HCI Design Decisions

Several HCI design decisions happen during the communication phase when the use cases are specified. Firstly, most interaction designers would argue that the use case scenarios are very much a deliverable of interaction design as they are of business analysts or requirements engineers. Further, in requirements documents used in the industry (and in SE literature) use case scenarios are accompanied by wireframes of the user interface, which are sometimes extremely detailed e.g. (Pressman, 2005 pp. 191-195). Usually, little or no HCI design effort would have taken place before these use case scenarios were written. Typically, an HCI practitioner may not be even involved at this stage of the project. Many of these HCI design decisions can be considered as premature and potentially inappropriate. Typically (and particularly in contracted software development projects following the waterfall model), once something gets written up in a requirements document it gets *"set in stone"* and changes, however desirable, become uphill after that.

This is *not* to imply that HCI design decisions should *not* be taken during the communication phase. In the spirit of the waterfall model, it is appropriate to make some key HCI design decisions during communication phase as long as they are done consciously and after following the HCI design process. However, there seems to be no acknowledgement in SE literature that requirements specifications already specify HCI design decisions. This objection perhaps stems from terminology differences. Most analysts would object to an allegation that some *"design"* already happened in the requirements specifications. In their minds, the term *"design"* is reserved to the software design activity during the modelling phase.

In a footnote, Pressman does admit that it would be reasonable to include *"interface analysis"* in requirements analysis *"since requirements analysis issues are discussed there"* but eventually prefers to place his section of interface analysis as a part of software design because *"interface analysis and design are intimately connected to one another, and the boundary between the two is often fuzzy"* (Pressman, 2005 p. 365). Pressman does mention several user and task analysis activities and user interface design activities but does not attempt to integrate these with the rest of software development process.

4.2.3 HCI Aspects Considered Superficial

There seems to be a tacit assumption in SE literature that HCI design is merely *"skin-deep"* and it is too early to involve HCI practitioners during the requirements specification phase. Some SE literature even explicitly warns against this. Even as late as during modelling, Rumbaugh et al. urge software engineers to *"concentrate first on information flow and control rather than the presentation format"* because, after all, *"the same program logic can accept inputs from command lines, files, mouse buttons, touch panels, physical push buttons or remote links, if the surface details are carefully isolated"* (Rumbaugh, et al., 1991 p. 172).

This approach at best only considers the skeleton and surface planes of Garrett's model of user experience discussed above. It ignores the existence of the structure plane of user experience, and gives much responsibility of the strategy and scope planes of the user experience to the customer rather than a professional designer. Usability evaluations often throw up issues that deeply affect the software architecture. John and Bass identify 27 such architecturally sensitive usability issues that are *"deeper than skin"* (John, et al., 2004). Such surface-led outlook to HCI design is criticised as trying to *"paint lipstick on a bulldog"* (Cox, 1997 p. 15).

4.2.4 Alternatives Not Considered before Design Decisions

SE literature does not mention or demonstrate the importance of considering alternative HCI designs. Considering alternatives as a part of the design process is a common practice in almost all design disciplines. This opens the possibility of combining solutions or at least choosing the most appropriate solution. This is of particular importance in the human-facing aspects of a product. Envisioning many solutions forces the designer to think differently deliberately. One approach designers use is creating rough ideas before creating detailed, fair drawings. Buxton calls this *"sketching user experiences"* (Buxton, 2007). Ideas rendered roughly are easier to critique and modify than fair, finished drawings. This approach is completely missing in SE literature, particularly in reference to HCI design decisions.

4.2.5 Responsibility of the Design Pushed on Customers

Designers and the software development organisation (and not users or customer organisation) need to accept the responsibility of the design. Statements in SE literature seem to suggest a tendency to the contrary. For example, "*As the analysis evolves, certain elements will become relatively stable, providing a solid foundation for the design tasks that follow. However, other elements of the model may be volatile, indicating the* ***customer*** *does not yet fully understand the requirements for the system.*" (Pressman, 2005 p. 196) (emphasis ours). We ask, what happens in cases where a customer does not have any internal core competence in software development, for example, an airline, a bank, or a hospital? Elsewhere, Pressman warns that "*if an organization does not understand how to manage and control software projects internally, it will* ***invariably struggle*** *when it outsources software projects*" (Pressman, 2005 p. 46) (emphasis ours). This sounds like *"if a patient does not understand how to manage and control her own health, she will invariably struggle when she gets admitted to a hospital"*.

The professional has holistic responsibilities, both stated and assumed. The design professional should strive to make the product easy to make, easy to use, easy to maintain, easy (on the environment) to discard, and yet safe, sustainable, empowering (or at least not disempowering), and appropriate, among other things that may matter in that context. These responsibilities are not for the customer to specify in a contract or a document. The customer may take final calls when offered choices, particularly when trade-offs are involved, but the designer needs to make holistic, considered recommendations first. The customers sign-off should be equivalent to an informed consent given by the patient to a professional doctor.

4.2.6 Inside-out View of Product Development

SE literature largely presents an "inside-out" view of product development. A large majority of concerns in SE literature are related to how we can make sure that we develop software optimally, maintain it well, ensure that it will not crash etc. Few concerns go beyond software development and into how we can respond to the external problem at hand.

This reminds us of the early days of the automobile industry. To get a car to work was a significant achievement in those days, and products were routinely designed inside-out. As the industry matured, this view changed. Product development and automobile engineering continue to be important in an automobile industry today, but over the years, the industry has grown in confidence to allow most cars of today to be built outside in. A similar trend can be seen in the consumer electronics industry.

Design is a goal-driven, creative, problem-solving process. The *problem* is necessarily external to the design process. As the field of software engineering matures and grows in confidence, it will become important to start looking at the problem outside more often than the internal problems.

4.3 Agile Process Models

While the waterfall model has come to represent linearity, agile process models have come to represent the iterative nature of software development. While early ideas of agile programming were expressed back in the 1970s (Baker, 1972), several process models that emerged in the last 15 years fall under the broad umbrella of *"agile"*. Pressman summarises seven agile process models: Extreme Programming, Adaptive Software Development, Dynamic Systems Development Method, Scrum, Crystal, Feature Driven Development, and Agile Modelling (Pressman, 2005 pp. 103-124). These process models may vary in their details, but they have several common elements best captured by the *"agile manifesto"* (Agile Manifesto, 2001):

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

The last point is particularly important. In agile processes, it is typical to solve a small part of the problem to begin with and to grow the solution in iterations. Agile processes believe that changes in software requirements will necessarily happen. Agile processes are designed to accommodate changes even late in the process to *"harness change for the customer's competitive advantage"* (Agile Manifesto, 2001). Promoters of agile processes use phrases such as *"embrace change"* or *"change is welcome"* to emphasise this. As changes may affect the software architecture, agile methods rely a lot on *"refactoring"* the architecture during iterations.

Fowler lists many reasons why requirements change, and in fact why they *"ought to be changeable"* (Fowler, 2005). Firstly, customers cannot recognise what options they have while specifying requirements. Even if they could, they cannot make an informed decision at this stage primarily because the cost to each new requirement cannot be predicted right up front. Fowler gives several reasons why costs cannot be predicted early: Software development *"is a design activity"*, and thus hard to plan and cost. Further, the basic ingredients of software keep changing rapidly. In addition, costs are dependent on the individuals involved and their experience. Finally, software is intangible and yet malleable. Only when they use an early version of some software do the customers really begin to understand which features are valuable and which are not (Fowler, 2005). Even if we could get an accurate and stable set of requirements early, Fowler believes that *"you are still doomed"*. The fundamental business forces in today's economy are so dynamic that every six months, new requirements are likely to emerge.

In agile processes, the main measure of progress is *"working software"*, and the proponents look down upon *"analysis paralyses"* and BDUF (Big Design Up Front). Agile methods deliver working software in small pieces, but frequently, sometimes as frequently as once a week, but no later than once in two months. This length of time forms a *"heartbeat"* for the project and helps maintain pace. Agile methods also insist that this needs to happen smoothly, without the developers working overtime.

Each iteration of an agile process follows a mini-waterfall within itself. Sufficient requirements are expressed, analysed, the software architecture is refactored if necessary, the code is written or re-written, tested and released. If some requirements could not be completed in the current iteration, these are carried over to the next iteration.

Agile methods do not plan a timeline for the whole project. Because new versions of the software are constantly being released, it makes it easier for everyone (including the customer) to see a *"momentum"* in the project. This makes it easier to estimate the time needed to achieve the overall vision of the project and to make course corrections.

While testing is important in all software process models, agile methods especially emphasise on testing. Agile methods suggest not only testing the current version of the product, but to set up automated testing procedures so that testing is frequent and when changes happen during iterations, the automated regression testing detects the breaks soon. Automated regression testing is particularly important because the software is released often and manual testing each time would be wasteful.

Agile methods depend a lot on teamwork and internal communication. It is believed that best architectures, requirements, and designs emerge from self-organising teams. Developers work alongside customers during the development. There is usually little documentation, but there is a lot of emphasis on face-to-face communication between team members. Pair-programming (programming done by two developers together) and daily *"stand-up"* meetings (that last no more than 15 minutes) help in maintaining communication going among team members.

HCI processes share several qualities with agile processes. HCI design is intrinsically an iterative process consisting of analysis, design, and usability evaluation. The problems found during the evaluation are fixed in the next iteration. Such iterations continue until no problems are found and user experience goals are met. Given this preference for iterations, agile methods seem a good fit for integrating HCI activities within the agile processes. The emphasis on people and deliverable products rather than documentation and planning are also common qualities. Just like agile programmers, designers are more of doers. The informality of the agile methods gels well with the informal culture of design. Designers are more at ease in face-to-face communication and visual presentation of ideas than with wading through long documents. Most critiques agree that there is potential to integrate user-centred activities with agile development. Nielsen acknowledges that agile methods hold promise for addressing the many ways in which traditional development methodologies erected systematic barriers to good usability practice (Nielsen, 2008).

However, despite the similarities, several HCI issues still emerge with agile process models, which we will discuss in the next section.

4.4 Human Computer Interaction Issues with Agile Processes

4.4.1 Software Engineers Are Asked to Design

The most important issue with agile process models is that they pay little attention to users, usability, and HCI design. It effect, agile methods do not acknowledge that HCI activities require a different set of specialised and important skills. This is reflected in the team composition. Agile teams primarily consist of software engineers, and working code is considered the primary deliverable. Anyone who does not deliver code (e.g. a designer) does not easily fit in culturally.

Several critiques have reflected this view. Blomkvist comments that though agile processes value people, skills, and teamwork in other areas, they do not regard that usability and interaction design skills as important (Blomkvist, 2005). Nielsen identifies *"threats"* of agile methods (Nielsen, 2008). The biggest threat, according to Nielsen, is that agile methodologies are developed by programmers to address the implementation side of software development, overlooking HCI design. While Nielsen is not against HCI design being performed by the same people who do the coding, he feels it must be recognised as a separate activity rather than leaving it to happen as a *"side effect of coding"*. Beyer et al. suggest that the separation of design from engineering in agile methods helps engineering good products, but does not help the engineers find answers to the basic design questions such as what the scope of the product ought to be, what the basic function and the structure will be and what the details of the user interface will look like (Beyer, et al., 2004). Constantine concludes that agile methods seem to be at their best in applications that are *"not GUI intensive"* (Constantine, 2002).

4.4.2 Users Are Asked to Design

To help design a new system, agile methods put representative customers or users in the team. While this may give a false feeling to the development team that the voice of users is being heard, this may not be the best way to create good designs.

This point is supported by many critics. Bayer et al. argue that there is no such thing as representative users. At best, they are a sub-set of users and often, they only represent themselves (Beyer, et al., 2004). Further, even real users are unable to articulate what they do and how, particularly when they are not in the context of that work, and certainly if they have not been doing the work for a while. Finally, users are not able to make design decisions for a new system. Users may not have the appropriate skills required to create visions of future systems. It has been known for a while that users cannot answer speculative questions and have a hard time even predicting how they will interact with future systems (Nielsen, 1993 p. 12). Such a user representative may be called upon to make all HCI design decisions, a skill he may not personally possess as users are not designers (Nielsen, 1993 pp. 12-13), (Blomkvist, 2005). Alistair Cockburn reportedly summarised that *"[user involvement] is not a weak point [in agile methods], it is an absence"* (Constantine, 2002).

Design of interactive systems requires a complex set of skills and it is inappropriate to assume that all representative users would have it. User should be involved, but not for making the design decisions. Skilled HCI practitioners can design good systems by observing users in their contexts, by involving them in participatory design activities, or by asking them to try out prototypes during usability tests.

4.4.3 Change is Managed Well, But Anticipated Poorly

Agile methods plan very little up front because it is assumed that the business needs and requirements will change any way. However, as Allen Cooper puts it, this is a self-fulfilling prophecy. Requirements change because planning is avoided (Cooper, 2008). Managing change is one of the strengths of agile methods. As a result, agile methods shun Big-Design-Up-Front. In doing so, they seem to be throwing out the proverbial baby with the bathwater. Agile methods do not seem to differentiate between elaborate planning and deeply understanding user needs, between software design and design for human beings, and between intra- and extra-lifecycle changes. They tend to club these in one basket and shun them equally.

We categorise changes to the HCI design into five types:

- Changes that arise because a **new user need or user problem** is discovered after requirements are frozen.
- Changes that arise because someone thinks of a **new idea** after the requirements were frozen.
- Changes that arise because something that was thought to be **technically feasible** turns out not to be so and a workaround is required.
- Changes that arise because late usability evaluations of early releases throw up **unanticipated usability problems** that were not captured on early prototypes. Some problems can go unnoticed in usability tests on low-fidelity prototypes. For example, effects of delays in the interface operation or page load delays. These could be caught only on a version of a working product.
- Finally, changes that could not have been anticipated. A new technology suddenly becomes popular, a war breaks out, a new law is passed, recession upsets the market dynamics, or a competitor launches an innovative feature that must be accounted for. We call these **extra-lifecycle changes**.

We posit that if this design process is followed, changes in all categories can be minimised. Particularly, it is important to differentiate between intra-lifecycle changes (the first four types) and extra-lifecycle changes. The intra-lifecycle changes are likely to arise within the lifecycle of the project and can be minimised by following the design process. For example, changes that arise of poor understanding of user needs or use context well can be almost eliminated through careful user studies. If sufficient efforts are put in ideation with multi-disciplinary teams, we can minimise (though perhaps not eliminate) 'new idea' type changes. If the product definition is evaluated by a multi-disciplinary team before requirements are written, chances are that feasibility issues will be caught early. If the early prototypes were sufficiently detailed, usability evaluations of products should throw minimal surprises.

Extra-lifecycle changes happen because of sudden and relatively unpredictable change in the environment beyond the project lifecycle. For example, a new technology can suddenly become popular. At the time of writing this (2010), 3G mobile telephony has recently been introduced in India. It is expected that 3G telephony will have an impact in India, possibly in the next 10 years. At this point, it will be fair to assume, though, that 3G telephony will be available only to a small number of subscribers for the next three years. However, if 50%+ cell phone subscribers in India start using 3G telephony by the middle of year 2012, it would be fair to consider that as an extra-lifecycle change that could not have been anticipated.

However, many changes in the software development are *not* of this nature. Some changes are requested for because the development team did not anticipate some needs of the user (e.g. *"voters need clear and immediate feedback when they cast their votes on an electronic voting machine"*). Other changes are requested for because the new product is incompatible with what users are already familiar with (e.g. *"change the shortcut for "save" to Control+S and throw a dialog box if the user tries to quit before saving"*). Some changes are requested for because after seeing the software, the users can clearly see what is missing that could have been easily done (*"send the borrower an email notification when her library book becomes due"*). Some changes are requested for when the company opens a new line of business (e.g. *"the state-owned airline finally decided to sell tickets online – like all their competitors"*). All these changes arguably arose from latent (but unexpressed) needs of users and businesses that are supposed to be studied before the product is designed. Many such changes, if not most could be anticipated through some relatively minor investments in HCI activities. Beyer et al. feel that though features and technologies might change rapidly, the underlying human work practice changes little (Beyer, et al., 2004). Lack of knowledge about work practice slows development teams, whereas deep understanding leads to speed in development.

Just because they manage change through refactoring, agile methods seem to give a licence to do a poor job at anticipating and containing change. Surely, unnecessary refactoring takes its toll on the product and the budget. Proponents of agile methods seem to do little introspection about the reasons for intra-lifecycle changes, which are the most common type of changes in projects.

Agile methods could do their promised good job at managing the un-anticipatable extra-lifecycle changes, when they occur. HCI activities can help in anticipating many of the intra-lifecycle changes that arise out of human needs and business processes.

4.4.4 Agile User Stories Are Not Interaction Design Scenarios

Agile teams use user stories to define, manage, and test features of a product. It is tempting to think of these as parallel to scenarios in interaction design and to think of stories as a direct link between HCI and agile methods. However, a closer look at tells a different *"story"*.

Agile user stories are written by customers, focus on the user interface of one feature, and are supposed to be about three sentences long (Wells, 1999). The length of the story is determined by the time it takes to implement it in code. User stories are chopped and churned during estimation. Anything that might take longer than 3 weeks to code needs to be broken down into smaller stories, anything that might take less than 1 week should be merged with something else. In terms of contents of the stories, almost any development task (such as database design or network configuration) qualifies as a *"story"*.

Scenarios in interaction design are lot richer than three-sentence-long user stories. They are created by designers to envision new products. A scenario may involve more than one feature and may involve one or more personas. Scenario narratives are never only three sentences long, are often accompanied by storyboards or videos, and only sometimes describe details of the user interface. The main purpose of a scenario is to explain the high-level impact of the future product on the life of the user in a particular situation (Cooper, et al., 2003 pp. 77-82). It is difficult to imagine how a scenario can be chopped or merged just so that it can be developed in three weeks.

There could be a link between scenarios and stories though. A scenario could be a starting point from which several agile stories are derived. While stories could be used to develop the individual features within the heartbeat of the project, the overarching scenario can help the team understand and keep track of the overall context.

4.4.5 Short Iterations

An important HCI issue is that breaking down product development into small parts and constant change can potentially undermine the totality of the user experience. While some HCI researchers have no issues with this, a few have critiqued this of agile methods (Constantine, 2002), (Nielsen, 2008). Piecemeal design could lead to lack of cohesiveness and allow inconsistencies to creep in. Maintaining a comprehensible and consistent user interface as new features are added becomes increasingly difficult.

Short iterations cause further problems as the usability team tries to maintain the heartbeat of the project. While *"discount usability methods"* such as heuristic evaluation are relatively easy to integrate in short iterations (Nielsen, 2008), the typical 1 to 3-week sprints makes it extremely difficult to integrate longer, richer activities such as contextual inquiry or detailed usability evaluation. The risk is that such activities will never be done at all.

4.5 Rational Unified Process Model

Rational Unified Process (RUP) is a process model, a software development approach and a process product that provides a customizable process framework (Kroll, et al., 2003), (Kurchten, 2004). RUP process model is an iterative, risk-driven, use-case driven, architecture centric process model. Its timeline is divided into four phases: Inception, Elaboration, Construction, and Transition. The phases of RUP are not just renaming of the phases in the waterfall model. Within each phase, there are typically one or more complete iterations. In each iteration, almost all activities related to software development are followed (Figure 4.2).

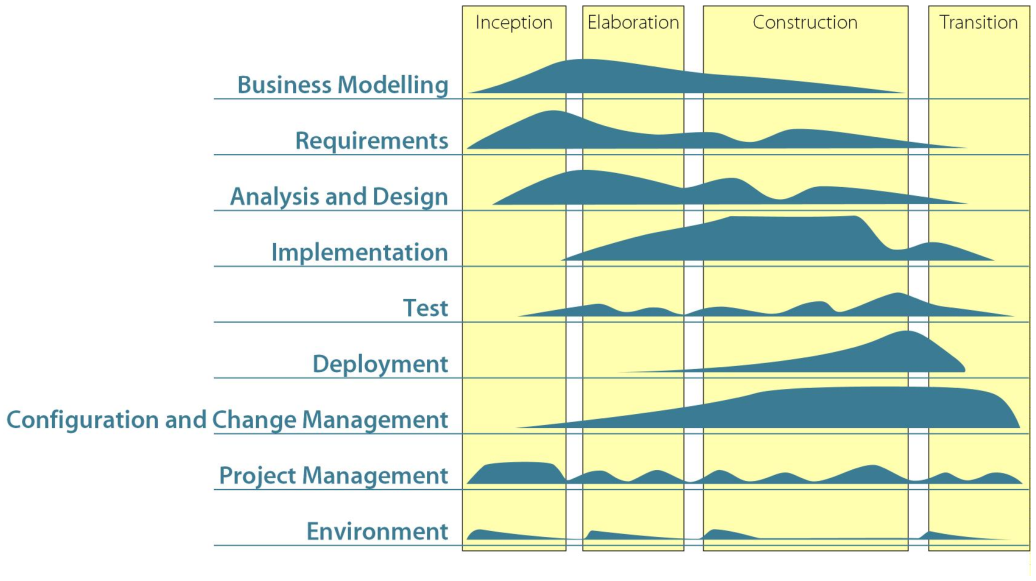


Figure 4.2: The two dimensions of RUP (Kroll, et al., 2003)

At the end of each RUP phase, certain risks are mitigated and a specific milestone is achieved. At the end of **inception** phase, the project achieves the Lifecycle Objective milestone. At this point, stakeholders decide if the project is financially feasible and worth investing into. At the end of the **elaboration** phase, the project achieves a Lifecycle Architecture milestone. At this point, the technical and software architectural risks are mitigated and a stable version of the software architecture is finalised. At the end of the **construction** phase, the project reaches an Initial Operational Capability Milestone, where all the logistical risks related to product development are mitigated. Finally, at the end of the **transition** phase, the project reaches the Product Release Milestone, where risks related to deploying the product to its user base are mitigated. The four phases of RUP are designed to attack major risks first and to accommodate the big changes early.

RUP recognizes a multi-disciplinary approach to software development. Its process model is two-dimensional (Figure 4.2). The horizontal axis represents the dynamic structure of the process along the time dimension, described by the phases. The vertical axis represents the static structure of the process, capturing how the various disciplines participate in the process. Each discipline has a set of well-organised workflows that define one or more roles, activities, and artefacts. For example, the requirements discipline consists of roles such as system analyst and requirements specifier, activities such as analyse the problem, understand stakeholder needs, define the system, manage the scope of the system, refine the system definition, and manage changing requirements. The artefacts coming from this discipline include a vision document, a use-case model, use cases, storyboards associated with use cases, (which will serve as the basis for user-interface prototypes), supplementary specifications, and a glossary (Kurchten, 2004). The graphs corresponding to the disciplines represent the contributions of each discipline on an ongoing basis during the phases. For example, the activities of the requirements discipline peaks during the inception phase, though it carries out through to the transition phase. (Unfortunately, as we will return to in the next section, RUP ignores the disciplines and roles related to HCI.)

Kroll and Kruchten characterise software engineering process models by considering two dimensions (Figure 4.3). The low ceremony / high ceremony dimension maps out the level of formalism that the process model requires. High ceremony processes rely on high levels of documentation, traceability, formal committees etc. Low ceremony processes produce minimal supporting documentation and little formalism in working procedure. The non-iterative / iterative dimension maps the level of iterations in the process. Processes on the non-iterative side of the dimension are linear with late integration and testing. Iterative approaches are risk driven with early implementation of an architecture and early integration and testing (Kroll, et al., 2003 pp. 49-65).

Kroll and Kruchten position the waterfall process model high on ceremony and low on iteration, while the agile process models are less on ceremony and more on iterations. As the agile processes mature and are used in larger projects, Kroll and Kruchten expect that they will be more formalised, with more guidance for successful project execution. They position the Capability Maturity Model (CMM) assessment framework on a scale that is even more formal than the waterfall model. They position the Capability Maturity Model Integration (CMMI) (Software Engineering Institute, 2006) assessment framework somewhat lower on the ceremony axis, and the one supporting iterative development. RUP, while always iterative, is quite flexible on the ceremony axis, and Kroll and Kruchten claim that this allows RUP to be adapted for a wider variety of projects, ranging from small, one-person, one-week projects, to large, multi-location, multi-person-year projects.

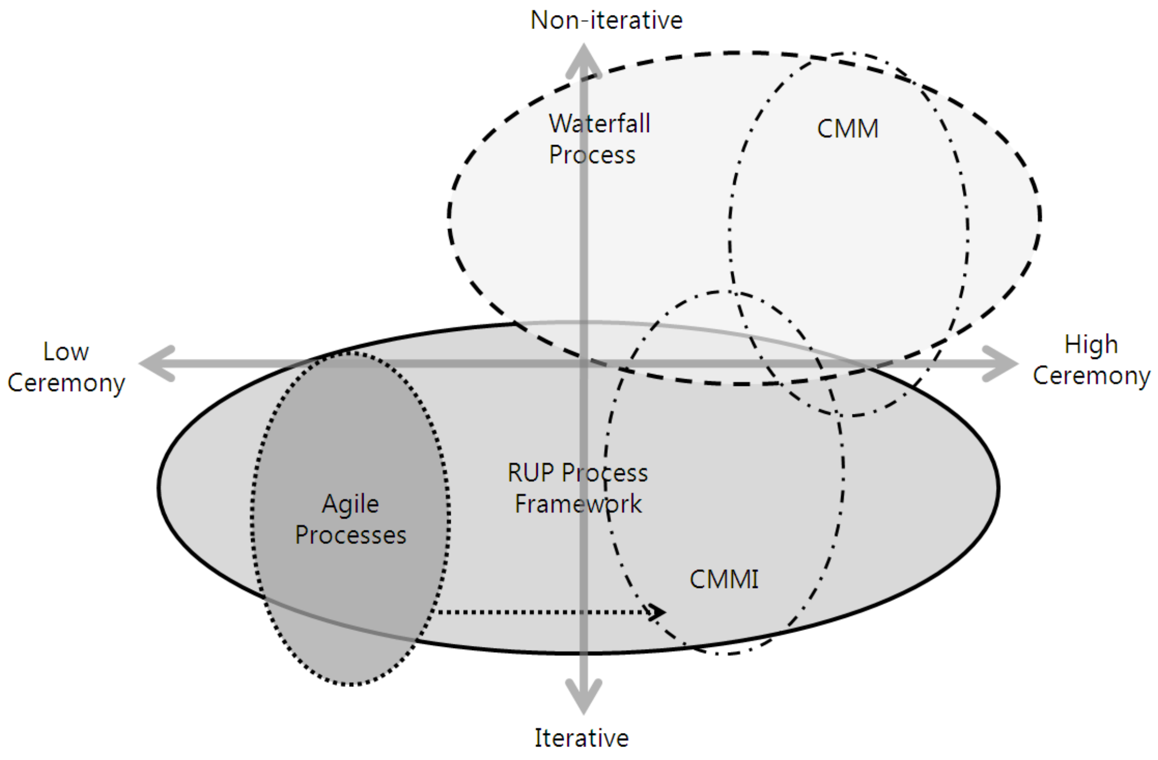


Figure 4.3: Process characterisation based on ceremony axis and iteration axis, adapted from (Kroll, et al., 2003 pp. 49-65).

RUP has several strengths relevant to software development in general and is and potentially amenable towards the integration of HCI activities. RUP intrinsically recognizes that software development is multi-disciplinary. RUP stresses on a milestone driven and iterative approach and manages risks well by attacking the most important risks first. The dynamic dimension describes an iterative process with four phases that is quite suitable for HCI design. RUP is flexible along the *"ceremony axis"*. It can be used by large, formal teams as well as small, informal ones. In spite of these strengths, RUP has several shortcomings from an HCI perspective.

4.6 Human Computer Interaction Issues with Rational Unified Process

4.6.1 Not User-Centred Enough

RUP has been critiqued for not allowing for sufficient user involvement. Göransson et al. felt that RUP is not a user-centred method (Göransson, et al., 2003). They point out to fundamentals in RUP that prohibits a true user-centred design (UCD) process. They also point to the difference between the iterations in RUP and UCD. Within iterations, RUP is very much like waterfall, while iteration in UCD is a refinement of a part of the system to achieve goals. Though RUP is multi-disciplinary, there are differences with UCD here too. In RUP, there is too much emphasis on generating artefacts and documents and project members are encouraged to get *"his or her work done"*. Little collaborative work across disciplines actually happens. In addition, they found little support to design the user interface within RUP.

4.6.2 Stakeholders Are Asked to Design

Like waterfall and agile, RUP assumes that we can simply ask the stakeholders what they want. Here stakeholders include customers, management, analysts, developers, testers, technical writers *"and other key people"* (Kroll, et al., 2003 pp. 94-102). It is generally assumed that stakeholders already know what needs to be created and can articulate it well. *"But how do we gather those needs? It is important that we actively* ***gather*** *all types of stakeholder requests…"* (Kurchten, 2004 p. 164). And *"typically, the domain and the subject matter experts* ***know*** *what this functionality is from the user perspective."* (Kroll, et al., 2003 p. 103). However, as noted above, users and stakeholders often do not have the ability to envision a future system, and anticipate and answer questions about it. Even if they could, this only gathers the expressed needs, and leaves out the latent needs. The requirements for such a system will inevitably change.

There is an acknowledgement that *"understanding what to build"* is a problem. *"This may sound* ***strange****, but the fact is that in many projects there is no common understanding of what needs to be built. Although all team members think they know, often each one has a completely different understanding than the next."* (Kroll, et al., 2003 p. 96).

Unfortunately, the creative design problem is merely viewed as an intra-team communication issue of bringing everyone on the same page.

4.6.3 HCI Practitioners and HCI Activities Are Not Represented

Like agile methods, RUP too does not recognise the roles of HCI related disciplines. While RUP is a multi-disciplinary framework with the static dimension dedicated to highlighting the roles of various disciplines, usability, interaction design, or information architecture skills are conspicuous by their absence.

RUP literature is at best muddled and at worst, misleading on when and how to design the user interface**.** RUP acknowledges the importance of usability and lists it as the first non-functional requirement (Kroll, et al., 2003 p. 307), (Kurchten, 2004 p. 159). Kruchten differentiates between *"user needs"* and *"system requirements"*. Though he does not link this discussion explicitly to usability, he insists that one must understand the *"what"* of a system (stated in requirements such as *"the system shall do xyz"*) and the *"why"* of the system (the user need) because this would help the team to do a better job of interpreting the requirements as well as to make tradeoffs. Unfortunately, *how* the user will interact with the system is not that well addressed.

On the other hand, RUP seems to warn against the *"risk"* of spending too much time on HCI design early on: *"It should be noted that there is also clear risk that users, UI designers, and analysts seek to perfect the UI prototypes, which could lead to analysis-paralysis. Your goal is not to finalize the UI; your goal is to get a reasonably good UI to facilitate effective communication with users on what the system should do, allowing you to finalize the use cases more rapidly."* (Kroll, et al., 2003 p. 307). They do not explain why *"finalizing use cases rapidly"* is a better goal than *"getting a perfected UI prototype early"*, particularly since we know that change in those use cases is inevitable.

There are some inconsistencies about when usability work is recommended by RUP literature. Kroll and Kruchten noted about the transition phase, the last phase of the RUP: *"At this point in the lifecycle, all major structural issues should have been worked out, and user feedback should focus mainly on fine-tuning, configuration, installation, and usability issues"* (Kroll, et al., 2003 p. 162). It very much sounds like another *"lipstick on* *the bulldog"* phenomenon. Too little, too late. Usability evaluations often throw up issues that deeply affect the software architecture (which needs to be frozen upon during the elaboration phase of the RUP). As noted above, John and Bass have identified 27 such architecturally sensitive usability issues that are *"deeper than skin"* (John, et al., 2004). Doing usability evaluations early (say during the elaboration phase, perhaps even during the inception phase) could certainly mitigate risks arising out of usability problems that affect software architecture. Doing them during the transition phase just brings in the risk of jeopardising software architecture or shipping an unusable product.

On the other hand, Kruchten seems to suggest that usability should be considered in the requirements discipline (which peaks during inception) (Kurchten, 2004). In the requirements discipline, Kruchten even notes that storyboards should be an artefact developed, which will be later used in user-interface prototypes and user interface design should happen in the analysis and design discipline, which peaks at the intersection of inception and elaboration phases, and in the first iteration of construction phase. In appendix A, Kruchten specifies user-interface designer as a developer role, who is involved in *"gathering usability requirements and prototyping candidate user-interface designs to meet those requirements"*. In appendix B, Kruchten specifies these artefacts associated with the role of user-interface designer, storyboard as a part of the requirements artefact set, and user-interface prototype and navigation map as a part of analysis and design artefact set.

However, Kruchten mentions no activities or roles related to usability or user-interface design in the disciplines of requirements or analysis and modelling. The roles associated with requirements discipline are *"system analyst"* and *"requirements specifier"* while the activities are analyse the problem, understand stakeholder needs, define the system, manage the scope of the system, refine the system definition, and manage changing requirements. The main roles involved in the analysis and design workflow are software architect and a [software] designer (who mainly defines classes) and optional roles include a database designer, a capsule designer and a [software] architecture and [software] design reviewer. Activities include define and refine software architecture, analyse [software] behaviour, design components, and design the database.

The inconsistencies in RUP literature lead to a lack of clarity about when HCI activities have been prescribed and which role would do them. Moreover, several HCI activities, including user studies, interaction design, and usability evaluation and associated deliverables of user models, product definition, usability problems, and refinements are missed out completely.

4.6.4 Implicit HCI Design during Inception and Elaboration

Kruchten noted that the main *input* to the design of a user-centred interface is *"the use-case model, the supplementary specifications and the story boards which are developed in conjunction with the users or their representatives and other key stakeholders"* (Kurchten, 2004 p. 177). As has been identified earlier (section 4.2.2, page 44), use-cases often over-specify interaction design decisions and it is important to follow HCI design process *before* use cases are specified. Perhaps change is inevitable because of this approach. Perhaps designers seek to perfect the UI prototype because that could stabilise the use cases and minimise changes.

In the examples that they give, much of HCI design happens during inception and elaboration phases, though it happens implicitly, unconsciously, and without involvement of the appropriate skills. All the four example projects by Kroll and Kruchten construct a user interface prototype or a mock-up of the most critical use cases during inception or elaboration phases, and at least three of these are for the purpose of better communication with the client (Kroll, et al., 2003 pp. 77, 106-107).

4.6.5 Change is Managed Well, But Anticipated Poorly

HCI issues related to change management with RUP are similar to those with agile process models discussed above. Both RUP and agile process models assume that change in software development is inevitable. Both do a good job of managing change by attacking major risks early. However, little is done to tackle the source of intra-lifecycle change, poor understanding of user needs and lack of design steps, to begin with. Clearly, RUP underestimates the change risk mitigation afforded by HCI design practices.

4.7 Summary

In this chapter, we presented a critique of three SE process models: waterfall, agile and RUP from an HCI design perspective. The main issues with the three process models are that HCI design happens prematurely and unsystematically, the HCI activities have not been integrated with SE process models, and the role of HCI practitioners has been left out of these process models.

In the next chapter, we will look at other gaps between HCI and SE as reported in literature and in industrial case studies analysed by us. In chapter 6, we look at some prior efforts to integrate HCI with SE, including prior work related to integrate HCI with the three process models discussed in this chapter. In chapter 7, we propose our extensions to these process models for integration of HCI activities with them.

5 Human Computer Interaction in Software Engineering: Identifying Gaps

5.1 Literature Review on Human Computer Interaction Related Gaps

SE evolved in the context of the IT industry with the objective of developing *"methods and procedures for software development that can scale up for large systems and that can be used to consistently produce high-quality software at low cost and with a small cycle time"* (Jalote, 1997 p. 13). Over the same period, computer software made its journey from high-end scientific equipment to pervasive, almost invisible consumer products of the new millennium that touch the lives of a large number of people. In the early 1990s, it was estimated that almost half of the code in systems and 37-50% of efforts are related to the system's user interface (Myers, et al., 1992). The share has probably gone up today as computing has become more pervasive. As a result, the importance of user-centred design approaches has gained ground.

The fields of HCI and SE have evolved independently in the past two and a half decades, apparently almost completely unaware of each other until recently. There still exist major gaps between HCI and SE, both in academic institutions and in the industrial practice. The joint working group 2.7 / 13.4 remarked, *"The standard curricula for each field makes little if any reference to the other field and certainly does not teach how to interact with the other field. There are major gaps of communication between the HCI and SE fields: the architectures, processes, methods, and vocabulary being used in each community are often foreign to the other community."* (IFIP WG 2.7/13.4 on User Interface Engineering, 2004). Deliverables of one group are not evidently useful to the other, hampering workflows. There is an apparent disconnect between the priorities of the two groups. These gaps could lead to communication and coordination problems, duplication of effort, compromises in the process, and eventually the quality of the output. There is a need to have shared processes, common techniques, nomenclature, checkpoints, and measures for success (Jerome, et al., 2005).

The main goal of HCI activities is the same as that of SE process models, viz. to deliver a high quality product to the end user to satisfy a human need or to solve a human problem. However, skills and techniques required for doing HCI activities are quite different from the ones required for doing the SE activities. As discussed in chapter 2 and 3, HCI activities utilise a specialised set of abilities, viz. broad-based designer aptitudes such as sensitivity, creativity, ability to synthesise form, visualise, sketch and present ideas, specialised HCI skills such as conducting user studies, analysing user needs, visualising scenarios, user interface design, information visualisation, prototyping, and usability evaluation, and knowledge in the areas such as cognitive psychology, ergonomics and social and cultural issues. These abilities are themselves derived from several disciplines and each may take a lifetime to master. Quite clearly, HCI is a specialisation distinctly different from traditional software engineering and having the people with these skills in the team is important.

As was discussed in the last chapter, merely having HCI skills in the team is not enough. Neither are mere intentions to do produce usable software. The HCI activities must have a process support within SE. Göransson et al. report an example where clear intentions to apply a user-centred design approach ran into several problems (Göransson, et al., 2003). This example is not an exception. As Mayhew says, it is a myth that if there is a usability guru or two around, their expertise will somehow find its way in the design of the product (Mayhew, 1999 pp. 3-4, 406). A cookbook approach that can relies only on general principles and guidelines may not work. Intentions, skills, principles, and guidelines must be supported by a process. The HCI practitioners must have process support before they can deliver good quality usable software.

Firstly, skills must be converted into actionable activities. It is a common complaint by HCI practitioners working in very process-compliant companies that they are seldom called upon to do user studies or carry out usability evaluations or even allowed to meet the users. This happens because the prescribed SE processes adopted by the organisation do not demand that these HCI activities need to happen.

Secondly, the activities must be done at the right time. For example, if requirements have already been frozen before the HCI person is involved in the project, and if the use cases already specify the strategy, scope and structure levels of user experience of the product, doing the activities related to user needs identification, interaction design or information architecture at this stage may be irrelevant.

Thirdly, the activities must result in the right kind of deliverables that must be used by the rest of the team in the intended manner. Though the final deliverable is the user interface design, the HCI team needs to produce several intermediate deliverables. Some of these are for use internally by the HCI team members, for example analysis reports of individual interviews, a description of a persona, or a set of screens tied together through a low-fidelity prototype. Some others are for stakeholders from marketing, technology, or business to respond, for example findings from an affinity diagram, or scenarios in the life of the persona. In addition, some others may be elements that need to be incorporated in the product as is, for example the information architecture, hierarchy, labels in a menu, colour, font, and size specifications, icons and layouts, or raw html pages. All these outputs must be planned for and used appropriately. Tools and techniques need to be put together to enable the HCI team to control their output depending on the results of their own evaluations. Such tools need to go beyond the usual separation of the presentation layer from the business logic.

Finally, there is a need for mutual trust and respect between the team members from both disciplines for each other's activities, deliverables, and decisions. Chamberlain et al. report a field study investigating user centred design in three projects using agile methods, which looked at (among other things) collaboration among team members (Chamberlain, et al., 2006). The problems of communication and power within the project existed between developers and designers as each group defended their discipline. If the HCI design needs to be changed for some reason, the HCI team must be consulted and given an opportunity to propose an alternative rather than implementing a design that was deemed right by the programmers. In turn, the HCI team must review the ongoing development work regularly to keep track of changes and respond to new situations quickly to match development cycles.

In their introductory chapter, Seffah et al. list a set of obstacles in integrating usability in SE (Seffah, et al., 2005 pp. 5-9). One obstacle is the deep-rooted myth that usability is not a central topic of SE. Usability activities are considered easily dispensable by a software project manager when the project is short on budget or time. Another obstacle is the ambiguity associated with usability, the different meanings it presents to different people. Claims about usability methods are hard to prove using classical scientific techniques because of the difficulty in collecting statistically valid empirical evidence. Further obstacles come because HCI activities represent a paradigm of outside-in and holistic development of products. The parts facing the users are developed first, the system internals later. Organisations used to SE paradigm of inside-out and modularised (or reductionist) design have a natural tendency to resist this major paradigm shift. The social and cultural differences between the people with HCI and SE backgrounds do not help.

Gaps exist also in HCI and SE research. Social and cultural differences play a role there as well. Sutcliffe is sceptical about whether synthesis will ever emerge in SE and HCI research for social rather than intellectual reasons. Disciplines tend to have self-perpetuating mechanisms since they form communities that mutually support individual survival via peer review. Both HCI and SE have well developed areas of research, but many researchers in either field rarely reference the other field, and the research seems to be motivated by quite different ideas (Suitcliffe, 2005).

Things are looking up though. Jerome and Kazman analyzed several trends in SE and HCI research and felt that there has been a significant research interest to bridge the gap, particularly since year 2000 (Jerome, et al., 2005). In recent times, there have been a series of workshops and events organised to promote interaction between HCI and SE researchers the IFIP working group 2.7 / 13.4 on User Interface Engineering (2004), the Human-Centred Software Engineering conferences (2007, 2008 and 2010), and the Workshops on the Interplay between Usability Evaluation and Software Development (2004, 2008 and 2009). Some of the proposals of these events are reviewed in chapter 6. However, these events are still restricted to special interest groups and attract only a fraction of attendance compared to the mainstream conferences in either area.

5.2 Gaps in Industry Practices

Jerome and Kazman analyse the gaps between SE and HCI in practice (Jerome, et al., 2005). From a survey of 63 HCI practitioners and 33 software engineers, they found that the state of practice is not very encouraging. They report that there is substantial lack of mutual understanding among software engineers and HCI practitioners and the two disciplines hardly follow each other. They also do not collaborate much in projects. 68% software engineers report that they made key software design decisions that affect the user interface without consulting HCI practitioners. Even greater proportion of HCI practitioners (91%) believe that software engineers were making key design decisions without consulting any HCI practitioners. When collaboration does occur, it usually happens too late. Only 1 out of 21 software engineers and 2 out of 60 HCI practitioners reported that they collaborated during the specifications phase. All other responses were divided between coding, testing, or after release, all stages too late according to Jerome and Kazman. On occasions where HCI practitioners and software engineers did collaborate, the main impediment in implementing the HCI recommendations was the non-availability of time in the projects. Infrequent contact leads to misperceptions and miscommunications between groups. There appear to be important differences in how software engineers and HCI practitioners answered the question regarding who did the testing of the usability of software. Software engineers thought quality assurance or software engineers themselves conducted these tests, while HCI practitioners thought that they handled it.

To get a first-hand experience of the gap between HCI and SE, we reviewed real-life projects in the Indian IT industry. The questions we were trying to answer were:

- How and when do HCI design decisions happen in the SE process?
- What role do the HCI practitioners play? How do they influence the SE process?
- What are the common objects between HCI and SE process in use today?
- How are the concerns raised above about HCI and SE integration handled?

This was a qualitative survey exploring several aspects. The projects were selected as they became available. The author participated in some projects in the role of a designer or usability evaluator, which allowed him to look *"under the hood"*. In addition, HCI practitioners in the industry were requested to contribute additional case studies. The author discussed the projects with these participants and reviewed available project documentation.

Many participants and organisations were concerned about confidentiality and information security policies within their organization and their contractual agreements with the clients. In spite of these difficulties, eight case studies could be analyzed, and five of these could be analysed in detail. All projects involved a medium to large Indian IT company (the vendor) providing software development services to its international client (the customer). All vendors were very process-aware with CMM level 5 or ISO 9000 certifications. Seven out of eight projects used a variant of the waterfall model while one used agile methods. None used the RUP, though all study contributors were aware of RUP. In seven out of eight cases, a professional HCI person was involved, though the extent of involvement in each project varied. The following is a summary of the lessons learnt from these.

5.2.1 HCI Inputs Are Not Taken During Requirements Specifications

The main issue that came out clearly across all case studies was that HCI inputs are needed early in the process before requirements are finalised. Use cases in requirements documents routinely over-specified the HCI design, including details such as the sequence and the contents of dialog boxes in the application. This over-specification happened possibly because there was a physical and cultural distance between the developers and users, the development teams were less familiar with the context of users, and the requirements specifiers wanted to have a control on the user interface. In one project, no HCI practitioner was ever involved throughout the project. In this project, many more HCI decisions were specified in the use cases (down to the size, location, label, and colours of menu items).

HCI design process was not followed during requirements specification and HCI skills were not used in most cases. Several participants complained that customers do not state many usability requirements explicitly or in responses to questionnaires. It was well known to the HCI practitioners among the participants that such requirements could be captured by ethnographic user studies though no such studies were carried out in most projects. Inappropriate HCI design decisions specified in use cases seemed to have a ripple effect in many areas of user experience. HCI practitioners remarked that once something is written up in a requirements document it gets *"set in stone"* and subsequent changes, however desirable, become uphill. The only way solve this problem seems to be to involve the HCI practitioners early in the project, certainly before freezing UI design, and possibly before freezing on requirements. Early investment in HCI design makes business sense for users, customers and the vendors. However, this happened in only 1 of the 8 cases.

One related business problem seemed to be particularly difficult to resolve to several participants. Requirements were typically specified before a project contract was drawn up between the vendor and the customer. Requirements were the basis on which project budgets and timelines are worked out. Rarely were these services paid for, and there was a push on the part of the vendors to go through this phase as soon as possible so that a concrete project materialised with minimum up-front investment. This has implications not only on the HCI decisions made during this phase, but also to the other commitments.

However, this does not seem to be an insurmountable problem and some vendors have managed to modify their business process to resolve this issue. One case study demonstrated a positive impact of early involvement of HCI practitioners. The project involved migration of a task management application for use by 22,000 field engineers of a telecom company from a laptop to a mobile phone. The goal was to make it more convenient for the field engineers to manage tasks and reduce their dependence on laptops while on the move. A multi-disciplinary team comprising of software engineers, business analysts and HCI practitioners set out to identify a business case for the project. The team created a proof of concept and justified the business case that went into development. The HCI practitioners were involved in analysing the current application, defining scope, and creating a proof of concept of the new application. Creation of an early prototype helped in selling the business case to the stakeholders and getting a buy-in from users. The prototype also influenced critical software architectural decisions where access technologies played an important role.

5.2.2 *Porting Projects Get Minimal HCI Inputs*

Every software project represents an opportunity to improve the user experience. Conversely, every project also represents a risk of degrading the user experience. This applies even to porting and migration projects (which were a majority in the cases studied). Less importance was given to requirements gathering in general and usability requirements in particular in such projects. It was assumed that most requirements were well-understood and had to be *"copied over"* from earlier version. However, such ports often involve a change of delivery platform, a change of context, or a change of users. Either of these can have a big impact on HCI design and the corresponding requirements.

In one of the cases, the quality of user experience deteriorated significantly. This project was to port a product from a legacy platform to a web-based application. The HCI team studied screenshots of the existing application and reproduced them as closely as possible in a web browser. They could not meet real users or observe them using the application. A particularly important HCI issue was missed out completely. The application was a frequent use product and the users often used keyboard shortcuts. This problem was discovered very late, after the client representative had signed off and the first version of the product was delivered to the users. The task completion times in the new software went up substantially, and the users rejected the product altogether.

5.2.3 *Client Representatives Take Design Decisions*

The above case demonstrates another issue, viz. the client representative. In the industry, a client representative routinely drives many HCI design and usability considerations. Such a person may have never been a user himself or may have moved out of that role a long time ago. His / her sign-off may not imply that the product is usable. This can be revealed only by usability evaluations with real users. In the above case, users in the client organization rejected the product even though the client representative had given his acceptance. The vendor had to go in for a major re-write to introduce the keyboard shortcuts. The project was delayed substantially.

5.2.4 HCI Skills Do Not Have Process Support

This point argued in the literature reviewed above was reinforced. Merely having skilled HCI practitioners on the team did not resolve all HCI issues automatically. They also needed process support to do their work. While most projects studied had some involvement of HCI practitioners, they still ended with unresolved usability issues that they knew could be solved. A multi-disciplinary team needs to work together. The team needs to be armed with appropriate user inputs and needs a common set of work-products and a common process to approach the product development holistically and add value. Role of each discipline needs to be mutually understood and respected, first within the team and then across the organizations.

5.2.5 Too Little and Too Late is Not Good Enough

In some projects, HCI practitioners were pulled in towards the end when too many obvious usability problems surfaced (as seen by the client contact person). One HCI practitioner described these as *"last straw"* projects, while another described his job as akin to *"running an ambulance service"*. In these situations, HCI practitioners worked under severe constraints. Typically, the project would have already spent its allotted budget and time and the HCI practitioners needed to hit the road running. They had no time to understand the scope of the project and no budget to do usability activities they wanted to do. Even if some HCI activities were done, most of the recommendations they could come up to improve the UI seemed too impractical to implement in the given situation. Few cosmetic changes would be made, mainly to satisfy the client representative, and the project would be *"pushed through"*.

5.3 Summary

In this chapter, we reviewed gaps between HCI and SE discussed in literature and as we observed from industrial case studies. The case studies confirmed the following areas of concern:

- There exist major gaps between HCI and SE practices. The practitioners of the two disciplines do not collaborate enough.

- HCI design does happen during requirements specification, particularly as a part of use case definition. Rarely is it done systematically or even consciously. People with HCI skills are not involved in a majority of cases.
- Customers or users are not designers. Customers and users do not state several usability related requirements explicitly, even in response to questionnaires. Therefore, questionnaires and surveys are not enough. Many user problems, needs, habits, and constraints can be captured only by direct observation.
- Usability issues do arise late in software development. Some of these issues are *"deeper than skin"* and affect the project scope. Often, such issues are left unresolved. Where they are resolved, this leads to late iterations and rework, shipping an unusable product or an outright rejection from users.
- The client representative is often called upon to make design decisions. In many occasions, clients are unaware of the value professional designers can add through an HCI design process, though ease of use and user friendliness is often an explicitly stated goal.

In the next chapter, we review current approaches to integrating HCI with SE.

6 Integrating Human Computer Interaction Activities with SE Processes

The first question to consider perhaps is why one should integrate HCI in SE and why not the other way around.

One reason perhaps is that non-integration of HCI with SE is a problem that affects HCI community more than the SE community. As Seffah et al. note, HCI is by no means considered a central topic in SE (Seffah, et al., 2005 pp. 4,8), but SE is certainly a necessity for HCI. A product cannot be built using HCI efforts only. SE efforts are essential part of building the design that the HCI effort creates. There are many examples where software engineers work on projects without ever involving an HCI practitioner. However, the converse is not true. HCI practitioners always work on projects, which (if not abandoned) will eventually be implemented by software engineers. Usually, it is the HCI practitioner who is invited to join a SE project (often too late).

Another reason could be cultural. In a survey that included people from both professions, Jerome and Kazman report that several HCI practitioners claimed that they collaborated with software engineers frequently, but software engineers believed that they had little or no contact with HCI practitioners (Jerome, et al., 2005). Their interpretation is that software engineers are reluctant to adopt HCI processes, while HCI people try to fit in.

Another, perhaps more reasonable, interpretation could be that there are lot more software engineers, and very few HCI practitioners. There were 878,000 software professionals employed in the Indian industry in the year 2006 (NASSCOM, 2006). Our guesstimate of the number of HCI practitioners in India at that time is 2,000 to 3,000 and no more than 5,000 to 8,000 at the time of writing this report. It is perhaps a good guess that HCI practitioners in India are outnumbered 1:300. Nielsen surveyed 31 development projects that had usability engineering activities to find how much usability effort was required in projects (Nielsen, 1993 pp. 6-8). Of the total project effort in person-years, median-sized projects reported using only 6.5% effort for usability. In an ideal situation, HCI practitioners asked for only 10% effort on an average and 21% in the top quartile. The ideal desired usability effort was independent of the project size (*r*=0.12), i.e. smaller projects required relatively more usability efforts, while larger projects did not require lot more efforts. Over the years since Nielsen's survey, project sizes have reduced and importance of usability may have increased. On the other hand, there has also been an increase in experience within the HCI community. In an ideal case, perhaps it is appropriate to expect 10% of the overall effort to be associated with usability and HCI related activities. Even in this case, HCI practitioners are outnumbered by a ratio 1:10.

It seems better if HCI activities are integrated in existing SE processes rather than the other way around. It certainly seems to be a good strategy for organisations, where HCI practices are not yet well established, but SE process models are. Göransson et al. express in exasperation, *"if you can't beat them, join them"* (Göransson, et al., 2003).

Literature on integration of HCI with SE can be classified as process approaches and non-process approaches. The non-process-based approaches include work in the area of modifying software architecture patterns to make it more responsive to usability concerns, extending SE artefacts to include usability, creating other boundary objects or techniques between the two disciplines, identifying patterns of integrating HCI activities with SE processes, and activity mapping. The process-based approaches are proposals that aim at integrating HCI and SE processes. These include new process model proposals, and proposals to integrate HCI activities into existing process models such as the waterfall, agile, and RUP.

6.1 Non-Process-Based Approaches

6.1.1 Improving Software Architecture

Commonly, a usability problem is discovered late in the process and it requires a modification that reaches too far into the architecture of the system to allow viable and timely changes. Building on their strong background in software architecture, researchers at the Carnegie Melon University propose an approach to make software architecture more flexible to tackle late arising usability issues. John et al. called these as *"deeper than skin"* architecturally sensitive usability issues and suggest methods to consider these during the earliest architectural design (John, et al., 2004). An example of a usability problem that is discovered late could be a command that takes too long to complete. A possible solution would be to give the option to the user to cancel such a command and return to the earlier state. However, this change trickles into many parts of the software architecture and is difficult to contain. John et al. identify 27 such architecturally sensitive usability scenarios. They provide ways to reason about the forces acting on the architectural design in these scenarios and a checklist of important software responsibilities and possible architecture patterns to satisfy these scenarios.

Adams et al. extended this work by defining a relatively smaller number of software tactics and linking them to accrued usability benefits (Adams, et al., 2005). An example of a tactic is *"separate data from the view of that data"*. In the scenario where the software should support international use, using this tactic could potentially result in several usability benefits, including accelerating error-free performance, reducing impact of slips, supporting problem solving, facilitating learning, preventing and accommodating mistakes and increasing confidence and comfort of users. Adams et al. identified a matrix of 9 benefits and 13 tactics, each cell of which lists one or more of 26 architecturally sensitive usability scenarios.

An alternative approach is presented by Juristo et al. (Juristo, et al., 2003). They take high-level usability attributes (e.g. learnability, satisfaction) and break them down into usability properties (e.g. guidance, feedback, user control). They link each usability property to one or more usability patterns (e.g. wizard, undo, alert, progress indication) and then define software architecture patterns to realise the usability patterns. This allows the software architect to select architecture patterns based on expressed high-level usability requirements. It is not clear at this time though, if it is feasible to integrate each of these patterns into all software architectures imaginable.

Though interesting and important, there are several limitations to these approaches. Firstly, the architectural patterns, benefits, tactics, and scenarios identified so far, though numerous, are far from exhaustive. An exhaustive list may not be even possible.

Secondly, each combination of a benefit - tactic - scenario needs to be rigorously proved or exceptions identified. Thirdly, this approach already adds to the already numerous things that a software architect needs to consider. These approaches make the software architecture flexible so that potential usability problem could be solved, but may not help design a usable product. The utility of these approaches is analogous to the utility of a good dictionary for writing a novel. Indispensible, as a reference, but it does not inspire the plot. Further, these approaches assume that the usability issues have been identified and the software architect will actually consider them, which may not happen unless the process actually demands it.

6.1.2 Extending SE Artefacts

As the Unified Modelling Language (UML) became a standardised collection of SE artefacts, some approaches to bridge the SE-HCI gaps focussed on extending SE work artefacts to accommodate HCI inputs.

Constantine and Lockwood propose essential use case modelling as a technique of reducing the number of built-in assumptions in use cases about the user interface that has yet to be designed (Constantine, et al., 1999). An essential use case is a simplified, abstract, generalised, technology-free, and implementation-independent description of one task or interaction that embodies the purpose underlying that interaction.

Nunes and Cunha propose the Whitewater Interactive System Development with Object Models (WISDOM) as a lightweight SE methodology that uses UML to support HCI techniques (Nunes, et al., 2000). They propose extensions to UML to incorporate descriptions of users, user behaviours, user intentions in performing supported tasks and a specification of the abstract and concrete user interface.

Da Silva et al. propose UMLi, an extension to UML that tries to overcome the limitations of UML in visualising widget containment (da Silva, et al., 2000). The structural elements are modelled in UMLi in a user interface diagram, a specialised version of class diagrams of UML, that allows specification of UI elements such as top level windows, inner windows, input widgets, display widgets, buttons etc. at an abstract level. The behaviour elements are modelled in extended versions of activity diagrams of UML.

Expressing HCI deliverables in UML or similar output could ensure easier flow of artefacts between HCI and SE activities and could potentially help integrate HCI with SE. Unfortunately, so far none of these proposals has become popular within the HCI or SE communities. One reason could be that merely integrating deliverables may not be enough. Perhaps it may be important to integrate the activities that generate these deliverables. Once that happens, perhaps these approaches are worth revisiting.

6.1.3 Boundary Objects and Other Communication Techniques

Some researchers look for identifying and creating shared objects (sometimes called boundary objects) or communication techniques between HCI and SE practitioners.

Sutcliffe feels that since models and scenarios are already used in some form by both SE and HCI disciplines, understanding these may help integrate SE and HCI within software development (Suitcliffe, 2005). He argues that reusable knowledge in the form of formal, generic models, claims, and design rationale should become part of the skill set for all system designers. Scenarios are used for creative exploration by both disciplines, so they too can become the common ground, though they need to be further integrated into development processes throughout the lifecycle.

Carter et al. suggest that for usability engineering to succeed, in addition to resulting in a usable system for end users, it must also be usable by the people who are involved in development (Carter, et al., 2005). They propose an approach to integrating documentation used to develop the resulting system. They build on Putting Usability First (PUF) methodology and transformed usability engineering requirements into software engineering specification by capturing user requirements and other contextual information into UML. They also identify high-level relationships between PUF and UML components. Users correspond to actors, content to class attributes, tools to class operations, tasks to essential use cases and scenarios to use case instances. It is hoped that expressing the PUF methodology in UML form can ensure the application is developed in a context rich information environment that minimises the occurrence of usability problems.

Kujala suggests a method to bridge the gap between informal user need descriptions and formal user requirements (Kujala, 2005). Kujala's solution was an intermediate work product: a user needs table that summarises task sequences in one column and related problems and possibilities in the other. This captures the user needs found during user interviews and helps transform them into use case descriptions that can be readily used as a part of the requirements specification by software engineers.

Pyla et al. identify communication gaps between practitioners on both sides. They proposed a constraint-based framework called the Ripple Framework that identifies connections and dependencies within and between HCI and SE lifecycles and provides a framework to represent artefacts generated at each stage (Pyla, et al., 2005). Their approach is to maintain HCI and SE separate but coordinated processes. It incorporates techniques to accommodate communication about design changes and sends possibly cascading messages to inform people on both sides.

Jokela presents a workshop technique to help multi-disciplinary teams arrive at quantitatively measurable and prioritised usability requirements (Jokela, 2009). In addition to generating requirements for the specific project, Jokela found that the workshops educates developers about usability and motivates the team towards user centred design.

Lee et al. propose a methodology of integrating scenario based design method with extreme programming, called XP+SBD (Lee, et al., 2007). They propose an incrementally expanded central design record that keeps track of design goals, scenarios, claims map, and user stories in XP, facilitates communication and coordination, gives the developers an overview, and yet does not limit or delay incremental software delivery. The same record can be used to track and manage usability evaluations by the usability group.

6.1.4 Patterns of Integration

Battle says that HCI practitioners lack a clear direction to integrate HCI with SE and hence need to negotiate their roles and scope of their involvement in software development on a case-by-case basis (Battle, 2005). Synthesised from experience of several practitioners, she describes four process patterns to integrate HCI in SE*: "A-foot-*

in-the-door for internal usability practitioners", *"a-foot-in-the-door for external usability practitioners"*, *"UCD focus on early definition and design"*, and *"UCD in every phase"*. These patterns capture experience of introducing HCI activities in SE in early, middle, and late parts of software development. As expected, both the foot-in-the-door patterns are opportunistic, but in different ways. Internal usability groups typically make few early interventions and work around the internal processes. External consultants are involved in late interventions, quick fixes, and long-term visions for the product. In both cases, the organisation typically moves to the pattern of UCD focus on early definition and design, where extensive intervention in the early phases and some interventions in the middle phases happen. The most mature organisations use the UCD in every phase with substantial intervention in each phase.

6.1.5 Activity Mapping

Given that it is not clear how to integrate HCI activities in SE, Ferre presents a way to integrate 7 HCI activities and 36 related usability techniques with activities in software development process (Ferre, 2003). He argues that one of the reasons why usability techniques are not regularly used in software development was the lack of integration with SE concepts, terminology, and process. Though Ferre does not talk about specific SE process models, he used the software engineering body of knowledge (SWEBOK) (IEEE Software Engineering Coordinating Committee, 2001) as a basis for software development activities. Ferre maps HCI activities with corresponding software development activities to integrate usability with SE (Table 6.1). His goal is *"to offer the average software developers a way to integrate usability activities and techniques into their existing software development process"*.

In summary, in our proposal in section 7.1, we build on Ferre's work of integrating HCI activities with SE activities. While the intention of mapping HCI activities on SE activities is excellent, and we agree with many of the mappings, we do not agree with one particular mapping: the HCI activity of interaction design is mapped to the design activity of software engineering (the row highlighted in Table 6.1). While the word 'design' is common, the two activities are not similar and certainly cannot be done at the same time.

Table 6.1: Ferre's mapping between software development activities and HCI activities.

Software development activities	**HCI activities**
Requirements elicitation	User analysis
Requirements analysis (develop product concept, problem understanding, modelling for specification for context of use)	User analysis Task analysis Develop product concept Prototyping
Requirements specification	Usability specifications
Requirements validation	Usability evaluation (walkthroughs)
Design	Interaction design
Evaluation	Usability evaluation (usability tests and follow up studies)

As per SWEBOK software design is *"the software engineering life cycle activity in which software requirements are analyzed in order to produce a description of the software's internal structure that will serve as the basis for its construction"* (IEEE Software Engineering Coordinating Committee, 2001). Clearly, this activity needs to be done after the requirements have been specified and validated, irrespective of how iterative the software engineering process model might be.

As defined by (IXDA, 2009), interaction design attempts to define the *"structure and behaviours"* of interactive products and services and user interactions with those products and services. As defined by (Garrett, 2003 p. 32), interaction design is the activity in which we define *"how the system behaves in response to the user"*. Either way, interaction design affects the scope and the requirements of the product. It is too late to consider interaction design in the design phase of software development. Much of interaction design and at least some user interface design need to happen before prototyping and usability evaluation activities, which are (rightly) mapped by Ferre to requirements analysis and validation activities of software development. This problem would become obvious if pre-requisites and deliverables of each activity were considered. In an update on this work (Ferre, et al., 2005) this problem seems to continue.

6.2 Process-Based Approaches

6.2.1 New Process Models

Constantine and Lockwood's usage centred design process (as opposed to user-centred design processes) is perhaps the best-know example of a HCI process model that is easy to integrate with SE (Constantine, et al., 1999). Key elements of usage centred design framework include a model-driven design process, iterative improvement, and metrics of quality.

Models in usage centred design are represented in a particularly abstracted form called essential modelling, the technique of abstracting in a technology-free, idealised and abstract description. Role model captures the relationships between users and their roles as they exist in the real world (usually captured in thing-doer or adjective-thing-doer forms e.g. leave-approver or late-bus-commuter). Operational model describes the real world context in which the system will be used such as the physical work context, features, and limitations of the equipment and devices, the interaction profile (e.g. frequency of use), information profile and other operational risk factors. Task model depends on the role model and has a central role in usage centred design, captures the structure of the tasks users will need to accomplish with the new system and is represented in the form of essential use cases and a use case map. Content model is derived from the task model, is a high-fidelity abstract model that specifies the tools and materials to be supplied to the user interface, the interconnections between them and the navigation map (or the conceptual model). Implementation model captures the concrete elements of the interface such as visual elements, widgets, descriptions of their operation, menus, layouts, information visualisation etc.

Usage centred design defines a set of activities to be carried out, but gives a lot of flexibility to reorganise the activities in a manner that suits the project. In that sense, it is not a process, but rather a collection of activities. Many of the activities are directed towards creation of the models mentioned above. The process begins with collaborative requirements dialog, domain modelling, and task modelling. This is followed by content and implementation modelling, concentric construction, architectural iteration, and interspersed by usability modelling activities. Running in parallel to all these are activities related to operational modelling, help system documentation, standards and style definition and object structure design. All the activities are highly iterative.

Though Usage Centred Design is not a SE process model, authors claimed that its model-driven approach dovetails into most SE methodologies easily. For example, essential use cases can be converted into the concrete use cases of UML. System responsibilities in essential use cases can be converted to responsibility based design or CRC cards. The abstraction and brevity of essential use cases allows an agile start to early object modelling (Constantine, et al., 2003).

6.2.2 Integrating HCI in the Waterfall Model

As discussed in section 4.1, the waterfall model remains popular and particularly so in outsourced software development. However, only a few attempts seem to have been made to integrate HCI activities in the waterfall model.

Dix et al. look at the waterfall model of software engineering as a means of understanding the structure of the design process (Dix, et al., 1998 pp. 178-189, 205-220). They also describe several HCI activities in detail. However, they do not give specific suggestions on integrating these HCI activities in the waterfall model. Similarly, several authors refer to the activities derived from the waterfall model as a starting point while identifying approaches to integration of HCI with SE (Ferre, 2003), (John, et al., 2004), (Ferre, et al., 2005) and (Pyla, et al., 2005). However, they do not detail how the waterfall process might look like after the integration, and how deliverables might flow from one phase to the other.

Costabile makes perhaps the most direct reference to include usability in the waterfall model (Costabile, 2002). Figure 6.1 summarises the recommendations from Costabile. The single-outline boxes in the figure are the phases of the waterfall model assumed by her, while the double-outline boxes are additions of HCI activities she suggests. She augments requirements specification and architectural design phases with user and task analysis. She argues for explicit inclusion of user interface design, scenarios, prototyping, and testing during architecture design and detailed design. She suggests that the product should be evaluated for usability and iterated upon *"from each phase [to] any of the previous phases"*.

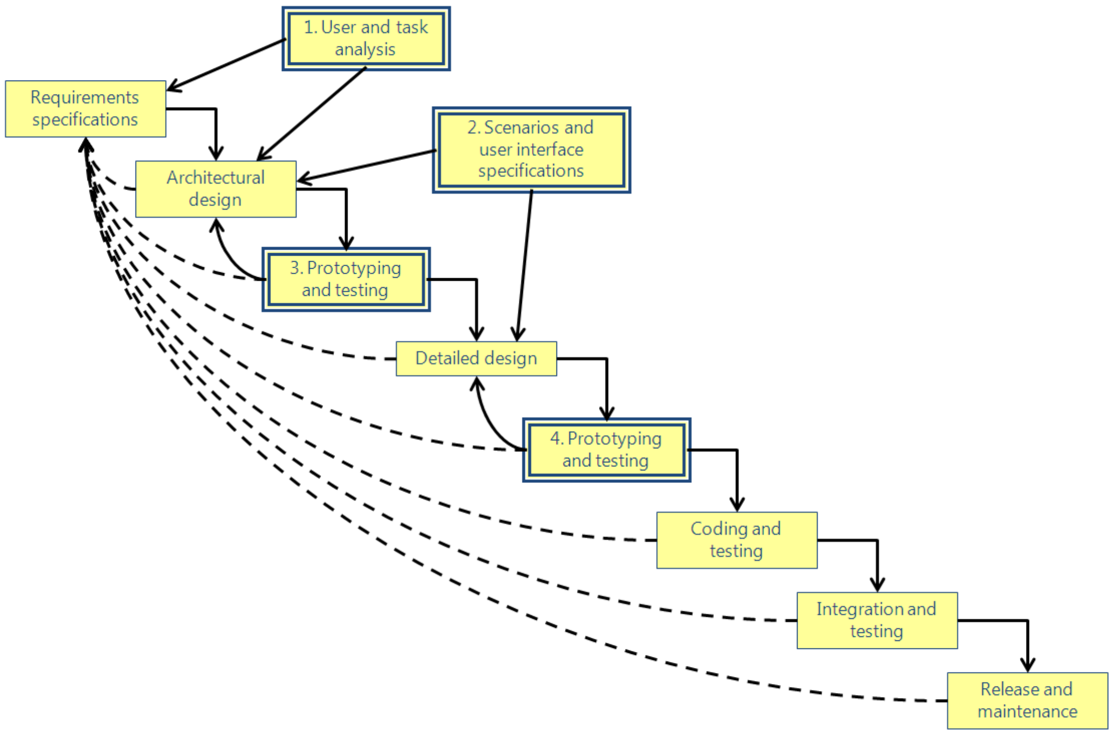

1. User and task analysis
Requirements specifications
2. Scenarios and user interface specifications
Architectural design
3. Prototyping and testing
Detailed design
4. Prototyping and testing
Coding and testing
Integration and testing
Release and maintenance

Figure 6.1: Revised waterfall cycle suggested by Costabile (2002).

In our proposal in section 7.2, we expand on Costabile's ideas. The most important similarity between our proposals is related to integrating user, task, and context analysis before requirements specification.

However, there are some crucial differences. Firstly, Costabile suggests that first-cut UI design and evaluation should happen in architectural design and detailed design phases. This is quite late. Use cases that are written in the requirements specification phase already specify too many HCI design decisions. This happens not only in real-life practice (as discussed in section 5.2.1, page 70), but is also formally suggested in a majority of SE literature (as discussed in section 4.2.2, page 44). Therefore, an iteration of HCI design and evaluation needs to happen before requirements are specified. Secondly, Costabile suggests, *"from each phase of the software life cycle it is possible to go back to any of the previous phases, in particular, go back and update the requirements specifications"*. These paths are marked in the dotted lines in Figure 6.1. While iterative approaches are often favoured in HCI literature, and some amount of iteration does occur in waterfall projects in practice, we posit that iteration in-the-large (such as going back all the way to requirements specification from testing) goes against the grain of the waterfall process model and is more likely to be discarded in real-life projects. Instead, in the next chapter we suggest smaller, tightly managed iterations throughout the process for the HCI activities.

6.2.3 *Integrating HCI in Agile Processes*

There is a lot of prior work in the area of integrating HCI and usability in the agile process models, which serves as the foundation for our proposal (discussed in section 7.3).

The usage centred design process (Constantine, et al., 1999) discussed above is claimed to integrate well with agile software development. Constantine draws many parallels between usage centred and agile methods (Constantine, 2002). Like agile, usage centred design is an iterative development process where roles and tasks are identified, prioritised, described, organised and refined. Like many agile methods, usage centred design uses index cards to model task cases to ensure that descriptions are brief and to the point. Also like in agile methods, usage centred design attempts to attack the high-priority tasks first. Further, Constantine insists, as in agile methods, one should not be dogmatic about the process and should do only things that actually help. However, there were some differences as well. The main difference was that usage centred design is not a software development process, but an *"adjunct to an effective agile process"* and the *"missing link back to the users"*. The other difference from most agile processes was that it relied heavily on model driven design. In spite of these differences, Constantine believed that usage centred design *"probably qualifies as an agile method even though it was not conceived as one"*.

Patton narrates an industrial case study of integrating usage centred design into XP (Patton, 2002). Patton reports that simple, informal techniques like preconception purge and using index cards to derive role models and task models and prioritising them worked very well. Mapping task cases to abstract prototypes made transition from user needs to a tangible design easy. On the other hand, people had some teething troubles with understanding user roles and daylong collaborations were tiring for participants.

However, the team managed to integrate many findings into their mainstream XP processes.

Beyer et al. propose a few steps to integrate their contextual design process to integrate them with agile processes (Beyer, et al., 2004). Before the iterations begin, set the project focus, do contextual inquiries, and build an affinity diagram to capture information about how tasks are done today. Explain findings and brainstorm design ideas with the whole team, build user stories and run a release planning process to organise stories into iterations. Do two quick iterations of design, prototyping, and testing user interfaces for the user stories for the first iteration and provide these user stories and user interfaces for implementation. During the first iteration, while the development team is developing the UI for the first iteration, design, and test the UI for the second iteration. Second iteration onwards, design the UI for the next iteration, but also test the working code from the last iteration (testing could be skipped for some iterations).

At an abstract level, Blomkvist attempts to integrate basic values and principles rather than specific processes (Blomkvist, 2005). He argues that integrating HCI activities in agile methods can *"act like cake frosting"* and might lead to a false belief that all usability problems have been managed. Blomkvist also argues that the other way round (integrating agile methods in HCI processes) has problems of its own. The resulting process may not become agile enough. He calls for a *"balanced"* approach and makes several suggestions so that agile values and practices deal with *"project management and organisational issues, as well as making coding more efficient"*, while user centred design focuses on *"design methods and user involvement"*.

After a review of three projects using agile methods and HCI activities, Chamberlain et al. listed principles about integrating user centred design into agile development. Basic needs of users must be discovered before any code is written, designers must *"feed"* users with prototypes and user feedback on a cycle that works for everyone, and designers, developers and the customer should be willing to work collaboratively (Chamberlain, et al., 2006). They conclude that all the three projects had some extent of design before coding. Functional specification in one case, significant user research and high level specification in another, and an entire sprint in the third case. Though formal evaluations were not done, they report that the project that did significant user research upfront, also reported best integration and fewest issues with collaboration.

In 2002, a famous dialog titled *"Extreme Programming vs. Interaction Design"* happened between Kent Beck (the inventor of Extreme Programming) and Allen Cooper (a well-known interaction designer and author of several books on the subject) that showed up gaps in the fundamentally different approaches and very little meeting ground between the two ends (Nelson, 2002). Since then, Cooper seems to have moved to appreciate agile methods much better. In 2008, Cooper gave a keynote talk at the Agile 2008 conference (Cooper, 2008) in which he proposed a method to integrate interaction design with agile methods. He divided software development into four steps: generating a big idea through insight and introspection, designing the form and behaviour of the product for its users, making the engineering decisions on how to build the technology, and then actually constructing it. Cooper suggests that while design and engineering need to be iterative and collaborative, the actual construction need not be. Cooper proposes one addition and one change to agile methods. The addition is that before design, engineering and construction starts, interaction designers needed to collect and analyse information about the users and stakeholders. The change is taking Fred Brooks' idea of planning to throw one away literally (Brooks, 1995) and splitting the software development in two parts. In the first part, design and engineering happen together as a single, collaborative, and iterative process. Next, the code could be entirely re-written in a linear construction activity.

Nielsen summed up three recommendations for integrating usability in agile development, which are quite similar to Beyer et al.'s: foundational user research before development starts, parallel track user experience work that stays one step ahead, and shorter usability activities (Nielsen, 2008).

Our proposal in section 7.3 for integrating HCI activities in agile process models agrees with several of these authors, particularly Beyer et al. and Nielsen.

6.2.4 Integrating HCI in Rational Unified Process

There have been three proposals to integrate HCI in RUP. Interestingly all three appeared in the year 2003.

Heumann suggested creating a UX model and storyboards within the discipline of the analysis and design disciplines of the RUP (Figure 4.2 on page 56) (Heumann, 2003). The storyboards and the UX model are based on the use cases, and comprise of details such as additional actor characteristics, usability guidance, usability requirements, and user experience elements such as screens and input forms.

Unfortunately, one problem that Heumann left unresolved is about the usability issues that have been captured in the use cases. Let us consider the example use case that Heumann himself uses. The use case is about how a buyer would make an incremental bid at an auction and four extensions to the use case: situations when the buyer places a bid when the auction was closed, when the buyer has pending payments, the entered bid was invalid, and if the buyer does not confirm the bid by accepting legal terms. Potential usability problems are discernable even in this brief summary, and at least three of the four extensions can be made unnecessary by suitable changes to the use case. *(The buyer should not be allowed to place a bid if the auction is closed or if the buyer is not eligible to bid for some reason. The system could allow the buyer to browse items, but should continuously inform him of the status of the auction and / or his ineligibility before he even places a bid and prevent him from placing a bid where he is ineligible. The system should not allow the buyer to place a wrong bid. The user interface could constraint the buyer from placing the wrong bid in some way.)* Instead of basing the storyboard on use cases, if the use cases are derived from storyboards after these have been evaluated for usability, the possibility that such usability problems creep in will be reduced.

The other problem is with the specification of the UX model. It comes across as too complex and non-intuitive a way to explore design and to enable divergent thinking. As discussed above, divergent thinking is a necessary first step before design. It might be possible to use this as a form of specification, but certainly not suitable for exploration, evaluation and modification, which is something a designer must necessarily do before specification.

Sousa et al. propose RUPi, an alternative process that integrates HCI concepts in RUP (Sousa, et al., 2003). They suggest changes to workers and artefacts in four workflows in Figure 4.2 (page 56): requirements, analysis and design, implementation and testing. Their suggestions are summarised in Table 6.2. While Sousa et al. add several important and relevant roles and artefacts to RUP, they do not suggest any changes to the dynamic dimension of RUP. Hence, it is not clear when these roles act and when the artefacts are produced and consumed within RUP. It is also not clear how the additions of workers and artefacts in RUPi relate with the different phases and milestones in the RUP.

Table 6.2: Additions to RUP of roles and artefacts by RUPi (Sousa, et al., 2003).

RUP Disciplines / Workflows	RUPi addition of Workers (Roles)	RUPi addition of Artefacts
Requirements	Ergonomist Human-factors expert	Scenario
	User-interface designer	User model Task model User-interface prototype
Analysis and design	User-interface designer	Refinement of task model Conceptual design of the UI Class model
Implementation	Implementer	Prototypes
Test	Test designer	Usability evaluation model

Göransson et al. propose a new discipline in the static dimension of the RUP called usability design, which will consist of five roles: usability designer, field study specialist, interaction designer, graphic designer, and usability evaluation specialist (Göransson, et al., 2003). They also fear that while they see benefits of these specialised roles, information could be lost when one role hands over information to the other role (e.g. when field study specialist hands over results to the interaction designer). Therefore, they suggest that the role of usability designer cover for other roles as well. Figure 6.2 demonstrates how the discipline of usability design integrates in the four phases and with respect to other disciplines.

Göransson et al. also suggest a workflow for this discipline. At the start of the project, a usability design plan is created and refined, during inception and early elaboration, user studies are done, and competitive products are analysed. As a next step, conceptual design, interaction design, detailed design, and user assistance materials are developed and iteratively evaluated for usability. A parallel activity monitors usability work on an ongoing basis.

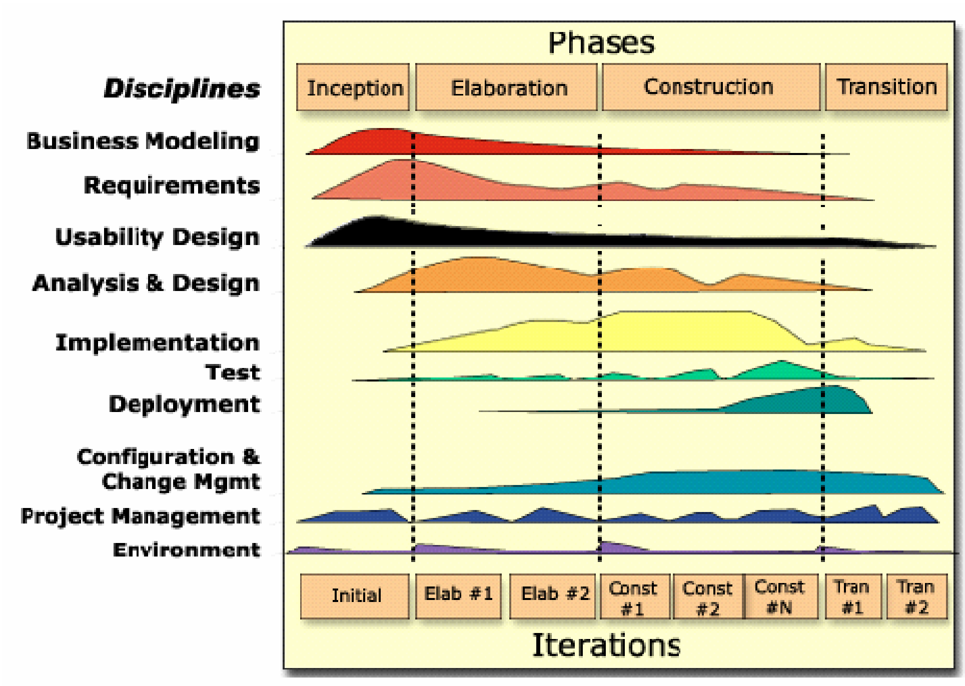


Figure 6.2: Integration of Usability Design discipline with RUP (Göransson, et al., 2003).

Our proposal in section 7.4 for integrating HCI activities with RUP is closest to the one by Göransson et al. While we agree with the main thrust of their proposal and most of the activities in their workflow, we differ over the choice of the disciplines and the details of its integration with RUP. The disciplines need to reflect more closely the professions in the field of HCI. The integration could reflect the iterative nature of RUP some more and synchronize better with its milestones, as we propose in section 7.4 below. In particular, we propose that more interaction design and evaluation should happen during the inception phase to mitigate the *"go-no go"* risks and the project achieves the Lifecycle Objective milestone at the end of that phase.

6.3 Summary

In this chapter, we reviewed current literature on integrating HCI with SE. Some authors take non-process approaches such as modifying software architecture, creating boundary objects, identifying patterns of integration etc. Other authors look at ways of integrating HCI activities with SE process models. Many of the proposals for integrating

HCI in SE process models are based on extensive experience of the respective authors. However, there is no hard evidence or a systematic study to show that integration of HCI activities in SE processes in the manner proposed leads to more usable products.

In the next chapter, we summarise the principles that could be used to integrate the HCI activities in SE process models and present our proposals for integrating HCI with waterfall, agile, and RUP process models.

7 Integrating Human-Computer Interaction Activities with Software Engineering

7.1 Approach to Integration

What we suggest is not a single, independent, prescriptive process model, but an approach for integrating HCI activities into popular SE process models. The approach recognizes the need for a variety in process models to satisfy the needs of a variety of products, project contexts, and team abilities.

Our approach is characterised by two important guiding principles:

- A truly integrated process should integrate and optimise HCI **activities** and HCI **deliverables** in the SE process. It should make HCI activities an explicit part of the SE processes, so that HCI design happens purposefully, and not by default.
- Integration should not disrupt **the core values of the SE process model** (for example, the *waterfall-ness* of waterfall model, the *agility* of agile models, and the *risk-driven approach, phases, and the milestones* of RUP). It follows that to integrate well HCI deliverables must be available before related decisions are required by that SE process model.

Before we list other principles, we elaborate these principles further.

In Table 3.1 (page 27), we identified HCI activities, methods, disciplines, and deliverables that should be integrated in a SE process model. In Table 7.1, we re-list the activities and deliverables and compare them against the typical SE activities and deliverables of the waterfall model, and identify the dependencies between the activities and the deliverables of HCI and SE. Literature reviewed in section 4.2 and experience from the case studies discussed in section 5.2 suggest that there is a need to specify some HCI design decisions in use cases as a part of the requirements specification activity. Therefore, high-level scenarios and a refined product definition should be available before the use cases and wireframes are determined. Similarly, the refined and detailed UI prototypes should be available before the detailed software design can be done. Thirdly, the minor tweaks and changes to the UI should be provided as required during the construction phase. These dependencies have been marked by arrows in the table. Other dependencies (such as functionality, strategy, and business model) have been omitted here for simplification, but these would also be relevant.

Table 7.1: The dependencies of the HCI activities and deliverables on the SE activities and deliverables in the waterfall model.

HCI Activities	**HCI Deliverables**
User studies, user modelling, market analysis	Analysis User models User needs, problems, goals and constraints Opportunities for design interventions Product goals
Ideation	Design ideas
Product definition	High-level scenarios Low fidelity prototypes, wireframes Business model Strategy, scope and structure of Garrett's model
Formative usability evaluation 1 and refinement	Refined and approved product definition and product goals Technology feasibility approval Business feasibility approval
Design detailing	Medium to high fidelity UI prototypes Structure, skeleton and surface of Garrett's model
Formative usability evaluation 2 and refinement	Usability problems Metrics Refined, detailed UI prototypes UI specification
Development support	Minor tweaks
Summative usability evaluation 3	Usability approval Metrics

SE Activities	**SE Deliverables**
Communication Project initiation Requirements gathering	Statement of need and feasibility, A bounded statement of scope of work Functional and non-functional requirements Quality assurance scenarios Detailed use cases Low fidelity prototypes
Planning Estimating Scheduling Tracking	Plans
Modelling Analysis Design	Architecture design Detailed design
Construction Code Test	Releases Bugs Bug fixes
Deployment Delivery Support Feedback	

We identify these additional guiding principles in our approach for integration:

- The process should recognise and support the involvement of **multi-disciplinary** teams. It should recognize that merely representing all professions in the team is not enough. The process should be a script that tells everyone in the team what to do when.
- The process should encourage **divergence and transformation** of the problem space before **converging** to a solution. The process should support the team to consider many alternatives for the HCI design before making decisions.
- The process should support **iterations**. If the SE process model does not allow for iterations, these iterations should be contained within the HCI activities. The purpose of the iterations is not to incrementally grow the product, but to solve the usability problems found in the early ideas and to explore alternative solutions. To keep the early investments low and the iterations short, the early prototypes could be rough or partial, and the early evaluations could be quick. Nevertheless, the iterative nature of the HCI design process should not be sacrificed.
- The professionals in the team should make **considered, holistic design recommendations** and these should not be merely arrived at by asking the business stakeholders. The team should engage the stakeholders in a language that they understand. The informed consent of the stakeholders to the design should not be a reason for the designers to abdicate their responsibility.
- In addition to managing change, the process should help **contain change** by pro-actively anticipating the reasons of change and then accounting for them through creative design solutions. Particularly, it should anticipate and contain intra-lifecycle change (that arises due to insufficient understanding of the context). By pro-active design, the process may help the team anticipate the extra-lifecycle change (that may arise due to the change in the external environment), and possibly even induce the appropriate, desired change.

In the following sections, we present our proposals for integrating HCI activities in the process models of waterfall, agile, and RUP using these principles.

7.2 Extended Waterfall Process Model

Using the description by Pressman (Pressman, 2005) as the baseline specification of the waterfall process model, here we propose an extended waterfall model. Figure 7.1 summarises our proposal. The changes are highlighted in red. Below, we describe the activities, methods, deliverables, and roles that need to be added to each phase of the waterfall model. We describe these in terms of the five phases used by Pressman: communication, planning, modelling, construction, and deployment.

In chapter 12, we empirically evaluate the effect of integrating each HCI activity with the waterfall process model (as described in this section) on usability goals achievement by analysing data from 50 projects from the industry that used the waterfall process.

7.2.1 *Communication Phase*

Pressman suggests that the communication phase comprises two activities: project initiation and requirements gathering (Pressman, 2005 p. 79). Requirements engineering includes activities of inception, elicitation, elaboration, negotiation, specification, validation, and management (Pressman, 2005 pp. 175-204).

Among these, the activities of requirements elaboration, specification, validation, and management are about converting informal descriptions into formal technical models of software features, functions, constraints, and quality requirements and managing them through the development process. These are mostly internal to software engineering and we propose no changes to them (though we merge them under a common title "requirements specification"). On the other hand, the activities of requirements elicitation and requirements negotiation entail interfacing with the customers and users, and this is where we suggest most changes.

Activities, deliverables, and roles in the modified communication phase of the waterfall model are detailed below. For each activity, we provide a reference to the HCI activities activity in Table 3.1 on page 27. We also link each new activity to its closest match in (Pressman, 2005) and point out the differences.

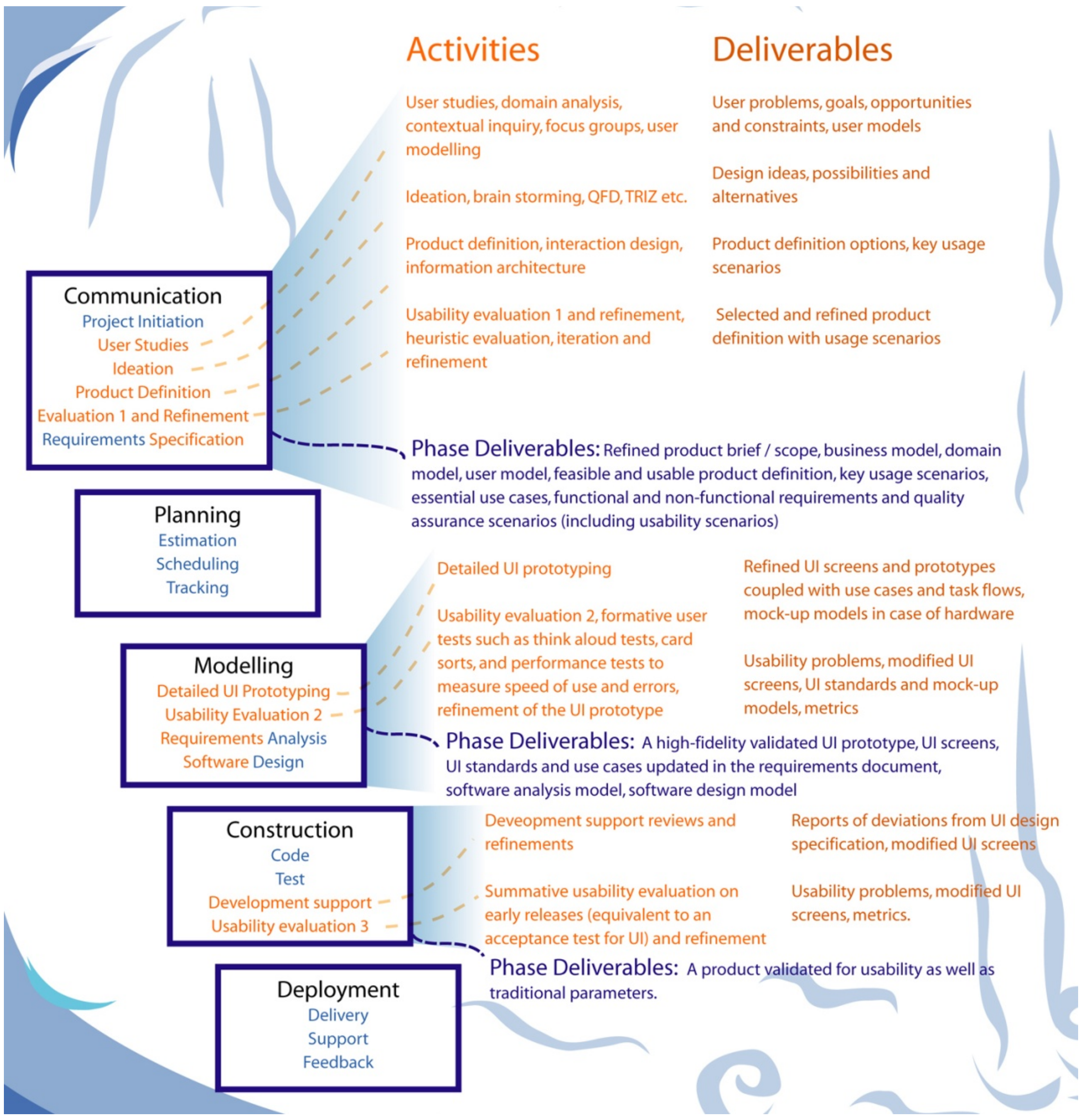

Activities
Deliverables
User studies, domain analysis, contextual inquiry, focus groups, user modelling
User problems, goals, opportunities and constraints, user models
Ideation, brain storming, QFD, TRIZ etc.
Design ideas, possibilities and alternatives
Product definition, interaction design, information architecture
Product definition options, key usage scenarios
Usability evaluation 1 and refinement, heuristic evaluation, iteration and refinement
Selected and refined product definition with usage scenarios
Communication
Project Initiation
User Studies
Ideation
Product Definition
Evaluation 1 and Refinement
Requirements Specification
Phase Deliverables: Refined product brief / scope, business model, domain model, user model, feasible and usable product definition, key usage scenarios, essential use cases, functional and non-functional requirements and quality assurance scenarios (including usability scenarios)
Planning
Estimation
Scheduling
Tracking
Detailed UI prototyping
Refined UI screens and prototypes coupled with use cases and task flows, mock-up models in case of hardware
Usability evaluation 2, formative user tests such as think aloud tests, card sorts, and performance tests to measure speed of use and errors, refinement of the UI prototype
Usability problems, modified UI screens, UI standards and mock-up models, metrics
Modelling
Detailed UI Prototyping
Usability Evaluation 2
Requirements Analysis
Software Design
Phase Deliverables: A high-fidelity validated UI prototype, UI screens, UI standards and use cases updated in the requirements document, software analysis model, software design model
Construction
Code
Test
Development support
Usability evaluation 3
Deveopment support reviews and refinements
Reports of deviations from UI design specification, modified UI screens
Summative usability evaluation on early releases (equivalent to an acceptance test for UI) and refinement
Usability problems, modified UI screens, metrics.
Phase Deliverables: A product validated for usability as well as traditional parameters.
Deployment
Delivery
Support
Feedback

Figure 7.1: The proposed changes to the waterfall model.

Project initiation: Before the project starts formally, the product manager or another business stakeholder requests that a new product to be built or an existing product be enhanced in some way. Relevant stakeholders are identified and interviewed, technology is reviewed, and a broad direction for the product is identified. The activity is quite informal, multi-disciplinary and at times multi-organisational. The activity involves business stakeholders, software engineers, users, and possibly HCI practitioners. The outcomes of this activity are an initial project brief, business goals, technology capabilities, and constraints. This activity is not new, and is quite similar to the activity described by Pressman as *"Initiating the Requirements Engineering Phase"* (pp 181 -184).

User studies: This corresponds to the HCI activity 1 in Table 3.1 (page 27). It involves methods such as user observation, contextual inquiry, focus groups, stakeholder interviews etc. followed by user needs analysis using techniques such as task analysis, affinity, and work modelling or similar. If the project happens to be a redesign project or a porting project, it may be a good idea to do a usability evaluation of the current version of the product.

The deliverables of these activities are user problems, goals, opportunities, and constraints. The findings could be documented with the help of user modelling techniques such as personas and scenarios, but a range of other user modelling techniques could also be used.

If a professional ethnographer is a part of the team, this is where he / she would contribute most, but an interaction designer or an information architect could also be involved. The product manager may not be present in all interviews, but may participate in the analysis, or stay informed about the findings.

We propose user studies as an activity partly in addition to, and partly in lieu of the traditional stakeholder interviews, business modelling and domain modelling activities suggested by Pressman as part of *"Requirements Elicitation"* (pp 184-191). The importance of user studies, requirements elicitation, and other activities would vary with the nature of the product. In business critical, life critical, and *"thick domain"* applications such as banking and financial services, call centre applications, office and factory automation, and healthcare products, user studies and business and domain modelling will play an important role. On the other hand, in consumer oriented products including social networking sites, consumer devices such as cell phones, cameras and washing machines, or entertainment products such as games and gaming consoles, user studies and competitive market analysis would be important. For information / content oriented products such as news sites or e-learning systems, content analysis and learner needs analysis would matter.

Ideation: This corresponds to HCI activity 2 in Table 3.1 (page 27). This activity is characterised by two classic designer attributes: problem solving and divergent thinking. It is best done by a multi-disciplinary team representing business, technology, and design. By a wide range of formal or informal creativity methods, the team generates several design ideas to cover a wide range of possibilities and alternatives. Some of these ideas may be wild and not feasible, while others might be very practical. At this early stage, all ideas are captured but none is evaluated and no decisions are made. Though ideation is listed as a separate activity, it may often happen during user studies and business analysis. As each user interview is analysed, as more problems are found, as more market information is identified, more ideas are generated. By pulling this out as an explicit activity, we are only trying to emphasise its importance.

Though ideation is a fundamental part of any new creation, and many software engineers would readily admit to its importance, Pressman does not suggest any specific activity related to ideation in the waterfall model. In that sense, it is a new activity.

Product definition: This corresponds to HCI activity 3 in Table 3.1 (page 27). Like ideation, problem solving is still a goal of this activity, though the focus now shifts to transformation from divergence. This activity involves creation of the high-level design that defines the holistic impact of the new product on the users' life. As Jones suggested, this would be the most creative and fun stage of the design process, but also the point where big blunders could be made (Jones, 1970 p. 66). This is the point where judgements of both values and technicalities need to be combined into high-level decisions about the solution. Unlike ideation, this is not a team activity. Rather, as Brooks suggested, it should be an outcome of one or at the most two minds acting *uno animo* (Brooks, 1995 p. 35).

The spirit of this activity roughly corresponds to what Pressman calls *"Collaborative Requirements Gathering"* within elicitation (pp 185-188), however, the team composition and expected outcomes are different. The problems are not *identified* during a meeting, as Pressman suggests. This was done during the user studies activity. Here, the focus is on decision-making. Several possibilities of high-level responses are considered and then the stakeholders narrow down on one approach of the new product (or a few alternatives).

The typical outcomes of this activity include decisions related to interaction design and information architecture, which describe how the product would fit in the users' lives. These decisions are documented in the form of a vision of the product, a conceptual model, and a few high-level scenarios that describe product usage and its effects on the persona's life (for example, the video about Apple Knowledge Navigator (Apple Inc., 1987)). The decisions could be accompanied by rough, low-fidelity prototypes such as wire-frames or mock-up models (for example, the video of Google Maps (Google Inc., 2007)). However, these are not converted into a set of formal use cases or quality requirements yet. A few alternative ways of implementing the interface are acceptable at this stage, even desirable.

Usability evaluation 1 and refinement: This corresponds to HCI activity 4 in Table 3.1 (page 27). The product definition should be evaluated for usability at this stage. This needs to be done in addition to the traditional technical, financial, time, and resource feasibility that Pressman suggested in *"Software Scope and Feasibility"* (pp 677).

A formal usability test with real users may not be required at this stage in most cases. Unless the interface is very new, we may not need detailed interface evaluations. Further, the prototype would have low fidelity, and it would not be easy to handle it during a usability test with real users, though some usability professionals do excellent usability tests even with paper prototypes (Snyder, 2003). In outsourced software development, there is also a practical issue. Budgets are tight as a formal contract for the software development would not be in place until requirements are frozen. Perhaps a discount method such as heuristic evaluation could be used to identify a list of potential problems with the proposed low-fidelity prototype and after scenarios. It is desirable to involve reviewers with varied backgrounds, including expertise in HCI, the domain, and / or the technology. The product definition should be refined to fix the problems found. The deliverable of this activity is a selected, feasible, refined product definition with high-level usage scenarios, possibly represented as essential use cases and a low-fidelity prototype.

Requirements specification: This is not a new activity though we propose to change the name of this activity from *"requirements gathering"* to *"requirements specification"*.

The activity of identifying what is required from the new product should be over by this stage, though it would have not been specified to the extent and in the format required in the SE process models, which happens now. This activity would include the requirements engineering activities of elaboration, specification, validation, and ongoing management, including defining quality requirements and specifying detailed use cases (Pressman, 2005 pp. 177-204). All deliverables of traditional communication phase, including a statement of need and feasibility, a bounded statement of scope of work, functional and non-functional requirements, quality assurance scenarios, detailed use cases, and the low-fidelity prototypes are documented. The main role in this activity is of the business analysts and software engineers. Here the role of the HCI practitioner is secondary, mainly to answer questions, clarify doubts, review the documentation to ensure that it does not deviate from the original intentions, and make changes if issues arise. The role of the business stakeholders is to understand what is being specified and give approvals.

Thus, this phase goes beyond *"communication"* and into *"definition"*, though we retain the title for sake of continuity. Table 7.2 summarises the activities, deliverables, and roles in the new Communication phase.

Table 7.2: Summary of activities, deliverables and roles in the new communication phase.

Activities	**Outcomes**	**Roles**			
		SE	**HCI**	**M**	**U**
Project initiation, stakeholder interviews, technology review	Initial project brief, business goals, technology capabilities and constraints		✓	✓✓	✓
User studies, domain analysis, contextual inquiry, focus groups, user modelling	User problems, goals, opportunities and constraints, user models	✓	✓✓	✓	✓✓
Ideation, brain storming, etc.	Design ideas, possibilities and alternatives	✓✓	✓✓	✓✓	
Product definition, interaction design, information architecture	Product definition options, key usage scenarios	✓✓	✓✓	✓✓	
Usability evaluation 1 and refinement, heuristic evaluation, iteration and refinement	Selected and refined product definition with usage scenarios	✓	✓✓	✓	✓
Requirements specification	Redefined product brief, project scope, domain analysis, product definition, functional requirements, non-functional requirements, task flows, use cases, quality assurance scenarios	✓✓	✓	✓	

SE – represents software engineering disciplines including business analysis, software architecture, programming, testing, quality assurance and project management
HCI – includes disciplines related to HCI including interaction design, information architecture, usability and ethnography
M – represents the product management, but depending on the context, it could also include marketing and / or client representatives
U – represents end users of the product, but in some contexts, it could also include people affected by the product (e.g. patients in a hospital)
✓– Partial involvement / responsibility
✓✓– Full, substantial involvement / responsibility

7.2.2 *Planning*

We do not propose new HCI activities be added to the planning phase of the waterfall model. However, we do suggest the use of HCI metrics to help in estimation, planning, and tracking of HCI activities. We review several existing HCI metrics and propose two new ones in chapters 10 and 11, some of which could help during planning.

7.2.3 *Modelling*

The modelling phase is a transition from the real world to the virtual. The modelling phase of the waterfall model comprises of the activities of analysis and design (Pressman, 2005 pp. 208-353). For the purpose of clarity, we rename them as *"requirements analysis"* and *"software design"*.

Requirements analysis activity consists of converting the requirements into an intermediate analysis model representing information, function, and behaviour. An analysis model is a stage between the real world and the virtual world represented by a design model. Requirements analysis focuses on formally defining a hierarchy of domain classes that mirror the real world, their responsibilities, the events that affect them and the relationships between them. The models are expressed formally in a language such as the UML.

Software design is about defining a high-level organisation of the software architecture and a low-level detailed component design including classes, and class interfaces of the proposed product. While analysis classes are about mirroring the real world, design classes are about deciding how these are to be implemented.

This is the least inter-disciplinary phase of the waterfall model. It is the time where the engineers and the HCI practitioners go and do their own bits. The only reason for one group to talk to the other during modelling would be if what they do changes the requirements in any way. Changes in requirements affect work of both disciplines. In that sense, the requirements document is an important boundary object between the two disciplines during the modelling phase. Managing requirements with both the groups during the modelling phase would be the main job of the product / project manager.

We do not suggest any changes to the requirements analysis and software design activities (other than an elaboration of the titles). In this phase, we propose that HCI practitioners complete the following design related activities and deliverables during modelling phase of the waterfall model:

Detailed UI prototyping: This corresponds to HCI activity 5 in Table 3.1 (page 27). It is the crux of HCI design activities where all the detailed design decisions are made and where the chief deliverable of the discipline flows out. This activity comprising of creating detailed mock-ups and prototypes. The deliverables of this activity are refined UI screens for use cases, and in case of hardware, the 3-D mock-up models, and form factors. Typically, this activity is done in two or more iterations with the next activity of usability evaluation between the iterations. The high-risk interfaces, and interfaces used in the most frequent or most critical scenarios are prototyped and evaluated first. The fidelity of the prototypes and the details of the use cases increase through iterations. As the UI details are finalised, these are updated into the requirements document and the software engineering team is updated about the additions.

Usability evaluation 2 and refinement: This corresponds to HCI activity 6 in Table 3.1 (page 27). It involves formative evaluation of the UI prototype with real users through techniques such as card sorting, think-aloud protocol, and performance tests. The outcome of this activity is a prioritized list of potential usability problems with the UI prototype, which is used during redesign. Another outcome could be user experience metrics.

A natural sequence and iteration is implied in the above two HCI activities. The outcomes of these two HCI activities are detailed and validated UI screens, UI standards, metrics, and use cases. In case of hardware products, the outcomes also include hardware specifications, form factor, and drawings. The outcomes of this activity are updated into the requirements specification document, but could sometimes be also consumed directly into the construction activity (e.g. HTML prototypes or CSS specifications). Since the HCI activities and SE activities have few interdependencies other than the requirements document, these could be scheduled in parallel. Table 7.3 summarises the activities, deliverables, and roles in the new Modelling phase.

Table 7.3: Summary of activities, deliverables, and roles in the new modelling phase.

Activities	**Outcomes**	**Involvement**			
		SE	**HCI**	**M**	**U**
Requirement analysis	Analysis models, domain classes, entity relationship models, data and control flow models, state diagrams, CRC models	✓✓		✓	
Software design, software architecture, detailed software design	Software architecture, design classes, class interfaces, collaboration diagrams, activity diagrams, component diagrams	✓✓		✓	
Detailed UI prototyping	Refined UI screens and prototypes coupled with use cases and task flows, mock-up models in case of hardware		✓✓		
Usability evaluation 2, formative user tests such as think aloud tests, card sorts, and performance tests to measure speed of use and errors, refinement of the UI prototype	Usability problems, modified UI screens, UI standards and mock-up models, metrics		✓✓	✓	✓✓

SE – represents software engineering disciplines including business analysis, software architecture, programming, testing, quality assurance and project management

HCI – includes disciplines related to HCI including interaction design, information architecture, usability and ethnography

M – represents the product management, but depending on the context, it could also include marketing and / or client representatives

U – represents end users of the product, but in some contexts, it could also include people affected by the product (e.g. patients in a hospital)

✓– Partial involvement / responsibility

✓✓– Full, substantial involvement / responsibility

7.2.4 Construction

The construction phase involves two activities in the waterfall model: coding and testing. These are the core activities in the waterfall model and are discussed extensively in Pressman and other SE literature.

We do not suggest any changes to the coding and testing activities. However, we propose two activities that could have significant impact on the delivered software:

Development support: This corresponds to HCI activity 7 in Table 3.1 (page 27). In spite of best design documentation, design flaws often creep in during implementation. This may happen not because of lack of intention, but perhaps because of ability and focus. Some people just do not notice design flaws. Often, the specifications of the user interface are not detailed or formal enough. Even where specifications are detailed, the variations within design elements could be too many to specify up front, and it could be easier to catch these defects in reviews. As the development progresses, the outcomes need to be reviewed for conformance with the specification and with the original design intention.

We suggest this as an explicit activity for two reasons. First is a practical reason. In many software development projects, once the UI prototype has been created, evaluated and found satisfactory and specifications have been detailed out, it is assumed that the HCI practitioners' job is done and they move on to other projects. Sometimes the other project is located in another office or another city. In such cases, at least some members of the original design team should be around and accessible to maintain continuity.

The second reason is related to change. As noted above, change is considered inevitable in software development. Development support activity is meant to support design changes late in the development cycle, however it should be noted that these changes would be superficial. If *deeper-than-skin* changes are required, it may be better to redo the modelling phase or perhaps even the communication phase (though this kind of iteration would be disastrous for many projects and is not envisaged in the waterfall model).

Usability evaluation 3: This corresponds to HCI activity 8 in Table 3.1 (page 27). A summative usability evaluation of a suitable early release with real users can go a long way in giving confidence to the design team and the product manager. At the very least, it can bring out the bad news early and help reprioritise the rest of the construction phase. Where possible, it is recommended that an independent group of usability experts carry out this activity, not the one that was originally involved in the HCI design.

At times, a summative usability evaluation could be a part of an acceptance test or a contractual requirement. In such cases, evaluating an early release will not suffice. The evaluation must be done at the end of the process, and usually by an external agency. If such a test is envisaged, the design team may need plan on an internal summative evaluation first to ensure that the product is ready for independent evaluation. Such evaluations are usually performance tests, where speed, errors, and satisfaction measures are collected. Deliverables of this activity would be usability problems and deviations from targeted performance goals accompanied by suggestions for improvements. If performance measures are taken, user experience metrics could be another outcome. If the process is followed well, there should be few severe problems at this stage. The activities and deliverables of this phase are summarised in Table 7.4.

Table 7.4: Summary of activities, deliverables and roles in the new construction phase.

Activities	**Outcomes**	**Involvement**			
		SE	**HCI**	**M**	**U**
Code, integration	Code releases	✓✓		✓	
Test	Test plans, test cases, bug reports	✓✓		✓	
Development support reviews and refinements	Reports of deviations from UI design specification, modified UI screens	✓	✓		
Usability evaluation 3 and refinements	Usability problems, modified UI screens, metrics		✓✓	✓	✓✓

SE – represents software engineering disciplines including business analysis, software architecture, programming, testing, quality assurance and project management

HCI – includes disciplines related to HCI including interaction design, information architecture, usability and ethnography

M – represents the product management, but depending on the context, it could also include marketing and / or client representatives

U – represents end users of the product, but in some contexts, it could also include people affected by the product (e.g. patients in a hospital)

✓– Partial involvement / responsibility

✓✓– Full, substantial involvement / responsibility

7.2.5 *Deployment*

We do not propose any development related HCI activities be added to the deployment phase of the waterfall model. However, in this phase the HCI design team may collect benchmark data in the field on usability metrics such as user performance, HCI effort estimation, and return on investment on HCI activities that may dovetail into user studies activity for the next version.

7.3 Extended Agile Process Model

By using the extreme programming description by (Beck, 2004), (Pressman, 2005) and HCI activities listed in Table 3.1 (page 27) as the base, we propose a modified agile process model.

There is a lot more clarity in the community about how HCI can be integrated with agile processes than there was at the beginning of our research. Much attention to this has come in the last few years, and many ideas that we thought were original, we find have been already echoed in the literature in recent times (discussed in section 6.2.3). While it is frustrating to miss the chance to express these ideas before others, it is also gratifying to know that there are researchers who agree with our views independently. Our proposal for integrating HCI in agile processes therefore is not entirely new, but a different expression. We link each recommendation to the 8 HCI activities that we identified in Table 3.1. In chapter 11, we present a metric to measure integration of HCI with agile process models and present some data from industry projects. Hence, this re-expression is still relevant from the perspective of this thesis.

Figure 7.2 captures an overview of our proposal for the integrated process. The process can be understood in four parts: the pre-iterations phase, the first HCI iteration, the second HCI iteration, and ongoing iterations after that.

7.3.1 Pre-Iterations Phase

Before designing even the first bit of the user interface, the HCI activities of user studies, ideation, product definition, and usability evaluation 1 must be performed. These correspond to HCI activities 1, 2, 3, and 4 in Table 3.1 (page 27). Like in the waterfall model, user studies should be done with the help of ethnographers and designers. After the user studies are done, ideation should be a multi-disciplinary activity where HCI and SE practitioners, business stakeholders, and user representatives are involved. This should be followed by product definition where the high-level vision of the product is accompanied by usage scenarios. These scenarios should be evaluated and if problems are found, those need to be fixed. (The details of each activity are quite similar to the waterfall model described above).

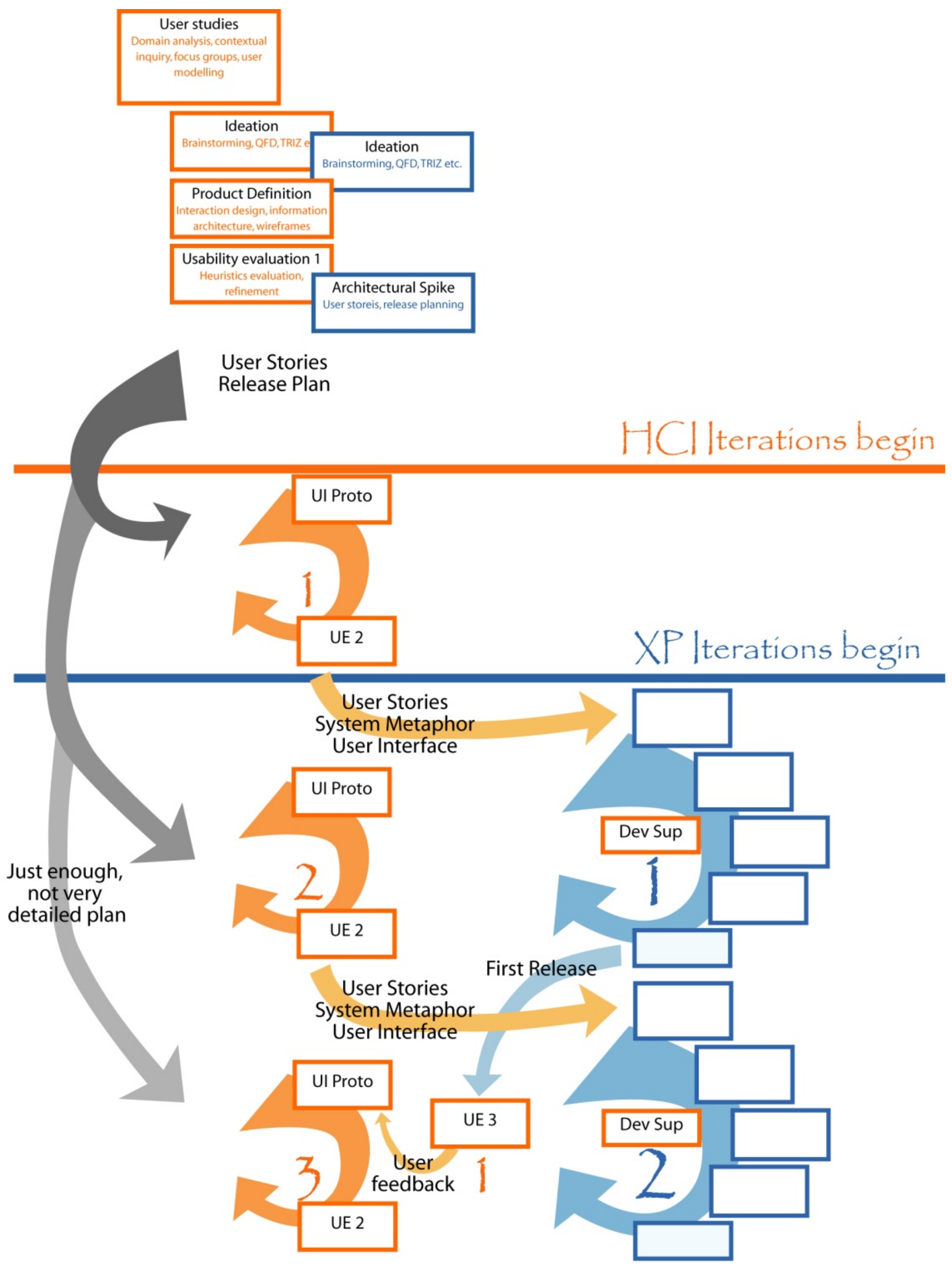


Figure 7.2: A schematic overview of HCI activities integrated in Extreme Programming. Activities in red boxes are the new HCI activities proposed, while blue are SE activities.

It will be impossible to squeeze all these activities in the typical agile iteration. These activities must therefore happen before the XP iterations begin. There could be two ways to manage the pre-XP activities. They could be scheduled just before the project kicks off. In this case, these activities must be done as quickly as possible so as not to delay the project start by too much.

Alternately, these could ongoing activities that an independent research group works on. Such a group can do user studies, ideation, and market analysis on an ongoing basis and propose several high-level product definitions periodically. These product definitions could be evaluated for feasibility and a few of these could be taken up for development. When such a product actually goes for development, it is important that some of the people who were involved in the original research break away from that research group and join the mainstream development. This will ensure that there is continuity of thought in the new team.

Once a product definition and corresponding scenarios have been decided upon, the scenarios can be broken down into user stories, which can be used for the first architectural spike and release planning in XP. The release plan will decide which user stories to build in the first iteration, and which ones to build later.

7.3.2 *The First HCI Iteration*

Agile process models do not specify all the requirements in one go. Agile processes distribute development in short iterations and start delivering bits of working code from the first iteration. This implies that the user interfaces for those bits also need to be delivered from the first iteration, in parts.

Once a high-level product definition is in place and the user stories for the first iteration have been identified, the HCI team should do the detailed design of the interface for those user stories, create a prototype, and evaluate it with users. These correspond to HCI activities 5 and 6 in Table 3.1 (page 27). It is particularly important to do the first iteration carefully, as it could set the direction for the rest of the project. Doing the first iteration carefully should be the HCI team's response to piecemeal design problem with agile methods discussed above. Original assumptions about user stories may be substantially challenged in the first iteration. Yet it must be done in the typical timeline of the development iteration of 2 to 3 weeks. There is no point in designing the entire detailed user interface because that would take away the flexibility of the agile process, and will need a lot more time. Because agile methods are good at managing change, it is also not necessary to get the first iteration absolutely right. It is sufficient to *not* be absolutely wrong.

At the end of the first iteration of HCI, the team should be ready with the evaluated user interface screens so that the first iteration of XP can start.

The first iteration of HCI activities should end before the first iteration of development and subsequently, HCI activities need to stay one iteration ahead. Some companies give more leeway to the HCI teams by letting them be three or four iterations ahead. However, in that case, user stories for development iterations also need to be planned ahead of time for those many iterations. Further, this makes the project more complex for the HCI team as they need to keep track of several versions of the product at a time.

7.3.3 The Second HCI Iteration

Once the first iteration of XP starts, the HCI team needs to work on UI design, prototyping, and evaluation for the user stories planned in the second iteration and be ready with more UI designs for the second iteration by the time the first iteration of XP ends.

At the same time, one more activity is added to the HCI team's tasks on an ongoing basis, viz. development support. This corresponds to HCI activity 7 in Table 3.1 (page 27). This activity is particularly important in context of agile methodologies. The HCI team may not have had enough time to design the first version as they would have liked and may not get time to thoroughly document all design decisions or changes to user stories. In any case, agile methods do not rely much on documentation. The HCI team members should regularly participate in the development team meetings, review ongoing work and provide clarifications or changes as required. This activity is quite compatible with the idea of pair programming in agile methods. This also keeps the HCI team grounded to the development momentum and quickly integrates UI design deviations that are introduced in the current iteration with user interface designs for the next iteration that the team is working on in parallel.

7.3.4 The Third HCI Iteration

In addition to iterative UI design and evaluation for the next iteration, and development support for the current iteration, a fourth strand of work gets added to the HCI team from the third HCI iteration, viz. summative usability evaluation of the release from the last iteration. This corresponds to HCI activity 8 in Table 3.1 (page 27).

Since agile processes release early and release often, it is also possible for HCI teams to evaluate early and evaluate often. This will be a summative evaluation, as iteration 1 would now be over and the development team would have moved on to handling stories in iteration 2. Of course, the HCI team would be working on user stories of iteration 3 and they have the opportunity to incorporate user feedback from the evaluation as usability backlog they found from usability evaluation of iteration 1. It may also be possible to influence the development team to fix problems by ongoing interaction during development support of the second development iteration.

It may not be necessary to do a summative evaluate in every iteration, but it would be important to do them in the first few iterations as early versions of the product are released, and after every two or three iterations thenceforth. In short projects with less than 3-4 iterations, it may be enough to plan one summative evaluation at the end of development. In longer projects, more evaluations may be needed. It may be a good idea to outsource this evaluation occasionally to a third party. It will act as a checkpoint for the entire team.

Figure 7.3 shows a zoomed in view of a typical iteration with all the four strands that will need to be maintained from the third HCI iteration onwards.

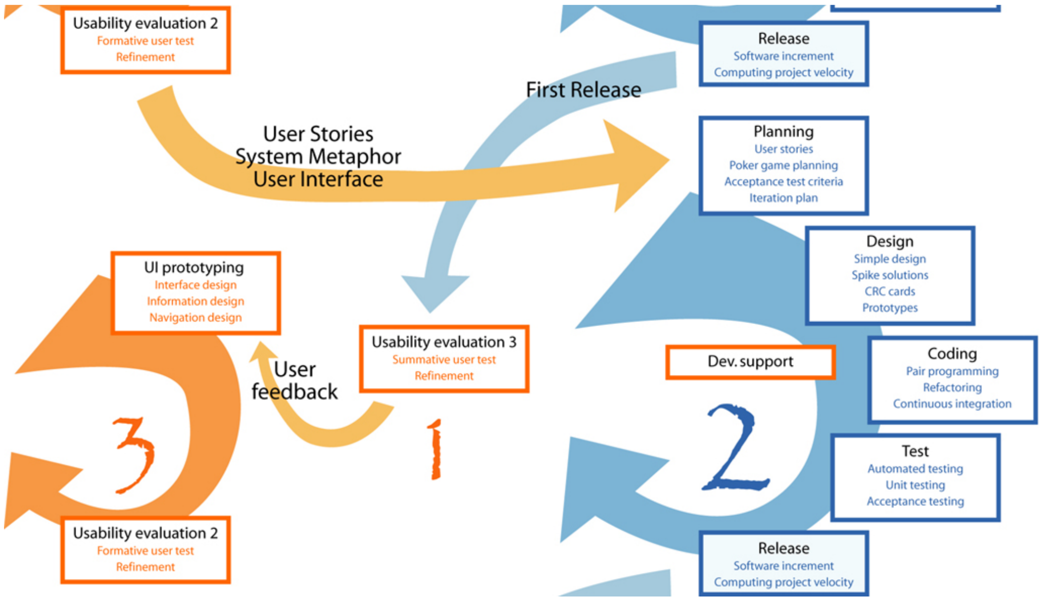


Figure 7.3: A close up view of iteration 2 of XP, where all three strands of HCI activities are visible.

HCI activities of design and evaluation work well in an iterative process model. However, while working with agile processes, the HCI group should also set up its iterations to match the iterations in the development process. Further, it is necessary for the HCI team to keep track of at least three versions at a time: the version delivered by the SE team from the last iteration (for summative evaluation), the development support for the current iteration, and design for the next iteration. Since the HCI team would be managing 3 versions at a time, things can get quite complex and it will need excellent management of resources and time. It is very important for them to keep track of what is happening in which version. While it may be tempting for some HCI practitioners to be 3-4 iterations ahead, it would mean the HCI team have to keep track of 5-6 versions of the user interface at the same time, making the picture a lot more complex.

In chapters 12, we empirically evaluate the effect of integrating HCI activities with the agile process models (as described in this section) on usability goals achievement by analysing data from 11 projects from the industry that followed agile processes.

7.4 Extended Rational Unified Process Model

While integrating HCI activities in RUP, it is important to consider both the static dimension of the RUP as well as the dynamic. The static dimension of the RUP consists of the disciplines, the workflows, and the roles. The dynamic dimension consists of the phases, the activities, the milestones, and the deliverables.

7.4.1 Disciplines, Workflows, and Roles

Three important *skills* can be said to be important in doing the HCI activities: the skill of empathising with the users and understanding their needs and problems (the divergence), the skill of imagining a solution (the transformation) and the skill of evaluating the solution (convergence). We propose three new disciplines to RUP that correspond to these three skills.

- The skill of understanding is represented by the discipline of **user studies**. In context of technology products, a growing number of ethnographers are hired. In addition, people from other disciplines use ethnographic techniques to understand users, their cultures, their needs, problems, and opportunities better. This discipline allows the teams to look at the problem as broadly as possible to answer the question *"what matters"*.
- The skill of creation is represented by the **design** discipline. In a given project, the design discipline may include specialists from any of these creative disciplines: interaction design, information architecture, information design, product design, visual communication design, technical writing, user interface design, and user interface development. In some projects, it may include other creative disciplines including instructional design, illustration, filmmaking, animation, music, and sound engineering. The design discipline becomes active in the second half of the divergence, works through the transformation until the first half of the convergence, and answers the questions *"how should we respond"* and *"how should the design be detailed"*.
- The skill of evaluation is represented by the **usability evaluation** discipline. Typically, practitioners from applied psychology, human factors and ergonomics backgrounds specialise in this discipline. They find usability problems in design at all levels by carrying out activities such as heuristic evaluation, cognitive walkthroughs, think aloud tests, card sorts, and performance tests. This discipline answers the question *"how are we doing"*.

It may be tempting to merge these three skills into one discipline as Göransson et al. have proposed and merely define these three as roles in the structure of the RUP (Göransson, et al., 2003). While this may not be grossly unacceptable, we feel that the above trifurcation better represents the current skills in the industry, the academic disciplines of the current HCI practitioners, and their aptitudes. We anticipate that, as interactive artefacts enter more fields and use multiple media, the roles within the design discipline will become even more specialised, and we will need to leave room to accommodate this future growth. As with other disciplines of RUP, an individual person may play multiple roles within or across disciplines. This is particularly true of practitioners with a lot of experience. In other cases, it is conceivable that the three disciplines are represented by specialist individuals working in tandem. At times, it is conceivable that one or more of these disciplines are outsourced to another company that specialises in those activities.

For clarity, we rename the *"requirements"* discipline as *"software requirements"*, and *"analysis and design"* as *"analysis and software design"*. We also extend the project management discipline to include both product and project management. Figure 7.4 is a modification of Figure 4.2 and summarises the integration of the disciplines with the phases and milestones of RUP.

7.4.2 Phases, Activities, Milestones, and Deliverables

Figure 7.4 also summarises the involvement of the above disciplines with the phases and milestones, the dynamic dimension of RUP.

At the end of the inception phase, the **lifecycle objective milestone** of the RUP must be met so that it results in a go/no-go decision for the rest of the lifecycle. Working towards this milestone, user studies, ideation, product definition, and usability evaluation 1 and refinement (HCI activities 1-4 in Table 3.1, page 27) must happen in the inception phase, followed by the technical and financial feasibility before the lifecycle objective milestone can be met.

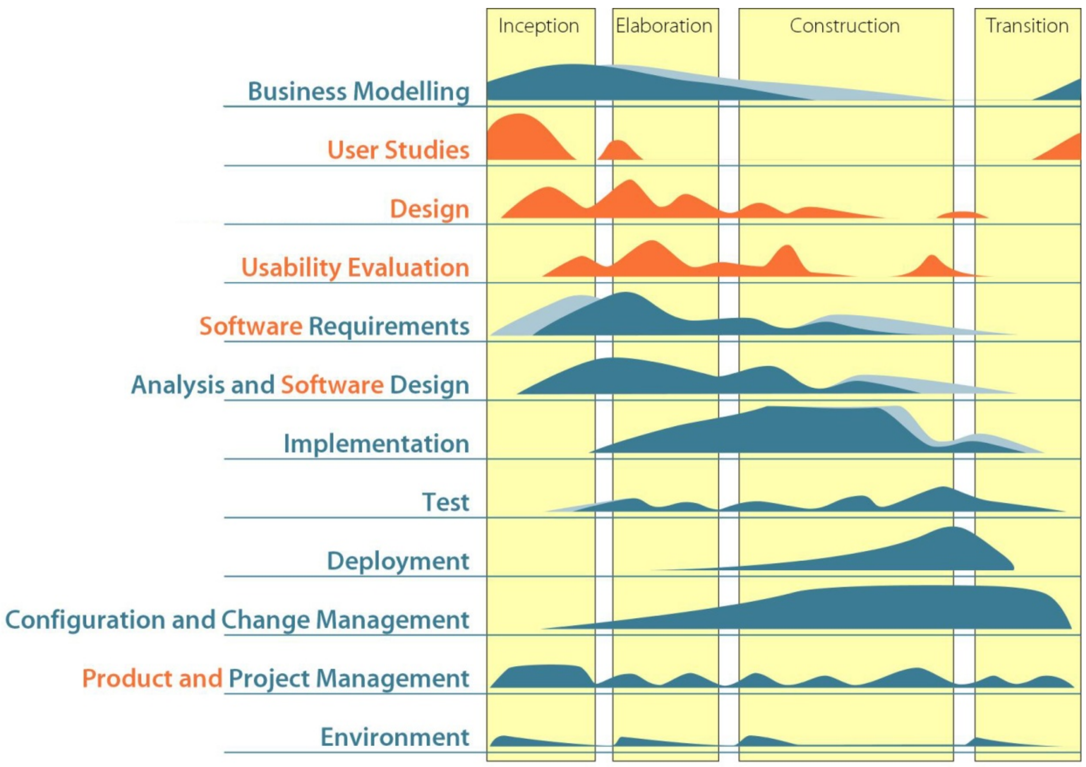


Figure 7.4: A proposal to integrate the three disciplines of user studies, design and usability evaluation with the RUP.

The user studies discipline will make its largest contribution in this phase in identifying user needs, problems, goals, and constraints. The design discipline will look for opportunities and brainstorm possibilities and alternatives. The usability evaluation discipline may participate in feasibility studies and heuristic evaluations of early concepts. These three disciplines, along with the traditional RUP disciplines of business modelling, software requirements, analysis and software design, and product and project management should work together to answer the two critical questions *"what matters?"* and *"how should we respond?"* in order to achieve the lifecycle objective milestone. At the milestone, the team should have a feasible product definition, with a clear understanding of the strategy and scope, cost, schedule, risks, plans and a go / no-go decision.

At the end of the elaboration phase, the **lifecycle architecture milestone** of the RUP must be met, where a stable architecture is agreed upon before going ahead. Fluctuating requirements can represent a major threat to the stability of the software architecture. A usable prototype gives a lot of confidence to the users, the stakeholders, and the developers and can go a long way in stabilising user requirements and hence the software architecture. The outcome of UI prototyping and iterative usability evaluation 2 and refinement (HCI activities 5-6 in Table 3.1, page 27) is a stable, usable prototype. Hence, these activities should be integrated with the elaboration phase of RUP.

Here, the user studies discipline still has a small role to play, discovering the benchmark data for later evaluations. The design and usability evaluation disciplines, along with the business modelling, software requirements and analysis, software design and product and project management disciplines are involved in tight iterations to answer the questions *"how should the product be designed and detailed?"* and *"how are we doing?"*. They may start with low-fidelity prototypes and formative evaluations and end up with a medium-fidelity prototype and a summative performance test. At the milestone, they are ready with a usable prototype and stable software requirements specification. This would help the requirements discipline to stabilise the requirements and the analysis and design disciplines to stabilise the software architecture ahead of the construction phase.

The implementation of the software starts during the **construction** phase. This is where development support (HCI activity 7 in Table 3.1) would be required. This would be provided by the design discipline and the usability evaluation discipline. The design discipline is involved in fixing usability problems found and in supporting development of production quality UI. The usability evaluation discipline is involved in evaluations of successive releases. An early version of the production code with the most important functionality would be available towards the end of the first iteration within the construction phase. This would be a good time to do the summative usability evaluation 3 (HCI activity 8 in Table 3.1), while there is still a chance to fix problems found. In some projects, another round of usability evaluation and refinement might be required at the end of the last iteration in construction, just before the **initial operational capability** milestone is reached and the product moves into the transition phase.

During the **transition** phase, the only involvement is thought of that of the User Studies discipline, where they will be gathering data from the field for the development of the next version.

In addition to resulting in a better user experience in the product, we expect that integration of the three new disciplines we described within the RUP will have an effect on reducing the load on the software requirements discipline (that is shown to stretch right into transition in Figure 4.2). It may also reduce the load of Business Modelling discipline, as the discipline of user studies can share the burden to an extent. Finally (and hopefully), since the contextual user needs would be identified early, this should have a reduction in the implementation and transition time. These hypotheses need to be empirically evaluated. Unfortunately, after a study of 61 projects in the Indian IT industry (reported below), we did not find even one that used RUP. In this report, therefore, we are obliged to leave this proposal as a matter of future research.

7.5 Summary

In this chapter, we proposed the principles of integration of HCI activities with SE process models. Based on these principles, we proposed extensions to the three SE process models with integrated HCI activities.

Having established the process model extensions, a research question naturally arises. How can one prove that a process model proposal is any good? We do this by answering the question, *"How well do these process models help a team achieve the usability goals of the product?"* We propose a metric to measure how well the team achieved its usability goals and another metric to measure how closely they followed our prescribed process models. If the two metrics correlate, we could conclude that the proposed process models help design teams achieve their usability goals.

As we started collecting data from the industry, we realised that several practitioners faced difficulty in expressing the usability goals of their products. To help them express usability goals systematically, we designed a usability goal setting tool that provides guidance on setting and evaluating usability goals.

In chapters 8 and 9, we present our usability goal setting tool and its evaluation. In chapter 10-12, we present our metrics and the validations of the process model proposals discussed in this chapter.

PART III – CORRELATING INTEGRATION WITH GOALS ACHIEVEMENT

8 Usability Goals Setting

In this chapter, we present a Usability Goal Setting Tool (UGT), a tool that helps design teams set and evaluate usability goals. We used UGT in the Usability Goals Achievement Metric (UGAM), described in chapter 10. Though we describe them sequentially in this report, UGT and UGAM evolved in parallel.

In our initial trials of UGAM with industry participants, we realised that many teams required assistance to spell out usability goals of their products. We observed that, at times, the HCI practitioners were not clear about the goals of the product they were designing. In some cases, the HCI practitioners were clear but the other stakeholders were not on the same page. This happened not only in new, unfamiliar projects, but also in cases where goals are being set retrospectively after the project was completed. Though most practitioners agreed that setting usability goals unambiguously is important, several teams needed help to do so.

We created UGT to help teams set usability goals. We wanted to develop a systematic method of setting usability goals. We know from experience that certain patterns of usability goals are predictable based on some input parameters. For example, if we know that a task is very frequent, certain usability goals are likely to be more important. On the other hand, if the product is targeted to non-tech-savvy users, other goals become important. UGT is a repository of such prior experience and a recommender system for future goal setting.

8.1 Usability Goals

8.1.1 User Goals, Business Goals, Product Goals, and Usability Goals

We differentiate between user goals, business goals, and product goals.

User goals are the goals which users possess, and at least some of these influence what the user does with a product and how (*"enjoy a peaceful vacation"* or *"ensure a secure future for my children"*). Cooper and Reimann further differentiate user goals between

end goals that represent tangible outcomes of using the specific product (*"find the best price" or "process the customer's order"*), **experience goals** that express how someone wants to feel while using the product or the quality of their interaction (*"feel competent and confident"* or *"have fun, or at least not be too bored"*), and **life goals** that represent deep drives and motivations and that can explain why the user is trying to accomplish the end-goals (*"be a good parent"* or *"ensure that my child has a good upbringing"*) (Cooper, et al., 2003 pp. 62-67). User goals and intents are investigated during user studies and modelled in user modelling techniques such as personas, scenarios, and work models (Beyer, et al., 1998 pp. 101, 197, 259-261).

Business goals define what the business stakeholders would like to achieve by developing the product (here we mean goals that drive the development of a product, not goals of running a company, though the two are connected). Business goals would be articulated by the product manager, probably based on marketing data, as well as information generated from prior studies of targeted users (*"create a social networking product among youth, make money through advertising"* or *"give away a cheap hardware, make money on content sales on an ongoing basis"*). If the product is a service to support the current business of the organisation, business goals could be articulated by other stakeholders (*"sell airline tickets online to save costs"* or *"save time by automating workflows in the organisation"*). We acknowledge that at times, goals of an organisation may not be related to making money; goals of an NGO (*"reach out to every child in distress"*), or a state funded university (*"encourage research collaboration across disciplines"*) for example. Nonetheless, for the purpose of this research though, we prefer to club these under business goals.

Product goals describe what the product should do and sets out benchmarks against which the design will be evaluated (*"the website will support a vacation end-to-end", "the system will enable a person living with HIV / AIDS to manage his / her regimen"*, or *"the electronic voting machine will enable all eligible voters to vote without help"*). Product goals may be about functionality, costs, reliability, and time as well as about usability and user experience. Product goals are derived from user goals and business goals.

UGT is about helping the team set up product goals. Further, it is about the subset of product goals related to the usability and user experience of the product. A usability goal is a commitment that a designer makes to his client or his management. Setting usability goals is an important step early in the design process. Setting goals before design gives the team a target to achieve. Usability goals help guide the design process, make the design activity tangible, and help evaluate the designs. In HCI design, often multi-disciplinary teams are involved, so setting goals early and getting an agreement from all stakeholders is all the more important. Usability goals can be understood easily, even by non-UX-professionals. Stakeholders outline the usability goals and HCI practitioners fine-tune them based on their knowledge and findings from user studies.

8.1.2 Prior Work about Usability Goals

There is considerable related work about goals, usability and user experience. Here we mention only the work that has directly influenced UGT.

Design for a need has been a part of traditional industrial design thinking. Charles Eames reportedly said, *"Design is a plan for arranging elements in such a way as best to accomplish a particular purpose"* (Eames). This emphasis on goals probably dates back to the days of Bauhaus school for design (1920s), and certainly from the days of the Ulm school (1950s). Archer explains that since design is necessarily associated with change, identifying goals means, *"defining the needs and pressures which constitute the driving force for change"* (Archer, 1965). According to Archer, the first step is to determine the goals of the design effort together with *"the essential criteria by which a good solution will be distinguished from a not so good solution"*. Jones adds that goal setting is a creative act that turns a complicated problem into a simple one by deciding what to emphasise and what to overlook. It happens after the divergence stage and at the beginning of the transformation stage, when the objective, the brief, and the problem boundaries are fixed, when critical variables are identified, when constraints are recognised, when opportunities are taken and when judgements are made (Jones, 1970 pp. 66-67).

In a typical industrial context, it may not be possible to meet all goals imaginable and it is necessary to prioritize. Archer talks about rank ordering sub-problems as a method of prioritizing goals and resolving conflicts (Archer, 1965). Cross talks about an objective tree method, organizing objectives into a hierarchy of higher and lower level objectives (Cross, 2000 pp. 61-76). The first product brief may be both brief and vague. Cross suggests that the designer expand this into a detailed list and organise it in a hierarchy of higher level and lower level objectives.

The closely related fields of usability, interaction design, and information architecture also emphasise the importance of goals. ISO 9241 defined usability as the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use (International Organization for Standardization, 1997). Shneiderman (Shneiderman, 2004) and Nielsen (Nielsen, 1993) agree on five high-level usability goals of a product: learnability, speed of use, error-free use, retention over time, and subjective satisfaction. Mayhew categorises usability goals as qualitative and quantitative, and as performance goals, preference goals, and satisfaction goals (Mayhew, 1999). ISO 9126-1 described usability in terms of understandability, learnability, operability, and attractiveness (International Organization for Standardization, 2001). Cooper and Reimann emphasise the importance of goals in usability and interaction design (Cooper, et al., 2003). Bevan summarises several other ways of organising usability measures, which could be looked upon as goals (Bevan, 2008).

Relatively recent research interest in human emotions has broadened the traditional focus of researchers from usability to user experience. However, approaches to user experience goals are not universal. There is no single accepted definition of what user experience is, and some researchers wonder whether it could be designed at all. McCarthy and Wright preferred to look at people's experience in terms of *"felt life"* (McCarthy, et al., 2004 pp. 12-21). They believed that people actively construct their experiences and that each person's experience is unique, rich, and difficult to communicate, let alone design. It is unclear in this approach, what the goals of the design efforts that create technology-mediated experiences ought to be. In a recent survey, Law et al. concluded that the concept of user experience is dynamic, context-dependent and subjective (Law, et al., 2009). While there may be no explicit agreement on a complete list of user experience goals, most researchers and practitioners favoured a goal-driven approach in design.

The next section describes the details of UGT. The first purpose of the UGT is to help set usability goals systematically and carefully. UGT also helps set a few common user experience goals. Further, UGT gives flexibility to the HCI practitioners to add or delete goals or to state them in another way.

8.2 Usability Goals Setting Tool

UGT aims at helping HCI practitioners and multi-disciplinary design teams to set goals for their interactive product. UGT is envisioned as a part recommender system and a part collaboration tool. It helps break down high-level goals into more concrete, measurable goal parameters. It helps prioritise goal parameters by assigning each a suitable weight. This section provides an overview of UGT. A detailed version of UGT, including step-by-step instructions, is described in Appendix I and on the UGT website (Joshi, 2009). Using UGT involves four steps: initiate, set goals, review goals and set evaluation guidelines (Figure 8.1). These steps have been explained below.

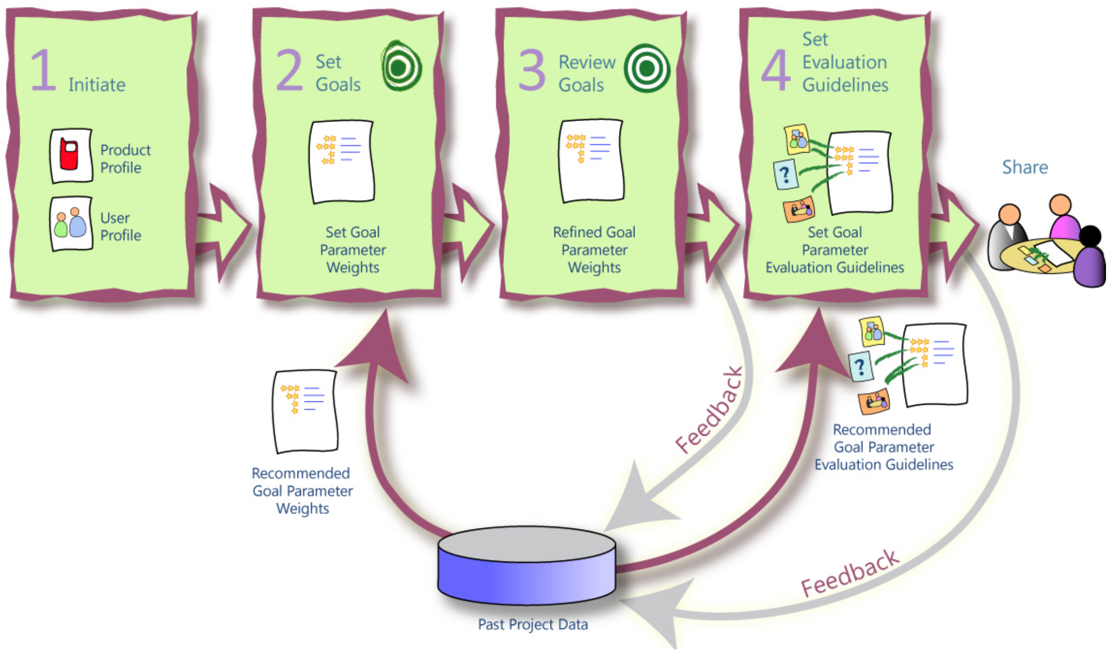

1 Initiate
Product Profile
User Profile
2 Set Goals
Set Goal Parameter Weights
3 Review Goals
Refined Goal Parameter Weights
4 Set Evaluation Guidelines
Set Goal Parameter Evaluation Guidelines
Share
Recommended Goal Parameter Weights
Feedback
Recommended Goal Parameter Evaluation Guidelines
Feedback
Past Project Data

Figure 8.1: An overview of UGT usage workflow.

8.2.1 *Initiate*

The HCI practitioner approaches UGT when he / she is clear about the product brief and has sufficient understanding about the domain, the problems, the context, and users.

UGT begins by capturing this understanding as product and user profiles. These form the input for UGT.

The HCI practitioner creates a product profile by answering the following questions:

- Name, version, and industry domain of the product
- Work practice domain: life critical; business critical; goal-oriented; casual; enabling technology; e-learning; information product; entertainment product
- Expected cost to the user: premium product; low-end product; not-free-but-I-don't-pay product; free with another product / service; free-to-use product
- Platform(s): desktop computers; web; mobile phones; interactive voice response systems / call centres; television; custom platform
- How many types of users will use the product? Which ones?

Next, the HCI practitioner creates one or more user profiles by answering the following questions for each user type:

- Age (pick a range)
- Lowest level of tech-savvyness targeted: Low (e.g. cannot save a contact, but can make a call on a mobile phone); moderate-low (e.g. can save a contact on a phone); moderate (e.g. checks mail every day); moderate-high (e.g. can install applications); high (e.g. writes programs, loves Linux). Pick the lowest level targeted.
- Range of frequency of use: once; once a year; once a month; once a week; once a day; few times a day; continuously. Pick a frequency or a frequency range.
- Product complexity: very complex; complex; moderate; simple; very simple. Pick any one.
- Nature of the market: mass market; wide market; niche market; internal application for trained staff; thick domain product for experts. Picks any one.
- Value addition of this product in user's life / motivation to use the product: no intrinsic motivation; I have many options; socialize; it entertains me; it's my hobby; I have been using it; it saves my time, informs me, or helps me do my job

better; it makes things accessible; it makes / saves money for me; I have no choice, I have to use it as a part of my job. Pick all that apply.

8.2.2 Set Goals

Once the HCI practitioner has defined the user and product profile, he sets goals. UGT recommends a list of potential goals and breaks them down into more granular goal parameters (Table 8.1).

A recommended list of goals ought to be generic enough to be applicable in a wide variety of projects, and yet granular enough to be interpreted consistently. While high-level usability goals such as learnability or ease of use are widely applicable, these are at times too generic and could be interpreted differently by different people. For example, one can interpret learnability either as *"it should take less time to train users"*, or as *"users should be able to learn to use the product on their own"*. This might make it difficult to evaluate the design consistently.

While recommending the goal parameters in Table 8.1, our attempt was to achieve a medium-level granularity. We wanted these goal parameters to be easy to interpret consistently and yet they should be applicable to a wide range of products. One should be able to define unambiguously how the design will be evaluated against these goal parameters.

To begin with, the high-level goals in UGT were derived from (Shneiderman, 2004 pp. 14-16) and (Nielsen, 1993 pp. 24-37, 79-85). An initial set of 20 goal parameters were derived from these goals based on our experience and supplemented by brainstorming with practitioners. These were improved upon and expanded to 30 goal parameters in Table 8.1 with the help of formative evaluations described in chapter 9.

The goal parameters recommended in UGT are not meant to be comprehensive and may not cover all aspects of user experience in every situation. They are suggestive of the kind of goals that the HCI practitioner *could* set, akin to a checklist. At all times, the practitioner has the freedom to add goal parameters, edit, or re-word the suggested goal parameters, and to group goal parameters differently if preferred. Yet, as we report below, the current list of goal parameters has both reasonable coverage and sufficient granularity.

Table 8.1: Goals and goal parameters in UGT.

	Learnability
1	**Findability**: options / data / information should be visible / easy to find
2	User should take less **time to learn**: (e.g. in < 10 minutes, in < 2 hours practice)
3	Users should be able to **learn on their own**
4	Product should be internally consistent
5	Product should be consistent with **other products**, older **methods** / past **habits**
6	Product should be consistent with the **earlier version**
7	User should remember / **retain** critical, but infrequent tasks
	Speed of use
8	User must be able to do **primary** task / **frequent tasks** quickly, easily, at all times
9	User should be able to **navigate** quickly and easily
10	Product should not load user's **memory** / product should not put **cognitive load**
11	**Flexibility**: user should **control** the sequence of tasks
12	User should be able to complete tasks in specific **time** / no. of **steps** / in less **efforts**
13	Product should be **personalised** for the user automatically
14	Product should be **localised** for specific market segments
15	A user should be able to **customise** the product for himself
	Ease of use
16	Interface should clearly communicate the **conceptual model**
17	**Intuitiveness**: User should be able to **predict** the next step / task
18	No **entry barrier**: user must be able to complete **critical first tasks**
19	Product should require no **unnecessary tasks**
20	Product should **automate** routine tasks / minimise user **task load**
21	Product should be **always on**, always accessible
	Ease of communication
22	Information architecture: well aggregated, well categorised, well presented
23	**Communication** should be clear / user should easily **understand** text, visuals
	Error-free use
24	Product should give good **feedback** / display its current **status**
25	Product should not induce errors
26	Product should tolerate user's errors / **forgiving** interface / should prevent errors
27	Product should help user **recover from errors** / help users **troubleshoot** problems
	Subjective satisfaction
28	User should **feel in control** of the product / **behavioural** appeal
29	User should feel **emotionally engaged** / brand / fun / reflective appeal / trust
30	User should find the product **aesthetically** appealing / **visceral** appeal

Shneiderman states, *"a clever design for one community of users may be inappropriate for another community"*, and *"an efficient design for one class of tasks may be inefficient for* *another class"* (Shneiderman, 2004 p. 14). Depending on the context, users and platform, some goals may be more important for a project, while others might be irrelevant.

In UGT, the HCI practitioner determines the priority of a goal parameter by assigning it one of the following weights:

- Extremely important / unique selling proposition (5)
- Very important (4)
- Important (3)
- Usual relevance / hygiene factor (2)
- Somewhat relevant (1)
- Irrelevant (0)

Weights express the relative importance of goals and parameters in the context of a project. For example, a product meant to be used several times a day by a call-centre agent is likely to have higher weight for speed related goal parameters. A one-time use product like a web site for visa application for a tourist might give importance to learnability and ease of use. On the other hand, a life-critical product to be used in an operation theatre is likely to rate highly error-free use and may sacrifice learnability.

While it may be tempting to set a high weight to each goal, goal-setters should be aware that it might not be necessary, practical, or even possible to achieve such a design. The weights should reflect the priorities of the business, the stakeholders, and the users. The weights help prioritize usability evaluation activity. The highest rated goals and parameters must be evaluated more thoroughly, while the lower weighted goals could be perhaps evaluated by a discount method.

In addition to suggesting a list of goal parameters, UGT also provides recommendations for assigning weights to these goal parameters based on user and product profile contents. For example, for the goal parameter 1, *"Findability: options / data / information should be visible / easy to find"*, UGT recommends assigning the following weights:

- **Assign weight 4-5** if the product is targeted to low-tech users, older users, or if the application complex, business-critical or life-critical.
- **Assign weight of 2-3** if the product is meant for frequent use (as users will get used to it after a while or learn shortcuts, though make sure that the frequent tasks are easy to access), or if the product is extremely simple and no findability issues are expected.
- Usually, findability is assigned a weight of 2 or more – it is almost always relevant.

Detailed recommendations for all the 30 goal parameters are described in Appendix I and on the UGT website (Joshi, 2009).

As shown in Figure 8.1, UGT has self-learning capabilities. The weights suggested by UGT are based on experience of past products with similar user and product profiles. The HCI practitioners may use the recommendations or choose to ignore them. These decisions are captured and fed back to enrich UGT further. As UGT collects data from more projects, it gives more accurate and relevant recommendations for weights. The current recommendations in UGT are derived from feedback of 15 think aloud sessions and manual inspections of those projects described in chapter 9. As data from more projects becomes available, UGT would give improved recommendations.

8.2.3 Review Goals

After the HCI practitioner has done one pass of assigning weights to goal parameters, the list of goal parameters is re-presented, this time sorted by their weights. Goal parameters with weights that deviate substantially from the recommended range are highlighted. The HCI practitioner and other team members review the weights and tweak them. This step was introduced because it was observed during the early evaluations that HCI practitioners over-assigned weights in the first pass, but preferred to tone them down during a review.

8.2.4 Set Evaluation Guidelines

At this step, the HCI practitioner is asked to suggest ideas for how the design will be evaluated against each goal parameter.

UGT suggests an evaluation technique for each goal parameter in UGT. It suggests at least one technique for a user-based test and one for a review-based evaluation. For example, for the goal parameter 1, *"Findability: options / data / information should be visible / easy to find"*, UGT suggests the following:

- **In a usability test**, ask users to do tasks that require them to find information or menu options. Ask them to think aloud while doing tasks. Look for confusions and unmet expectations.
- **In a performance test**, give users a goal to achieve that requires them to locate options in different parts of the product. Count errors or the amount of time it takes them to do so.
- **During a review**, look for data / options that are not organized consistently with the users' mental models. Findability problems can be hard to spot unless the reviewers are familiar with the users' abilities and mental models.

Techniques for evaluating all the 30 goal parameters are described in Appendix I and on the UGT website (Joshi, 2009). These are based on the formative evaluation reported in chapter 9, and further brainstorming. Again, the HCI practitioner has a choice to use the suggested evaluation ideas. As shown in Figure 8.1, these decisions are mined and fed back into UGT.

Once the HCI practitioner and other team members are satisfied with goal parameter weights and evaluation guidelines, they can share this data with a wider group (including client, marketing professionals, product managers etc.). They give their inputs and after negotiations, the weights for the goal parameters are finalised. Future versions of UGT may have options for online collaboration, negotiation, and tracking. These steps of UGT have been envisaged, but not prototyped or evaluated yet. All studies reported here either were conducted by one person per project, or were shared with the team face to face or over emails.

8.3 Summary

In this chapter, we presented Usability Goals Setting Tool (UGT), a tool to help HCI practitioners set usability goals. UGT suggests 30 medium-level granularity goal parameters. Depending on the user needs and the context of the project, the HCI practitioner assigns a weight of 0 to 5 to each goal parameter, and sets guidelines to evaluate it. Based on previous experience, UGT recommends goal parameter weights and evaluation guidelines, but the HCI practitioner has the freedom to go beyond the recommended weights and guidelines. The HCI practitioner may add, delete, combine, or modify suggested goal parameters to suit the needs of the project. The decisions are fed back into UGT to improve its recommendations.

In the next chapter, we describe formative and summative evaluation of UGT. UGT is used for goal setting in the Usability Goals Achievement Metric, described in chapter 10.

9 Evaluations of the Usability Goals Setting Tool

9.1 Formative Evaluation

UGT was formatively evaluated and improved upon in think-aloud sessions. The purpose of the formative evaluation was to find the HCI practitioners' qualitative responses to a tool such as UGT and to make UGT as complete as possible, so that it could be applied in many types of projects. During the formative evaluations, UGT was continuously modified to improve its coverage and completeness.

A paper prototype of UGT was developed. Interaction design students and HCI practitioners were invited to use the prototype to define the product and user profiles, and set and review goals for a project that they had recently completed, while thinking aloud. They were given the freedom to add or edit choices at each step. The meaning of each goal parameter was carefully explained with examples where necessary. These explanations and meanings have been captured in Appendix I and on the UGT website (Joshi, 2009). Once the participant had finalized the goal parameters and weights, he / she was asked to describe a usability test that he / she will use to evaluate the design against the important goal parameters (the ones that were assigned weights of 3 or more). Then, the participant was asked to comment about UGT, probed on whether UGT expressed his / her ideas sufficiently, and whether the goal setting activity was useful.

Totally, 15 people participated in the formative evaluation. Of these, 4 were senior HCI practitioners from the industry and 11 were interaction design students who had recently finished a masters-level programme. The participants suggested many changes. UGT was continuously modified during and after each session to incorporate these changes. Goal parameters were added, re-worded, split, merged, or regrouped to fit the contexts. Yet, each goal parameters were kept generic enough so that it can be applied in a wide range of products and expressive enough to suit the needs of the individual products.

The high-level goals of learnability, ease of use, speed of use, error-free use, retention, and subjective satisfaction had two important changes from (Shneiderman, 2004 pp. 14-

16) and (Nielsen, 1993 pp. 24-37, 79-85). Retention did not turn out to be important enough for most projects for it to be an independent high-level goal. Therefore, it was merged as a medium-level goal parameter under learnability. A new high-level goal was introduced, ease of communication. This emerged to be important in cases of the information and communication oriented products such as web sites and learning tools. After the formative evaluation, the total number of goal parameters went up from 20 to 30. Most changes were discovered in the early evaluations. No major changes happened in the last two evaluations and only two changes happened in the last four evaluations. Table 8.1 (page 128) lists the modified goal parameters that emerged from the formative evaluation.

The formative evaluation also generated qualitative feedback about UGT. Many participants had a positive response to the activity. Participants agreed that information required for product and user profiling is usually available early. One participant remarked, *"These were some of the things that I should have done earlier. Had I done them, the product might have been better."* Several participants also thought that the activity helped them prioritize goals, made many things clear, and gave many insights. It helped them consider goal parameters that they had not considered earlier. A participant remarked, *"Though I had set goals for these projects earlier, it was not so systematic"*.

Participants liked the activity of setting evaluation guidelines for each goal early. It made them think through the goal carefully at the initial stage and gave them better control over the project. None of the participants had thought of setting evaluation guidelines in this way. One participant was undecided about if he could share the goals with non-HCI practitioners as he felt it required a lot of background about HCI. On the other hand, another participant remarked, *"this will be helpful to bring the team on the same page"*. Another participant thought she could use UGT to pitch for new projects, as it makes relevant interaction design activities tangible and *"communicates why we are charging so much"*.

After the formative evaluation, product profiles, user profiles, and goal parameters of the 15 projects were manually inspected to identify patterns that predict the weight of goal parameters. Though not all, many goal parameters were found to be somewhat related to the product and user profiles. Detailed qualitative recommendations for assigning weight were developed for each goal parameter, included in Appendix I and on the UGT website (Joshi, 2009).

During the formative evaluation, participants were also asked to describe a usability evaluation that they will use to test the design against goal parameters. Brainstorming was done to generate more evaluation ideas for each goal parameter. At least one user-based test idea and one review-based evaluation idea was generated for each goal parameter, also included in Appendix I and on the UGT website (Joshi, 2009).

9.2 Summative Evaluations

After the UGT had stabilized from the formative evaluations, two summative evaluations were done with help of 49 HCI practitioners from the industry. The purpose of the summative evaluation was to evaluate if the tool was useful, if it helps the practitioners understand the context of their project, and if it helps them think of more usability goals than they had thought of earlier.

The first evaluation was during a nine-day professional course on interaction design taught by the author. It was attended by participants from mixed backgrounds such as graphic design, product design, user interface design, e-learning, engineering etc. Most had no formal education in HCI or a related field. Their experience as a HCI practitioner ranged from 0 to 3 years. Participants were introduced to the UGT. Each participant was then asked to set goals for an industrial project that they had worked on. After the goal-setting exercise, participants were asked to fill out a post-test questionnaire. Participants were also asked to suggest more goal parameters relevant to their projects.

A second, similar evaluation was done with participants who had more experience and formal backgrounds in HCI or design. These participants had a mean experience of 7 years and HCI related experience of 3.5 years.

The post-test questionnaire was filled by 34 participants in the first test (the relatively inexperienced group) and 15 participants in the second test (more experienced group).

Table 9.1 summarizes the questions and responses to the two post-test questionnaires. The lighter rows represent the data from the first evaluation (relatively inexperienced group) and the darker rows represent the data from the second evaluation (relatively experienced group).

Table 9.1: Post-test responses to UGT by 34 relatively inexperienced participants (lighter rows) and 15 more-experienced participants (darker rows).

	Disagree (-2)	(-1)	(0)	(1)	Agree (2)	Mean rating	n	D	α <
This was a useful exercise.	0	0	2	3	29	1.79	34	.65	.01
	0	0	1	4	10	1.60	15	.53	.01
This exercise helped me understand the context of my project better.	1	0	1	5	27	1.68	34	.59	.01
	0	0	0	8	7	1.47	15	.60	.01
This exercise made me think about relevant user experience goals that I had not considered earlier.	1	0	2	10	21	1.47	34	.51	.01
	1	0	1	5	8	1.27	15	.47	.01
I wish I had done this goal-setting exercise while I was doing my project.	0	0	0	8	25	1.76	33	.60	.01
	0	2	6	2	5	0.67	15	.27	- -
Had I done this exercise during the project, it would have helped me do my job better.	0	1	2	6	24	1.61	33	.52	.01
	0	0	2	4	9	1.47	15	.47	.01
Had I done this goal-setting exercise during the project, it could have led to better user experience.	0	0	4	5	24	1.61	34	.52	.01
	0	2	3	4	6	0.93	15	.27	- -
I will do this exercise when I am doing my next project.	0	0	0	10	24	1.71	34	.60	.01
	0	0	2	5	8	1.40	15	.47	.01
I think it is possible to involve my colleagues / boss / clients in this exercise.	0	0	4	9	21	1.50	34	.48	.01
	0	1	1	8	5	1.13	15	.47	.01

The Kolmogorov-Smirnov test can be used to determine whether the distribution of the sample across ranks of an ordinal scale is a good fit to a hypothesised uniform distribution (Kurtz, 1983 pp. 167-170). The D values of Kolmogorov-Smirnov test for all questions for inexperienced participants and questions 1, 2, 3, 5, 7, 8 for experienced participants are significant at $\alpha = 0.01$. While the mean ratings to all questions are positive, the inexperienced group consistently gave higher ratings.

We can conclude that both groups agreed that the exercise was useful (Q1), that it helped them understand the context of the project better (Q2), and that the exercise made them think about goals that they had not considered earlier (Q3). Both groups believed that UGT could have helped them do their job better (Q5), wanted to use the technique in their next project (Q7), and felt that they could involve their colleagues in the exercise (Q8). The inexperienced group wished that they had done the goal-setting exercise during their projects (Q4), and felt that it would have led to a better user experience (Q6).

9.3 Additional Analyses of Goal Parameters

The formative evaluations helped improve UGT and the summative evaluations helped establish its usefulness. Additional data was collected from industry projects to evaluate the coverage, relevance, granularity, and internal validity of goal parameters. Data was explored to identify the relationship between the weights and scores of goal parameters. The relationship between the product and user profiles and the goal parameter weights was also explored with the aim of making quantitative recommendations for weights of goal parameters in UGT.

Data was collected from 65 industrial projects. Projects represented a wide variety of the Indian IT industry including company size, geographical spread, type of business (contracted software development companies and product companies), and process model used (waterfall and agile).

For each project, data was collected about user profiles, product profiles, and goal parameter weights as described above. In addition, participants were also asked to rate the performance of their final designs against each goal parameter on a scale of 0-100 where 0 represents the worst possible score of user experience on account of that goal parameter, 25 represents a score that is quite bad, though not the worst possible, 50 represents an undecided score, 75 represents a good enough score, and 100 represents the best possible score of user experience design on account of that goal parameter. Participants were encouraged to use information from usability tests, reviews, and user feedback where available while giving their scores for each goal parameter.

Table 9.2 summarises goal parameter weights (means and standard deviations) and goal parameter scores (means and standard deviations) across the 65 projects.

Table 9.2: Number of projects that assigned a particular weight to a goal parameter, goal parameter weight means and standard deviations, number of projects that assigned a particular score to a goal parameter, and goal parameter score means and standard deviations. The numbers refer to the goal parameter numbers in Table 8.1. (n = 65)

	Projects out of 65 with weights						**Weights**		**Projects out of 65 with scores**					**Scores**	
	0	**1**	**2**	**3**	**4**	**5**	**Mean**	**SD**	**0**	**25**	**50**	**75**	**100**	**Mean**	**SD**
1	0	0	3	13	30	19	4.00	0.83	0	1	16	38	10	71.92	16.83
2	1	5	12	24	17	6	3.06	1.13	0	7	22	31	5	63.08	19.82
3	5	6	15	14	19	6	2.83	1.40	5	5	16	36	3	60.38	24.56
4	1	4	17	19	17	7	3.05	1.16	2	4	18	28	13	67.69	24.09
5	5	5	19	18	8	10	2.75	1.41	4	7	31	17	6	55.38	24.40
6	37	3	8	6	6	5	1.32	1.75	34	1	14	14	2	30.38	34.09
7	10	12	11	17	10	5	2.31	1.52	8	7	26	22	2	51.15	25.93
8	3	3	5	14	28	12	3.49	1.28	4	2	10	33	16	71.15	25.86
9	1	1	12	11	30	10	3.51	1.11	0	7	11	32	15	71.15	22.63
10	1	2	23	14	17	8	3.05	1.18	4	10	15	33	3	58.08	25.04
11	7	8	16	19	11	4	2.48	1.37	6	9	22	25	3	53.85	25.86
12	4	1	13	20	15	12	3.18	1.33	2	3	22	31	7	64.62	21.60
13	15	10	12	6	17	5	2.23	1.70	19	9	12	21	4	43.08	33.80
14	17	12	8	10	9	9	2.14	1.79	20	5	18	13	9	44.62	35.77
15	29	9	6	8	9	4	1.55	1.73	33	6	9	13	4	30.38	35.21
16	0	5	8	18	21	13	3.45	1.17	0	4	20	37	4	65.77	17.44
17	1	4	11	25	18	6	3.12	1.10	0	3	27	27	8	65.38	19.11
18	7	6	10	19	16	7	2.80	1.47	6	8	20	25	6	56.54	27.34
19	3	2	19	26	12	3	2.78	1.08	1	9	16	32	7	63.46	23.00
20	10	7	12	16	16	4	2.51	1.51	9	8	22	23	3	51.15	27.75
21	10	6	16	10	10	13	2.66	1.70	13	6	11	26	9	54.62	33.92
22	0	1	3	24	18	19	3.78	0.98	0	3	15	34	13	71.92	19.52
23	0	0	5	14	29	17	3.89	0.89	0	3	17	31	14	71.54	20.19
24	0	0	10	21	22	12	3.55	0.97	1	5	19	27	13	67.69	23.27
25	3	0	17	24	14	7	3.03	1.16	3	6	22	30	4	60.00	22.88
26	3	6	20	24	10	2	2.58	1.10	4	7	28	20	6	56.54	24.72
27	2	5	19	18	16	5	2.86	1.20	3	7	24	24	7	59.62	24.48
28	1	2	9	28	19	6	3.23	1.01	3	2	15	40	5	66.15	21.39
29	8	12	10	15	12	8	2.54	1.58	11	7	24	17	6	50.00	29.97
30	2	6	18	21	14	4	2.78	1.17	2	9	20	26	8	61.15	24.62

9.3.1 *Coverage, Relevance, and Granularity*

While recommending goal parameters in UGT, our aim was to achieve a reasonable level of granularity so that the goal parameters are concrete, specific, measurable, and interpreted unambiguously. At the same time, we wanted the goal parameters to be widely applicable, relevant to a large number of projects, and internally reliable. We evaluated the data from projects to verify if this was indeed the case.

Coverage is the extent to which the goal parameters suggested by UGT suffice the goal-setting needs of a project. One method of evaluating if UGT has enough coverage was to look at how many new goal parameters were added by practitioners. If UGT lacked coverage, many participants would feel the need to define additional goal parameters before they were satisfied that the usability goals of their product have been expressed sufficiently. Throughout the study, the participants had the freedom to add, edit, merge, or regroup goal parameters if the suggested ones did not suit their needs. This was indicated to them at the beginning of the study. To remind the participants, the UGT form always had blank lines under each group of goal parameters and at the bottom as shown in Table 8.1 (page 128). After assigning goal weights, the participants were encouraged to express goals relevant to their projects beyond the ones listed in UGT.

A majority of the participants did not specify additional goal parameters. The goal parameters currently recommended in UGT could express goals in 59 out of 65 projects to the satisfaction of the practitioners in those projects. Practitioners of only 6 projects added goal parameters. Among these, 4 specified one additional goal parameter each, and 2 specified 2 additional goal parameters each. Though these goal parameters were relevant to those particular projects, there was no overlap between them and were difficult to generalise to other project contexts. We can conclude that currently UGT has a reasonable coverage and is applicable to a majority of projects in the industry.

Relevance is the number of goal parameters that could be removed from UGT without hampering coverage significantly. Table 9.2 lists the number of projects that assigned a particular weight to each goal parameter. The 30 goal parameters got a reasonably wide spread of weights within 0-5. The mean weights across goal parameters across projects was 2.85, and standard deviation was 1.44 (n = 65 x 30 = 1950). Mean weights for individual goal parameter varied from 1.32 to 4.00, standard deviation varied from 0.83 to 1.79. Out of the (65 projects x 30 goal parameters=) 1,950 cases, a weight greater than 0 is assigned to 1,757 (90%) cases and in only 193 (10%) of the 1,950 of the cases, the weight was set to 0 (irrelevant). As we can see in Table 9.2, weight 0 is not restricted to a few goal parameters, but spread across several. Further, even goal parameters weighted irrelevant by a large number of projects (goal parameter 6 and 15, for instance) have been weighted important (3 or more) by many other projects. We can conclude that the goal parameters suggested in UGT are relevant to projects.

Granularity is the extent to which UGT helps break down high-level goals into concrete goal parameters. Our aim was to achieve a good balance of granularity. We wanted the goal parameters to have sufficient granularity to allow practitioners to express themselves precisely and yet not so fine, that they cannot differentiate between adjacent elements. If the list of suggested goal parameters were too long, it could hamper the usability of UGT itself.

We could consider the granularity of UGT in two ways, the granularity of the goal parameters, and the granularity of the scale (weights 0-5). HCI practitioners' behaviours during the weight assignment gave pointers to granularity of the scale. During the study, the participants seemed comfortable with the scale. Assigning a specific meaning to each point on the scale seems to have helped. Once they got used to it, the participants could easily differentiate between the adjacent points on the scale and could justify why one goal parameter would have a weight of 2 while another would have a weight of 3.

Granularity of the goal parameters is harder to evaluate. In some cases, a few of the goal parameters could be split up further and made more granular, e.g. goal parameter 29 "user should feel emotionally engaged / brand / fun / reflective appeal / trust". On the other hand, since none of the participants felt the need for doing so for their projects though they had a choice, it is perhaps not necessary.

Our objective behind splitting high-level goals into goal parameters is to make goal setting less ambiguous and consistently interpretable. During the study, participants could interpret the goal parameters unambiguously. Occasionally, they needed to refer to the UGT documentation and go through examples for some of the goal parameters. Nevertheless, once they understood the meaning, they could unambiguously interpret the goal parameter in the context of their project, determine its importance, and think of ways to evaluate that goal parameter. We believe that we have achieved a right balance of granularity in UGT goal parameters. Of course, in case a particular project needs more granular goal parameters, UGT always allows for the flexibility.

9.3.2 Internal Consistency Reliability

In chapter 10, we propose a metric that uses goal parameter weights and scores. When variables are used as components in this manner, their internal consistency reliability is important. Hatcher explains that internal consistency reliability is the extent to which individual variables are measuring the same underlying construct of interest. Cronbach's alpha evaluates the variation accounted for by the true score of the underlying construct. Cronbach's alpha above 0.7 is considered an acceptable measure of internal reliability. To test if any variable was varying differently than the other variables, and therefore did not measure the same underlying construct, the variable is dropped and alpha is calculated again with remaining variables. If the alpha increases substantially, then the variable must be deleted to measure the underlying construct (Hatcher, 1994 pp. 131-140).

The overall internal reliability for the 30 goal parameters in the 65 projects using Cronbach's alpha was 0.7870 for the weights, and 0.8733 for the scores. Both values indicate an acceptable level of internal reliability. Each of the 30 goal parameters was deleted by turns and Cronbach's alpha was re-calculated for the remaining 29 goal parameters. If any of the goal parameter weights was varying drastically differently from others, it would have led to a big increase in the alpha and that goal parameter would be considered dispensable. The resulting alphas did not vary much, from 0.7720 to 0.7969 for weights and from 0.8629 to 0.8780 for scores (Table 9.3). In case of only a few goal parameters (highlighted in the table) the alpha increased marginally (maximum increase was 0.0099).

We can conclude that the 30 goal parameters are internally consistent and are essential to measure the same construct.

Table 9.3: Cronbach's alphas of goal parameter weights and scores for 65 projects arrived at by deleting one goal parameter out of 30 at a time.

	Goal parameter deleted (see Table 8.1 for longer descriptions)	Cronbach's alpha of remaining 29 goal parameter weights	Cronbach's alpha of remaining 29 goal parameter scores
1	Findability	0.7918	0.8694
2	Less time to learn	0.7801	0.8669
3	Learn on their own	0.7858	0.8722
4	Consistent: internally	0.7856	0.8692
5	Consistent: with other products	0.7969	0.8698
6	Consistent with earlier version	0.7832	0.8763
7	Retain infrequent tasks	0.7774	0.8725
8	Do primary tasks quickly	0.7771	0.8629
9	Quick and easy navigation	0.7810	0.8665
10	Memory / cognitive load	0.7761	0.8665
11	Flexibility / user control	0.7824	0.8735
12	Complete tasks in time	0.7771	0.8654
13	Auto-personalised	0.7783	0.8740
14	Localised	0.7750	0.8764
15	User can customise	0.7862	0.8753
16	Conceptual model	0.7749	0.8663
17	Intuitiveness	0.7742	0.8678
18	No entry barrier	0.7720	0.8674
19	No unnecessary tasks	0.7818	0.8712
20	Minimise task load	0.7787	0.8708
21	Always on	0.7724	0.8780
22	Info architecture	0.7836	0.8686
23	Clear communication	0.7834	0.8661
24	Feedback	0.7765	0.8663
25	Not induce errors	0.7883	0.8665
26	Forgiving interface	0.7850	0.8652
27	Help error recovery	0.7777	0.8649
28	Users feel in control	0.7836	0.8676
29	Emotional engagement	0.7800	0.8677
30	Aesthetic appeal	0.7884	0.8712
	Cronbach alpha for all 30 goal parameters	0.7870	0.8733

9.3.3 *Justification of Weighted Goal Parameters*

In chapter 10, we propose Usability Goals Achievement Metric (UGAM), a metric that uses weighted average of goal parameter scores for representing a metric for usability goal achievement. Intuitively, it feels right to derive a summary measure in which usability goals important in the given context are given more weightage. However, opinion is not clear in literature about whether or not one could use weighted averages in computing summary measures of usability. This analysis was done to validate our decision.

Sauro and Kirklund propose a usability metric calculated by averaging equally weighted standardised values for task time, errors, completion, and satisfaction (Sauro, et al., 2005). They claimed that giving these considerations equal weight was fine because all the pairs of the four usability scores correlated between 0.3 and 0.5, and because the principle component analysis yielded only one un-rotated component with roughly equal coefficients for all four variables (varying from 2.12 to 2.40). However, this conclusion has some validity issues.

Firstly, their analysis was based on only 4 projects, all financial domain applications, presumably targeted to similar type of users. Three of the applications were Windows based while one was web-based. Tasks, domains, users, contexts, and platforms vary a lot more and this conclusion cannot be generalised. If we consider projects with more varied situations usability scores may not correlate so much. Secondly, Sauro and Kirklund ignored several other aspects that can contribute to usability, such as learnability, flexibility, customisability, ease of use, or aesthetics that might be important in other contexts. Thirdly, it is possible that task time, errors, completion, and satisfaction are goals at too high a level of granularity for weights to matter. However, each of these could be made up of several goal parameters, as we call them in UGT, which in turn could be carrying more or less weight in different contexts.

In another study done across a large number of projects of more variety, Hornbæk and Law found medium to low correlations among efficiency, effectiveness, and satisfaction (Hornbæk, et al., 2007).

In a more recent study, Sauro and Lewis distinguished between post-task satisfaction (user satisfaction rating just after task completion) and post-test satisfaction (rating after a usability test comprising of several tasks) (Sauro, et al., 2009). From 2,286 users in 97 datasets, they reported that the correlations computed from the task time, completion and errors correlated moderately with post-task satisfaction, but the correlations dropped substantially with post-test satisfaction. This dataset was much wider than the Sauro and Kirklund study. It consisted of usability studies conducted from 1983 to 2008, and included a wider set of products such as printers, accounting and human resources software, websites, and portals. However, even in this set, most tasks were closed-end productivity oriented activities (such as create an expense report, install paper in a printer, review employee performance reports, check status of a submitted report). In this study, already, the coefficients of the first un-rotated component of the principal component analysis vary somewhat more than the earlier study (from 0.63 to 0.82), though only five variables have been used. In this study, the common factor analysis indicated two factors, what Sauro and Kirklund call an objective factor with completion rates, errors, and time loading strongly, and a subjective measure with post-test and post-task satisfactions loading strongly.

There are some pointers that un-weighted summary measures may not be the best to cover a wide base of tasks, contexts, and situations. We choose to use weighted average of goal performance score (as discussed in chapter 10). This analysis was done to validate our decision.

If projects score similarly across a majority of goal parameters, it may not be necessary to give each goal parameter a different weight. On the other hand, if projects do not score similarly across goal parameters (i.e. projects score well against some goal parameters, but not so well against other goal parameters), differential weights to goal parameters could be justified.

We analysed the goal parameter scores data from the 65 projects. Spearman's rho correlations were calculated between each pair of goal parameter scores for 65 projects. A total of (30*29/2 =) 435 pairs of correlations are possible between the 30 goal parameter scores. Of these, for 241 pairs (55.4%) the Spearman's rhos were not significant ($p > 0.05$). For the remaining 194 pairs (44.6%) the correlations were significant ($p < 0.05$), but 55 (13%) of these pairs had a Spearman's rho between -0.21 to +0.3. As many as 299 pairs (68.7%) either had an insignificant rho, or had a rho between -0.21 to +0.3. Only 136 pairs (31.3%) had a significant rho greater than 0.3, of which only 11 pairs (2.5%) had rho greater than 0.5, and only 4 pairs (0.9%) had rho greater than 0.6.

This justifies our strategy in chapter 10 of assigning weights to goal parameters in proportion to their importance while calculating UGAM.

9.3.4 Goal Parameter Achievement

In an ideal situation, design teams would like to score well on all goal parameters. In practical situations, when resources are scarce, design teams would still like to do well on high-weighted goal parameters and may not mind doing moderately or even badly on low-weighted goal parameters. Design teams would certainly not like to end up doing badly on high-weighted goal parameters.

This analysis was done to test the following hypothesis: *Design teams naturally concentrate on high-priority goals, thus goal weights and goal scores are positively correlated.* If this hypothesis were to be true, a tool like UGT would not be necessary. The hypothesis was tested in three different ways.

First, an analysis was done on mean weights and scores. Pearson's correlation coefficient (r) measures the degree of relationship between two interval variables (Kurtz, 1983 pp. 253-288). Pearson's correlations on mean weights and mean scores of the 30 goal parameters (Table 9.2 on page 138, n = 30) was calculated. A significant positive and strong correlation resulted (r = 0.957, n = 30, $p < 0.0005$). We can conclude that mean goal parameter weights and mean goal parameter scores correlate very well.

Second, an analysis was done between all weight-score pairs. Spearman's rho (r) measures the degree of relationship between two variables if one or both the variables are ordinal (Brace, et al., 2003 pp. 90-92). Both the weights and scores are ordinal variables. The 30 goal parameter weights and goal parameter scores for 65 projects resulted in 1,950 weight-score pairs. Spearman's rho between all pairs of goal parameter weights and goal parameter scores reports significant, but moderate correlation (*rho* = 0.391, n = 1,950, $p < 0.0005$).

Crosstabs summary was generated for these 1,950 weight-score pairs (Table 9.4). The rows in the crosstabs represent the weights and the columns in the crosstabs represent the scores. The number in a cell represents the number of goal parameters with the corresponding weight and score. The cells with highest percentage of cases in each row are in ***bold-italics*** letters. In an ideal case, practitioners would want to do well on most goal parameters. In that case, most bold-italics lettered cells of Table 9.4 would be in the right-most column (score = 100). This is not the case. In a practical case, practitioners would certainly want to do well on high weighted goal parameters. In that case, the bold-italics lettered cells of rows for weights 3-5 would be in the right most column of the cross-tabs (score = 100). This is not the case for either of the rows. A particular matter of concern is that goal parameters weighted 3, 4 and 5 (the high-weighted goal parameters) scored less than 50 (undecided or worse) in 39%, 36% and 33% of projects respectively (the cells marked gray in Table 9.4).

From the analysis all the 1,950 weight-score pairs, we can conclude that individual goal parameter weight-score pairs correlate only moderately and 37% of the goal parameters that are marked important perform poorly.

Table 9.4: SPSS crosstabs output for 30 goal parameter weights and goal parameter scores for 65 projects. The highest percentage cell for each row is in bold-italic letters. The gray cells are a matter of concern.

Goal Parameter Weights	Goal Parameter Scores					Total
	0	25	50	75	100	
0	***144 (75%)***	9 (5%)	19 (10%)	19 (10%)	2 (1%)	193 (100%)
1	21 (14%)	35 (24%)	***44 (30%)***	37 (25%)	10 (7%)	147 (100%)
2	17 (5%)	41 (11%)	142 (39%)	***146 (40%)***	23 (6%)	369 (100%)
3	10 (2%)	37 (7%)	157 (30%)	***254 (49%)***	60 (12%)	518 (100%)
4	11 (2%)	28 (6%)	138 (28%)	***256 (51%)***	65 (13%)	498 (100%)
5	1 (0%)	18 (8%)	57 (25%)	***97 (43%)***	52 (23%)	225 (100%)
Total	204 (11%)	168 (9%)	557 (29%)	809 (42%)	212 (11%)	1,950 (100%)

Third, the correlation of weights and scores for each goal parameter was explored. Spearman's rhos were calculated to explore the relationship between weights and scores for each goal parameter across the 65 projects. Table 9.5 lists the rhos and significances for the 30 goal parameters. Significant rhos are highlighted ($p < 0.05$). The table is sorted in the descending order of mean weights assigned across 65 projects. If there had been a real correlation between the weights and scores of a goal parameter, we expected that a sample size of n = 65 projects should have revealed it. It is interesting to note that the weight-score rhos are not significant for half the goal parameters, typically for goal parameters with high mean weights. The Pearson's correlations between the goal parameter weights and the weight-score Spearman's rho is significant and strongly negative (r = -0.815, n = 30, $p < 0.0005$). The weight-score correlation seems to increase and become significant as the mean weight drops.

Table 9.5: Goal parameter weights (mean and SD), Spearman's rho for goal parameter weight and score and its significance (n = 65 projects).

	Goal parameters (see Table 8.1 for longer descriptions)	**Mean weights (0-5)**	**Spearman's rho for weight - score**	**Significance of rho *p* < or =**
1	Findability	4.00	0.17	0.186
23	Clear communication	3.89	-0.19	0.138
22	Info architecture	3.78	-0.12	0.335
24	Feedback	3.55	-0.02	0.901
9	Quick and easy navigation	3.51	0.09	0.482
8	Do primary tasks quickly	3.49	0.34	0.006
16	Conceptual model	3.45	0.03	0.834
28	Users feel in control	3.23	0.17	0.181
12	Complete tasks in time	3.18	0.17	0.186
17	Intuitiveness	3.12	0.23	0.070
2	Less time to learn	3.06	0.22	0.074
4	Consistent: internally	3.05	0.39	0.001
10	Memory / cognitive load	3.05	0.02	0.897
25	Not induce errors	3.03	-0.04	0.725
27	Help error recovery	2.86	0.21	0.092
3	Learn on their own	2.83	0.54	0.0005
18	No entry barrier	2.80	0.28	0.021
19	No unnecessary tasks	2.78	0.07	0.558
30	Aesthetic appeal	2.78	0.33	0.007
5	Consistent: other prods.	2.75	0.38	0.002
21	Always on	2.66	0.55	0.0005
26	Forgiving interface	2.58	0.35	0.004
29	Emotional engagement	2.54	0.53	0.0005

20	Minimise task load	2.51	0.48	0.0005
11	Flexibility / user control	2.48	0.33	0.007
7	Retain infrequent tasks	2.31	0.24	0.059
13	Auto-personalised	2.23	0.71	0.0005
14	Localised	2.14	0.64	0.0005
15	User can customise	1.55	0.74	0.0005
6	Consistent with earlier version	1.32	0.77	0.0005

From the analysis of correlation of weights and scores for each goal parameter, we find that weights and scores are better correlated for low-weighted goal parameters, but not so well correlated for high-weighted goal parameters. We can conclude that design teams achieve better scores when a typically low-weighted goal parameter gets an occasional higher weight, but do not achieve similar high scores when a typically high-weighted goal gets an even higher weight.

One possible interpretation could be that the high-weighted goal parameters were *"latent"*. The HCI practitioners gave a higher weight to these goal parameters when they saw them listed in UGT during the study, but they were not particularly conscious of these goal parameters during the project. Potentially latent goal parameters with low weight-score Spearman's correlations (rho < 0.15) are the following. These 8 goal parameters have a mean weight of 3.4 and a mean score of 66.2:

- 23 - Communication should be clear
- 22 - Information architecture should be well aggregated, categorised, presented
- 25 - Product should not induce errors
- 24 - Product should give good feedback / display its current status
- 10 - Product should not load user's memory / put cognitive load
- 16 - Interface should clearly communicate the conceptual model
- 19 - Product should require no unnecessary tasks
- 9 - Users should be able to navigate quickly and easily.

By implication, the low-weighted goal parameters could be *"explicit"*. These are more readily expressed by the stakeholders (such as clients or product managers) and would be specifically addressed during the project and in acceptance tests. Potentially explicit goal parameters with high and significant weight-score Spearman's correlations (rho > 0.50, $p < 0.05$) are:

- 6 - Product should be consistent with earlier version
- 15 - A user should be able to customise the product for himself
- 13 - Product should be personalised for the user automatically
- 14 - Product should be localised for specific market segment
- 21 - Product should be always on, always accessible
- 3 - Users should be able to learn on their own
- 29 - User should feel emotionally engaged with product / brand / product should be fun / reflective appeal / trust.

These 7 goal parameters have a mean weight of 2.2, which is significantly less than the latent goals mentioned above ($p < 0.0005$). Their mean score is 44.8.

9.3.5 UGT as a Recommender System

UGT has the potential to become a system for mining prior experience and recommending usability goals based on values of input variables. The current UGT recommendations for assigning weights to goal parameters were arrived through personal experience, feedback of 15 think aloud sessions with practitioners, and manual inspections of the input parameters and the weights assigned in the projects. This analysis was done to explore the possibility of quantitatively predicting goal parameter weights based on the product and user profiles.

For this purpose the independent variables related to product and user profiles of the above 65 projects were treated as input variables. The data from the product profile was encoded as these 18 independent variables:

- If it is the **First version** of the product (0/1, 1 for 36/65 projects)
- If the product was **Paid** for (0/1, 1 for 20/65 projects), **High end** (0/1, 1 for 10/65 projects), **Low end** (0/1, 1 for 10/65 projects), **Not free but I don't pay** (0/1, 1 for 31/65 projects) or **Free to use** (0/1, 1 for 14/65 projects)

- Criticality of the product **Life critical** (0/1, 1 for 10/65 projects), **Business critical** (0/1, 1 for 21/65 projects), **Goal oriented** (0/1, 1 for 25/65 projects), **Learning product** (0/1, 1 for 13/65 projects), **Information product** (0/1, 1 for 11/65 projects), **Casual use** (0/1, 1 for 15/65 projects)
- **Criticality** was also separately encoded as an ordinal scale of 1-5, assuming life critical products = 5, business critical = 4, goal oriented / learning products = 3, information products = 2 and casual use products = 1.
- Platform of the product: **Desktop** (0/1, 1 for 20/65 projects), **Web** (0/1, 1 for 47/65 projects), **Mobile** (0/1, 1 for 8/65 projects), **IVR** (0/1, 1 for 1/65 projects), **Custom** (0/1, 1 for 3/65 projects)

The data from the user profile were encoded as these 36 independent variables:

- Age of targeted users was encoded as: **Lower bound of age** (years), **Upper bound of age** (years), **Age range** (years), and whether the target audience included **Kids** (0/1, 1 for 4/65 projects), **Youth** (0/1, 1 for 49/65 projects), **Middle aged** (0/1, 1 for 48/65 projects), **Elderly** (0/1, 1 for 16/65 projects)
- Lowest level of **tech-savvyness** of targeted users was converted into an ordinal scale of 1-5. If targeted users had lowest level of tech-savvyness equivalent to those *"who cannot save contacts on a mobile phone"*, they were assigned tech-savvyness = 1 (4/65 projects), *"can save contacts on a mobile phone*" = 2 (11/65 projects), *"checks mail every day*" = 3 (36/65 projects), *"can install applications*" = 4 (11/65 projects) and *"loves Linux"* = 5 (3/65 projects).
- Frequency of use was encoded as **Once use** (0/1, 1 for 7/65 projects), **Yearly use** (0/1, 1 for 4/65 projects), **Monthly use** (0/1, 1 for 9/65 projects), **Weekly use** (0/1, 1 for 20/65 projects), **Once a day use** (0/1, 1 for 25/65 projects), **Few times a day use** (0/1, 1 for 24/65 projects) and **Continuous use** (0/1, 1 for 11/65 projects).
- Frequency was also encoded as four ordinal scale variables: **Highest frequency** (1-7) indicating the highest targeted frequency (assuming Once = 1 and Continuously = 7), **Lowest frequency** (1-7) indicating the lowest targeted frequency from the above, **Frequency range** (1-7) indicating the range of

frequencies above targeted (highest - lowest) and **Frequency mid** (1-7) indicating the targeted median frequency.

- **Complexity** was converted into an ordinal scale of 1-5, assuming very simple = 1 (3/65 projects), simple = 2 (18/65 projects), moderate = 3 (30/65 projects), complex = 4 (10/65 projects) and very complex = 5 (4/65 projects).
- The width of the market including **Wide** market (0/1, 1 for 22/65 projects), **Niche** market (0/1, 1 for 22/65 projects), **Internal** market (0/1, 1 for 19/65 projects) and **Expert** market (0/1, 1 for 4/65 projects)
- **Market Width** was also separately encoded as an ordinal scale of 1-3 where wide market = 3, niche or expert market = 2 and internal market = 1.
- Motivation was encoded as **No motivation** (0/1, 1 for 2/65 projects), I have **options** (0/1, 1 for 10/65 projects), **Socialise** (0/1, 1 for 6/65 projects), **Cool** lifestyle (0/1, 1 for 7/65 projects), It **entertains** me (0/1, 1 for 7/65 projects), It **informs** me (0/1, 1 for 21/65 projects), It makes things **accessible** (0/1, 1 for 11/65 projects), It **saves time, makes things convenient** (0/1, 1 for 40/65 projects), It **makes / saves money** (0/1, 1 for 13/65 projects) and I **have no choice** (0/1, 1 for 20/65 projects).

Spearman's rho correlations were explored between the above (18+36 =) 54 input variables and the weights assigned by participants to the 30 goal parameters (output variables). Among the (30 goal parameters x 54 input variables =) 1,620 possible correlations, 179 were significant (p < 0.05), either positive or negative. Table 9.6 lists the significant correlations.

Table 9.6: Significant correlations (p < .05) between goal parameter weights and input parameters.

	Goal parameters	Significant positive correlations	Significant negative correlations
1	Findability	Middle aged (0.30), Age range (0.29), Upper bound age (0.27)	*Custom platform (-0.32)*
2	Less time to learn		I have no choice (-0.32)
3	Learn on their own	Market width (0.32), Paid (0.30), Learning (0.28), Youth (0.28)	Internal market (-0.34), I don't pay (-0.34), I have no choice (-0.32), Tech savvyness (-0.32), Complexity (-0.29), Lower bound age (-0.28), Few times a day use (-0.27)
4	Consistent: internally	Criticality (0.30), Makes money (0.28), Upper bound age (0.27), Niche Market (0.27), Middle aged (0.25)	Tech savvyness (-0.34), Once a day use (-0.30), Cool lifestyle (-0.28), Casual (-0.28), Free to use (-0.28)

	Goal parameters	**Significant positive correlations**	**Significant negative correlations**
5	Consistent: with other products	I don't pay (0.30), Niche Market (0.28), Frequency range (0.26)	Wide market (-0.30), Free to use (-0.29), Socialise (-0.29), Casual (-0.28), Cool lifestyle (-0.25)
6	Consistent: earlier version	Monthly use (0.35), Criticality (0.25)	First version (-0.66), Makes things accessible (-0.26)
7	Retain infrequent tasks	I don't pay (0.36), Criticality (0.35), Life critical (0.30), *Expert market (0.29)*	
8	Do primary tasks quickly	Tech savvyness (0.34), Lowest frequency (0.26), Complexity (0.26), Makes things accessible (0.24)	*Kids (-0.36)*, Once use (-0.35), Learning (-0.34)
9	Quick and easy navigation	Life critical (0.37), Complexity (0.30), Middle aged (0.30), Upper bound age (0.27), *Expert (0.25)*	Internal market (-0.29)
10	Memory / cognitive load	I have no choice (0.30)	Free to use (-0.32), Frequency range (-0.26)
11	Flexibility / user control	Life critical (0.40), Continuous use (0.28), Criticality (0.27)	Mobile (-0.25)
12	Complete tasks in time	Tech savvyness (0.35), Continuous use (0.33), Saves time, makes things convenient (0.29), Lowest frequency (0.29), Life critical (0.27), Criticality (0.26), Frequency mid (0.26), Complexity (0.25)	Learning (-0.39)
13	Auto-personalised	Mobile (0.31), I have options (0.27), Lowest frequency (0.26), Casual (0.25)	Once use (-0.35)
14	Localised	Market width (0.31), Free to use (0.29), I have options (0.29)	Internal market (-0.32), I don't pay (-0.26), First version (-0.24)
15	User can customise	Casual (0.35), Saves time, makes things convenient (0.26)	
16	Conceptual model	Informs me (0.33), Criticality (0.25), *Custom platform (0.25)*	Web (-0.32), Continuous use (-0.28), Tech savvyness (-0.26), First version (-0.26), Free to use (-0.25)
17	Intuitiveness		*Kids (-0.31)*, Weekly use (-0.25)
18	No entry barrier	Makes things accessible (0.39), Informs me (0.33), Saves time, makes things convenient (0.32), Low end (0.29), Youth (0.27)	First version (-0.29), Entertains me (-0.28)
19	No unnecessary tasks	Saves time, makes things convenient (0.30), *Expert market (0.27)*, I have no choice (0.25)	Once a day use (-0.31)
20	Minimise task load	Tech savvyness (0.27), Web (0.26), *Yearly use (0.26)*, Continuous use (0.25)	Entertains me (-0.30)
21	Always on	Youth (0.38), Low end (0.26), *Yearly use (0.25)*, Monthly use (0.25), Criticality (0.25)	Entertains me (-0.32), First version (-0.29), *Kids (-0.29)*
22	Info architecture	Makes things accessible (0.37), Low end (0.32), Mobile (0.28)	
23	Clear communication	Low end (0.31), Informs me (0.29), Mobile (0.27), Makes things accessible (0.24)	Few times a day use (-0.35), Lower bound age (-0.32), Tech savvyness (-0.25)
24	Feedback	Mobile (0.36), Informs me (0.26), Wide market (0.25)	Weekly use (-0.32), Frequency range (-0.30), Few times a day (-0.25)
25	Not induce errors	*Expert market (0.27)*, Informs me (0.25)	Tech savvyness (-0.40), Free to use (-0.34), Socialise (-0.34), Wide market (-0.31)
26	Forgiving interface		High end (-0.29)
27	Help error recovery		Few times a day use (-0.28)
28	Users feel in control	Desktop (0.29), Casual (0.29), Cool lifestyle (0.27)	Web (-0.31)

	Goal parameters	Significant positive correlations	Significant negative correlations
29	Emotional engagement	Wide market (0.48), Market width (0.46), Casual (0.44), Socialise (0.40), Cool lifestyle (0.35), Age range (0.34), Life critical (0.33), Youth (0.31), Free to use (0.25)	I have no choice (-0.48), Business critical (-0.48), Lower bound age (-0.42), Goal oriented (-0.30), Internal market (-0.29), Highest frequency (-0.28), Makes money (-0.27), Complexity (-0.25)
30	Aesthetic appeal	Wide market (0.49), Market width (0.47), Cool lifestyle (0.41), Socialise (0.38), Casual (0.37), Entertains me (0.34), Free to use (0.24)	Business critical (-0.45), Continuous use (-0.42), I have no choice (-0.38), Criticality (-0.36), I don't pay (-0.34), Lower bound age (-0.33), Tech savvyness (-0.32), Internal market (-0.30), Complexity (-0.28)

The causality in many of the correlations in Table 9.6 is obvious. For example, the weight for goal parameter 29 *"User should feel emotionally engaged / brand / fun / reflective appeal / trust"* is positively correlated with the variable *"Wide market"* (rho = 0.48, $p < 0.0005$) but negatively correlated with the variable *"I have no choice"* (rho = -0.48, $p < 0.0005$). Products targeted to the wider market must be emotionally engaging, but such an engagement may not be as important in case the users have no choice but to use the product.

However, one needs to use discretion while interpreting the causality of some of the correlations. Collinearity between input variables can lead to correlations that are difficult to explain theoretically. For example, it is not obvious why the weight of goal parameter 5 *"Product should be consistent with other products, older methods, past habits of users"* is negatively correlated with the variable *"Wide Market"* (rho = -0.30, $p = 0.02$). This could have happened because in the current sample the variable *"Wide market"* is strongly positively correlated with the variable *"Casual use products"* (rho = 0.46, $p < 0.0005$). Many projects that addressed wide markets were also meant for casual use where consistency with other products may be less important. Out of the 179 correlations, the causality of 5 correlations seems to be non-obvious to us. These have been <u>underlined</u> in Table 9.6.

Another concern is very small samples in certain variables (custom platform was used in 3 projects, there were 4 projects with kids as target audience, 4 projects were meant for yearly use, there were 4 expert market projects). Out of the 179 correlations, 11 correlations have very small samples (though they have statistical significance of $p < 0.05$). These have been *italicised* in Table 9.6.

The balance 158 correlations are largely obvious and useful for goal setting.

Incidentally, each goal parameter weight seems to be correlated significantly with at least one input variable. The following input variables correlate with 5 or more goal parameters and could be considered *"important inputs"* for goal setting. The numbers of goal parameters with significant correlations are shown in brackets:

- Tech-savvyness score of users (9)
- User motivation related variables: I have no choice (6), Leads to a cool lifestyle (5), Makes things accessible (5), Informs me (5)
- Type of market variables: Wide market (5), Internal market (5)
- Cost variables: Free to use (8), I don't pay (5),
- Criticality: Criticality ordinal variable (8), Casual use products (7), Life critical products (5)
- Frequency variables: Continuous use products (5)
- Product complexity ordinal variable (6)
- First version products (5)
- Mobile platform (5)

When the dependent variable is binary, a logistic regression results in a model that gives an estimate of the dependent variable as a function of covariates. A logistic regression allows us to derive the odds ratio of a covariate, which is the ratio of odds of success of the dependent variable per unit increase in the covariate (Walpole, et al., 2007 pp. 523-527).

To predict the goal parameter weights with the help of project and user profiles, logistic regressions were performed. For this regression, data was simplified. The goal parameter weights were re-encoded as binary variables (0 = weight from 0 to 2, 1 = weight from 3 to 5). The recoded goal parameter weights were considered as dependent variables, and the user and product profile variables that had significant rhos in Table 9.6 were considered as covariates and were entered into a logistic regression with backward stepwise elimination using the likelihood ratio test. If the model or any of its coefficients were not significant, the least significant predictor was eliminated and the remaining variables were entered again until the model and all predictors became significant. Significant models ($p < 0.05$) that gave improved predictions with significant predictor variables ($p < 0.05$) emerged for 19 of the 30 goal parameters (Table 9.7).

The odds of success of the dependent variable are enhanced by the factor of the odds ratio per unit rise in the covariate. For example, the odds that the goal parameter 3 *"users should be able to learn on their own"* would be important (i.e. have weight 3 to 5) is predictable from three user profile variables, viz. few-times-a-day use (odds ratio = 0.23), complexity (odds ratio = 0.23), and internal market (odds ratio = 0.26) (column 4 of Table 9.7). This goal parameter had a weight of 3-5 in 60.9% of cases, which can be considered the original prediction accuracy (column 5 of Table 9.7). The model improved the prediction accuracy to 73.4% (column 6 of Table 9.7).

Similarly, goal parameter 29 *"product should be emotionally engaging"* is predictable from goal oriented product (odds ratio = 0.17), business critical application (odds ratio = 0.21), and wide market product (odds ratio = 15.57). The model improved the prediction accuracy from 53.8% to 81.5%.

Table 9.7: Significant logistic regression models that gave improved predictions of goal weights with significant predictor covariates, the odds ratios of predictor covariates, the original prediction accuracy, and the improved prediction accuracy with the model.

	Goal parameters	**Model *p* <=**	**Predictor variables (odds ratio, *p*)**	**Original prediction accuracy %**	**Model prediction accuracy %**
1	Findability	--	--	95.3	--
2	Less time to learn	--	--	72.3	--
3	Learn on their own	0.0005	Internal market (0.26, 0.04), Complexity (0.44, 0.02), Few times a day use (0.23, 0.02)	60.9	73.4
4	Consistent: internally	0.024	Middle aged (3.86, 0.03)	67.2	70.3
5	Consistent: with other products	0.015	I don't pay (3.49, 0.02)	55.4	64.6
6	Consistent: earlier version	0.0005	First version (0.06, 0.0005)	73.8	75.4
7	Retain infrequent tasks	0.004	I don't pay (4.39, 0.01)	50.8	67.7
8	Do primary tasks quickly	0.004	Lowest frequency (1.70, 0.02), Learning (0.22, 0.05)	83.1	84.6
9	Quick navigation	--	--	78.1	--
10	Memory / cognitive load	0.004	I have no choice (5.92, 0.01)	60.0	61.5
11	Flexibility / user control	--	--	52.3	--

	Goal parameters	Model *p* <=	Predictor variables (odds ratio, *p*)	Original prediction accuracy %	Model prediction accuracy %
12	Complete tasks in time	0.002	Learning (0.13, 0.003), Criticality (1.66, 0.05)	72.3	78.5
13	Auto-personalised	0.009	I have options (5.13, 0.03), Casual (4.53, 0.02)	56.9	69.2
14	Localised	0.003	I have options (7.08, 0.03), Internal market (0.24, 0.04)	56.9	66.2
15	User can customise	0.011	Casual (4.75, 0.01)	67.7	72.3
16	Conceptual model	0.005	Continuous use (0.20, 0.04), First version (0.17, 0.03)	80.0	81.5
17	Intuitiveness	--	--	75.4	--
18	No entry barrier	0.001	Saves time, makes things convenient (4.14, 0.02), First version (0.22, 0.02)	64.1	76.6
19	No unnecessary tasks	0.003	Once a day use (0.29, 0.02), Saves time, makes things convenient (3.50, 0.02)	63.1	69.2
20	Minimise task load	0.026	Web (3.53, 0.03)	55.4	64.6
21	Always on	0.001	Criticality (1.83, 0.02), Youth (4.65, 0.03), Monthly use (9.58, 0.05)	50.0	73.4
22	Info architecture	--	--	93.8	--
23	Clear comm.	--	--	92.2	--
24	Feedback	--	--	84.6	--
25	Not induce errors	0.0005	Tech savvyness (0.28, 0.01), Free to use (0.15, 0.01)	69.2	76.9
26	Forgiving interface	--	--	55.4	--
27	Help error recovery	--	--	60.0	--
28	Users feel in control	--	--	81.5	--
29	Emotional engagement	0.0005	Wide market (15.57, 0.002), Goal oriented (0.17, 0.02), Business critical (0.21, 0.03)	53.8	81.5
30	Aesthetic appeal	0.0005	Market Width (2.45, 0.03), Business critical (0.15, 0.004), Continuous use (0.14, 0.04)	59.4	76.6

9.4 Summary

In this chapter, we presented evaluations of UGT. The formative evaluations were used to make UGT complete so that it can be used in a wide variety of projects. Participants in summative evaluations found UGT useful and more systematic than their current methods of setting usability goals. It helped them prioritize goals and pushed them to consider goals that they had not considered earlier. Setting evaluation guidelines helped them think through their goals more thoroughly. While both experienced and inexperienced HCI practitioners found the tool valuable, the inexperienced found it more so. While the 30 goal parameters currently recommended in UGT may not be comprehensive, they provide a substantial starting point for goal setting.

Quantitative analysis of data from 65 projects showed that goal parameter weights have granularity, and internal reliability. A large majority of participants could express their usability goals without the need of defining additional goal parameters – pointing to a reasonable coverage of UGT.

We found that only 31.3% of the goal parameter score pairs had a correlation greater than 0.3, and only 2.5% pairs had a significant correlation greater than 0.5. A large majority of goal parameter scores are not correlated. This justifies our decision to use weighted averages of goal scores for usability metric calculation, as discussed in the next chapter.

This analysis strongly re-established the need for a tool like UGT. The goal parameter weights and scores correlated only moderately. A particular matter of concern is that 37% of the high-weighted goal parameters scored 50 or less out of 100 (undecided or worse).

Teams seem to achieve better scores when a typically low-weighted goal parameter gets an occasional higher weight, but do not achieve similar high scores when a typically high-weighted goal parameter gets an even higher weight. We identified 8 goal parameters that could be interpreted as *"important but latent"*. Perhaps teams are not mindful of these parameters during the projects.

Quantitative explorations indicate that some user and product profile variables correlate with the weights assigned to of some of the goal parameters. Logistic regressions showed that these variables could be used to improve the predictions of weights. Current data can predict weights of 15 of the goal parameters with 75%+ accuracy.

In the next chapter, we present usability goals achievement metric (UGAM) to measure how well the team achieved its usability goals. In chapter 12, we establish a correlation of UGAM with process metrics.

10 Usability Goals Achievement Metric

While there are numerous usability metrics to evaluate specific products, only a few allow organizations to track progress across projects, and none that tell how well the team achieved its usability goals. We propose Usability Goal Achievement Metric (UGAM), a product metric that measures how well the design of the product achieves its usability goals. The objective is to provide a summary measure of usability goal achievement independent of the domain, context of use, platform, or the software development process, so that one can evaluate the impact of the process across projects.

10.1 Product Metrics in Human Computer Interaction

Metrics are thoroughly discussed in SE literature. Fenton and Pfleeger describe measurement as *“the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules”* (Fenton, et al., 2002 p. 5). Pressman highlights the subtle difference between measurement and metrics. Measurement occurs as the result of collection of one or more data points, while a metric tries to relate the measures in some way (Pressman, 2005 p. 466). IEEE Standard Glossary defines a metric as *“a quantitative measure of the degree to which a system, a component, or a process possesses a given attribute”* (IEEE, 1993).

Product usability metrics could be defined as a quantitative measure of the degree to which a product possess a given usability attribute. Tullis and Albert say that usability metrics, like all other metrics, are based on a reliable system of measurement. Using the same set of measurements each time, the thing measured should result in comparable outcomes (Tullis, et al., 2008 p. 7). Further, all usability metrics must be observable and quantifiable in some way.

Though the word *“metric”* is seldom used in practice of usability, several measures are often used to evaluate the user experience quality of products. Seconds taken to withdraw money from an ATM, the number of keystrokes required to enter a word in a complex script, the number of errors made while carrying out a banking transaction, or the percentage of users that abandon the shopping cart on checkout are examples of quantitative measures of the user experience afforded by a product. Tullis and Albert summarise several such metrics that can be used to evaluate products, viz. task success, time on task, errors, severity of errors, efficiency, and learnability (Tullis, et al., 2008 pp. 63-97).

These metrics and their combinations have proved to be very useful within the contexts of particular projects. It is tempting to use the same metrics or their combinations to compare across projects. However, this may not make sense because product goals may be drastically different across projects.

For example, consider an ATM that allows bank customers to withdrawal cash from their accounts. **Learnability** and **ease of use** would be important goals for this product. The designer of the ATM needs to ensure that every customer who attempts to use the ATM for the first time is successful in completing the transaction without help. Another important goal could be to **minimise the errors** (neither the bank, nor the customer would like to lose money). If the design of a product can achieve these two goals well, it will ensure that most users will use the ATM and stop queuing for withdrawals at the bank branch, thereby reducing cost per transaction for the bank. **Speed of use** would be only moderately important. It should not take inordinately long time to withdraw money from an ATM but perhaps saving a few seconds is not very important. If the first three goals are achieved well and many people start using ATMs, the bank could simply add more ATMs (as adding ATMs is cheaper than adding bank employees).

On the other hand, imagine a product with the same feature (withdraw cash from an account) targeted to a teller at the bank branch. Its goals would have to be optimised for frequent use. **Error-free use** could be still important (the bank still does not want to lose money on wrong transactions), but **speed of use** could be a close second (the teller's time is very important, the bank would like to minimise the queue, and minimise the number of tellers it needs). We could perhaps add a third goal, viz. the product should **minimise fatigue** of the user (the teller is going to do this transaction day-in and day-out for several customers). For this product, **learnability** and **ease of use** perhaps would not be so important and could be sacrificed if it is necessary to achieve the other three goals (the tellers are few, and could be always trained), but these two goals will be nice to have, provided the other three goals are met.

Imagine that these two products have been evaluated through usability tests and measures of learnability, ease of use, speed of use, error-free use, and fatigue are available. How could we make an apple-to-apple comparison across the two products? In the case of the ATM, the fatigue factor does not matter at all as the customer is expected to use the ATM only once or twice in a month. Similarly, for the teller, learnability and ease of use do not matter much if these goals were achieved at the cost of speed of use, error-free use, or fatigue factor.

To generalise, how could we make apple-to-apple comparison across projects varying in domains, platforms, and contexts? What we need is a summary measure that takes into account the achievement of usability goals of the products.

10.1.1 Prior Work on Summary Measures

While several research papers discuss metrics related to usability and HCI, here we focus only on those that give a summary measure. There are many examples of summary measures of usability.

One approach is to measure the satisfaction of the targeted users through questionnaires. Data may be collected just after users have used the product for the first time or after the users have been using a product for a while. Since the targeted users will be familiar with the context of use, the assumption could be that they would take into consideration other goals of the product while expressing their satisfaction.

Brooke suggested System Usability Scale (SUS), which presents the user with 10 statements to which users agree or disagree on a 5-point rating scale, where 1 represents strongly disagree, 3 represents neither agree nor disagree, and 5 represents strongly agree (Brooke, 1996). Five of these questions are positively worded (e.g. *"I think I would like to use this system frequently"*), while five are negatively worded (e.g. *"I found the system unnecessarily complex"*). The metric is calculated by aggregating scores (adjusted for negativity) for the ten statements and scaling the result to 100.

Other satisfaction scales have been developed, with longer questionnaires. Chin et al. proposed a Questionnaire of User Interface Satisfaction (QUIS), which consists of 27 positively worded 9-point rating scales divided into five categories: overall reaction, screen, terminology and system information, learning, and system capabilities (Chin, et al., 1988). Kirakowski and Corbett proposed Software Usability Measurement Inventory (SUMI) a questionnaire with 50 questions each on a 3-point scale, some positively worded, some negatively (Kirakowski, et al., 1993). Lewis proposed Computer System Usability Questionnaire (CSUQ) with a set of 19 positively worded questions on a 7-point scale (Lewis, 1995).

Tullis and Stetson compared satisfaction scores on two web portals by evaluating them on five scales (including SUS, QUIS and CSUQ) (Tullis, et al., 2004). One web portal was found to be doing consistently better than the other on all scales. They then explored the data to identify the scale that gave a similar result for fewest users, and found SUS, the simplest of the scales, to be the best predictor with fewest users in their study.

Lin et al. proposed the Purdue Usability Testing Questionnaire (PUTQ) that went a step further (Lin, et al., 1997). Based on eight HCI considerations (compatibility, consistency, flexibility, learnability, minimal action, minimal memory load, perceptual limitation, and user guidance), the questionnaire asks 100 questions and derives a single average score for usability. The evaluator has an option to assign a weight (0 or 1) to each question, and thus decide whether to include the question in the scale. While the approach does lead to a single usability score and gives some control to the evaluators, the selected eight considerations seem to be a mix of usability goals and heuristics that achieve those goals. Secondly, the weight for parameters is to be assigned by the evaluator during the evaluation (without consulting other stakeholders). Thirdly, the eight considerations and the questions listed under each of them seem to be limiting and do not leave room for context-specific goals that could be important (such as *"reduce fatigue"* or *"user should be able to do it right the first time"*).

It is possible to use one of the user satisfaction scores as a product metric for comparing across projects. However, user satisfaction alone may not completely reflect all the business concerns in the area of usability such as speed of use, errors, time-based performance, learnability etc. Sauro and Kirklund suggest that to increase the meaningfulness and strategic influence of usability data, analysts need to be able to represent the entire construct of usability as a single variable without sacrificing precision (Sauro, et al., 2005). Sauro and Kirklund propose a single, standardized, and summated usability metric for each task by averaging standardized values for task time, errors, completion, and satisfaction. Their calculation is based on the equal weight for each of these considerations. As discussed in section 9.3, our data suggests that there is a justification for using weighted means in usability goal scores.

Lewis suggests a rank-based system to assess competing products based on user's objective performance measured and subjective assessment by the evaluators (Lewis, 1991). The metric is useful for a relative comparison between competing like products with similar tasks but it cannot be used to compare unlike products.

McGee identifies several issues with scales, viz. they may have pre-defined ranges, they may be too sensitive or too insensitive or too dependent on specific tasks done during the evaluation (McGee, 2004). McGee derives a single usability scale based on a method called Usability Magnitude Estimation (UME) (McGee, 2004). Users make evaluations using ratio estimates on any scale that they want by comparing the given tasks to a standardised reference task. McGee reduces the data using geometric averaging to allow true ratio comparison. However, McGee does not suggest how to derive a single measure for usability from measures for the different tasks. Further, this work is completely dependent on the technique of usability evaluation. This is not always practical (in a company striving to move up the HCI maturity ladder, for example). Other limitations of this method are that it relies only on perception of users and it ignores perspectives of other stakeholders, particularly the goals of business stakeholders.

Constantine and Lockwood suggest an essential suite of metrics that measure quality of designs or prototypes (Constantine, et al., 1999 pp. 426-442). Interesting metrics among these are *essential efficiency*, a ratio of the essential (i.e. fewest) number of steps that should have been in the product and the actual number of steps required by the design, *task concordance*, an index that captures whether more frequent tasks are easier to do than less frequent tasks, and *task visibility*, that measures if the interfaces show users exactly what they need to know or do. While these metrics are meant to be used with a specific product on an absolute scale, they could potentially also be used to compare these attributes across products. However, these still are only specific attributes, which may not be equally important in all project contexts.

The wider notion of user experience (as opposed to usability) is attracting the attention of the academia and the industry. Usability parameters are typically related to the processing of information or completion of tasks. However, affective reactions and emotional consequences play an important role in the overall user experience (Mahlke, 2005). In some product contexts, visceral, behavioural, and reflective elements (Norman, 2004), aesthetics (Tractinsky, et al., 2000), enjoyment (Jordan, 2000), and creativity (Swallow, et al., 2005) may need to be considered. As discussed earlier, McCarthy and Wright say that people actively construct their experiences and that each person's experience is unique, rich, and difficult to communicate, let alone predict, design, or measure (McCarthy, et al., 2004 pp. 12-21). Researchers are struggling to arrive at even a definition for user experience. In a recent (2009) survey of 275 respondents from the user experience profession, Law et al. concluded that the concept of user experience was dynamic, context-dependent and subjective and they found that it is not easy to understand, define, or scope it (Law, et al., 2009). If measuring usability is hard, it is harder still to characterise user experience in terms of metrics. We have not come across any summary measures of user experience.

10.1.2 Critiques on Usability Metrics

Several authors have commented on and cautioned against use of metrics in usability, particularly summary measures.

Constantine and Lockwood argue that no simple number can completely represent subtle and complex issues of usability of a software system and a quick look at the numbers can never substitute a careful study of the interface (Constantine, et al., 1999 p. 419). Numbers can sometimes create *"an illusion of understanding"* and metrics may be given undue importance when balanced against *"sound judgements and well-informed opinions"*. However, they do feel that usability metrics have several uses but these should augment and support other forms of design review and evaluation, not supplant them.

Hornbæk and Law argue against the validity of a single, summative measure because it either relies solely on users' perceptions or is arbitrary in including or excluding constituent usability parameters from its summation procedure (Hornbæk, et al., 2007). They argued that attempts to reduce usability to one measure lose important information.

Gulliksen et al. are critical of measurement in the area of usability and user experience (Gulliksen, et al., 2008). They are concerned that given the difficulties in measurement, people may measure only those aspects that are easy to measure, and these measures may turn out to be *"eternal truths"*. They find that organisations had trouble in interpreting metrics, drawing conclusions, and turning them into action items. They reflect Lockwood and Constantine's concern that once something is numerically assessed everyone may focus on the measurement alone and ignore the complexity behind the work situation. Gulliksen et al. are not totally opposed to measurement per se, but are merely pointing to limitations that the measurement induces.

The main thrust of all critique against summary measures of usability seems to be that they have limitations, and should not be seen as a substitute of qualitative judgements in design situations. We acknowledge these limitations. Yet we believe that a total rejection of summary measures would not be appropriate either. If interpreted within the scope of what they are supposed to measure, summary measures of usability can enable comparison across projects. UGAM, which we present in section 10.2, is designed to measure the achievement of usability goals expressed by the design team and the stakeholders.

10.1.3 Shortcomings of Current Metrics

Though some of the metrics discussed above could be used as a product metric to compare across products, none of them is entirely suitable. UGAM, which we present in section 10.2, addresses these limitations.

Firstly, many of the metrics rely only on a *pre-identified* set of questions that may not be equally applicable in all situations. For example, in a usability test that the author witnessed, a user was rating the statement of SUS, *"I think I would like to use this system frequently"* in the context of a configuration interface for a home-router. The user remarked, *"Hopefully, not"*. In the context of this product, a low-score to this statement is not necessarily bad, because the product is not supposed to be used frequently and this question is not relevant. Some questionnaires take care of this issue to an extent by adding an option *"not applicable"*. While this allows the users to skip questions, it gives little control to the stakeholders and evaluators. PUTQ goes a step ahead by allowing evaluators to decide if a question is relevant or not.

Second issue is *control*. Metrics discussed above do not allow stakeholders to specify whether a goal is relatively more important than another in their context. For example, while speed of use is relevant in the design of an ATM, ease of use and error-free use are a lot more important. In an application for a call centre agents, efficiency and effectiveness may be important, but the product should also be pleasant to use. For a computer game, the player's satisfaction may be of utmost importance, and if the game manages to make gamers happy (even) at the cost of efficiency or effectiveness, the game ought to be considered successful. If efficiency, effectiveness, and satisfactions measures for all these products are flatly aggregated (as Sauro and Kirklund suggest above), each successful application might show up as mediocre one. As Hornbæk and Law found, their scores may not correlate, but they need not. The weights in UGAM give a finer control to HCI practitioners and stakeholders to express what is important.

Thirdly, no metric allows stakeholders to *add goals* especially relevant to their product. For example, the product manager of a home router wanted to ensure that their product had a smooth *"out of the box experience"*. The HCI practitioner of a device to be used in mining applications wanted his screen to be readable *"both in blinding sunlight and pitch darkness"*. A large telecom company wanted to ensure that its employees *"did things right, the first time"*. A requirement for a call centre application was to allow the user to *"complete successfully a large number of calls with minimal fatigue"*. In UGAM, we allow HCI practitioners to add, edit, delete, or merge goal parameters to suit their needs.

A related issue is *obsolescence*. Many metrics give too much importance to the usability concerns that were important at the time they were proposed. Statements such as *"reading characters on the screen"* or *"I sometimes wonder if I am using the right command"* were perhaps more critical in the days of command prompts, green screens and limited typefaces. On the other hand, contemporary usability issues arising out of social interactions on the web, ubiquity of the mobile phone or multi-modal interactions (touch, gesture, voice) are missing in these scales. By staying flexible and incorporating new goal parameters from time to time, UGAM can avoid obsolescence.

None of the summary metrics mentioned above attempt to measure the experience of a product explicitly with reference to its goals. Many are too complex to compute practically on an on-going basis in the industrial practice. Most lack the flexibility required to serve the needs of a wide variety of projects or to mature with time.

10.1.4 Need for a Goal-based Technique-neutral Metric

Firstly, there is a need for a metric that takes into account *the usability goals*. The purpose of a process model is to help the development team achieve goals. Our interest is not to measure the absolute notion of usability or user experience, but to measure the extent to which a process helps the team achieve the usability and user experience goals. While several authors have argued for a goal-based approach in HCI design, there seems to be almost no work on a goal-based approach in measuring the usability of the product.

A metric based on usability goals will have several advantages. Fenton and Pfleeger emphasise the importance of goals in a metric: *"a measurement program can be more successful if it is designed with the goals of the project in mind"* (Fenton, et al., 2002 p. 84). As we discussed in chapter 8, goals are an important way for the stakeholders to express the desired user experience. Depending on the goals of the project, HCI practitioners make different design decisions to optimise resources, sometimes sacrificing one attribute of usability to achieve another, more important one. Goals are (and should be) available early in a project, a plus when it comes to metric calculation in a practical situation. Moreover, goals factor in the variability in the project, and enable comparison across projects.

Another benefit of a goal-based, flexible approach is that it can go beyond the notion of usability and into other areas. While it may be difficult to define user experience universally, within the specific context of a project, it is usually possible for a design team to agree about the issues that would affect users' experience the most. Additional overlapping goals, such as business goals or safety goals could also be integrated and measured similarly.

Further, there is a need for a *technique-neutral* product metric. A common issue with metrics discussed above is that they can be evaluated *"only with users"*. While this is desirable, it may not be possible in a company striving to move up the HCI maturity ladder. In such companies, frequent usability evaluations with real users in may not be practical. Many project teams in Indian IT companies would consider themselves lucky to get an opportunity to evaluate their design with real users once in the lifecycle. Discount usability methods such as heuristic evaluation (Nielsen, 1993 pp. 155-163) are more practical in such contexts. While it may not be an ideal situation, it is a necessary step before the company achieves higher levels of usability maturity, and it is the phase where the usability practitioners could use a lot of support from metrics. The metric needs to be *technique-neutral*, i.e. it should be computable either from review-based usability evaluation technique or from user-based usability evaluation technique.

A flexible, goal-based, technique-neutral approach to product metrics can help measure usability issues that are important in the context of a project and at the same time allow comparison across projects. We present our proposal for the same in the next section.

10.2 Usability Goals Achievement Metric

We propose Usability Goals Achievement Metric (UGAM), a product metric that measures the quality of user experience. UGAM is product metric on a scale of 0-100, where 100 represents the best user experience possible and 0 represents the worst. The motivations are:

- to measure the user experience of a product in reference to its usability goals

- to develop a flexible metric that can be applied across a variety of projects, irrespective of domain, context, platform, process model, and usability technique
- to develop a flexible metric that will mature with the organization
- to compute the metric with minimal additional costs and efforts.

UGAM consists of the following conceptual elements:

Goals and goal parameters: These are similar to the ones discussed in UGT in the last chapter. High-level usability goals are broken down into a set of goal parameters at a level of granularity that make them easy to evaluate. A list is suggested in Table 8.1 above, but the HCI practitioners and the stakeholders have the freedom to derive additional, relevant parameters that express their usability goals better. In theory, these could be drastically different from the ones we suggested. Goal parameters could be added, removed, or combined according to the context of the project, the needs of the users, the vision of the stakeholders, and / or to fit the terminology that makes the product development team comfortable. The initial list is meant to give users a starting point, while the flexibility is meant to allow the metric to mature with the experience of the organizations using UGAM.

Weight: As described in UGT, each goal parameter is assigned a weight between 0-5 where 0 indicates that the goal is not relevant, 1 indicates that the activity is somewhat relevant, 2 indicates the typical importance (the hygiene factor), 3 indicates the goal parameter is more important than usual, 4 indicates that it is very important and 5 represents that it is extremely important. UGT also provides guidance to set weights based on previous experience.

Score: To compute UGAM, each goal parameter is assigned a score between 0-100, where 0 represents the worst possible user experience design on account of that parameter, 25 represents that the design is quite bad, though not the worst, 50 represents an undecided state, 75 represents that the design was good enough, though not exceptional and 100 represents the best possible user experience design rated against the goal parameter.

A parameter may be assigned a score by one of the following ways:

- By linking it directly to user performance in a usability test or in real life. E.g. assign score 100 for *complete tasks in specific time* (goal parameter 12) if the average transaction time from 'card inserted to cash and card withdrawn' in the field is reported to be less than 45 seconds; assign score 75 if it is 45 to 90 seconds; 50 if it is 90 to 150 seconds; 25 if it is 150 to 240 seconds; and 0 if it is greater than 240 seconds. Assign score 100 for *findability* (goal parameter 1) if all users can find 95% of the critical options; assign score 75 if all users can find 85% of the critical options; assign score 50 if all users can find only 75% of all critical options; and so on.
- By using evaluators' assessment from a qualitative usability evaluation. E.g. after a think-aloud test, evaluate how confused the users were about the *conceptual model* (parameter 16). If most users were confused most of the time, assign score 0; if most users were confused at one time or other, assign score 25; if some users were confused at some times, assign score 50; if only very few users were confused very few times, assign score 75; if no user ever had any confusion, assign score 100.
- By reviewers' rating after a heuristic evaluation of the product.

Guidelines: Goal setters should also set guidelines to help the evaluators interpret goals consistently in the context of the project. Scores of some goal parameters can be directly linked to the findings of the usability evaluations while scores of other parameters may need to be assigned by the evaluators or reviewers. Guidelines tell the evaluators how to assign scores or how to link scores to specific performance measures. Even in the case where the goal setters themselves will be evaluators, it is a good idea to set guidelines because there could be a large time gap between goal setting and the actual evaluation.

The process for computing UGAM for a product has these steps:

Goal Setting: Early in the project, typically just after user studies but before design, an HCI practitioner and stakeholders identify goals and parameters for each goal, assign weight to goal parameters, and decide evaluation guidelines.

Scoring: After design, and immediately after each usability evaluation, one or more independent evaluators assign a score to each goal parameter. To begin with, multiple evaluators should assign scores independently. If there is a large difference in their scores for a goal parameter (> 50), the evaluators should discuss the parameter. They have the opportunity to converge their scores before the average is calculated.

UGAM Calculation: UGAM is the sum of the weighted average of the scores of all goals

$$UGAM = \frac{\sum(W_p \times S_p)}{\sum W_p}$$

where W_p is the weight of each goal parameter and S_p is the score of that parameter. In case multiple evaluators are involved, a simple average across evaluators is deemed the score.

Example

Table 10.1 shows an example UGAM calculation for an industrial project. The team first allocated weights to each goal parameter (shown in column 2). The evaluators then evaluated the product and assigned a score for each goal parameter (shown in column 3). The weighted average for the goal parameter scores under each goal resulted in the score of the higher-level goal (shown in column 4) and the weighted average of all goal parameter scores resulted in the UGAM score (shown in the last row of column 4). In this example, the team could see that by focussing on the high-weighted but low-performing goals (e.g. ease of use) and goal parameters (e.g. users should be able to learn on their own), they could improve their UGAM score substantially.

Table 10.1: An example UGAM calculation.

Goals and Goal Parameters	Weight	Score	Goal Score
Learnability			**69.44**
Findability: options / data / information should be visible / easy to find	4	75	
User should take less time to learn: (e.g. < 2 hours of practice)	3	50	
Users should be able to learn on their own	4	50	
Product should be internally consistent	5	100	
Product should be consistent with other products, older methods / past habits of users	2	50	
Product should be consistent with the earlier version	0	-	
User should remember / retain critical, but infrequent tasks	0	-	
Speed of use			**65.00**
User must be able to do the primary task / the most frequent tasks quickly, easily, at all times	0	-	
User should be able to navigate quickly and easily	4	75	
Product should not load user’s memory / product should not put cognitive load on a user	2	75	
Flexibility: user should control the sequence of tasks	1	50	
User should be able to complete frequent / critical tasks in specific time / no. of steps / in less efforts	0	-	
Product should be personalised for the user automatically	0	-	
Product should be localised for specific market segments	3	50	
A user should be able to customise the product for himself	0	-	
Ease of use			**50.00**
Interface should clearly communicate the conceptual model	2	25	
Intuitiveness: User should be able to predict the next step / task	3	50	
No entry barrier: user must be able to complete critical tasks	0	-	
Product should require no unnecessary tasks	2	75	
Product should automate routine tasks / minimise user task load	0	-	
Product should be always on, always accessible	0	-	
Ease of communication			**75.00**
Information architecture: Information should be well aggregated, well categorised, well presented	3	75	
Communication should be clear / user should easily understand text, visuals	4	75	
Error-free use			**63.89**
Product should give good feedback / display its current status	3	75	
Product should not induce errors	3	75	
Product should tolerate user’s errors / forgiving interface / should prevent errors	1	25	
Product should help user recover from errors / help users troubleshoot problems	2	50	
Subjective satisfaction			**75.00**
User should feel in control of the product / behavioural appeal	3	75	
User should feel emotionally engaged with product / brand / fun / reflective appeal	1	0	
User should find the product aesthetically appealing / product should have a visceral appeal	3	100	
UGAM score			**66.81**

UGAM is a summary measure, but it also creates a usability profile and allows drill-down to goal level and goal parameter level. Calculation of UGAM could be a part of every usability evaluation of the project. The UGAM scores of formative evaluations will help track the progress of the project. UGAM score from the final summative evaluation will establish whether the product meets quality criteria. UGAM can be used to track the usability of a product across versions. Managers could use UGAM to track the performance of several projects at a time and to identify the poorly performing projects and plan to assign their best people to those projects.

In case the product is meant to be used by multiple user profiles, separate goals should be set and separate UGAM should be calculated for each profile (as users belonging to different profiles are likely to have different experiences with the same product).

10.3 Comparing Usability Goals Achievement Metric to Traditional Usability Studies

Would UGAM scores agree with ratings by traditional usability evaluators? In some cases, UGAM may be calculated in non-ideal situations, for example after a review of products rather than a usability evaluation, or by people with limited experience in usability. How would UGAM scores perform in these situations? We conducted an experiment with the help of students to compare UGAM scores with the ratings of an evaluator after a traditional usability evaluation.

10.3.1 The Products

In this experiment, we evaluated nine products from three diverse product categories. Product categories included three SMS input methods for Hindi, three CD-writing applications and three cricket websites.

The three SMS input methods for Hindi were based on currently available phones of Nokia, Sony Ericson, and a new system. The targeted audience was Hindi-speaking users who had studied between standards 7th to 10th of schooling, who had been using a mobile phone from 6 months to 2 years and who had never learnt to input Hindi text in any system. The evaluators were asked the following key questions: Which system is the easiest for the beginner? Which encourages self-learning? Which is the best after practice? Overall, which system is the best? Are there any specific problems and suggestions for improvement of in each design?

The three CD-writing products that were evaluated were Roxio, Nero, and the Windows Vista native CD-writing application. The chosen user segment consisted of middle-aged, medium tech-savvy users who wished to take regular backups of their work without any help.

The three cricket websites were cricinfo.com, cricketnext.com, and cricbuzz.com. The chosen user segment consisted of office-going, medium tech-savvy users who are cricket fans, who play cricket and know the game and the cricket terminology well, who always follow matches of their country and occasionally follow matches between other countries, who have access to the internet, use it for occasional searches, but had never visited the three sites in question.

10.3.2 Method

The experiment was carried out with participation of two student teams working independently. One team comprised of 9 students doing their masters in interaction design, who were attending a course on usability evaluation (**Team 1**). The other team comprised of 34 computer science and engineering students, a mix of masters and fourth year undergraduate students, who were attending an introductory course on human-computer interaction (**Team 2**). A persona description of targeted users and a scenario were provided to the evaluators in each case.

Team 1 was the control group and represented the traditional usability evaluation techniques. These students had an aptitude for interaction design and had previously attended several courses related to design, including courses on interaction design and user studies. This exercise was done during a course on usability evaluation. This team was divided into three groups. Each group evaluated one product from each category using three different usability evaluation techniques as a part of the course.

The Hindi SMS input products were evaluated by Team 1 using a **performance test**. This was the most rigorous of the three tests. Each group recruited five users. Each user was given a five-minute orientation of the phone model. Then the user was shown 18 cards one by one. Each card had one Hindi word written on it. The words were sequenced in the increasing order of difficulty. The user was asked to input the words on the phone. The user was encouraged to type each word without help. If the user could figure out how to type the word, the task was considered *"successful without help"*. If the user could not type the word on his own, help was provided. If the user could type the word with help, the task was considered *"successful with help"*. If the user gave up or could not type the word in spite of help, it was counted as an unsuccessful task.

The three CD-writing applications were evaluated by Team 1 with the help of a discount usability evaluation method of **heuristic evaluation**. Each product was reviewed by a sub-group of three evaluators. Each person in the group evaluated the product independently and listed out usability problems. The three evaluators then met for a debriefing session and combined their lists. They also came up with suggestions to improve the design.

The three cricket websites were evaluated by Team 1 by the technique of **think-aloud test**. Each sub-group recruited four to six users in a wide age range. Each user was given a set of tasks and was asked to think aloud while trying to perform the task. Tasks included finding the latest score of an ongoing cricket match, reviewing the summary of a match that got over recently, finding the next match that you can see with friends at a nearby stadium or on TV, finding the rules of the (then new) league cricket match series and finding and comparing statistical data of two cricket players. As each user performed the tasks, their problems, confusions, frustrations, and comments were noted.

Each evaluation was spread over six days (four working days and two days of a weekend for buffer time). Though each product was evaluated by a sub-group, the sub-groups worked together, shared their work often, agreed on common goals within a category, and had a common test protocol and a common test setup. The three sub-groups recruited users together and presented findings to each other on the last day of each evaluation. After the presentations, all the students were asked to rate each product on a scale of 0 to 10 based on the findings, where 0 represented the worst user experience and 10 represented the best.

Team 2 was the test group and only used UGAM for evaluation. Team 2 students came from a computer science discipline and they were only half-way through their first course on HCI. They were taught the technique of calculating the UGAM in a 1.5 hour-long session and did the actual UGAM evaluation in the next session, also lasting about 1.5 hours.

This team computed UGAM for the same nine products described above and against the same briefs. The 34 students were divided into nine groups. Each group was asked to evaluate one product. Each group worked independently, without sharing any material such as goals, parameters, weights, or scores with other groups. Team 2 had much shorter time to complete the assignment and could carry out only a brief inspection of the product. In all respects viz. background in interaction design, knowledge of usability evaluation techniques, time available for evaluation, access to users, interaction with other groups, and sharing findings about other product evaluations Team 2 was at a disadvantage compared to Team 1. Of course, they had UGAM to help them.

10.3.3 Findings

Table 10.2 to Table 10.4 summarize the ratings by Team 1 and UGAM scores by Team 2. Table 10.2 also lists the findings from the performance test, namely the task completion rates of users without and with help.

To determine the relationship between the ratings by Team 1 and UGAM scores by Team 2 for the 9 products, two-tailed Pearson's correlation was performed. It was found that there is a significant positive correlation r = 0.68, $p = 0.04$ between ratings by Team 1 (mean = 5.44, SD = 1.55, n = 9) and UGAM by Team 2 (mean = 65.71, SD = 11.12, n = 9). With the help of UGAM, Team 2 could predict 46% of the variation in the ratings by Team 1 ($r^2 = 0.46$).

Table 10.2: Performance test findings, ratings by Team 1 and UGAM scores by Team 2 for Hindi text input on mobile phones

	New System	**Nokia**	**Sony Ericson**
Findings by Team 1 (average number of words typed by a user, number of users = 5, total words attempted = 18)			
Without help	9.6 (53%)	8.0 (44%)	3.4 (19%)
With help	8.0 (44%)	2.4 (13%)	4.2 (23%)
Total	17.6 (98%)	10.4 (58%)	7.6 (42%)
Rank	1	2	3
Ratings by Team 1 (control group, scale 0-10)			
Average rating	7.6	6.5	3.7
SD	0.8	0.8	0.8
Rank	1	2	3
UGAM by Team 2 (test group, scale 0-100)			
UGAM score	69.5	67.5	48.9
Rank	1	2	3

Table 10.3: Heuristic evaluation ratings by Team 1 and UGAM scores by Team 2 for CD-writing applications.

	Roxio	**Nero**	**Vista**
Ratings by Team 1 (control group, scale 0-10)			
Average rating	6.5	4.8	3.6
SD	0.9	1.2	1.1
Rank	1	2	3
UGAM by Team 2 (test group, scale 0-100)			
UGAM score	80.5	75.8	60.2
Rank	1	2	3

Table 10.4: Ratings by Team 1 after think aloud tests and UGAM scores by Team 2 for cricket websites

	cricinfo.com	**cricketnext.com**	**cricbuzz.com**
Ratings by Team 1 (control group, scale 0-10)			
Average rating	6.6	6.1	3.6
SD	2.4	0.6	0.5
Rank	1	2	3
UGAM by Team 2 (test group, scale 0-100)			
UGAM score	63.1	75.6	50.5
Rank	2	1	3

Within product categories, the ranking by Team 1 ratings and by UGAM scores tallied in the categories for Hindi text input (performance test) and CD-writing software (heuristic evaluation). In the case of the Hindi text input product category (performance test), UGAM scores also tallied with the ranking by the results of the performance test. On the other hand, for the cricket web sites (think-aloud protocol), the bottom ranked product tallied, but the first two sites were exchanged. This is perhaps because the first two product categories are goal-driven and task-oriented and the goals in Table 8.1 were expressive enough. On the other hand, cricket websites are information spaces to be explored at leisure, user goals, tasks are not so clear, and evaluations may tend to be more subjective. Perhaps the difference can also be attributed to the different techniques used by the Team 1. Even so, the top two sites (cricinfo.com and cricketnext.com) got close scores in both the ratings and UGAM. Further, the standard deviation of the rating for cricinfo.com was unusually high, pointing to disagreement within the Team 1 rating.

It can be concluded that despite constraints (time available for evaluation, access to users, background in interaction design, knowledge of usability evaluation techniques, interaction with other groups, and sharing findings about other product evaluations), the team using UGAM could reasonably mimic ratings by traditional usability evaluators.

A limitation of this experiment was that it was done by university students. Evaluations by professionals could be different. Further, the experiment involved evaluation of only 9 products (though the findings were statistically significant). It will be interesting to follow up this study with professional assessments that are more detailed.

10.4 Summary

In this chapter, we presented UGAM, a summary metric to measure achievement of usability and user experience goals. UGAM is goal-based, technique-neutral, flexible metric. UGAM gives control to the stakeholders by allowing them to define the priority of usability goal parameters by assigning weights and defining evaluation guidelines. UGAM is calculated by weighted average of goal parameter scores. Though UGAM is a summary metric, it also creates a usability profile of the product that allows drill-down to goal and goal parameter levels.

In an experiment, we evaluated 9 products with the help of 2 teams. Team 1 sub-groups worked together, for longer time, and used traditional usability techniques to rate the products. Team 2 sub-groups worked separately, in a shorter time, and used only a review and UGAM. The final ratings of both groups across categories correlate significantly.

In the next chapter, we present Index of Integration (IoI) a process metric to measure how closely a team followed a prescribed process model during a project.

11 Index of Integration

11.1 Process Metrics

While there have been several proposals to define product metrics of usability, there seem to be no metrics to measure the quality of HCI design process followed or the integration of HCI activities with SE processes.

There is one method though, to evaluate the HCI maturity of an organisation. The Usability Maturity Model (UMM) presents an ordered scale that helps categorise an organisation in six levels of maturity with respect to human-centeredness: incomplete, performed, managed, established, predictable, and optimising (Earthy, 1999). Each level is recognised by certain process attributes each of which is recognised by a few practices. For example, the *"managed processes"* level is measured by *"performance management"* and *"work product management"* attributes. The work product management attribute is further divided into four practices: *"identify requirements"*, *"identify the activities needed"*, *"manage the configuration of the work products"*, and *"manage the quality of the work products"*. During an evaluation, the organisation is measured against each practice on a 4-point scale: no evidence of achievement of the defined practice, some achievement of the defined practice, significant achievement of the defined practice, and full achievement of the defined practice. The scores are summated over practices and attributes to determine the usability maturity of the organisation. While UMM could be useful to evaluate the maturity of the organisation, it may not be useful to evaluate the compliance with a particular process model in a particular project.

Software engineering has a strong tradition of using historical data to help software engineers understand and improve their performance. Examples of such data-driven procedures, process frameworks, or process development approaches are the Personal Software Process (Humphrey, 1995), Team Software Process (Humphrey, 1999) and the Capability Maturity Model Integration (CMMI) (Software Engineering Institute, 2006).

However, even in the field of SE, most process metrics are indirect measures. Pressman says *"we measure the efficacy of the process indirectly"*, mainly from metrics based on the outcomes that can be derived from the process (Pressman, 2005 p. 651). Such outcomes could be measures of errors uncovered before release of the software, defects delivered to and reported by end-users, work products delivered, human effort expended, calendar time expended, schedule conformance, and effort and time spent performing generic software engineering activities. Defect removal efficiency (DRE) is a measure of particular interest, defined as

$$DRE = \frac{E}{E + D}$$

where E represents the number of errors found in the software before it was delivered and D represents the number of defects found after delivery.

Software Engineering Institute measures process performance in a similar way of the actual process results achieved characterized by both process measures (effort, cycle time, and defect removal efficiency) and product measures (reliability, defect density, and response time) (Software Engineering Institute, 2006).

Such indirect measures are useful if we can assume that the process will necessarily lead to the desired outcomes. When the indirect measure is low, we can assume that process was not followed. When the indirect measure is high, one can assume that the process was followed. On the other hand, if we need to establish that following the process leads to the desired outcomes, we need to evaluate process compliance more directly. We propose one such direct measure in the next section.

11.2 Index of Integration

We propose Index of Integration (IoI) as an empirical process metric, nominally on a scale of 0-100, where 100 represents the best possible integration of HCI activities in the software development activities as prescribed in the process model used as the base, and 0 represents the worst possible integration.

The IoI metric consists of the following conceptual elements:

Software Engineering Phases: These are the broad phases as described in a software engineering process model such as waterfall (communication, planning, modelling, construction and deployment) and RUP (inception, elaboration, construction and transition). In case of agile process models, where *"a minute of software development looks like a decade of software development"* (Nelson, 2002), there would still be some work being done before the software development iterations begin, some during, and some after they end.

HCI Activities: HCI activities for each phase are prescribed in the process model assumed as the base. These could be organisation-specific or based on a published recommendation. To demonstrate IoI, we use the recommendations that we prescribed as extensions for the waterfall (page 96) and agile (page 107) process models.

Weight: The weights have a similar meaning as in UGAM. Each HCI activity will have a given weight on the scale of 0-5 where 0 indicates that the activity is not relevant, 1 indicates the activity is somewhat relevant, 2 indicates that the activity is typically important, 3 indicates the activity is more important than usual, 4 indicates that the activity is very important and 5 indicates that the activity is extremely important. The weights could be assigned by three entities.

- While prescribing a process model, the **process prescriber** should recommend weights or weight ranges for the prescribed HCI activities. The prescriber may also give suggestions for when to give a higher weight, and when to give a lower weight than prescribed. In the next section, we prescribe weights for our process proposals.
- While adopting a process model, an **organisation** may review weights recommended by the process prescriber and adapt them to its needs.
- The same could be done by a **development team** adopting a process for a specific project (though, in this case it can no longer be a range, but just one number for each activity). For example, in a project, if the domain or users are new to the team, it may be very important to do user studies (weight = 5). On the

other hand, if the team is already very familiar with the context and the domain and if they have a lot of experience designing similar products, user studies may not be of high importance for that project (weight = 2).

All three should review the weights from time to time to accommodate their experiences.

Score: During evaluation, each activity will be assigned a score. The score is given on a rating of 0-100, where 100 represents the best case situation i.e. the activity was done in the best possible manner, with the highest fidelity, in the most appropriate phase of software development and with the best possible deliverables; 75 represents that the activity was somewhat toned down, but was still well-timed and well-executed; 50 represents that the activity was done with some shortcuts or perhaps was not timed well; 25 represents that the activity was done with many shortcomings; and 0 represents the worst case situation where the activity was not done at all.

Activity evaluation guidelines: These spell out considerations that help the evaluation of each activity. Guidelines may define the techniques used to carry out activities, the skill and experience levels of the people doing the activities, the deliverables, and other parameters that affect the fidelity of the activity. An example is shown below.

Example

Following are the guidelines for the HCI activity of *"contextual user studies and user modelling, competitive product analysis"* in Table 11.1 below:

1. Both organizational data gathering and user studies were done before requirements were finalized
2. User studies were done in the context of the users by the method of contextual inquiry
3. User studies were done with at least 20 users in each profile
4. User studies were done by people with experience in user studies in a similar domain of at least 2 projects
5. The findings including user problems, goals, opportunities and constraints were analyzed, documented and presented in an established user modelling methodology such as personas, work models, affinity diagram or similar

6. Competitive / similar products and earlier versions of the products were evaluated for potential usability problems by at least using discount usability evaluation methods such as heuristic evaluation and were benchmarked
7. User experience goals were explicitly agreed upon before finalizing requirements

100 = All the above are true, the activity was performed exceptionally well

75 = At least five of the above are true, including point 7, or all the above are true, but point 3 had fewer than 20 users per profile, the activity was performed reasonably well

50 = At least three of the above are true, including point 7, the activity was done with some shortcuts and / or perhaps was not timed well

25 = Only two of the above are true, the activity was done poorly with many shortcomings

0 = None of the above are true, the activity was not done.

Appendix IV gives detailed guidelines for evaluating all HCI activities listed in Table 11.1 and Table 11.2. The same is also available on the IoI website (Joshi, 2009).

The process for computing IoI for a project has the following steps:

Company HCI Process Prescription: The HCI group adopts an existing process or prescribes a process of its own. The prescription should include what HCI activities should be done in which phase of the SE process, the deliverables expected from each activity, suggested weights for each activity, and suggested activity evaluation guidelines. As it often happens, an organisation may follow not one SE process, but several. In that case, the HCI activities need to be integrated with each SE process. They also suggest a weight for each HCI activity and guidelines to score each activity.

Project HCI Process Definition: After getting a project brief and after understanding the domain, the users and the project context, the HCI practitioner fine-tunes the weights for the prescribed HCI activities. He / she should consult his teammates and stakeholders before finalizing the weights. This step should happen in the beginning of the project. The weights should be reviewed every time activities are reviewed or when plans change.

Process Evaluation: After the project has been completed (or in case of highly iterative processes, when a major release has been done), independent UX practitioners review the HCI activities, evaluate them for process compliance and assign a score for each activity. In case of multiple evaluators, an average across evaluators is deemed the activity score. In case there is a lot of divergence in scores of a particular HCI activity (> 50), the activity is discussed and reviewers are given a chance to change their score before an average is taken.

IoI Calculation: IoI is found by computing the weighted average of the scores of all activities:

$$IoI = \frac{\sum(W_a \times \mathrm{S_a})}{\sum \mathrm{W_a}}$$

where W_a is the weight for a particular HCI activity and S_a is the score for that activity.

11.2.1 Recommended Weights for HCI Activities in the Waterfall Model

In this section, we recommend weights for the HCI activities that we prescribed for the extended waterfall model (section 7.2, page 96). Table 11.1 lists our weight recommendations. Below, we give our rationale for these weights. We derived these by reviewing literature discussed in chapter 2, by using our personal experience, and with inputs from managers of HCI groups in two IT companies. These weight recommendations are applicable in a majority of project contexts. However, they may vary in specific cases (some examples of which we point out below). We present a statistical validation of these recommendations in chapter 12.

In the beginning of a project, it is very important to understand the context of the user and the market scenario. Hence, the activities related to **user studies** and competitive product analysis is recommended a weight of 3 to 4. The weight can increase if the team is especially unfamiliar with the domain and the context, and can decrease if the team is very familiar with the domain and the context.

Table 11.1: IoI weight recommendations for HCI activities in the waterfall model.

SE Phase / HCI Activity	Recommended weights
Communication	
User studies: contextual user studies and user modelling, competitive product analysis	3 – 4
Ideation: with a multidisciplinary team (HCI, tech, biz)	2
Product definition: IA / wireframes with a multidisciplinary team (HCI, tech, biz)	1 – 3
Usability evaluation 1: early formative eval. and refinement before freezing requirements	1 – 3
Modeling	
UI prototyping	4 – 5
Usability evaluation 2: formative eval. and refinement of prototype before development	4 – 5
Construction	
Development support: ongoing reviews by usability team during development	3
Usability evaluation 3: summative evaluation of an early version	1 – 3

Generating ideas is important. However, since user studies may also generate many ideas, the importance of explicit **ideation** is somewhat less. We therefore recommend it a weight of 2. However, sometimes, extensive user studies may not be done if the product is not based on contextual data but on ideas (for example, a toy, a game, or an interactive installation). In such cases, the weight of ideation will go up.

Product definition is recommended a weight of 1 to 3 because we feel this activity can vary in importance. In situations where the product is very innovative or particularly unpredictable, the weight of this activity can go up. On the other hand, if the product is very predictable and what needs to be done is clearly understood by all, the weight can go down.

Detailed UI prototyping is the crux of the HCI design process as the main deliverables of HCI professionals come forth from it. This activity is therefore recommended a weight of 4 to 5.

Usability evaluations are critical in getting the design of interactive products right. In our framework (chapter 3), we identified three occasions where usability evaluation can be done: just after the product definition, after detailing out the user interface, and after an initial version of the working product becomes available. Of these, the first two are formative (aimed at improving the design), while the last one is summative (aimed at ensuring that all goals have been met).

The formative evaluations are important as they directly affect the design. Between the two formative evaluations, we expect **usability evaluation 2** of the user interface to be more important as it will evaluate the design with many of its details in place. This evaluation is therefore recommended a weight of 4 to 5. We assume that if this formative evaluation was done well, the importance of doing the other two usability evaluations will be less.

Usability evaluation 1 of the product definition will usually have to be done on a very low fidelity prototype under very tight deadlines. In many contexts, a contract for the project is signed only after the requirements are finalised. Hence, we recommend a low weight of 1 for this step. In practice, the situations may vary somewhat. There may be opportunities (e.g. high-fidelity prototype was available early) and reasons (e.g. to demonstrate ideas to investors) to give more importance to the first formative evaluation. In this case, the weight of this usability evaluation can be increased and correspondingly, the weight for the next usability evaluation can be decreased.

Usability evaluation 3 of an early release is a summative evaluation and is expected to have little impact on design. Hence, it is also recommended a weight of 1. However, in projects where user is expected to do critical tasks, this step may gain weight of up to 3.

Finally, we reckon that a lot depends on the continued contact between the HCI professionals and the development teams after the activity of detailed UI prototyping has been completed. Unanticipated UI changes may arise late in the project. In many companies, the HCI professionals are a shared resource and they keep moving from one project to the next before the development in the earlier project is over. To emphasise the importance of **development support** and reviews of design changes during software development, we recommend this step a weight of 3.

11.2.2 Recommended Weights for HCI Activities in the Agile Model

Table 11.2 lists our initial weight recommendations for the HCI activities that we prescribed for the agile models we described above (section 7.3, page 107). Below, we give our reasons for these weights.

Table 11.2: IoI weight recommendations for HCI activities in the agile process model.

SE Phase / HCI Activity	**Recommended weights**
Before iterations start	
Contextual user studies and user modelling, competitive product analysis, ideation	3 – 4
Product definition/ IA / wireframes with a multidisciplinary team, evaluation	3
Detailed UI prototyping for 1st iteration	5
Usability evaluation (formative) and refinement of prototype for 1st iteration	3
During iterations	
Detailed UI prototyping for next iteration	4
Usability evaluation (formative) and refinement of prototype for next iteration	2
Development support reviews by usability team for current iteration	3 – 4
Usability evaluation (summative) of last iteration	1
After the last iteration	
Usability evaluation (summative) of release version	3

Before the iterations start, it is very important to understand the context of the user and the market scenario. Hence, the activity related to **user studies,** competitive product analysis, and ideation is recommended the weight of 3 to 4. The weight could increase if the team is especially unfamiliar with the domain, and could decrease if the team is very familiar with the domain and users.

Product definition and a preliminary evaluation where a multi-disciplinary team contributes, is recommended a weight of 3 because we thought it was an important activity. In the context of agile processes, this activity is more important than in the waterfall model because it will generate the user stories that support the creation of the first release plan. In contexts of projects where the product is very innovative or particularly unpredictable, the weight of this activity could go up. On the other hand, if the product is very predictable and what needs to be done is clearly known, the weight could go down.

As discussed above, the first two activities may not always be a part of the development project. Some companies may manage the first two steps as a separate project. Some companies may even run several such projects independently and then choose a particular product definition for development. Care needs to be taken while evaluating these activities. It should not matter whether the activities were done as a part of the current project or a separate one, so long as deliverables smoothly flow across the projects and a continuity of personnel is maintained.

As mentioned, it is important that HCI activities in agile processes stay one iteration ahead of software development activities. This is of particular importance for the first iteration, as this would set the 'tone' for the rest of the design iterations. **Detailed UI prototyping** of the first iteration user stories is therefore recommended a weight of 5. In addition, it is important to do a formative **usability evaluation** of the design before the development iterations begin. Therefore, this activity is recommended a weight of 3.

During the iterations of agile software development, it is still important for the HCI team to stay one iteration ahead. It is very important to design the **detailed UI prototype** for user stories of the next iteration during the current iteration (recommended weight of 4). It is expected that these intermediate prototypes would be **evaluated** for usability, though its rigour could be less than that of the first usability evaluation (recommended weight of 2). In the spirit of agile development, where there is lot of emphasis on team interaction and rather than documentation. We therefore assign higher weight for **development support** than in the waterfall model (recommended weight of 3 to 4). The HCI practitioners should continuously interact with the development team to ensure that design intentions are communicated and change requests are handled immediately.

Since releases will be available after the project enters the 2nd iteration of XP, it will be a good idea to do **summative evaluations** of the latest working versions of the last iteration. It is not necessary to do these after each iteration; doing one after 2 to 4 iterations may be good enough (recommended weight of 1).

Once the iterations are over, it is important to do a thorough **summative evaluation** on the final product (recommended weight of 3). Alternately, if the project is long, this evaluation may be planned once or twice a year, and not only at the end. It may be a good idea to involve an external evaluator for this evaluation to get fresh eyes and fresh ideas.

Example

Table 11.3 shows calculation of IoI for an example project in an industry project following the waterfall model. First, the project manager and the HCI practitioners working on a project fine-tuned the recommended weights for the project context (column 3). Three reviewers comprising of some project insiders and outsiders reviewed and rated the HCI activities in the project (column 4). Weighted average scores for HCI activities for each phase is presented (column 5). The weighted average for all activities, i.e. the IoI, is presented in the last row. In this example, it is clear that the biggest process improvements in this project were required in the communication phase. By comparing the IoI scores, phase scores and activity scores in several projects, the team can deduce where it needs to be putting in most of its efforts in process improvement.

Similarly, Table 11.4 shows calculation of IoI for an example project that uses our recommendations for agile methods described above.

Table 11.3: An example IoI calculation for the integration of HCI activities in the waterfall model.

Phases and HCI Activities	Recommended weights	Assigned weights	Activity Score	Phase Score
Communication				**50.00**
Contextual user studies and user modelling, competitive product analysis	3 - 4	4	25	
Ideation with a multidisciplinary team (HCI, tech, biz)	2	3	50	
Product definition / information architecture / wireframes with a multidisciplinary team (HCI, tech, biz)	1 - 3	3	50	
Usability evaluation (formative) and refinement of product definition	1 - 3	4	75	
Modelling				**75.00**
Detailed UI prototyping	4 - 5	5	75	
Usability evaluation (formative) and refinement of prototype	4 - 5	4	75	
Construction				**78.57**
Development support reviews by usability team	3	4	100	
Usability evaluation (summative)	1 - 3	3	50	
IoI Score				**64.17**

Table 11.4: An example IoI calculation for the integration of HCI activities in the agile model.

Phases and HCI Activities	Recommended weights	Assigned weights	Activity Score	Phase Score
Before iterations start				**60.71**
Contextual user studies and user modelling, competitive product analysis, ideation	3 – 4	4	50	
Product definition / IA / wireframes with a multidisciplinary, team, evaluation	3	4	50	
Detailed UI prototyping for 1st iteration	5	3	100	
Usability evaluation (formative) and refinement of prototype for 1st iteration	3	3	50	
During iterations				**56.67**
Detailed UI prototyping for the next iteration	4	4	100	
Usability evaluation (formative) and refinement of the prototype for the next iteration	2	4	25	
Development support reviews by usability team for the current iteration	3 – 4	3	50	
Usability evaluation (summative) of the earlier iteration	1	4	50	
After the last iteration				**75.00**
Usability evaluation (summative) of release version	3	4	75	
IoI Score				**60.61**

11.3 Industry Feedback on Usability Goals Achievement Metric and Index of Integration

A qualitative evaluation was done to get preliminary feedback from practitioners on UGAM and IoI. The goal of this evaluation was to try the two metrics in an industrial context, to get feedback on whether such metrics are necessary and useful, and to collect qualitative feedback to improve the metrics. We also wanted to measure the time it would take to compute the metrics in real-life situations.

UGAM and IoI were computed retrospectively for three projects in two large contracted software development companies. In each case, the metrics computation was done by HCI practitioners from the project, independent HCI practitioners, and project stakeholders. These stakeholders were familiar with the software artefact that was developed and the process that was followed. At the end of the metrics computation, feedback was taken from participants about the metrics.

It typically took about 3 hours to calculate both IoI and UGAM for a project. The time included explaining the two metrics, weight assignment, and scoring. This seemed to be the optimum time; longer meetings were difficult to schedule. The projects performed similarly in IoI and UGAM scores. The one project that had a high UGAM value also had a high IoI value.

Participants, particularly project stakeholders, were at home with the activity of metric calculation. To them, the activity brought HCI closer to SE. It created a lot of buy-in for HCI activities from the non-HCI stakeholders. One project stakeholder said, *"I never thought we could think so much [about user experience]."* The activity was more successful in projects where several stakeholders from the project participated as it stimulated discussion among stakeholders. While the participants appreciated the organizational perspective, the metrics seemed of less use to the projects as the projects were already over. Participants suggested that metrics should be calculated mid-way through the project while course correction was still possible.

Specifically, UGAM helped the HCI practitioners and project stakeholders to make goals explicit. One HCI practitioner remarked, *"Had we done this earlier, I would have known* *where to focus."* The teams adjusted weight to suit goal parameters to their project. They confirmed that this flexibility is indeed desirable. Though parameter evaluation guidelines helped, more details were desired (these were added subsequently in UGT as noted in Appendix I). Giving examples of HCI goals helped participants set goal parameters and weights. One stakeholder remarked, *"Without these inputs it would have been difficult to [assign weight and scores]."*

In case of a few UGAM parameters, divergent scores emerged for some parameters. Variations were observed in parameters for which the evaluation guidelines were not understood well or were interpreted differently by evaluators. In such cases, it was felt that it was better to let participants discuss the parameter and change ratings to converge scores if they so desire. (Please note, at this stage UGT had not been developed fully, the evaluators had been given only the preliminary list of goals and only sketchy guidelines.)

Initially, evaluators were free to assign any score from 0 to 100. Reducing the number of steps in scoring a parameter (for example, 0-25-50-75-100) and assigning each of the steps a concrete meaning helped reduce the variation in the scores. Detailed guidelines would help in reducing the divergence further.

Computing IoI was useful for project stakeholders as they could see the importance of HCI activities in the SE context. The HCI activities integrated in SE process models were acceptable as suggested. Though they were explicitly asked, none of the project stakeholders wanted changes to the prescribed HCI activities, their weights, or the evaluation guidelines.

An important feedback was a need for process models specifically targeted to redesign projects. Process models typically discuss new product development. Some participants asked for process models specifically targeted to redesign and porting projects. Given that many industry projects are of the type *"next version of X"* type, process models must be specifically adapted for them.

Walking through the activity evaluation guidelines helped in scoring, as not all stakeholders were aware of all HCI activities. Some project stakeholders felt that IoI should be computed before computing UGAM as this minimizes bias.

The metric descriptions presented above are a result of iterative modifications that reflect the feedback and lessons learnt from this qualitative evaluation.

11.4 Summary

In this chapter, we presented IoI, a metric to measure how closely the HCI activities were integrated with the SE process of a project as prescribed in a base process model. The weights for the HCI activities and guidelines for evaluating them are prescribed by the process prescriber, and fine-tuned as they are adapted by the organisation and the project team. At the end of the project, evaluators score each prescribed activity against the guidelines. IoI is a weighted average of HCI activity scores. Industry feedback on UGAM and IoI received during a qualitative evaluation indicates that both metrics were found to be helpful.

In the next chapter, we establish a correlation between IoI and UGAM, and demonstrate that adherence to the extended process models we proposed for waterfall and agile processes in chapter 7 help teams achieve their usability goals.

12 Measuring Effectiveness of Integration of Human-Computer Interaction in Software Development Processes

Having proposed process models for integrating HCI in waterfall, agile, and RUP in chapter 7, the research question we were dealing with was, *"How can we prove that our process model proposals worked and consistently led to more usable products?"*

The research becomes non-trivial for many reasons. Firstly, software products vary a lot. They are targeted to different users, to be used in different contexts, with different frequencies, and are deployed on different platforms. Secondly, design and development processes vary a lot. Teams follow different software development processes (waterfall, agile, RUP), and follow them to a different extent. Further, skills, creativity, knowledge, and experience of the team members vary. All this affects the outcomes of the project and makes it difficult to measure the quality of the process.

Seffah et al. state that empirical evidence of the acceptance, effects, and value of proposed usability engineering techniques is usually missing (Seffah, et al., 2005 p. 9). To empirically evaluate the value of a specific technique, it would be necessary to evaluate the same project repeated under conditions employing the technique verses *not* employing the technique, while controlling for skill, motivation, SE approach, and other possible differences between the two teams. Further, this challenging experiment would have to be repeated with different project teams, different software engineering frameworks, and on different projects in order for the results to achieve statistical validity. Assuming that n = 10 projects would give us the statistical validity required, and assuming that each project would have 10 control conditions, and further assuming that on an average, it costs ₹ 10,000,000 to do the project once, the budget of such an experiment would be in excess of ₹ 1,000,000,000, quite clearly well beyond the scope and budget of *our* research.

Assuming that we cannot repeat projects and control for parameters, how can one tell whether a process model helps a team achieve its goals, and whether it consistently leads to usable products? This brought us into the area of metrics and measurement. Though there is plenty of evidence of the a return on investment of usability activities in general (Bias, et al., 2005), there is no direct evidence that shows that better integration of HCI activities in SE processes will lead to better products at less cost. Further, there is no evidence that some HCI activities are more important than others are.

Assume that it is possible to express numerically the extent to which a process model is followed (a process metric like IoI). Further, assume that it is possible to express numerically the extent to which a team achieves its product goals (a product metric like UGAM). By following a prescribed process model to the extent X a project could achieve its goals to the extent Y, if we can demonstrate that for every X_1 that is greater than X_2, Y_1 is greater than Y_2 in most cases, we can conclude that the process model in question works; i.e. it helps the design team to achieve its goals. A correlation between the product metric and the process metric can also demonstrate the return on investment on following that process model; if a higher process metric consistently leads to a higher product metric, it makes sense to invest in that process.

Our primary hypothesis is that UGAM and IoI should correlate for our extended process model proposals. Our studies confirm this, as shown in this chapter.

12.1 Correlating Usability Goals Achievement Metric and Index of Integration

We wanted to explore whether IoI, a process metric, has an effect on UGAM, a product metric in industrial projects. Further, we wanted to compare UGAM and IoI with the practitioners' judgements about process quality and usability of the delivered product, and with an established metric such as SUS (Brooke, 1996). We also wanted to explore the impact of each HCI activity on goal achievement. To gather data, we carried out two studies (A and B) with two groups of participants from the Indian IT industry.

12.1.1 Procedure for Study A

The first group attended a training programme, a nine-day professional course on human-computer interaction design conducted by the author. It was attended by 35 people from various companies, mostly companies involved in software development. Participants came from mixed educational backgrounds such as graphic design, product design, web design, user interface design, e-learning, engineering, product management, and ergonomics. Industry experience of participants varied between 2-15 years but HCI-related experience was much less, typically 1-3 years. Many participants had an aptitude for design and a few of them even had formal design education.

A study to correlate UGAM and IoI was conducted during the course (Study A). Before the study started, the participants had attended four and a half days of the course during which they had learnt about conducting and analyzing contextual interviews, affinity diagrams, personas, conceptual models, layers of user experience, design process and user experience goals. After this, participants were invited to participate in the UGAM and IoI calculation for a project on which they had worked professionally. Participants were informed that it was optional to contribute the project data to the study and that they could remain anonymous in their submission if they so desired. Participants were distributed paper-based version of the UGT (Appendix I), UGAM (Appendix II) and IoI (appendix III).

Participants were then taught the method of using UGT, calculating UGAM, and IoI. During UGAM calculation, participants were walked through each goal parameter and were shown examples to explain its meaning. Similarly, during IoI calculation, they were walked through each HCI activity and its implication to the SE process. To begin with, the participants filled out the product profile and user profile as described above. If the product was supposed to be used by multiple user profiles, the participants were asked to set goals and calculate metrics for the most important user profile / the primary persona of the product. Next, the participants set weights for the goal parameters for their products using UGT. During the goal setting calculation, participants were walked through the goal parameters and were shown examples to explain their meaning. Then the participants rated the product against each goal parameter to calculate UGAM.

Finally, the participant calculated IoI by assigning weights and ratings to HCI activities done during the projects. They were walked through each HCI activity and its implications to the SE process. If waterfall process was used in their projects, the waterfall form (upper half of Appendix III) was used and if any of the variants of the agile processes was used, the agile form (lower half of Appendix III) was used.

After the participants had handed in the filled out forms, their choices were recorded and metrics were computed. Preliminary calculations were done using Microsoft Excel. Further statistical analysis of the data was done in SPSS.

12.1.2 Procedure for Study B

The second study (Study B) was done with a similar objective. It was done in more controlled conditions in one-on-one settings with the participants rather than in a group. This study was done with participants who had more experience and formal background in HCI than Study A. This study collected additional data about projects.

Before the study, a one-day tutorial on user experience metrics was announced on a mailing list of HCI practitioners in India. The tutorial was conducted five times on weekends in five different cities of Mumbai, Pune, Bangalore, Chennai, and Hyderabad in the months of January and February 2009.

The tutorial was free to attend, but it was open only to participants who had a few years of experience in HCI or a related area. The participants filled out a registration form for attending the tutorial and those who had no experience in HCI were screened out. The qualifying participants had an overall mean experience of 7 years and mean HCI related experience of 3.5 years. Participants were informed that after the tutorial they might be invited to participate in a research study in which would need to give 9-12 hours of their time, though this was not made a condition to attend the tutorial.

The tutorial gave a general overview of several user experience metrics, including SUS (Brooke, 1996), UGT, UGAM, and IoI. As an exercise, the participants calculated UGAM and SUS scores for popular products such as Gmail, Yahoo! Mail, and iPod.

After the tutorial, participants were chosen randomly and were invited to participate in the study through an individual email. Each participant who agreed was interviewed in two or three sessions.

The first session was a short briefing session over the phone, during which the participant was told the purpose and procedure of the study. During the study, the participant would calculate metrics for projects that he had worked on. Each participant was requested to contribute two projects. The participant was encouraged to contribute not only projects that he thought went well, but also projects that he thought did not go so well. If necessary, candidate projects were discussed and two projects were selected. The participant was promised confidentiality and was invited to participate in the second session.

The second session was scheduled at a time convenient to the participant. The participant was told that the session might last two to three hours. Many of the sessions for early participants were scheduled as face-to-face meetings. Later, some sessions were also scheduled on the phone.

At the start of the second session, the participant was reminded about the purpose and the procedure. Then, the participant's perceptions about the quality of user experience delivered in the product (D-UX) and the quality of the HCI process followed during the project (D-PRO) were recorded. The participant was asked to rate the user experience delivered for the project on a scale of 1-5, where 1 represented the worst possible user experience and 5 represented the best. He was asked to rate the quality of the HCI process followed for the project on a scale of 1-5, where 1 represented the worst and 5 represented the best.

Each participant was also requested to report the nature of the project, whether it was a contracted software development service for a client or a project in a product company. In addition, the participant was also requested to provide project management related information such as project size, effort, and cost. In most cases, participants did not have this information readily at hand. They were requested to provide this information later, but only few participants could actually get access to this information.

Then the participant was asked to fill out the SUS form (Brooke, 1996). A variation in the procedure had to be made for the purpose of this study. Normally, SUS is supposed to be filled out by potential users of the product after they have used the product and the score is averaged across users. Due to logistical constraints, it was not possible to follow this procedure in this study. Instead, the participant was asked to answer each SUS question as his best estimate of the expected average answer from users. Participants were encouraged to use information from usability tests, reviews, and user feedback where available to give their best answer.

Next, the participant calculated UGAM and IoI. He was handed over printed forms for UGAM and IoI calculation and was asked to fill them out while constantly thinking aloud. (For sessions conducted on the phone, the participants were sent a soft copy of the forms in advance.) The participant was not left alone to fill out the form, but was walked through the forms as he filled them out. Each question was verbally explained and the participant was encouraged to ask questions. Clarifications and examples were given where necessary. This was done to ensure that the participant understood each question clearly. The participant set weights for the goal parameters and scored the product against each goal parameter to calculate UGAM (Appendices I and II). Finally, the participant calculated IoI by assigning weights and scores of HCI activities done during the projects (Appendix III). If waterfall process was used, the waterfall form (upper half) was used and if any of the variants of the agile processes was used, the agile form (lower half) was used.

As the participant filled out the form, his choices were simultaneously recorded in Microsoft Excel with a template to compute the metrics. However, the participant was not told the results until he had finished computing all metrics for that project. After the session, the participant was mailed the file that showed the metrics that were calculated. The participant was also reminded to provide project management related information.

If the participant agreed, the calculations of the metrics for the second project were continued in the same session. Else, a similar third session was scheduled.

After calculating the metrics for the two projects, the participant was asked to review the scores and was given an opportunity to make changes to ratings or weights. The participant was asked to compare particularly the ratings across the two projects.

12.1.3 Analysis and Main Findings

35 participants participated in Study A, of which 21 participants submitted their data for both UGAM and IoI.

141 participants registered for the 5 tutorials in Study B. Of these, 83 participants qualified and attended the tutorials. Of these, 33 participants were requested to participate in the study. Most agreed, but only 23 could be scheduled. Between them, the 23 participants contributed 40 projects. 16 participants contributed 2 projects each, 6 participants contributed 1 project each, and 1 participant contributed 3 projects. Typically, it took between one to two hours for a participant to do the calculations for one project. The metrics calculation activities for the second project were usually faster than for the first project and the participants required much less help.

The participants came from a wide variety of companies including four large (25,000+ employees) contracted software development companies with, four relatively smaller contracted software development companies, four multi-national companies with large product development centres in India, a large, internationally popular internet company and five smaller product development companies.

A combined analysis of the 61 projects from Study A and Study B is presented below. Table 12.1 lists the raw data for the UGAM and IoI scores for the 61 projects. It also lists the type of process model followed in each project. Out of the 61 projects, 50 projects followed the waterfall model, while 11 projects followed agile process models. Of the 40 projects in Study B, 26 were carried out as part of a contracted software development service for a client while 14 were projects in product companies (this data was not collected in Study A).

Table 12.1: Raw UGAM and IoI scores for 61 projects in two studies

IoI	UGAM	Process	Type
92.39	69.93	Waterfall	Service
89.29	80.90	Waterfall	Product
80.43	71.43	Waterfall	
79.63	85.39	Waterfall	Service
79.31	80.65	Agile	Service
78.26	86.67	Waterfall	
77.17	71.40	Waterfall	
76.09	73.61	Waterfall	
75.83	74.68	Waterfall	Service
73.75	71.70	Waterfall	Product
72.00	75.33	Waterfall	Service
70.95	68.67	Waterfall	Service
69.17	77.32	Agile	Product
69.17	77.32	Agile	Product
67.31	63.51	Waterfall	Service
67.24	68.00	Agile	Product
66.30	64.08	Waterfall	
66.18	85.14	Agile	Service
65.63	68.66	Waterfall	Service
65.38	75.73	Waterfall	Service
65.22	70.71	Waterfall	
64.58	75.63	Waterfall	Product
64.17	66.81	Waterfall	Service
64.13	58.59	Waterfall	
63.39	78.78	Waterfall	Product
63.00	61.44	Waterfall	Service
62.96	80.16	Waterfall	Product
62.96	74.39	Agile	Product
61.46	68.88	Waterfall	Product
61.00	70.79	Waterfall	Service
60.61	70.99	Agile	Service
58.70	67.56	Waterfall	
58.33	70.74	Waterfall	Service
57.61	78.19	Waterfall	
57.50	74.65	Waterfall	Service
56.00	66.67	Waterfall	Product
55.47	77.22	Agile	Service
55.43	77.59	Waterfall	
55.36	62.26	Waterfall	Product
54.57	65.18	Waterfall	
51.92	56.32	Waterfall	Service
51.09	57.77	Waterfall	
50.00	65.41	Waterfall	Service
50.00	58.33	Waterfall	
50.00	52.96	Waterfall	
50.00	44.20	Waterfall	
46.74	65.91	Waterfall	
46.55	48.42	Agile	Service
46.05	42.47	Waterfall	Service
45.83	61.43	Waterfall	Service
41.30	46.52	Waterfall	
38.54	55.36	Agile	Service
38.04	57.39	Waterfall	Product
38.04	53.05	Waterfall	
37.50	59.23	Agile	Service
36.96	33.42	Waterfall	
35.87	44.21	Waterfall	
33.93	46.55	Waterfall	Service
31.00	47.30	Waterfall	Service
29.35	54.58	Waterfall	
26.85	56.92	Waterfall	Product

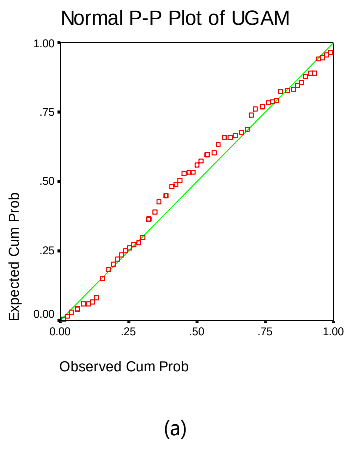
Normal P-P plots (Figure 12.1) drawn for UGAM and IoI show that assumptions of normality are not grossly violated for either metric. Therefore, to determine the relationship between UGAM and IoI, a two-tailed Pearson's correlation was performed. There is a significant positive correlation (r = 0.752, *p* < 0.0005 two-tailed) between UGAM (*M= 65.82, SD= 11.92, N=61*) and IoI (*M = 58.38, SD= 14.93, N= 61*). A simple linear regression model gives the best fitting relationship between a predictor variable and a criterion variable (Kurtz, 1983 pp. 254-286). A simple linear regression was performed to determine if IoI significantly determines the scores of UGAM. A significant model emerged (Table 12.2, column 2), with predictor IoI accounting for 56% of the variance in UGAM (adjusted $r^2 = 0.56$), which was very significant ($F = 76.862$, $p < 0.0005$). The coefficients (Table 12.3, column 2) show that IoI demonstrated significant effects on UGAM ($\beta = 0.601$, $p < 0.0005$). The t-statistic for the slope was also significant $t = 8.767$, $p < 0.0005$. The 95% confidence interval for IoI β is from 0.463 to 0.738. Thus, we can conclude that there was a positive significant relationship between IoI and UGAM.

UGAM and IoI were analysed using ANOVA of the means (Table 12.4). A small significance value indicates that a linear relationship exists between UGAM and IoI ($F = 69.94$, $p < 0.0005$).

Thus, the expected value of UGAM can be represented by this equation:

$$UGAM = 0.601 * IoI + 30.763$$

Figure 12.2 shows the normal p-p plot for the regression standardised residuals for the above equation. The normality assumption for the regression standardised residuals for UGAM is not grossly violated. We can conclude that the relationship between IoI and UGAM is strong, positive, and linear. This is also evident in the curve plotted between observed IoI and UGAM drawn against theoretical linear distribution between the two variables (Figure 12.3).



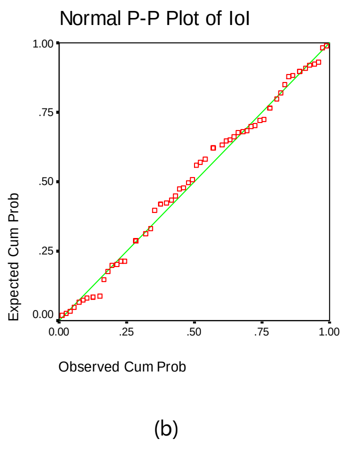


Figure 12.1: Normal P-P Plot for User Experience Goal Achievement Metrics (a) and for Index of Integration (b) (N= 61).

Table 12.2: Model summary for UGAM regressed on IoI

		Overall (N = 61)	Waterfall (N = 50)	Agile (N = 11)	Service (N = 26)	Product (N = 14)	Service Waterfall (N = 19)	Product Waterfall (N = 10)
R		0.752	0.753	0.809	0.747	0.825	0.769	0.841
R^2		0.566	0.567	0.655	0.559	0.681	0.591	0.707
Adjusted R^2		0.558	0.558	0.617	0.540	0.654	0.567	0.671
Standard error of the estimate		7.9209	7.9012	7.1171	7.9147	4.7463	7.2775	5.1896
Change Statistics	R^2 change	0.566	0.567	0.655	0.559	0.618	0.591	0.707
	F change	76.862	62.924	17.084	30.370	25.617	24.607	19.326
	df1	1	1	1	1	1	1	1
	df2	59	48	9	24	12	17	8
	Significance F change	0.000	0.000	0.003	0.000	0.000	0.000	0.002

Table 12.3: Regression coefficients for UGAM regressed on IoI

			Overall (N = 61)	Waterfall (N = 50)	Agile (N = 11)	Service (N = 26)	Product (N = 14)	Service Waterfall (N = 19)	Product Waterfall (N = 10)
Constant	Un-std coeff.	B	30.763	30.992	29.270	32.245	43.796	31.691	43.958
		Std error	4.126	4.409	10.172	6.365	5.554	7.031	6.131
		T	7.457	7.030	2.878	5.066	7.885	4.508	7.169
		Sig.	0.000	0.000	0.018	0.000	0.000	0.000	0.000
	95% conf. int. for B	Lower bound	22.508	22.128	6.260	19.109	31.695	16.858	29.819
		Upper bound	39.018	39.856	52.280	45.382	55.898	46.524	58.097
IoI	Un-std coeff.	B	0.601	0.582	0.693	0.576	0.445	0.559	0.439
		Std error	0.068	0.073	0.168	0.104	0.088	0.113	0.100
		T	8.767	7.932	4.133	5.511	5.061	4.961	4.396
		Sig.	0.000	0.000	0.003	0.000	0.000	0.000	0.002
	95% conf. int. for B	Lower bound	0.463	0.434	0.314	0.360	0.254	0.321	0.209
		Upper bound	0.738	0.729	1.072	0.791	0.637	0.796	0.669

*Table 12.4: ANOVA of the means for UGAM * IoI (N = 61)*

		Sum of squares	df	Mean square	F	Significance
Between groups	(Combined)	7972.475	52	153.317	2.224	0.115
	Linearity	4822.351	1	4822.351	69.943	0.000
	Deviation from linearity	3150.124	51	61.767	0.896	0.632
Within groups		551.572	8	68.947		
Total		8524.048	60			

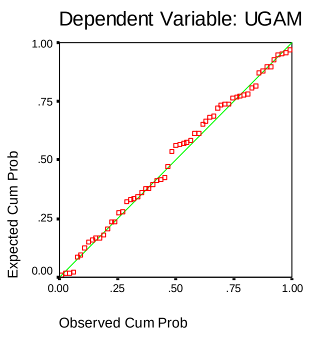


*Figure 12.2: Normal p-p plot of regression standardised residuals for UGAM = 0.601 * IoI + 30.763.*

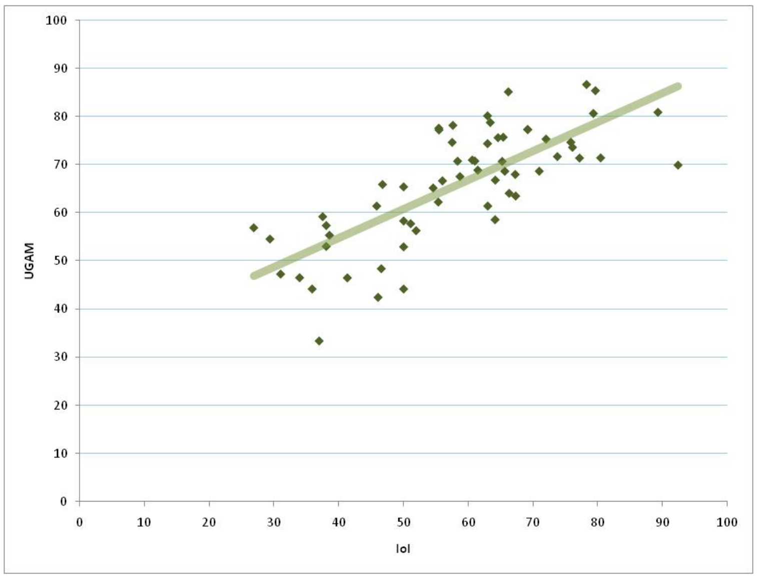


Figure 12.3: IoI vs. UGAM scatter plot against linear curve (R^2 = 0.56) (N = 61)

Table 12.2 and Table 12.3 report further findings from regression analyses on subsets of the data only considering the projects that followed:

- the waterfall models (N = 50, column 3)
- the agile process models (N = 11, column 4)
- the known contracted service projects (N = 26, column 4)
- the known product company projects (N = 14, column 5)
- the known contracted service projects that followed the waterfall model (N = 19, column 6)

- the known product company projects that followed the waterfall model (N = 10, column 7).

All models were significant (Table 12.2, *p* <= 0.003) and consistently returned significant positive coefficients, with all pairs of the 95% confidence interval limits for IoI coefficients positive (Table 12.3, *p* <= 0.003). Thus, it could be concluded that IoI and UGAM are positively correlated in software projects that use either waterfall model or agile model and by organisations that are involved in contracted service projects as well as those involved in product development.

It is interesting to note that projects following agile process models (UGAM = ***0.69****IoI + 29.3) have a somewhat higher coefficient of IoI compared to that of projects following waterfall process models (UGAM = ***0.58****IoI + 31.0) (Table 12.3, Figure 12.4). UGAM gains more in agile projects per unit increase in IoI than in waterfall projects. Agile process models return a better user experience than in waterfall process model and the difference increases with better integration.

Similarly, it is interesting to note that the constant is somewhat higher and the coefficient of IoI is somewhat lower for the subset of product company projects (UGAM = ***0.45****IoI + <u>43.8</u>) compared to the subset of services company projects (UGAM = ***0.58****IoI + <u>32.2</u>) (Table 12.3, Figure 12.5). Product company projects score a higher UGAM compared to service company projects at low IoI scores. This implies that process integration is somewhat less important in product companies than in service companies, possibly because there are other factors at play in product companies (such as high stakeholder involvement, immediacy of business impact of poor usability, or simply high levels of skills, talent, and experience).

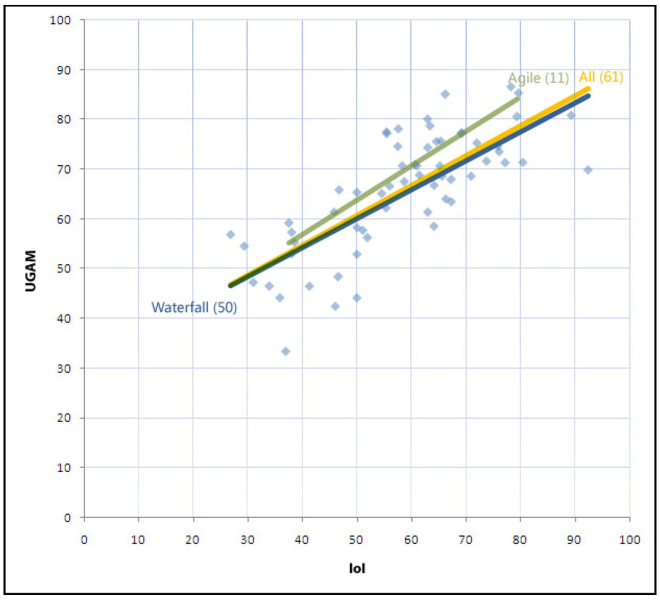


Figure 12.4: The difference between agile projects (N = 11) and waterfall projects (N = 50).

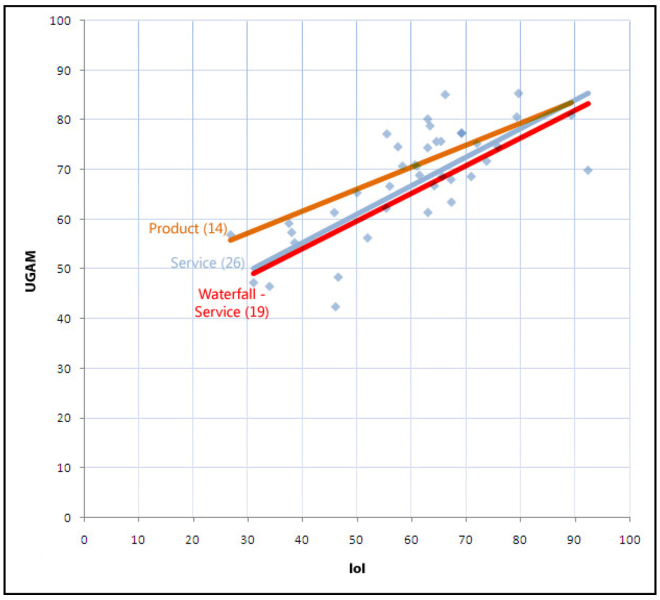


Figure 12.5: The difference between product projects (N = 14), service projects (N = 26) and service projects following the waterfall model (N = 19).

An important limitation of the two studies was that UGAM and IoI calculations were done by only one person from each project. Ideally, the metrics should be calculated by averaging inputs from several evaluators, both internal and external. This could not be done due to logistical constraints. We had assured the participants that the data would be kept confidential. To an extent, this would have helped overcome this limitation and allowed participants to be critical of their own work.

Another limitation of these studies is that all projects came from the Indian IT industry. The practices in the IT industry in India could be different from those in other countries. For example, in India the most popular process models are waterfall and agile. In our studies, none of the participants reported using RUP in their projects. Additional studies will be required in other geographies before the results can be generalised. Yet, within these limitations, we believe we have been able to get a wide category of projects for our studies, including services and products, and projects following the waterfall and agile process models.

12.2 Validation of Usability Goals Achievement Metric and Index of Integration

Additional metrics were collected with the purpose of validating UGAM and IoI in study B. The participant was asked to rate his perception of the user experience delivered for the project on a scale of 1-5 (D-UX). He was also asked to rate the quality of the HCI process followed for the project on a scale of 1-5 (D-PRO). In addition, the participant was asked to fill out a SUS form for the project (Brooke, 1996) with the variation described above. Table 12.5 lists the five metrics collected for each of the 40 projects. Normal P-P plot (Figure 12.6) shows that normality assumption is not grossly violated for all five metrics D-UX, D-PRO, SUS, UGAM, and IoI. This analysis was done to test the hypotheses: *UGAM is correlated with SUS and D-UX positively. IoI is correlated with D-PRO positively.*

Table 12.5: D-UX, D-PRO, SUS, UGAM and IoI for the 40 projects in study B.

D-UX	D-PRO	SUS	UGAM	IoI
2.5	3	85.0	62.3	55.4
4.5	4	95.0	71.7	73.8
4	5	95.0	69.9	92.4
3	2	67.5	74.7	57.5
3	5	50.0	68.7	70.9
4	4	90.0	85.1	66.2
2	1.5	30.0	46.6	33.9
4	3	82.5	66.8	64.2
4	3	72.5	63.5	67.3
3	4	75.0	70.7	58.3
4	2	82.5	55.4	38.5
4	3	92.5	78.8	63.4
4	4	70.0	68.9	61.5
5	5	87.5	80.2	63.0
4	2	47.5	56.9	26.9
3.5	3.5	72.5	77.2	55.5
2.5	3	35.0	48.4	46.6
4	3	90.0	74.4	63.0
4	3.5	92.5	68.7	65.6
3	3	40.0	42.5	46.1
4	4	90.0	75.6	64.6
3	3	60.0	56.3	51.9

3	2	52.5	47.3	31.0
4	3	82.5	74.7	75.8
5	4	97.5	85.4	79.6
4	4.5	70.0	77.3	69.2
4	4	72.5	68.0	67.2
4	4	82.5	77.3	69.2
4.5	4	87.5	80.7	79.3
2.5	3	47.5	61.4	63.0
4	4	75.0	70.8	61.0
4	3	70.0	61.4	45.8
3.5	2	77.5	57.4	38.0
4	3	72.5	71.0	60.6
2	2	55.0	59.2	37.5
4	4	95.0	75.3	72.0
3	3	65.0	65.4	50.0
4	3	80.0	66.7	56.0
5	5	82.5	80.9	89.3
4	3	75.0	75.7	65.4

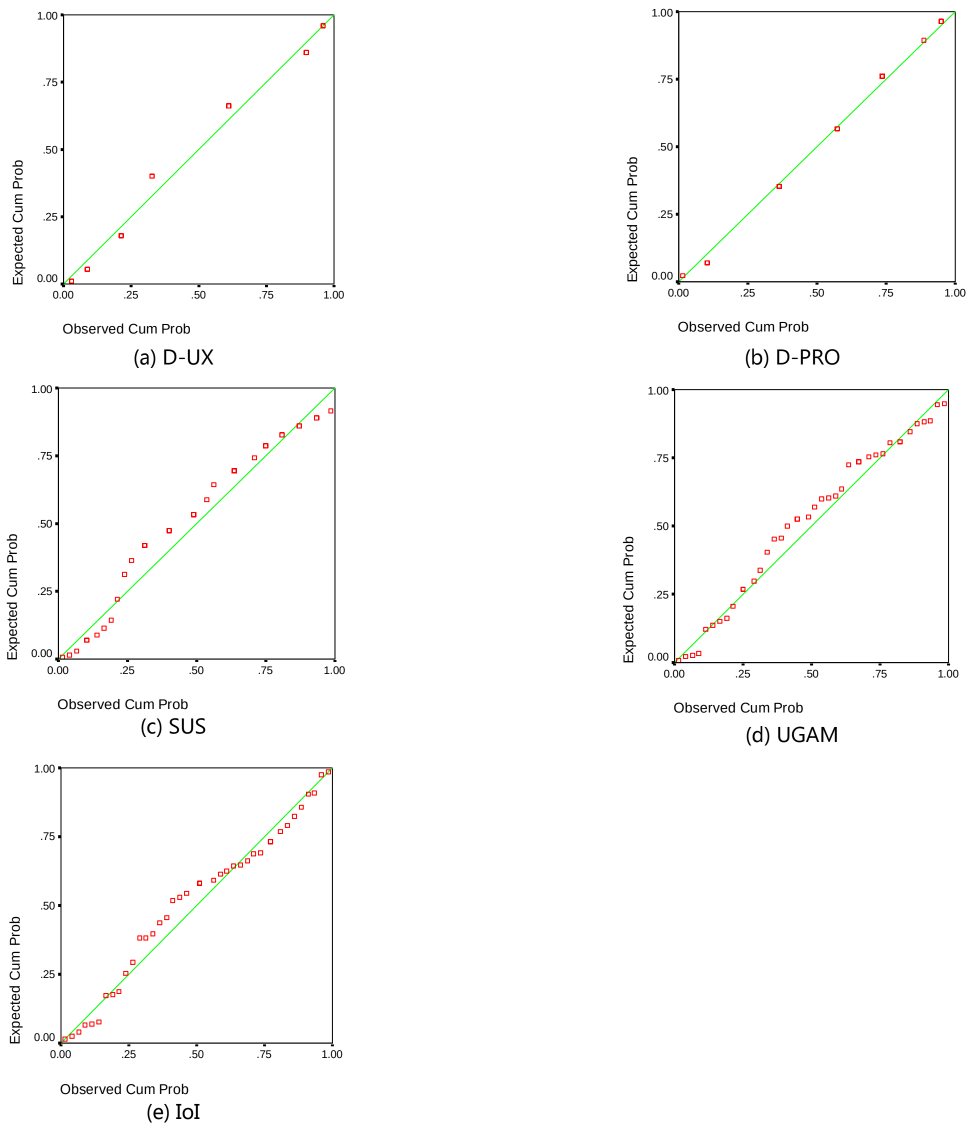


Figure 12.6: Normal P-P Plot for (a)D-UX, (b) D-PRO, (c) SUS, (d) UGAM and (e) IoI based on 40 projects in study B.

To determine the relationships between SUS and UGAM, a two-tailed Pearson's correlation was performed (Table 12.6). There is a strong, significant positive correlation between UGAM (M= 67.98, SD= 10.71) and SUS (M = 73.56, SD = 17.55) (r = 0.753, N = 40, *p* < 0.0005 two-tailed).

Table 12.6: SPSS output of Pearson Correlations between product metrics UGAM and SUS for study B (N = 40).

		SUS
UGAM	Pearson Correlation	0.753**
	Sig. (2-tailed)	0.000
	Sum of Squares and Cross-products	5515.100
	Covariance	141.413
	N	40

** Correlation is significant at the 0.01 level (2-tailed).

To determine the relationship between UGAM and D-UX, a two-tailed Spearman's rho correlation was performed (Table 12.7). There is a strong significant positive correlation between UGAM and D-UX (rho = 0.660, N = 40, *p* < 0.0005 two-tailed).

Table 12.7: SPSS output of Spearman's rho among product metrics UGAM and D-UX for study B (n= 40).

		D-UX
UGAM	Spearman's rho	0.660**
	Sig. (2-tailed)	0.000
	N	40

** Correlation is significant at the 0.01 level (2-tailed).

We can conclude that UGAM is positive correlated with both SUS and D-UX, and UGAM scores are not in dissonance with other product metrics, including the judgement of the HCI practitioners. This constitutes an additional validation of UGAM apart from the one described in section 10.3.

To determine the relationship among the process metrics IoI and D-PRO, a two-tailed Spearman's rho correlation was performed (Table 12.8). There is a strong significant positive correlation between IoI and D-PRO (rho = 0.746, N = 40, *p* < 0.0005).

We can conclude that IoI is positive correlated with D-PRO, and IoI scores are not in dissonance with the judgement of the HCI practitioners. This constitutes a validation of IoI.

Table 12.8: SPSS output of Spearman's rho among process metrics IoI and D-PRO for study B (n= 40).

		D-PRO
IoI	Spearman's rho	0.746**
	Sig. (2-tailed)	0.000
	N	40

** Correlation is significant at the 0.01 level (2-tailed).

12.3 Comparing Metrics with Practitioners' Judgements

This analysis was done to test the following hypotheses: *UGAM measures the quality of user experience better than the practitioner's judgement alone. IoI measures the quality of HCI process better than the practitioner's judgement alone.*

By merely comparing a product metric with another product metric, we cannot determine whether one is any better than the other. The two product metrics may disagree on scores for individual projects because of validity issues with *either* of them. On the other hand, if we assumed that process metrics are correlated with product metrics, we could compare two product metrics based on how well each of them correlates with one or more process metrics.

To determine the relationships between the product metrics UGAM, SUS and D-UX and the process metrics IoI and D-PRO, two-tailed Spearman's rho correlations were performed (Table 12.9). There are significant positive correlations between all pairs (p <= 0.004).

By comparing across rows of Table 12.9, we can observe that among the three product metrics, UGAM has the highest correlation with either process metric. Correlations of UGAM with the process metrics (0.732 and 0.654) are consistently higher than the correlations of D-UX with the process metrics (0.597 and 0.498, respectively).

Thus, we could accept that UGAM measures the quality of user experience better than the practitioners' judgements alone.

Table 12.9: SPSS output of Spearman's rho between product metrics D-UX, SUS and UGAM and process metrics D-PRO and IoI for study B (N= 40).

		IoI	D-PRO
UGAM	Spearman's rho	0.732**	0.654**
	Sig. (2-tailed)	0.000	0.000
	N	40	40
SUS	Spearman's rho	0.627**	0.448**
	Sig. (2-tailed)	0.000	0.004
	N	40	40
D-UX	Spearman's rho	0.597**	0.498**
	Sig. (2-tailed)	0.000	.001
	N	40	40

** Correlation is significant at the 0.01 level (2-tailed).

Similarly, we can compare two process metrics based on how well each of them correlates with product metrics. We can observe from Table 12.9 by comparing across columns that IoI has consistently higher correlations with the three product metrics (0.732, 0.627, and 0.597) than D-PRO (0.654, 0.448, and 0.498 respectively).

Thus, we could accept that IoI measures the quality of HCI process better than the practitioner's judgement alone.

12.4 Validation of Recommended Weights

In Table 11.1 (page 187), we had made recommendations of weights for HCI activities of the waterfall model. We had derived these recommendations qualitatively. This analysis was done to validate our recommendations, and to test the following hypothesis: *Some HCI activities affect UGAM more than other HCI activities.* We also wanted to indentify the HCI activities that affect UGAM most, and their extent. The subset of 50 projects using the waterfall model was used for this analysis (Table 12.1, page 204).

12.4.1 Weights Assigned by Participants

Table 12.10 lists the averages and standard deviations of weights for HCI activities actually assigned by participants in these projects (n = 50). Participants have not deviated substantially from our recommendations.

Table 12.10: Weights of the HCI activities recommended for the waterfall model, and the average and the standard deviation of weights actually assigned by participants to those HCI activities (N = 50).

SE Phase / HCI Activity	Recommendations	Assigned weight average	Assigned weight SD
Communication			
Contextual user studies and user modelling, competitive product analysis	3-4	3.7	0.8
Ideation with a multidisciplinary team (HCI, tech, biz)	2	2.5	0.7
Product definition / IA / wireframes with a multidisciplinary team (HCI, tech, biz)	1-3	3.1	0.7
Usability evaluation (formative) and refinement of product definition	1-3	2.0	1.1
Modeling			
Detailed UI prototyping	4-5	4.5	0.6
Usability evaluation (formative) and refinement of prototype	4-5	3.8	0.8
Construction			
Development support reviews by usability team	3	3.2	0.8
Usability evaluation (summative)	1-3	1.9	1.0

12.4.2 *Weights Derived from Regression Analysis*

The score of each HCI activity in IoI is a measure of the quality and fidelity of that HCI activity in a project. UGAM is a measure of the quality of user experience delivered by a project. The UGAM score is arrived at independently of the scores of HCI activities. If we can find the relative effect of the scores of HCI activities on the UGAM scores, this could be a way of evaluating the impact of HCI activities on the usability.

In the subset of 50 projects using the waterfall model, the normality assumption of the UGAM scores did not seem to be violated (Figure 12.7). Separate simple linear regressions were performed assuming the scores of each of the eight HCI activities to be the predictor variables and UGAM to be the criterion variable (Table 12.11). In case of each HCI activity, a significant model emerged and the activity score had a positive significant Pearson's correlation with UGAM ($0.56 > r > 0.33$, $0.32 > r^2 > 0.11$, $0.30 >$ adjusted $r^2 > 0.09$, $22.399 > F > 5.796$, $p <= 0.02$, two-tailed). All coefficients were positive. All lower bounds of the 95% confidence intervals of the coefficients were positive.

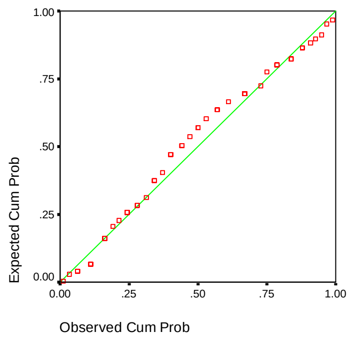


Figure 12.7: Normal P-P plot for UGAM scores of projects using waterfall model (21 projects in study A and 29 projects in study B, N=50).

Table 12.11: Summary of simple linear regressions on UGAM as criterion variable and the ratings of individual HCI activities as predictor variables on merged project scores (N = 50). The top four correlating activities have been highlighted.

									95% Confidence Interval for B	
	R	**R square**	**Adjusted R square**	**F**	**Sig.**	**B**	**t**	**Sig.**	**Lower Bound**	**Upper Bound**
User studies	0.445	0.207	0.190	12.517	0.001	0.221	3.538	0.001	0.095	0.346
Ideation	0.384	0.148	0.130	8.326	0.006	0.190	2.886	0.006	0.057	0.322
Prod Def	0.406	0.165	0.148	9.481	0.003	0.227	3.079	0.003	0.079	0.375
UE 1	0.351	0.123	0.105	6.748	0.012	0.162	2.598	0.012	0.037	0.287
UI Proto	0.564	0.318	0.304	22.399	0.000	0.299	4.733	0.000	0.172	0.426
UE 2	0.534	0.285	0.270	19.126	0.000	0.249	4.373	0.000	0.134	0.363
Dev Support	0.532	0.283	0.268	18.967	0.000	0.216	4.355	0.000	0.116	0.315
UE 3	0.328	0.108	0.089	5.796	0.020	0.134	2.407	0.020	0.022	0.246

We can conclude that all HCI activities recommended for the waterfall model in Table 11.1 affect UGAM positively and all activities are necessary. We can also see that scores of some HCI activities have a larger effect on UGAM than other activities. The strongest correlations, largest adjusted r^2 values, and largest coefficients were observed for the HCI activities of user interface prototyping, usability evaluation of the user interface and refinement, development support, and user studies, user modelling, competitive product analysis. This validates our 3+ weight recommendations for these activities (Table 11.1, page 187). This also justifies the 3+ average weight assigned by participants to these activities (Table 12.10).

The adjusted r^2 value in a simple linear regression is the measure of the proportion of variability in the criterion variable explained by the fitted model (Walpole, et al., 2007 p.

431). We could derive weights of the HCI activities for IoI in proportion to the adjusted r^2 values from Table 12.11, as we show below in column 4 of Table 12.13.

Brace et al. suggest that a multiple regression can be used to compare the relative contribution of each predictor variable to the criterion variable and assess the strength of the relationship (Brace, et al., 2003 pp. 212, 213, 222). The standardised coefficients of the predictor variables in a multiple regression allow us to do this relative comparison.

Using the stepwise method, a multiple regression was performed assuming the scores of the eight recommended HCI activities as the predictor variables and UGAM as the criterion variable. The most significant model returned these values: $r = 0.784$, $r^2 = 0.614$, adjusted $r^2 = 0.580$, $F = 8.533$, $p < 0.005$. The top four HCI activities identified above (UI prototyping, development support, usability evaluation 2 and user studies) also emerged as significant predictors in this model (Table 12.12). The scores on these four HCI activities predicted 58% of variation in UGAM. These four HCI activities had a positive, significant coefficient ($p <= 0.023$) and the lower bound of the 95% confidence interval for all coefficients was positive. The variance inflation factors (VIFs) of all predictor variables are below 1.4, indicating that there is no multi-collinearity among the predictor variables.

We could derive weights of the HCI activities in proportion to these standardised coefficients as shown in column 5 of Table 12.13. The weights derived from simple linear regressions and the stepwise multiple regression were in consonance with our original recommendations and with the weights assigned by the participants, as summarised in Table 12.13.

The main insight from this analysis perhaps was the derived weight for the development support HCI activity, which seems to be even higher than the one recommended by us and the one assigned by the participants. This activity has not received enough attention previously. This could be a matter of future investigation.

Table 12.12: The most significant model in the SPSS output of stepwise multiple linear regression on UGAM as criterion variable and the ratings of HCI activities as predictor variables (n = 50).

R	R Square	Adjusted R Square	Std. Error of the Estimate	Change Statistics				
				R Square Change	F Change	df1	df2	Sig. F Change
0.784	0.614	0.580	7.702	0.073	8.533	1	45	0.005

	Unstandardised Coefficients		Standardised coefficients	t	Sig.	95% Confidence Interval for B	
	B	Std. Error				L. Bound	U. Bound
(Constant)	33.794	4.024		8.398	0.000	25.690	41.889
UI Prototyping	0.133	0.057	0.253	2.346	0.023	0.019	0.247
Dev Support	0.123	0.040	0.306	3.064	0.004	0.042	0.204
Usability Eval 2	0.154	0.048	0.332	3.208	0.002	0.057	0.250
User studies	0.138	0.047	0.286	2.921	0.006	0.043	0.233

Table 12.13: A comparison of our recommended weights, average weights assigned by participants, weights derived by scaling up adjusted r^2 values from simple linear regressions (SLRs) and from scaling up the standardised coefficients of the stepwise multiple regression (MR).

HCI Activity	Recommended weights	Assigned weights	Derived weights scaled from	
			SLRs	MR
User studies, user modelling…	3 – 4	3.7	3.1	4.3
Ideation with a multidisciplinary team	2	2.5	2.1	-
Product definition	1 – 3	3.1	2.4	-
Usability evaluation 1…	1 – 3	2.0	1.7	-
User interface prototyping	4 – 5	4.5	5.0	3.8
Usability evaluation 2 of the UI…	4 – 5	3.8	4.4	5.0
Development support…	3	3.2	4.4	4.6
Usability evaluation 3…	1 – 3	1.9	1.5	-

The scores for the four HCI activities predict 58% of variation in UGAM compared to 56% predicted by IoI (the adjusted r^2 value in Table 12.2, column 3, page 206). This implies that it should be possible to construct an IoI-like process metric weighted as in Table 12.13 that measures the performance of only these four activities and predict UGAM in a similar way if not better. Though these four activities statistically turn out to be significant, we cannot conclude that the other HCI activities are irrelevant. More data (than the 50 projects reported here) could return models with significant coefficients for other activities, though we expect the weights of those activities to be lower than the top four activities.

A possible critique of our method for the purpose of this analysis could be that we showed the recommended weights to the participants before they assigned theirs. While we could have asked participants to assign weights without giving them our recommendation, it must be noted that that neither the recommended weights, nor the weights assigned by participants played a role in the regression analyses, which are based on the UGAM scores and activity scores alone. The weights derived from the regression analyses validate both the recommended and the assigned weights.

Another possible critique could be about our assumption that the scores of HCI activities are independent variables. Although the activity scores are naturally related (teams likely to score well on some HCI activities are likely to score well on others), it was essential to use them as predictor variables as it is the only way to establish their effect on usability. We minimised the bias by prescribing guidelines for evaluating each activity. The low VIFs imply that the assumption that the HCI activity scores are independent variables was acceptable for the purpose of the stepwise multiple regression.

Knowing which HCI activities are important would be useful in many contexts, particularly when resources are scarce and tradeoffs need to be made. Designers can use the rigorous, higher fidelity methods on activities that are more important, and make do with discount methods on less important activities.

We used our framework of HCI activities, the waterfall model, UGAM as the product metric, and projects from the Indian IT industry to find the relative contribution of various HCI activities. Our results may be generalised within these choices. Other researchers could use other frameworks, other process models, other product metrics, and / or other contexts in a similar way to identify the activities that matter in those contexts.

12.5 Summary

In this chapter, we presented data from 61 industrial projects and its analysis to validate the extended process models for waterfall and agile processes presented in chapter 7, validate the metric UGAM presented in chapter 10, validate the metric IoI presented in chapter 11, and validate the weights of HCI activities in the extended waterfall process model presented in section 11.2.

We validated the extended process models by demonstrating the correlation between UGAM and IoI. We found that IoI has different effect on UGAM in waterfall projects compared to agile projects, and in product company projects compared to service company projects.

We validated UGAM by demonstrating the correlation between UGAM and the practitioner's judgements about the usability of the product delivered by their projects and with SUS. We validated IoI by demonstrating the correlation between IoI and the practitioner's judgements about the quality of the process followed by their projects. UGAM and IoI had higher correlations with other metrics than the practitioner's judgements.

We validated our recommended weights in the extended waterfall process with the help of individual regressions of the HCI activity scores on UGAM, and a stepwise multiple regression of the HCI activities scores on UGAM. Analysis showed that development support seems to have a somewhat larger impact on UGAM than expected.

The next chapter presents our conclusions.

13 Conclusions

13.1 Findings and Contributions

Figure 13.1 summarises our contributions. Our contributions are aimed at improving and planning HCI activities in development processes and at helping in setting and scoring HCI goals. We have proposed a framework to organise HCI activities and use it as a baseline to extend SE process models. We have defined Index of Integration (IoI) to compare the process followed in the project with our prescribed process models. We have developed Usability Goals Setting Tool (UGT) to help set goals and Usability Goals Achievement Metric (UGAM) to help evaluate the product against those goals. By demonstrating the correlation between IoI and UGAM in industry projects, we have validated our process model extensions for waterfall and agile processes.

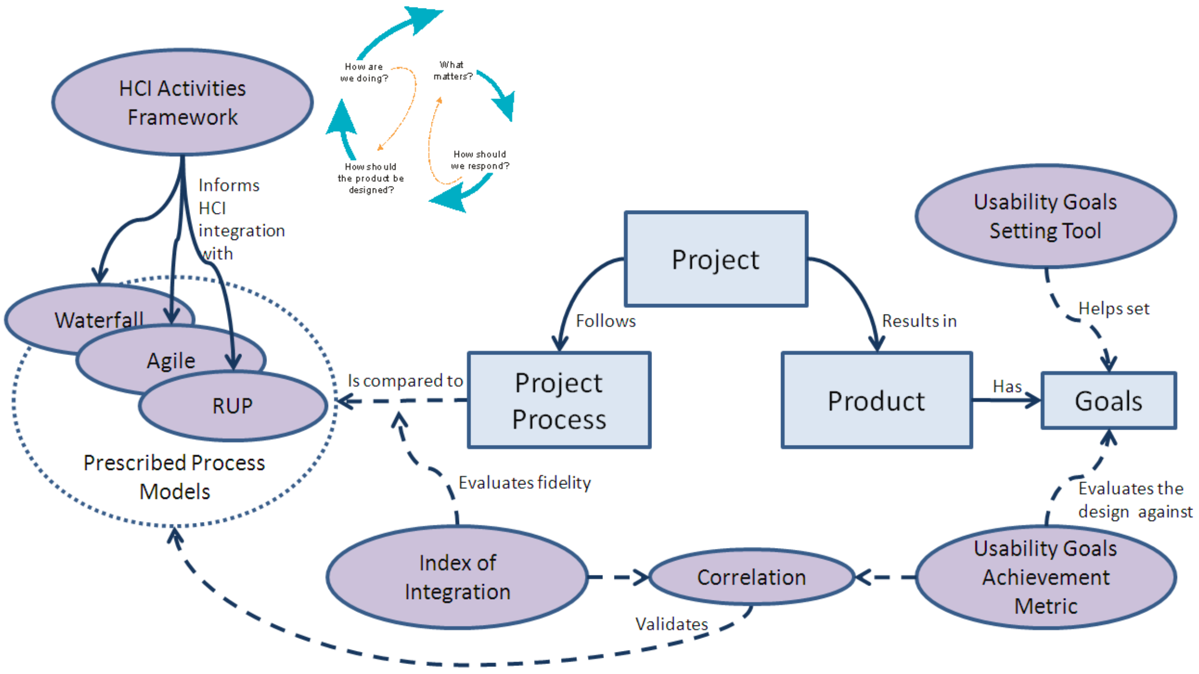


Figure 13.1: Summary of contributions.

Drawing from prior work, we have developed a framework consisting of HCI activities, methods, deliverables, and skills. We have used it as the basis for integrating HCI activities in SE process models of waterfall, agile, and RUP. While doing so, our focus has been to integrate all the essential HCI activities without disrupting the core values of the SE process models.

We have developed UGT to help the development teams set usability goals. We have identified 30 usability goal parameters, and related goal setting and evaluation guidelines, which give a starting point for HCI practitioners to set usability goals. Both experienced and inexperienced HCI practitioners have found the tool useful and said the tool helped them understand the context of the project better and made them think of goals that they had not thought of earlier.

Data from 65 industry projects established the need for such a tool. More than a third of the high-weighted goal parameters scored undecided or worse. Further, higher weighted goal parameters have a lower weight-score correlation. Projects achieve better scores for a typically low-weighted goal parameter when it gets an occasional higher weight, but do not achieve similar better scores for a typically high-weighted goal parameter when it gets an even higher weight. We have identified 8 high-weighted goal parameters, which did not have a good weight-score correlation. Our interpretation is that these goal parameters do not receive adequate attention. Stakeholders may not ask for these explicitly, but these have a high impact on the usability of the product.

We have developed Usability Goals Achievement Metric (UGAM), a goal-based, technique neutral, flexible summary metric. UGAM is calculated by a weighted average of goal parameter scores. There are several usability metrics in literature. In addition, many authors have emphasised the importance of usability goals. However, we have not come across a usability metric that emphasises goal achievement and gives the development team control to add, delete, or modify usability goals depending on the context.

With an experiment involving evaluation of 9 products by two groups of students, we have demonstrated that even in adverse conditions of less time, less exposure to usability, and no access to users, UGAM scores correlate with the ratings of evaluators after traditional usability studies.

We have developed Index of Integration (IoI), a process metric that evaluates the process followed in a project as compared to a prescribed process model. IoI is calculated by weighted average of scores of activities prescribed in a base process model. The weights and activity evaluation guidelines are recommended by the process prescriber and fine-tuned by the organisation and the project team adapting the process model.

In our literature search in the HCI or SE fields, we have not come across any method to validate a process model. This research has demonstrated a way to evaluate the impact of adherence to our prescribed process models on usability goals achievement. By analysing data from 61 projects from the industry, we have established that there is a strong positive correlation between UGAM and IoI, where IoI was calculated by using our process model proposals as the base. We found that the relationship between IoI and UGAM is linear (UGAM = 0.60*IoI + 30.8). The correlation between IoI and UGAM validates our extensions for integrating of HCI activities with waterfall and agile process models.

We found that the subset of projects following agile process models has a higher coefficient of IoI (UGAM = ***0.69****IoI + 29.3) compared to that of the subset of projects following waterfall process models (UGAM = ***0.58****IoI + 31.0). This indicates that at similar levels of process integration, projects following agile process models could result in more usable products than those following waterfall process model. Unfortunately, we could not collect data about the costs and efforts involved in this study. Assuming the cost and efforts of achieving high IoI in the two process models is similar, agile process models would return better usability as compared to the waterfall process model at similar levels IoI.

We also found that the constant is higher and the coefficient of IoI is lower for the subset of product company projects (UGAM = ***0.45****IoI + **<u>43.8</u>**) compared to the subset of services company projects (UGAM = ***0.58****IoI + **<u>32.2</u>**). This indicates that integration of HCI activities could be less important in product companies as they achieve higher UGAM levels even at low IoI levels, perhaps because of factors other than the process play an important role in product companies (such as high stakeholder involvement, immediacy of business impact of poor usability, or simply higher levels of skills, talent, and experience). This also indicates that service companies risk shipping unusable products more than product companies do, unless they achieve higher IoI.

In our study, the correlation between IoI and the product metrics was better than the correlation between HCI practitioners' judgement of the quality of the process followed and the product metrics. Similarly, the correlation between UGAM and the process metrics was better than the correlation between HCI practitioners' judgement of the user experience delivered and the process metrics. Among all process metric - product metric pairs, IoI-UGAM correlation was the strongest, with IoI predicting 56% of the variation in UGAM. We can conclude that among the metrics tested, UGAM and IoI are better than the HCI practitioner's judgements for process quality followed and user experience delivered.

There has been no method to establish the differential contributions of HCI activities in a process. This research has demonstrated that different HCI activities have a different impact on usability goals achievement in the waterfall model. Using regression analysis, we have identified four HCI activities in the waterfall model that make the biggest impact and account for 58% of the variation in UGAM. These activities are user studies, user interface design, usability evaluation of the finished user interface, and development support. It is important to note that development support activity makes a bigger impact than expected. After the interfaces are evaluated and delivered, project managers could plan for ongoing interaction between HCI practitioners and software developers during development.

The two metrics UGAM and IoI have their limitations that we acknowledge. As discussed in chapter 10, there are arguments in literature against summary measures. Some data is always lost in such measures, and one should not try to read too much in them. Yet, decision makers need measures to make decisions, and while this could be a risk in the context of usability, it may be an opportunity. Summary measures are particularly useful for comparisons across projects. Such comparisons can help the team understand what works and what does not and learn from experience.

UGAM and IoI have an organizational perspective and allow comparison across projects. UGAM and IoI are flexible and are based on weighted averaging of self-set targets. This ensures that important issues in the project count for more. Further, UGAM and IoI retain more information than just a summary value. Both create a profile and allow a drill-down to constituent components, which point to specific areas where corrective action may be taken.

Perhaps an important limitation of UGAM comes from the ephemeral nature of a user's experience. Any attempt to embody such an abstract phenomenon numerically is bound to be subjective. In spite of this limitation, we believe that UGAM can be useful. UGAM is not meant to measure the entire abstract notion of user experience, but the extent to which the HCI practitioners are achieving the goals they set. The HCI practitioners set goals and evaluate products according to their understanding of the context, users, and business goals. If their judgements (or judgements of the stakeholders who influence them) are accurate, UGAM will be a reasonable measure of usability goal achievement. We found that breaking up notions of usability into medium-granularity goals parameters helped evaluators focus on one issue at a time. The medium-granularity goal parameters and explicit evaluation criteria helped reduce subjectivity. Linking parameter scores to performance metrics and averaging across several evaluators can further reduce subjectivity.

Calculation of UGAM could be a part of every usability evaluation of a project. The UGAM scores of intermediate formative usability evaluations can help track the progress of the project. The UGAM score from the final evaluation can establish whether the product meets quality criteria or not. Service companies can use UGAM to assure customers a minimum level of usability goals achievement. Product companies can use UGAM to track usability across versions. Managers can use UGAM to track the performance of several projects at a time, to identify poorly performing projects early, and to plan corrective action.

An important limitation of IoI is that it does not measure the absolute process quality, but the extent to which the project follows the prescribed process. There are no widely accepted process models that integrate HCI with SE processes today. In conjunction with

UGAM, and other product metrics, IoI may be used to verify the effectiveness of new and current process model proposals (as we have demonstrated).

HCI practitioners can use IoI to evaluate the fidelity to a prescribed process model. Service companies can use IoI to assure customers a level of process compliance. Customers can specify a minimum IoI requirement against a specified process model in their request for proposals. Vendors can benchmark IoI in its previous projects against that process model and use historical data to prepare estimates for the new project.

13.2 Areas of Future Research

We have demonstrated a way to evaluate the impact of two process models (our extensions of waterfall and agile processes) on product metrics. Future research could use this approach to establish the effectiveness of other process models (such as our extension of the RUP).

IoI affects UGAM more in agile projects as compared to waterfall projects and in service company projects as compared to product company projects. Using a similar approach, future research could analyse other subsets of interest such as products targeted to specific user profiles (e.g. illiterate users), products on specific platforms (e.g. mobile phones), or products in specific domains (e.g. healthcare).

Our research showed that IoI explains 54% to 67% of variation in UGAM. Future research could investigate the effect of factors such as HCI practitioner experience, skill, and creativity and organisational maturity. These factors make a significant impact on the quality of HCI activities and ultimately on the quality of user experience delivered.

We found that 4 HCI activities have the highest impact on usability goals achievement in the waterfall model. Future studies focussing on specific contexts can be run on similar lines to identify high-impact HCI activities and methods in those contexts.

37% of high-weighted goal parameters were not achieved and the average UGAM score was 65.8. Similarly, average IoI scores were 58.4. In our studies, we collected data retrospectively, i.e. after the projects were over. In future, it will be interesting to study whether prospective use of tools such as UGT, UGAM, and IoI help the team achieve its goals better.

We found that high-weighted usability goal parameters tend to have low weight-score correlations. The exact causes of this phenomenon could be a matter of future investigation.

We analysed data manually to identify patterns in user profiles, product profiles, and goal parameter weights. Quantitative explorations indicate that some user and product profile variables correlate with the weights assigned to of some goal parameters. Logistic regressions with current data could predict weights of 15 goal parameters with 75%+ accuracy. A future study with a larger sample size can perhaps increase the confidence, accuracy, and coverage of such predictions.

Appendix I: Usability Goals Setting Tool

(This tool has been reproduced here for easy reference. It can also be viewed at http://www.idc.iitb.ac.in/~anirudha/ugt.htm.)

A goal is a commitment that a designer makes to his client or his management. Setting goals before design gives the team a target to achieve. Goals help guide the design process, make the design activity tangible, and help evaluate the design. Setting goals early and getting an agreement from stakeholders is important in the design process.

Unfortunately, design teams are not always clear about the goals of the product they are designing. Sometimes some team members may be clear, but others in the team may not be on the same page. Teams often need help in setting usability goals.

This document describes Usability Goal Setting Tool (UGT). This tool was developed to help HCI practitioners set usability goals and evaluate alternative designs against them. By HCI practitioners, we mean interaction designers, usability experts, user interface designers, information architects, or anyone else who is involved in design and evaluation of interactive products. In this document, we call anyone who uses UGT as a goal setter.

In this document, we differentiate between *user goals* (*"enjoy a peaceful vacation"* or *"ensure a secure future"*), *business goals* (*"capture the youth market"*, or *"save costs by pushing sales online"*) and *product goals* (*"the website will support planning and having a vacation end-to-end"* or *"the EVM should enable all users to vote without external help"*). Here, we focus on product goals. Further, we focus on the subset of product goals related to the usability and user experience of the product.

How to Use this Tool?

UGT helps goal setters specify high-level usability goals and break them down into concrete, measurable goal parameters. Before starting to use UGT, the goal setter should have a clear product brief and should have developed sufficient understanding about the domain, the problems, the context, and the users.

There are four steps in UGT: initiate, set goals, review goals, set evaluation guidelines, and share.

Initiate

UGT begins by helping the goal setter create a product profile (product domain, platform, cost etc.) and one or more user profiles (age, technology savvyness, expected frequency of use etc.). The goal setter should fill out Forms 1 and 2 for this purpose. If the product is targeted to multiple user profiles, the goal setter should fill out a copy of Form 2 for each of them.

Set goals

A design appropriate for one context may be very inappropriate for another. Depending on the context or the targeted user profile, some goals may have high priority, while others might be irrelevant.

UGT presents a list of 30 goal parameters in Form 3. The goal setter should review this list and identify a priority for each. In UGT, the goal setter expresses priority to a goal parameter by assigning it a weight from 0 to 5. The meanings of the weights are:

- **Unique selling point** (weight 5): This goal parameter is the unique selling proposition of the product. This goal parameter is among the top 3-4 user experience qualities of this product. The success of this product depends on doing well on this parameter. The tasks it affects may be frequent, critical, or both. The design must be thoroughly evaluated against this goal parameter.
- **Very important** (weight 4): This goal parameter is one of the top 6-7 user experience qualities of this product. The tasks it affects may be frequent, critical, or both. The design must be evaluated explicitly against this goal parameter before release.
- **Important** (weight 3): This parameter is important - among the top 10-12 user experience qualities of this product. The tasks it affects may be either frequent or critical. The product should be evaluated against this goal parameter explicitly unless the evaluators are convinced that the outcome would be positive. In any case, evaluators must report any adverse findings related to these goal parameters.
- **Usual relevance** (weight 2): This goal parameter is of the usual, vanilla relevance. This goal is met commonly and easily in products of this kind. It is not among the top user experience qualities of this product – not meeting this goal would not be a showstopper. The tasks it affects are neither frequent nor critical. The evaluation of this goal parameter is discretionary – evaluators may choose to skip evaluating the design against this goal parameter. However, if they do come across any issues related to this goal parameter, they should report them.
- **Somewhat relevant** (weight 1): This goal parameter is only somewhat relevant to the product. The tasks it affects are neither frequent nor critical. It is unlikely that the product is evaluated against this goal parameter.
- **Irrelevant** (weight 0): The goal parameter is not relevant to the product.

To help the goal setter assign weights, this document explains each goal parameter listed in form 3, gives examples where the goal parameter may be relevant, and makes recommendation for their weights for different product and user contexts based on experience in past projects. (In the automated version of the UGT, inputs from Forms 1 and 2 would be used to generate a set of recommended weights. In this paper version, Forms 1 and 2 serve as a reminder and as a means of looking up the recommendations.)

The list of goal parameters suggested in UGT is broad-based and should apply in many situations. However, it is not supposed to be comprehensive and may not cover every aspect of user experience that may be relevant to each project. The goal setter has the freedom edit or re-word the suggested goal parameter, to group goal parameters differently, and to define new goal parameters. Similarly, the goal setter has the freedom to assign a different priority than the one suggested here.

Please note that if the product is targeted to multiple user profiles, user experience goals for each user profile will be somewhat different – so the goal setter will need to fill out separate forms for each of them.

Review goals and set evaluation guidelines

After setting weights of goal parameters, it is important to review them. In the first pass, there is a tendency among goal setters to over-assign priority. As rule of thumb, goal setters should assign weight of 3 or more to less than 10-12 goal parameters.

The goal setter can take an average of the weights that he has assigned to check against these rules of thumb:

- If the average weight assigned is less than 1.9, the goal setter has perhaps assigned weights too low. During the review, the goal setter should consider giving higher weight to some goal parameters – push the 3s to 4s and 4s to 5s.
- If the average weight assigned is between 1.9 to 2.4, the goal setter has been on the lower side, but not by too much. He has the room to move up a bit if desired, though he does not have to.
- If the average weight assigned is between 2.4 to 3.4, the goal setter has been balanced. During the review, the goal setter should ensure that the relative weights of goal parameters are right.
- If the average weight assigned is between 3.4 to 3.9, the goal setter seems to think that every goal parameter is important (average weight = 3), but it could be the case in some projects. The goal setter may consider pushing weights of some goal parameters, but he does not have to.
- If the average weight assigned is more than 3.9, the goal setter has perhaps assigned weights too high. During the review, the goal setter should consider giving lower weight to some goal parameters – push the 5s to 4s and 4s to 3s.

During the review, the goal setter should write a justification to explain the priority of each goal parameter in the context of the project. The goal setter could also suggest how the design may be evaluated against each goal parameter. Some evaluation ideas are suggested along with the description of each goal parameter in this document.

Share

The goal setter should share or present the goal parameter priorities, the justification for the priorities in the context of the project, and suggestions for evaluating the design against each goal parameter to all stakeholders. A round of discussion and negotiations could follow. This will help bring the stakeholders on the same page about what they are trying to build. (The online version of UGT may have features to share and negotiate goals online. For now, this could be done by sharing the forms on emails.)

Form 1: Product Profile

A brief description of the product:

Business goals:

What is the current version of the product?	1st / 2nd / 3rd / 4th . . . . . / nth
What is the industry domain of the product?	
What is the work practice domain of the product? (Pick all that apply.)	❑ Life critical ❑ Business critical ❑ Goal oriented ❑ Learning ❑ Information ❑ Casual ❑ Gaming ❑ Enabling technology for disabled
What is the cost positioning of the product to the end user compared to other products in its category? (Pick one.)	❍ Premium product ❍ Low end product ❍ Not free, but I don't pay ❍ Free with other product or service ❍ Free to use
What platform(s) does the product use? (Pick all that apply.)	❑ Desktop ❑ Web ❑ Mobile ❑ IVR ❑ TV ❑ Custom

How many user profiles will use the product? Give a short (2-3 word) persona description for each profile and fill out user profile document for each profile.

User Profile 1:	Persona 1:
User Profile 2:	Persona 2:
User Profile 3:	Persona 3:

Form 2: User Profile

Please fill this document separately for each user profile. We will use the information in this form to recruit appropriate users and to design a usability test.

User Profile: ____________, Persona: ___________

What is the range of age targeted?	
What is the **lowest** level of tech-savvyness expected among the targeted users? (Pick the lowest level.)	❍ Like a programmer who loves Linux ❍ Like a tech-savvy manager who can install a piece of software from a CD ❍ Like an accountant who checks mail every day ❍ Like an office peon who can save contacts on a phone ❍ Like a grandmother who cannot save contacts on a phone
What is the expected frequency of use? (Pick one. If different people within the target are expected to use the product at different frequencies, pick a range.)	❍ Once in a lifetime ❍ One a year ❍ Once every month ❍ Once every week ❍ Daily ❍ A few times a day ❍ Continuously (>10 times a day)
What is the expected complexity of the product? (Pick one. The numbers in the brackets denotes number of screen types, tasks, or states that these users encounter with the product. If the product has several user profiles, count only those parts that this user profile is likely to use.)	❍ Very Complex (50+) ❍ Complex (30-50) ❍ Moderate (11-30) ❍ Simple (4-10) ❍ Very simple (3 or less)
What is the nature of the market? (Look at the examples and pick the closest option.)	❍ Mass market (everyone in the society – e.g. electronic voting machine) ❍ Wide market (large sections consisting of many user profiles – lawyers and doctors and the college youth) ❍ Niche market (a specific group of users, usually determined by a profession or lifestyle – farmers or gamers or the elderly) ❍ Internal application for trained staff (intranet) ❍ Thick domain product for experts (requires not only training but also expertise – radiologists)
What value does the product add to the user? Why would the user use the product? What is the motivation? (Pick all that apply, but be critical.)	❑ I have no intrinsic motivation to use it ❑ I have options, including doing things manually ❑ To socialize ❑ It gives me a cool lifestyle ❑ It entertains me ❑ I have been using it, I am used to it ❑ It informs me ❑ It makes things convenient, saves my time, helps me do my job better ❑ It makes things accessible ❑ It makes money for me / it saves me money ❑ I have no choice, I have to use it

Describe any of the following aspects of the targeted users that are relevant:

- Demographic variables – profession(s), education, gender, income etc.
- Inclusion criteria – any requirements that qualifies a targeted user (e.g. bank account, salaried job, net connection at home)
- Exclusion criteria – any situations that disqualifies them from using the product (e.g. no mobile phone, no internet access, below 18 years of age)

Form 3: Goal Parameters and Weights

	0	1	2	3	4	5
Learnability						
Findability: options / data / information should be visible / easy to find						
User should take less time to learn: (e.g. in < 10 minutes, in < 2 hours practice, in < 2nd attempt)						
Users should be able to learn on their own						
Product should be internally consistent						
Product should be consistent with other products, older methods / past habits of users						
Product should be consistent with the earlier version						
User should remember / retain critical, but infrequent tasks						
Speed of use						
User must be able to do the primary task / the most frequent tasks quickly, easily, at all times						
User should be able to navigate quickly and easily						
Product should not load user's memory / product should not put cognitive load on a user						
Flexibility: user should control the sequence of tasks						
User should be able to complete frequent / critical tasks in specific time / no. of steps / in less efforts						
Product should be personalised for the user automatically						
Product should be localised for specific market segments						
A user should be able to customise the product for himself						
Ease of use						
Interface should clearly communicate the conceptual model						
Intuitiveness: User should be able to predict the next step / task						
No entry barrier: user must be able to complete critical tasks						
Product should require no unnecessary tasks						
Product should automate routine tasks / minimise user task load						
Product should be always on, always accessible						
Ease of communication						
Information architecture: Information should be well aggregated, well categorised, well presented						
Communication should be clear / user should easily understand text, visuals						
Error-free use						
Product should give good feedback / display its current status						
Product should not induce errors						
Product should tolerate user's errors / forgiving interface / should prevent errors						
Product should help user recover from errors / help users troubleshoot problems						
Subjective satisfaction						
User should feel in control of the product / behavioural appeal						
User should feel emotionally engaged with product / brand / product should be fun / reflective appeal						
User should find the product aesthetically appealing / product should have a visceral appeal						
Other product goals						

Guidelines for Assigning Goal Parameter Weights and Evaluation

1. *Findability*

What is it?

Users should find it easy to find menu options, buttons, links, information etc. If these things are easy to find, learning to use the product becomes easy. The information pyramid should be inverted, bringing up as much information as possible without overwhelming the user. The things that users need frequently or in critical situations should be visible up-front, on the home page or in the main menu. Things that are not used often should still be placed in a location that is predictable and easy to find. Screens should not be cluttered and important elements should catch the users' attention and not be lost.

How to assign weights?

- **Assign weight 4-5** if the product is targeted to low-tech users, older users, or if the application complex, business-critical or life-critical.
- **Assign weight of 2-3** if the product is meant for frequent use (as users will get used to it after a while or learn shortcuts, though make sure that the frequent tasks are easy to access), or if the product is extremely simple and no findability issues are expected.
- Usually, findability is assigned a weight of 2 or more – it is almost always relevant.

In a study with 65 industrial projects, participants assigned an average weight of **4.00** (standard deviation **0.83**) for this goal parameter.

How to evaluate this goal parameter?

- **In a usability test**, ask users to do tasks that require them to find information or menu options. Ask them to think aloud while doing tasks. Look for confusions and unmet expectations.
- **In a performance test**, give users a goal to achieve that requires them to locate options in different parts of the product. Count errors or the amount of time it takes them to do so.
- **During a review**, look for data / options that are not organized consistently with the users' mental models. Findability problems can be hard to spot unless the reviewers are familiar with the users' abilities and mental models.

2. *User should take less time to learn*

What is it?

Users mentally assign a time limit for learning a product in a given category. A new driver expects to learn to drive a car in two weeks. A tech-savvy college youth expects to figure out the options in his a new mobile phone in an hour since purchasing it. A frequent flier hopes to figure out how to book an airline ticket in less than ten minutes on a new airline. For a ticket vending machine at the railway station, it should take less than a minute for a first-time visitor to buy his ticket. In some products, it is very important that the product can be learnt within this time limit. In other products, it is relatively less important.

How to assign weights?

- **Assign weight 4-5** if the product is business critical or an enabling technology in a public place and it is meant to be used by users without supervision
- **Assign weight of at least 3** if the product is targeted to low-tech savvy users, and even for users who are medium-tech savvy if the product is complex.
- **Assign weight of at least 2** if the product is not time-critical e.g. an entertainment product or a product to be used in a social context, where peers are present to help. Sometimes products targeted to captive audience are also assigned a low weight (e.g. users have no choice, but to use it such as intranet applications for employees), though this may not be a wise strategy in all cases.
- Usually, less time to learn is assigned a weight of 2 or more.

In a study with 65 industrial projects, participants assigned an average weight of **3.06** (standard deviation **1.13**) for this goal parameter.

How to evaluate this goal parameter?

- **In a usability test**, after a predetermined demonstration or a training session (the duration of which needs to be determined) and perhaps after a practice session, ask the user to do benchmark tasks. Observe user errors, confusions, frustrations, and difficulties. Estimate the time the user takes before he is confident using the product.
- **In a review**, invite new evaluators each time and evaluate the time they take to learn to use the product. Even so, users' learning time is likely to be underestimated during a review.

3. *Users should learn to use the product on their own*

What is it?

In many cases, we cannot put a user through a training programme and users must figure out how to use the product on their own. At times people may have a general understanding or expectation of how this product works (e.g. a home accounting application). At other times, the product concept may be completely new to them (e.g. the first buyer of the Nintendo Wii in a neighbourhood in 2008). Internal users can be usually trained to use the product (e.g. call centre applications), but such a training programme may not cover all situations and there would be a need to learn a few things on the job.

How to assign weights?

- **Assign weight of 5** if the product is to be used in an emergency (like the emergency door of the airplane).
- **Assign weight 3-4** if the product is targeted to medium to low tech-savvy users particularly if these users may be alone while using it for the first time. Also if the users will be paying for this product themselves (expectations would be high), if the users are likely to be impatient to learn the product, or if it is a learning product with high cognitive load.
- **Assign weight of 0-1** for products meant to be learnt through a training programme only (e.g. an internal call centre application, a critical task product such as a healthcare product in a ICU to be used by a trained nurse, or an exceptionally complex, frequent use product like an electrical power plant) or if the product is simple to use or the tasks are casual and users are likely to learn to use this product from a friend (e.g. entertainment and socializing products at a party).
- **Assign weight of 2** for most other products.

In a study with 65 industrial projects, participants assigned an average weight of **2.83** (standard deviation **1.40**) for this goal parameter.

How to evaluate this goal parameter?

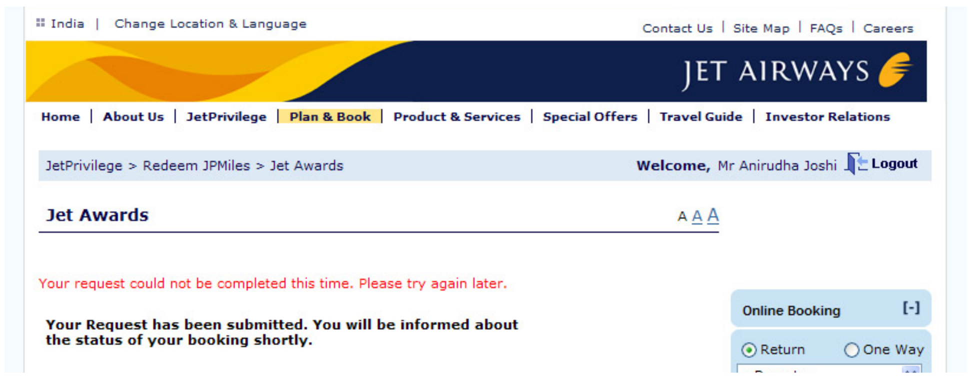
- **In a usability test**, give the users a goal to achieve and ask them to try to use the product on their own to figure out if they can use it.
- **During a review**, invite new reviewers (who are using the product for the first time) and pay particular attention to problems faced by the new reviewers.

4. *Product should be internally consistent*

What is it?

Internal consistency is very important. It is easiest to understand it by looking at examples of internal inconsistency.

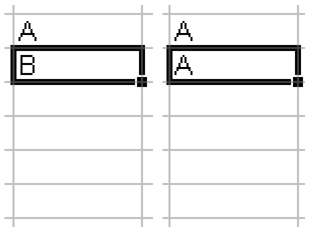
Look at the following messages (one in red, the other in black) that came when the user was trying to get an award ticket on Jet Airways.


India | Change Location & Language
Contact Us | Site Map | FAQs | Careers
JET AIRWAYS
Home | About Us | JetPrivilege | Plan & Book | Product & Services | Special Offers | Travel Guide | Investor Relations
JetPrivilege > Redeem JPMiles > Jet Awards
Welcome, Mr Anirudha Joshi
Logout
Jet Awards
A A A
Online Booking [-]
Return
One Way

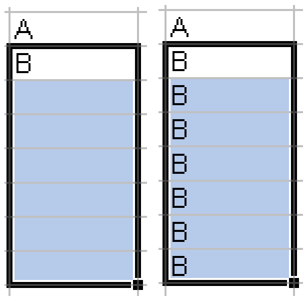
Your request could not be completed this time. Please try again later.

Your Request has been submitted. You will be informed about the status of your booking shortly.

Another example comes from Microsoft Excel. If the user types "A" in one cell and "B" in the cell below, then selects the cell with B in it and type Control + D, "A" gets copied over in the cell where "B" was earlier, as shown in the image below:



On the other hand, instead of selecting only the cell with B in it, if the user selects the cell with B in it, and a few more cells below it before typing Control + D, now "B" gets copied over as shown in the image below:



Users may tolerate minor inconsistencies in a non-critical task. However, if there are too many inconsistencies users can get frustrated and confused and will give up experimenting with the product. This increases errors and hampers learnability.

How to assign weights?

- **Assign weight 4-5** if the product is life-critical, business-critical or a premium product, or if it is targeted to older or low-tech savvy users.
- **Assign weight 2-3** for most other products.
- Most products will not have a weight of less than 2 for internal consistency.

In a study with 65 industrial projects, participants assigned an average weight of **3.05** (standard deviation **1.16**) for this goal parameter.

How to evaluate this goal parameter?

- **In a usability test**, ask users to perform a set of tasks and probe them to see if there are internal inconsistencies that cause confusion in the users' minds.
- **During a review**, inspect the interface of the product and look for internal inconsistencies.

5. *Product should be consistent with other products / past habits of users*

What is it?

People sometimes have deep-rooted habits and often transfer them from products that they have been using in the past. For example, savings bank account holders routinely scan their passbooks, which are designed in a particular format. If the ATM transaction slips are laid out in a similar format, it will become a lot easier for new users to read them. In the 1980s, Apple Macintosh standardised interface elements across applications such as menus (File, Edit, View, Options etc.), sub-menus (File menu has New, Open, Save options; Edit menu has Undo, Cut, Copy, Paste options) and keyboard shortcuts (Command + S, Command + Z, X, C, V). This was an important reason of the success of the Macintosh, and later the applications running on the Windows platforms (which had similar interfaces). These menu options are still to be found in most applications after 25 years, including some of the web applications.

How to assign weights?

- **Assign weight 3-5** for products expected to have a wide frequency range (infrequent to frequent), or if the product is targeted to higher age groups (50+) or to low-tech savvy users. These users hate changing their habits.
- **Assign weight 1-2** only if there is a high motivation for users to change their habits (e.g. if the product saves users' money and time, or if the product is to be used continuously, and a small change in habits can give many long-term gains), or if the product is meant to be strategically differentiated from earlier products.
- **Assign weight of 0** if the product is for young children – users who may have very few past habits.

In a study with 65 industrial projects, participants assigned an average weight of **2.75** (standard deviation **1.41**) for this goal parameter.

How to evaluate this goal parameter?

- **In a usability test**, ask users to perform a set of tasks and probe for inconsistencies with what people do in real world or older habits.
- **In a field study**, introduce the product in the life of a few users and see where it contradicts past habits in real life usage.
- **During a review**, look for design elements in the product that might be inconsistent with users' current habits. These problems are hard to spot in a review unless the reviewers have intimate contextual knowledge about the users' current habits.

6. *Product should be consistent with earlier the version*

What is it?

Loyal users turned their backs on some products when they launched a new version that was drastically different from their older version.

Autodesk Animator Pro (a DOS based application) was a market leader in 2-D animation until the mid-1990s. The product had a different user interface than the typical desktop applications of the time, and users who tried to learn on their own did not do that well. Nevertheless, the product had a loyal base of graphic designers who had invested time in learning to use it. Then Animator Pro for Windows was launched which tried to bring the interface closer to Windows applications, and the product died as competitive products took over.

Netscape Navigator was a free browser with 90% market share. Unfortunately, it regularly *"expired"* its installation and forced the users to download newer versions. What was worse – they changed the menu hierarchy in almost every version. Competitive products squeezed out the product in no time.

Microsoft Office made a big shift from the Apple menu structure when it moved from Office 2003 to Office 2007, and it was a painful first step for many users. MS Office survives, perhaps because competitors did not seize on the opportunity. It was a risky move that Microsoft could pull off. However, Microsoft was careful enough to carry over all keyboard shortcuts from the previous version into the new version.

How to assign weights?

- **Assign weight 4-5** if the current version of the product already has an established user base that will carry forward to the new version, or if the product supports critical or infrequent tasks.
- **Assign weight 2-3** if the current version of the product has many usability complaints from the current users or if the current user base is smaller than the one targeted with the new version.
- **Assign weight 0-1** if it is the first version of the product, or in cases where users have a very high motivation to switch to the new version (e.g. it makes things accessible in a way they were not before).

In a study with 65 industrial projects, participants assigned an average weight of **1.32** (standard deviation **1.75**) for this goal parameter.

How to evaluate this goal parameter?

- **In a usability test**, ask users of earlier version to do critical tasks with the new version. Ask users to think-aloud, and look for any problems, hesitations, confusions etc.
- **In a performance test**, look for problems that occur repeatedly (e.g. use of wrong keyboard shortcut or looking for items in the wrong menu.)
- **During a review**, inspect the earlier version first and look for inconsistencies that have crept in the newer version.

7. *User should remember / retain critical, but infrequent tasks*

What is it?

Many users may use the product intermittently. Some products are meant to be used intermittently (e.g. booking an airline ticket, filing a tax return, restoring lost data). Parts of an otherwise regularly used product may be used intermittently (e.g. making a PDF, doing a mail-merge, changing the IP address). Often, these intermittent tasks are also critical. Users can forget how they used the product last time (sometimes in a matter of hours) and may need to relearn it all over again. In such cases, the interface can be designed with cues to remind the user so that he can pick up the paths again.

How to assign weights?

- **Assign weight 4-5** if the product has critical, infrequent tasks (anything less frequent than once a week), or if the product is targeted to low tech-savvy users.
- **Assign weight 3** or more if the product is complex or for internal use, as such products often have some critical infrequent tasks.
- **Assign weight 0-1** if the product is one-time use, or if the product has no critical tasks (e.g. information-only web sites that would be accessed by users only once).
- **Assign weight 2** in most other cases.

In a study with 65 industrial projects, participants assigned an average weight of **2.31** (standard deviation **1.52**) for this goal parameter.

How to evaluate this goal parameter?

- **In a usability test**, train the users on how to do some tasks. Ask the user to repeat the task after a gap of hours / days / weeks. Look for elements that users have forgotten or are difficult to re-learn. If the product is to be used less frequently than once a month, thoroughly test first-time usability. Include time to learn in the task completion time in such cases.
- **During a review**, look for memory clues and elements that might be confused with each other. These problems are hard to spot in a review.

8. *User must be able to do primary tasks quickly, at all times*

What is it?

Add a computer to something, and chances are that it becomes harder to use. Computers allow us to add many useful features to products, but sometimes these features get added at the cost of the primary or the frequent tasks. Traditional landline phones could be cut off and stowed away just by putting them down. With the mobile phones, the users need to deliberately press the disconnect button and lock the keypad. When the users switched on the traditional cable TV, a TV channel started and the user were able to switch channels and change the volume. TVs connected to set top boxes show a complicated menu when switched on. The traditional film camera was always on, always accessible. A digital camera takes much longer to boot up, perhaps because it is loading features that the user does not need immediately.

How to assign weights?

- **Assign weight 3-5** if the product has one or two primary or frequent tasks, particularly if the users can do these tasks quickly in current products, or if the product is targeted to tech-savvy users (they are particularly keen on speed). If a product is to be used continuously, its most frequent task must be the quickest.
- **Assign weight 0-1** if the product is being used very infrequently (e.g. once a year), and where speed of use is not critical (e.g. exploratory toys for children, learning products or one-time use products).
- **Assign weight 2** in most other cases.

In a study with 65 industrial projects, participants assigned an average weight of **3.49** (standard deviation **1.28**) for this goal parameter.

How to evaluate this goal parameter?

- **In a field deployment**, look for primary and frequent tasks, and investigate if users take too long to do them or find them difficult to complete.
- **In a usability test**, it is difficult to evaluate this goal in the lab, unless the frequent tasks are already known. In that case, task completion times of those tasks.
- **In a review**, identify frequent tasks and evaluate if users are likely to take a long time or find difficult to do. If the product has a primary task, make sure that it can be done at all times.

9. *User should be able to navigate quickly and easily*

What is it?

Navigation refers to the mechanisms that allow the user to move from one part of the product to another. Typical navigation mechanisms on the web are left-navigation, top-navigation, menu-based navigation, and in-line links. Sometimes these elements become unnecessarily clunky – sub-menus take too long to pop-out, and hide off unnecessarily before the user has seen them. This can be a minor distraction in infrequent tasks, but can get on the user's nerves if he needs to use them frequently.

How to assign weights?

- **Assign weight 4-5** if the product needs to be used frequently, if it has critical tasks, if it is complex, or if it is targeted to an expert market. Products targeted to older people need to be smooth to operate as they may have lower levels of dexterity.
- **Assign weight 0-2** for products with few screens or in games where navigation has been deliberately slowed down. Products that have no menus or screens to navigate to can even have 0 weight.
- **Assign weight 3** in most other cases.

In a study with 65 industrial projects, participants assigned an average weight of **3.51** (standard deviation **1.11**) for this goal parameter. Although the average weight assigned was high, this was one of the parameters with a poor weight-score correlation in the study. Practitioners should pay special attention to this parameter as it could be a latent usability goal – stakeholders or users may not explicitly ask for it, but it could affect usability of the product substantially.

How to evaluate this goal parameter?

- **In a usability test**, observe if the users face any difficulty while navigating from one screen to another, while doing benchmark tasks.
- **During a review**, evaluate if any navigation takes too many steps or if the interface is clunky in its response. Compare multiple navigation options by using modelling with GOMS or KLM analysis.

10. *Product should not load user's memory / put cognitive load on user*

What is it?

The short-term memory of human beings can hold 7+/- 2 items at a time. Any time the product asks the user to input arbitrary information (such as PNR numbers or product activation codes) it taxes the user's short-term memory, even when such information is handy. If the user needs to pick out information from a cluttered screen (e.g. find the highest value in a randomly ordered list) it adds to his cognitive load. If the user needs to keep track of information that is changing too fast, or if the user is supposed to finish a critical task in a limited amount of time, this can add to the cognitive load.

How to assign weights?

- **Assign weight 3-5** if the product requires critical data entry (such as credit card details), provides critical information dynamically (a stock ticker), or is a frequent use application (e.g. a product for a call centre), or if the product needs to be used in an attention deficit situation (e.g. while driving a car or while giving a lecture).
- **Assign weight 0-2** if the product is simple and does not attract too much attention, or if it is a casual use product in a stress-free environment (such as a socialising or entertainment product).

In a study with 65 industrial projects, participants assigned an average weight of **3.05** (standard deviation **1.18**) for this goal parameter. Although the average weight assigned was high, this was one of the parameters with a poor weight-score correlation in the study. Practitioners should pay special attention to this parameter as it could be a latent usability goal – stakeholders or users may not explicitly ask for it, but it could affect usability of the product substantially.

How to evaluate this goal parameter?

- **During a think-aloud test**, probe hesitations. Probe when user stops articulating thoughts to find if there was an additional cognitive load on the user. Look for tasks where user needs to memorise things, where user prefers to write things down, or where users miss information that was presented.
- **During a review**, look for things that are likely to load the user's memory or points where important information is cluttered or difficult to understand.

11. Flexibility – User should control the sequence of tasks

What is it?

The computer is a tool in the hands of the user. The user, not the computer, should control the sequence of tasks – particularly if the user is an expert, tech-savvy user using the product frequently. On the other hand, too much flexibility could overwhelm a novice user – a step-by-step wizard is a typical interface where the novice is happy to relinquish control and speed to improve the ease of use.

How to assign weights?

- **Assign weight 3-4** if the product needs to be used frequently by tech-savvy, expert or niche market users. Products with life-critical, safety-critical, or business-critical tasks also tend to get higher rating. Rarely will this goal parameter get a weight of 5, unless users are very tech-savvy.
- **Assign weight 0-2** if it is a simple product, a casual use product, or a product to be used by low tech-savvy users.

In a study with 65 industrial projects, participants assigned an average weight of **2.48** (standard deviation **1.37**) for this goal parameter.

How to evaluate this goal parameter?

- **In a think-aloud test**, recruit users with varying needs. Probe to see if different users might have preferred another sequence than the one dictated by the product.
- **During a review**, play roles of different personas and investigate if the product is forcing sequence of tasks that may be inappropriate.

12. User should be able to complete tasks quickly, in time

What is it?

Steve Jobs reportedly told his engineers – *"In a few years, I bet five million people will be booting up their Macintoshes at least once a day. Well, let's say you can shave 10 seconds off of the boot time. Multiply that by five million users and that's 50 million seconds, every single day. Over a year, that's probably dozens of lifetimes. So if you make it boot ten seconds faster, you've saved a dozen lives. That's really worth it, don't you think?*"

We always have an idea about how long an activity should take. A web page should download in 15 seconds. It cannot take 15 minutes to complete one transaction at the ATM. Typing must be faster than writing on paper. The most frequent tasks must take the least amount of time, number of clicks, or efforts. In some products, there might be critical tasks that must be done quickly.

How to assign weights?

- **Assign weight 5** if the product is meant for frequent use, has repetitive tasks, or if the main promise is that the product saves time
- **Assign at least 3-4** if product targeted to tech savvy users, or is life-critical.
- **Assign weight 0-2** for most other products. It tends to be on the lower side if the product is not task oriented (e.g. a learning product) or if the user does not care about saving time.

In a study with 65 industrial projects, participants assigned an average weight of **3.18** (standard deviation **1.33**) for this goal parameter.

How to evaluate this goal parameter?

- **In a performance test**, measure task completion time. It is typical to benchmark speed of use against competitive products or previous version of the same product.
- **In a review**, look for number of steps involved, and whether each step is taking the user closer to achieving his goal. For detailed analysis, use KLM or GOMS.

13. *Product should be personalised for the user automatically*

What is it?

Products need to be automatically personalised for the user. Many popular products already do that. Gmail remembers all email addresses the user ever typed and lists them by the frequency of use. Chrome automatically lists the most frequently visited web pages in a new tab. PhotoShop remembers in which corner of the screen the user left the tools. Some phones automatically add all entries from the phonebook into the T9 dictionary. Some TV models automatically remember the volume setting of each channel.

How to assign weights?

- **Assign weight 3-4** for products targeted to a wide or mass market, if different users are expected to have different usage patterns, and particularly if the product is to be used frequently (e.g. a mobile phone). Rarely will a product have weightage 5 on this goal parameter, unless the product is about personalised service.
- **Assign weight 0-1** if all users are expected to behave exactly in the same way. If the product is to be used only once, or it has no possibility of personalisation this goal will be irrelevant.
- **Assign weight 2** in most other cases.

In a study with 65 industrial projects, participants assigned an average weight of **2.23** (standard deviation **1.70**) for this goal parameter.

How to evaluate this goal parameter?

- **In a field deployment**, probe contextually to detect opportunities for personalisation.
- **In a usability test**, ask users to do several tasks in different scenarios. Look for actions that the user does repeatedly to identify opportunities for improvement. Look for information or settings that the user needs to supply repeatedly (e.g. language setting on the ATM, address on an e-commerce web site).
- **In a review**, invite domain experts and representative users. Explicitly look for opportunities for personalisation that have been missed out.

14. *Product should be localised for specific market segments*

What is it?

Products must be localised for specific groups – geographical, linguistic, socio-cultural, or professional. Imagine meeting scheduler that displays only GMT, an English-only product for home use in the Japanese market, a calendar in India without marking the Diwali holidays, or a common accounting tool for doctors and plumbers. Typically, localisation of many products is limited to language translation of menus (and often, the translations are quite poor). However, localisation needs to go far beyond translation. Currencies and calendar holidays are obvious needs. Further, icons, metaphors, and aesthetics need subtler considerations from one culture to another. Social aspects (such as privacy expectations) or professional interests also vary a lot from one culture to another.

How to assign weights?

- **Assign weight 3-5** if the product is targeted to mass or wide markets, or if there are wide differences in usage across user segments within the target audience, but individual users within segments behave similarly. Web-based products and business critical products often require localisation.
- **Assign weight 0-1** if usage of products is expected to be quite similar across user segments – this could happen in internal applications for trained staff or thick domain applications to be used by experts.
- **Assign weight 2** in most other cases.

In a study with 65 industrial projects, participants assigned an average weight of **2.14** (standard deviation **1.79**) for this goal parameter.

How to evaluate this goal parameter?

- **In a field deployment**, do contextual inquiries to detect opportunities for localisation. Look for differences in usage (e.g. one user group uses one feature repeatedly, while the other one doesn't) and probe for reasons.
- **In usability tests**, invite multiple user groups to discover localisation opportunities.
- **In a review**, diverse, knowledgeable reviewers – make sure all user groups of interest are represented. Look for opportunities for localisation that have been missed out.

15. A user should be able to customise the product for himself

What is it?

It is important to start by saying that in most situations, a majority of users have no interest in customising the product. In order to be able to customise a product, the user needs to form a conceptual model of how the product works, and a majority of users cannot do that for a majority of products. Only tech-savvy users customise a frequent-use product after they have become competent performers. Among these, many customise only superficially (e.g. changing a wallpaper or a ring tone), usually in casual use products. A few do basic functional customisation (e.g. setting up shortcuts on the desktop, defining style sheets or changing the layout of the dashboard), and rare few do deeper customisation.

How to assign weights?

- **Assign weight between 2-4** if the product is complex, frequent use, or targeted to tech-savvy users, or if the product is for a casual use and has customisation opportunities. Rarely, if ever, this goal parameter is assigned a weight of 5.
- **Assign weight of 2** if the product is frequent use, but is either not complex or it is supposed to be used by non-tech savvy users.
- **Assign weight 0-1** in most other cases.

In a study with 65 industrial projects, participants assigned an average weight of **1.55** (standard deviation **1.73**) for this goal parameter.

How to evaluate this goal parameter?

- **In a field deployment** done on an early version of the product with tech-savvy users, do contextual interviews to ensure that all opportunities for customisation have been exploited. Also, ensure that unused customisation features are not coming in the way of other operation.
- **In a think-aloud test**, recruit competent performers. Give them customisation tasks to suit their own needs and probe confusions about the conceptual model clarity.
- **In a review**, invite competent performers on older versions or competing products to participate. Look for opportunities for customisation. Ensure that customisation features do not come in the way of other operation.

16. Interface should clearly communicate the conceptual model

What is it?

While a majority of users normally do not develop a conceptual model of the product, they need to have it to do new tasks, troubleshoot problems, or plan a complex series of tasks. The one thing the product's interface should surely not do is communicate the wrong conceptual model. Though it does sound like an obvious thing to say, many usability problems are found related to confusing interfaces communicating the wrong conceptual model.

How to assign weights?

- **Assign weight 3-5** for mass or wide market products, products targeted to low tech-savvy niche markets, critical products or products on new, custom platforms.
- **Assign weight of 2** only if the product is an internal application, the users are moderately tech-savvy, or a lot of product training is planned. Generally, this goal parameter is not assigned a weight of less than 2.

In a study with 65 industrial projects, participants assigned an average weight of **3.45** (standard deviation **1.17**) for this goal parameter. Although the average weight assigned was high, this was one of the parameters with a poor weight-score correlation in the study. Practitioners should pay special attention to this parameter as it could be a latent usability goal – stakeholders or users may not explicitly ask for it, but it could affect usability of the product substantially.

How to evaluate this goal parameter?

- **In a think-aloud test**, look for confusions caused by the lack of conceptual model clarity. During the test, find out whether users can do new tasks without training, recover from errors, and solve problems.
- **During a performance test**, find out how many users can complete tasks successfully, with or without help. Count errors caused by conceptual model problems.
- **In a review**, invite experienced HCI practitioners. Look for potential miscommunication of the conceptual model.

17. *Intuitiveness: user should be able to predict the next step*

What is it?

Users approach a product with some goals. Intuitive designs anticipate users' goals and provide an appropriate interface. With an intuitive product, users can easily predict what to do next, why to do it, and how to do it.

How to assign weights?

- **Assign weight 3-5** if the product is targeted to low tech-savvy users, or if the product is business critical or complex
- **Assign weight 2** in simple products or medium complexity products targeted to tech-savvy users. Generally, this goal parameter is not assigned a weight of less than 2.

In a study with 65 industrial projects, participants assigned an average weight of **3.12** (standard deviation **1.10**) for this goal parameter.

How to evaluate this goal parameter?

- **In a think aloud test**, probe users about their expectation of the next step, why, and how they would do it. Look for confusions / misunderstandings.
- **In a review**, invite experienced HCI practitioners. Look for steps that are not intuitive or are difficult to predict.

18. *No entry barrier: user must be able to complete critical first tasks*

What is it?

If the user cannot start using the product, he could end up not using it ever. An entry barrier is the crucial first step in product usage that prevents users from getting started. Some entry barriers are related to infrastructure (e.g. dependence on a particular browser or Flash version, having a particular type of mobile phone etc.). Other entry barriers are related to the procedure (e.g. long procedure to get an ATM card). Yet other entry barriers are plain old usability problems (e.g. user cannot navigate beyond the first menu, user did not know that he could touch a touch screen, or an airline that site requires the user to create an account before the he can check out the airfare).

How to assign weights?

- **Assign weight 3-5** if the users have a choice of not using the product (e.g. mass or wide market products), products with moderate or high complexity, products designed for improving accessibility, or products designed for saving time.
- **Assign weight 0-1** if the product is being used in a controlled environment, where infrastructural and procedural issues do not matter (e.g. a call centre) by trained staff (so first-time usability issues matter less), or if the product is a first-version proof of concept of a future technology.
- **Assign weight 2** most other cases.

In a study with 65 industrial projects, participants assigned an average weight of **2.80** (standard deviation **1.47**) for this goal parameter.

How to evaluate this goal parameter?

- **In a field deployment**, do contextual interviews with potential users who do not use the product to investigate if there are any entry barriers that prevented them. After an early version of the product is deployed, investigate why users choose not to use the product.
- **In a review**, invite HCI practitioners with experience in similar products. Compare the product with known entry barriers of other products of the same type.

19. Product should require no unnecessary tasks

What is it?

Though obviously not desirable, unnecessary tasks do slip in products often. Intranets ask their users to log in several times – each time in a different sub-system. Airlines ask users to be mindful of the web check-in rules (such as the time before departure when web check-in is allowed) *after* the user has input the PNR. Dialog boxes ask users to first click Apply, and then click OK. Unnecessary tasks are particularly undesirable when they hamper work frequently.

How to assign weights?

- **Assign weight 4-5** if the product is to be used frequently, and for products that are meant to save time, or if it is a work-related product targeted to busy people (including expert markets and internal staff).
- **Assign weight 0-1** rarely, only if the product is used very infrequently, or for non-critical tasks.
- **Assign weight 2-3** for most other products.

In a study with 65 industrial projects, participants assigned an average weight of **2.78** (standard deviation **1.08**) for this goal parameter. Although the average weight assigned was moderately high, this was one of the parameters with a poor weight-score correlation in the study. Practitioners should pay special attention to this parameter as it could be a latent usability goal – stakeholders or users may not explicitly ask for it, but it could affect usability of the product substantially.

How to evaluate this goal parameter?

- **In a field deployment**, evaluate in a contextual study whether any unnecessary tasks have crept in.
- **In a review**, look for any unnecessary steps. In a more rigorous review, model task as a sequence diagram and write user intents for each step. Identify tasks that have no underlying user intents. Write the task as an essential use case and compare it with actual use case.

20. Product should automate routine tasks / minimise user task load

What is it?

Machines are good at doing repetitive tasks while humans find them drudgery. In some cases, automation is the stated, high-level purpose (e.g. office automation, factory automation, or an automatic washing machine). In other cases, automation happens more subtly (e.g. autocorrect as you type, automatic alarm that reminds the patient to take a pill). Automation involves the risk of delegation – automate too much and the user may lose the feeling of control or get irritated (e.g. Microsoft's paperclip?).

How to assign weights?

- **Assign weight 4-5** if the product involves very high-frequency use (things done daily), has many routine tasks, or is targeted to tech-savvy users.
- **Assign weight 2-3** if the product involves low-frequency routine tasks (e.g. yearly tax payment reminder).
- **Assign weight 0-1** for products that do not involve a routine task. This goal parameter is not very relevant to information oriented websites, one-time use applications, and casual use applications.

In a study with 65 industrial projects, participants assigned an average weight of **2.51** (standard deviation **1.51**) for this goal parameter.

How to evaluate this goal parameter?

- **In a contextual interview**, investigate if there are any opportunities for automation of the routine tasks that are performed by the users today.
- **In a field deployment**, observe early users and investigate if any tasks need to be done routinely.
- **In a review**, look for tasks that might end up being done routinely (with or without the product) and can be potentially automated.

21. *Product should be always on, always accessible*

What is it?

Being always on and accessible could be crucial for some life-critical products – the pacemaker must be always on, so must be the oxygen supply in an ICU and the signal system in the railways. Business critical products too need to be always on, always accessible. If an airline website is not working for a day, the planes still need to take off but with fewer passengers. Similarly, for a taxi-driver, the mobile phone has become business-critical. Extending accessibility of the product may involve providing an extra battery pack, emergency talk-time when minutes run out, or synchronising the same calendar on multiple devices seamlessly.

How to assign weights?

- **Assign weight 5** for life-critical products or products that affect lives of millions of people in some way.
- **Assign weight 3-4** for frequent use business-critical products, infrequent but critical products, very popular websites, or products targeted to the "young and restless" generation.
- **Assign weight 0-1** for entertainment or casual use products.
- **Assign weight 2** in most other cases.

In a study with 65 industrial projects, participants assigned an average weight of **2.66** (standard deviation **1.70**) for this goal parameter.

How to evaluate this goal parameter?

- **In a field deployment**, interview users to investigate if the product was needed but was not accessible.
- **In a review**, invite domain experts and frequent users, and look for occasions when the product might not be on or accessible.

22. *Information architecture*

What is it?

Information should be well aggregated, categorised, and presented. Information architecture is very important in information-oriented products, particularly if large amount of information needs to be organised and presented to a diverse audience. The information architecture should reflect the thought-process or goals of the user (e.g. users look for birthday cakes and birthday candles in the same rack, a prospective student wants to choose a discipline in which to study at the university) or structures in the real world (e.g. business destinations, holiday destinations, places of pilgrimage) rather than internal organisation of code or structure of the organisation represented (e.g. visa department, foreign exchange department, train ticket department, air ticket department, hotel department etc.).

How to assign weights?

- **Assign weight 5** if there is a lot of, or complex information to communicate to a wide market (e.g. websites of a large organisations, intranets with many features, large ecommerce websites)
- **Assign weight 4-5** for large learning domain products with many choices (e.g. learning management systems, encyclopaedia), business-critical and goal-oriented products
- **Assign weight of at least 3** for most other products. Rarely will the weight for this parameter be below 3.

In a study with 65 industrial projects, participants assigned an average weight of **3.78** (standard deviation **0.98**) for this goal parameter. Although the average weight assigned was among the highest, this was one of the parameters with a poor weight-score correlation in the study. Practitioners should pay special attention to this parameter as it could be a latent usability goal – stakeholders or users may not explicitly ask for it, but it could affect usability of the product substantially.

How to evaluate this goal parameter?

- **In a usability test**, do a reverse card-sort and see if users classify information in the same way as the product.
- **During a think-aloud test**, probe for confusions, particularly if users are lost or if users look for data in the 'wrong' place.
- **During a performance test**, find out how many users can locate data, with and without help. Count errors induced due to information architecture problems.
- **In a review**, invite domain experts. Look for potential information architecture problems.

23. Communication should be clear

What is it?

The text should be well written, the labels should not be confusing, the graphics should be clear. Well-organised information architecture helps users reach the right page, but once the user has reached the right place, he should easily understand the content and make his decisions. The shopper should be able to choose his products; the prospective student and the tourist should be able to make up their mind; the learner should understand the concepts.

How to assign weights?

- **Assign weight 4-5** for business critical products, informative products, goal-oriented products, low frequency usage products and to products targeted to low tech-savvy users. Higher weight should go to a combination of criticality of the task, reduced attention span, and low education.
- **Assign weight 2-3** for most other products. Generally, this goal parameter is not assigned a weight of less than 2.

In a study with 65 industrial projects, participants assigned an average weight of **3.89** (standard deviation **0.89**) for this goal parameter. Although the average weight assigned was among the highest, this was one of the parameters with a poor weight-score correlation in the study. Practitioners should pay special attention to this parameter as it could be a latent usability goal – stakeholders or users may not explicitly ask for it, but it could affect usability of the product substantially.

How to evaluate this goal parameter?

- **In a think-aloud test**, probe for miscommunications of text and visuals. Probe to see if users are able to make 'correct' decisions.
- **In a review**, invite domain experts. Look for potential miscommunication of information.

24. Product should give good feedback / display its current status

What is it?

Giving appropriate feedback solves half the usability problems. At all times, users must know where they are, what they have been doing and what options they have. Users hate waiting, but when waiting is inevitable, wait cursors, rotating circles and percentage bars keep them pacified and tell them how long must they wait. Users are disturbed all the time so it is best to assume that users may not be paying attention to the product – users may not be looking at the screen when the feedback message "comes and goes". Feedback should be based on common conventions of a conversation. A modal dialog box is like a rude interruption – most of the times, feedback should be calm, though it should draw attention to itself when the house is on fire. Several modes can be used for feedback, including text, visuals, audio, and touch. Over time, good feedback helps the users build a conceptual model.

How to assign weights?

- **Assign weight 3-5** for goal-oriented products, business-critical products, learning products, wide or mass market products, products meant to be used in low-attention situations (e.g. products used on the move).
- Assign weight of at least **2** for all other products.

In a study with 65 industrial projects, participants assigned an average weight of **3.55** (standard deviation **0.97**) for this goal parameter. Although the average weight assigned was high, this was one of the parameters with a poor weight-score correlation in the study. Practitioners should pay special attention to this parameter as it could be a latent usability goal – stakeholders or users may not explicitly ask for it, but it could affect usability of the product substantially.

How to evaluate this goal parameter?

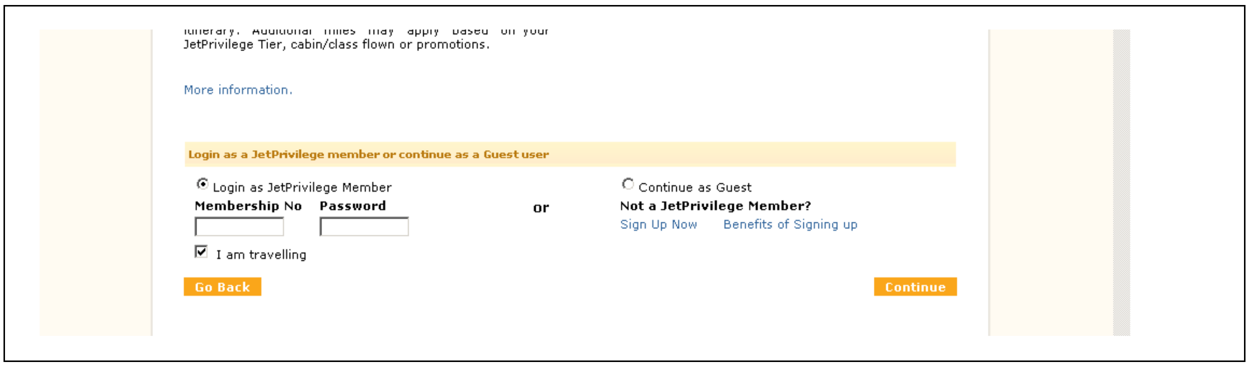
- **During a think-aloud test**, probe to find if users notice the feedback, understand it, and can tell the status correctly. In case of errors, observe if users notice the error and can decide the next course.
- **In a review**, look for potential feedback, error messages, and status display problems.

25. *Product should not induce errors*

What is it?

An error is an action of the user that does not achieve his goal. No one designs a product to induce errors, but flaws do slip in.

For example, in the Jet Airways booking page, if the user has not already logged in as a frequent flier member, the site gives you an opportunity to do so just after selecting flights. One thing that many users are used to doing after they enter their user ID and password is to click on the button below the two fields. In this case, if the user clicks on the button below the fields, it selects "Go Back" (which is the same as cancel) and throws the user back to the airline home page cancelling the transaction.



JetPrivilege Tier, cabin/class flown or promotions.

More information.

Login as a JetPrivilege member or continue as a Guest user

Login as JetPrivilege Member

Membership No Password

I am travelling

or

Continue as Guest

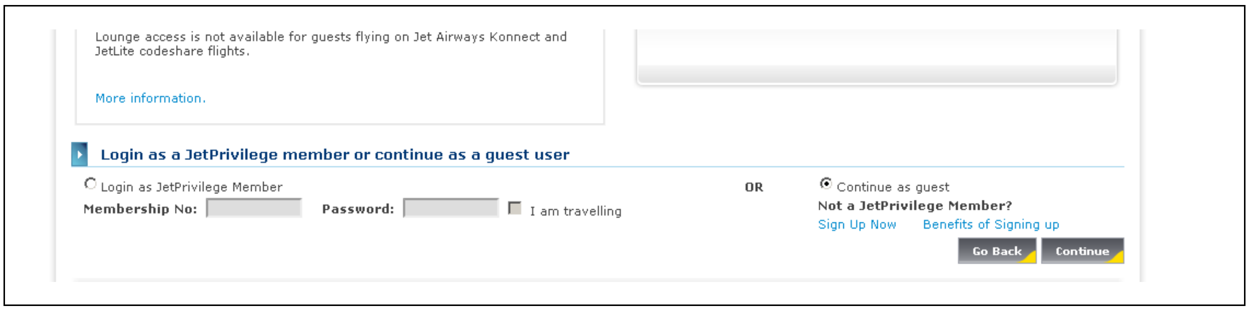
Not a JetPrivilege Member?

Sign Up Now Benefits of Signing up

Go Back

Continue

The above error can be easily fixed by a slight change in the layout as seen in the more recent version below.



Lounge access is not available for guests flying on Jet Airways Konnect and JetLite codeshare flights.

More information.

Login as a JetPrivilege member or continue as a guest user

Login as JetPrivilege Member

Membership No: Password: I am travelling

OR

Continue as guest

Not a JetPrivilege Member?

Sign Up Now Benefits of Signing up

Go Back Continue

Actually, the website does not need a Go Back button. All browsers come pre-installed with a back button. The Go Back button just induces an error unnecessarily. Non-tech savvy users are particularly susceptible to such errors.

How to assign weights?

- **Assign weight 3-5** if the users of the product are not very tech-savvy, or if the product involves life-critical, business-critical, or goal-oriented tasks.
- **Assign weight of 2** in most other cases. Generally, this goal parameter is not assigned a weight of less than 2.

In a study with 65 industrial projects, participants assigned an average weight of **3.03** (standard deviation **1.16**) for this goal parameter. Although the average weight assigned was high, this was one of the parameters with a poor weight-score correlation in the study. Practitioners should pay special attention to this parameter as it could be a latent usability goal – stakeholders or users may not explicitly ask for it, but it could affect usability of the product substantially.

How to evaluate this goal parameter?

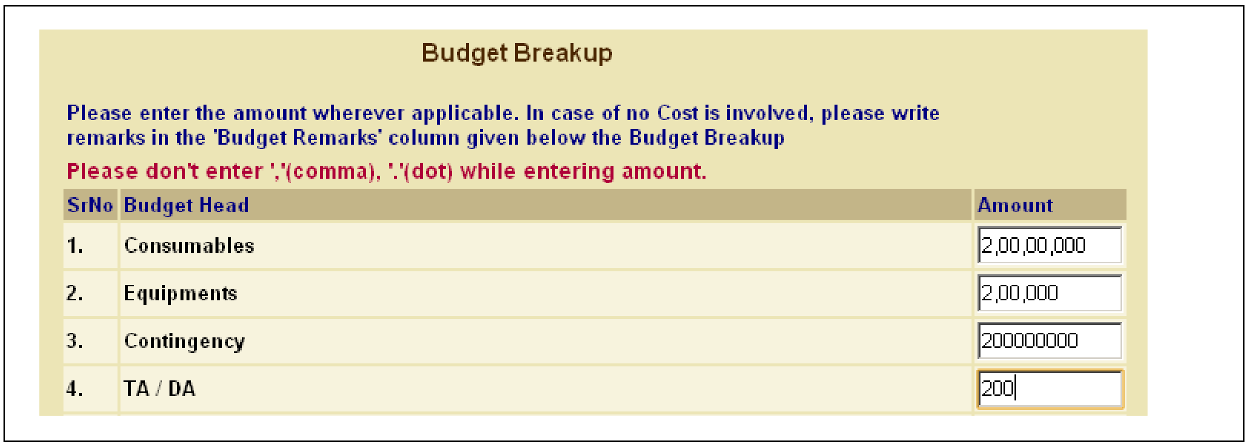
- **During a usability test**, ask users to perform typical tasks and observe the errors commonly made by users – see if any errors are induced by the design and seek out alternative designs that will avoid these errors.
- **In a review**, invite HCI practitioners with experience on a similar platform. Look for potential errors that could be induced because of the design.

26. *Product should prevent users from making errors, forgive / tolerate user errors*

What is it?

It will be nice if the product can prevent users from making errors. As per rules, the user can only withdraw an amount less than Rs 50,000 from an ATM each day. But the ATM interface allows the user to type in Rs. 50,00,00,000. It also allows the user to type in Rs. 503 or Rs. 550 but it will not dispense these amounts. These errors could have been easily prevented by a change of the interface.

When it is not possible to prevent errors, the product should try to tolerate user errors and try to guess the user's intention. When the user mistypes a search query, Google asks, *"Did you mean **isosceles**?"*. If the user prefers to type commas to make sure that large amounts are correct, allow him and then just ignore them. Avoid giving a stern warning like the one below (for example, try to guess the budget for the Contingency row in the image below). Even better, put the commas in automatically as the user types the number.



Budget Breakup

Please enter the amount wherever applicable. In case of no Cost is involved, please write remarks in the 'Budget Remarks' column given below the Budget Breakup

Please don't enter ','(comma), '.'(dot) while entering amount.

SrNo	Budget Head	Amount
1.	Consumables	2,00,00,000
2.	Equipments	2,00,000
3.	Contingency	200000000
4.	TA / DA	200

How to assign weights?

- **Assign weight 3-5** if the users of the product are not very tech-savvy, or if the product involves life-critical, business-critical, or goal-oriented tasks.
- **Assign weight 2** in most other cases.

In a study with 65 industrial projects, participants assigned an average weight of **2.58** (standard deviation **1.10**) for this goal parameter.

How to evaluate this goal parameter?

- **During a usability test**, ask users to perform typical tasks and observe the errors frequently made by users.
- **In a review**, invite HCI practitioners with experience on a similar platform. Look for potential errors that could be prevented.

27. *Product should help user recover from errors*

What is it?

Users do not make errors; they do tasks by imperfect approximations. If an error is possible, someone will make it in spite of best design effort. A good product helps users recover from errors. First, users need to recognise that an error has happened. Next, the user needs help to recognise the cause of the error condition and figure out an easy method to recover from the situation. Undo is the users' best friend. It allows users to explore the cyberspace, in the security of the knowledge that they can always retrace their steps.

How to assign weights?

- **Assign weight 3-5** if the users of the product are not very tech savvy or if the product involves life-critical, business-critical, or goal-oriented tasks.
- **Assign weight 1** if the product does not involve many tasks (e.g. an information-only web site with very few pages), or if the task is very frequent, the users are very tech-savvy, and they can figure out how to recover from errors on their own.
- **Assign weight 2** in all other cases.

In a study with 65 industrial projects, participants assigned an average weight of **2.86** (standard deviation **1.20**) for this goal parameter.

How to evaluate this goal parameter?

- During a think-aloud test, probe into the process of error recovery to identify if the product is actually helping the user in error recovery.
- During a performance test, count the percentage of users who are able to recover from errors, and the percentage of errors they can recover from.
- In a review, look at error recovery paths and analyse if they fit all users intents.

28. User should feel in control of the product / behavioural appeal

What is it?

Users often look forward to doing some tasks. It is a joy to cut vegetables with a well-balanced knife, or to cook ones' favourite breakfast. It is a joy to write on a nice white paper with a smooth ink pen. It is a joy to drive a well-maintained car on a smooth road when there is no traffic. On the other hand, no one likes a chair that creaks, a pen that scratches, traffic, or clunky menus. Some products feel nice, smooth, and reliable – other products feel clunky, scratchy, and unreliable. Norman calls this the behavioural appeal of the product.

How to assign weights?

- **Assign weight 3-5** for a task-oriented product meant to be used frequently, for a long period, a product meant to be used by low-tech savvy users, or a casual use product that is meant to afford a *"cool lifestyle"*.
- **Assign weight of 2** in most other cases.

In a study with 65 industrial projects, participants assigned an average weight of **3.23** (standard deviation **1.01**) for this goal parameter.

How to evaluate this goal parameter?

- **In a usability test**, observe / probe to find out if the user felt a loss of control at some stage while using the product. At the end of a test, ask the user to rate the sense of control in comparison with other benchmark products.
- **In a review**, look for potential elements of the design where user may feel a loss of control.

29. User should feel emotionally engaged / fun / brand / reflective appeal / trust

What is it?

Occasionally, users get emotionally attached to a product. Usually the reason is not material in the product, but something they associate with the product. It could be an heirloom pen, or the first birthday gift from a boyfriend. Sometimes it is the association of the good fun one had with the product (usually while sharing it with someone). Sometimes, it is not one person, but a group (e.g. social networking sites) or a country. Sometimes, it is about one's achievement (the bat with which I scored my first century). Sometimes, it is the brand association (why else would one drink brown water that does one no good and damages one's teeth?). At other times, it is an issue of trust (a person who went bankrupt will be a poor brand ambassador for a bank).

How to assign weights?

- **Assign weight 4-5** for a product that is a casual, wide market, lifestyle, or entertainment product (music players, cameras, game consoles, social networking sites, brand sites), or if there is a need to motivate the user to use the product.
- **Assign weight 3** if the product is life-critical or business-critical.
- **Assign weight 0-2** if it is an internal application, an application for which the user does not pay personally, or a product where user has no other choice.

In a study with 65 industrial projects, participants assigned an average weight of **2.54** (standard deviation **1.58**) for this goal parameter.

How to evaluate this goal parameter?

- **In a usability test / field deployment**, survey the users about their emotional engagement, enjoyment, brand appeal etc. Ask them if it were a product they would feel proud to own / be a part of. Ask them to benchmark their answers against other products with which they do feel emotionally engaged.
- **In a review**, invite practitioners experienced in branding and emotional engagement, and ones that are very familiar with the targeted audience.

30. User should find the product aesthetically appealing / visceral appeal

What is it?

Each user may have a different notion of what is beautiful. For some, beauty is raw, raunchy, and bold. For others, it is subtle, graceful, delicate, elegant, and dignified. Nevertheless, all users love things they consider beautiful. Each time one goes to a shop to buy a dress, one tries to buy the one that looks the best (cost being constant). Visceral appeal is the most immediate, first reaction when one sees, hears, smells, feels or tastes something – other considerations come later.

How to assign weights?

- **Assign weight 4-5** for premium products, but also for wide-market, free to use, consumer-facing web sites, for brands, social products, or entertainment products.
- **Assign weight 2-3** for frequently used internal applications, niche market applications or task-oriented applications.
- **Assign weight 1** if the product is used very frequently, used in a very critical context, or if the user has no choice but to use the product.
- **Assign weight 0** if the product is invisible, intangible, inaudible, and has no interface with the user.

In a study with 65 industrial projects, participants assigned an average weight of **2.78** (standard deviation **1.17**) for this goal parameter.

How to evaluate this goal parameter?

- **In a usability test / field deployment**, survey the users with questions about the aesthetic appeal of the product. Ask them to benchmark it against products that suit their aesthetic taste.
- **In a review**, discuss the product aesthetics with artists, designers, and critiques.

This tool is prepared by Anirudha Joshi, IIT Bombay. It can be downloaded from http://www.idc.iitb.ac.in/~anirudha/ugt.htm. It is continuously under improvement. A paper describing this tool has been published for the 4th Workshop on Software and Usability Engineering Cross-pollination: Usability Evaluation of Advanced Interfaces (EVAL 2009), Uppsala, Sweden in conjunction with INTERACT 2009. http://www.interact2009.org/?q=node/55

Please send feedback or comments to anirudha@iitb.ac.in
Version 3.3. September 15, 2010.

Appendix II: Usability Goals Achievement Metric Form

0 – Reject, downright bad. 25 – Mostly bad, but nominal good points. 50 – Undecided, equally good and bad. 75 – Industry standard, good enough, but not exceptional. 100 – Outstanding, superb, trend-setter.

	0	25	50	75	100
Learnability					
Findability: options / data / information should be **visible**					
User should take less **time to learn**: (e.g. in < 10 minutes, 2 hours practice, 2nd attempt)					
Users should be able to **learn on their own**					
Product should be internally consistent					
Product should be consistent with **other products**, older **methods** / past **habits** of users					
Product should be consistent with **earlier version**					
User should remember / **retain** critical, but infrequent tasks					
Speed of use					
User must be able to do the **primary** task / the most **frequent tasks** quickly, easily, at all times					
User should be able to **navigate** quickly and easily					
Product should not load user's **memory** / product should not put **cognitive load** on a user					
Flexibility: user should **control** the sequence of tasks					
User should be able to do freq. / critical tasks in specific **time** / number of **steps** / less **efforts**					
Product should be **personalised** for the user automatically					
Product should be **localised** for specific market segments					
A user should be able to **customise** the product for himself					
Ease of use					
Interface should clearly communicate the **conceptual model**					
Intuitiveness: User should be able to **predict** the next step / task					
No **entry barrier**: user must be able to complete **critical first tasks**					
Product should require no **unnecessary tasks**					
Product should **automate** routine tasks / minimise user **task load**					
Product should be **always on**, always accessible					
Ease of communication					
Information architecture: Information should be well aggregated, well categorised, presented					
Communication should be clear / user should easily **understand** text, visuals					
Error-free use					
Product should give good **feedback** / display its current **status**					
Product should not induce errors					
Product should **tolerate user's errors** / forgiving interface / should **prevent errors**					
Product should help user **recover from errors** / help users **troubleshoot** problems					
Subjective satisfaction					
User should **feel in control** of the product / **behavioural** appeal					
User should feel **emotionally engaged** with product / brand / product should be **fun** / **reflective** appeal					
User should find the product **aesthetically** appealing / product should have a **visceral** appeal					
Other product goals					

Appendix III: Index of Integration Form

0 – This step was not done | 25 – This step was done only nominally, there were several flaws in the technique | 50 – This step was done partially or by short-cut methods | 75 – Standard techniques were used to do this task, good enough, but not exceptional | 100 – Best in class, innovative techniques were used

For Waterfall Process Models

Communication	**Rec**		**0**	**25**	**50**	**75**	**100**
Contextual user studies and user modelling, competitive product analysis	3 - 4						
Ideation with a multidisciplinary team (HCI, tech, biz)	2						
Product definition / IA / wireframes with a multidisciplinary team (HCI, tech, biz)	1 - 3						
Usability evaluation (formative) and refinement of product definition	1 - 3						
Modeling							
Detailed UI prototyping	4 - 5						
Usability evaluation (formative) and refinement of prototype	4 - 5						
Construction							
Development support reviews by usability team	3						
Usability evaluation (summative)	1 - 3						
Other activities							

For Agile Process Models

Before iterations start	Rec		**0**	**25**	**50**	**75**	**100**
Contextual user studies and user modelling, competitive product analysis, ideation	4						
Product definition/ IA / wireframes with a multidisciplinary team, evaluation	3						
Detailed UI prototyping for 1st iteration	5						
Usability evaluation (formative) and refinement of prototype for 1st iteration	3						
During iterations							
Detailed UI prototyping for next iteration	4						
Usability evaluation (formative) and refinement of prototype for next iteration	2						
Development support reviews by usability team for current iteration	3						
Usability evaluation (summative) of last iteration	1						
After the last iteration							
Usability evaluation (summative) of release version	3						
Other activities							

Appendix IV: Index of Integration Guidelines

(These guidelines have been reproduced here for easy reference. They can also be viewed on http://www.idc.iitb.ac.in/~anirudha/ugt.htm.)

Evaluation Guidelines for HCI Activities in Waterfall

The following table summarizes the Phases, HCI Activities and Activity Evaluation Guidelines for the waterfall model. It should be used by the evaluators while computing IoI scores.

HCI Activities	Activity Evaluation Guidelines
Phase: Communication	
Contextual user studies and user modeling, competitive product analysis	1. Both organizational data gathering and user studies were done before requirements were finalized 2. User studies were done in the context of the users by the method of contextual inquiry 3. User studies were done with at least 20 users in each profile 4. User studies were done by people with experience in user studies in a similar domain of at least 2 projects 5. The findings including user problems, goals, opportunities and constraints were analyzed, documented and presented in an established user modelling methodology such as personas, work models, affinity diagram or similar 6. Competitive / similar products and earlier versions of the products were evaluated for potential usability problems by at least using discount usability evaluation methods such as Heuristic Evaluation and were benchmarked 7. User experience goals were explicitly agreed upon before finalizing requirements 100 = All the above are true, the activity was performed exceptionally well 75 = At least five of the above are true, including point 7, or all the above are true, but point 3 had fewer than 20 users per profile, the activity was performed reasonably well 50 = At least three of the above are true, including point 7, the activity was done with some shortcuts and / or perhaps was not timed well 25 = Only two of the above are true, the activity was done poorly with many shortcomings 0 = None of the above are true, the activity was not done.
Ideation	1. Ideation was done explicitly before requirements were finalized 2. Ideation was done by an established method such as brainstorming, participatory design, method cards, thinking hats, synectics, QFD and TRIZ 3. At least one experienced representative of the following disciplines participated: technology, business, design, user, project / client stakeholder 100 = All of the above are true, the activity was performed exceptionally well 50 = At least two of the above are true, the activity was done with some shortcuts and / or perhaps was not timed well 25 = Only one of the above is true, the activity was done poorly with many shortcomings 0 = None of the above are true, the activity was not done.
Product definition	1. Product definition was documented and presented explicitly by an established method such as personas, storyboards, scenarios etc. 2. At least two alternatives for high-level interaction design and information architecture were considered before finalizing functional, non-functional and content requirements 100 = All the above are true, the activity was performed exceptionally well 50 = At least one of the above is true, the activity was done with some shortcuts and / or perhaps was not timed well 0 = None of the above are true, the activity was not done.

HCI Activities	Activity Evaluation Guidelines
Usability evaluation and refinement of product definition	1. The product definition was evaluated for potential usability problems by at least using discount usability evaluation methods such as Heuristic Evaluation. 2. The usability problems found were addressed by necessary changes in the product definition as well as in the functional, non-functional and content requirements 100 = All the above are true, the activity was performed exceptionally well 0 = None of the above are true, the activity was not done.
Phase: Modeling	
Detailed UI prototyping	1. (If it was not done in the earlier phase) the product definition was evaluated for potential usability problems by at least using discount usability evaluation methods such as Heuristic Evaluation. 2. (If it was not done in the earlier phase) the findings usability problems found were addressed by necessary changes in the product definition as well as in the functional, non-functional and content requirements 3. A UI prototype, covering at least 50% of the use cases, all the frequent and critical use cases among them, was developed before coding began. 4. The UI prototype included interface design, information design and navigation design. 5. The designers of the UI prototype had experience with at least 2 projects in a similar domain and on a similar platform. 100 = All the above are true, the activity was performed exceptionally well 75 = Most of the above are true, the activity was performed reasonably well, though not exceptionally 50 = Numbers 3-4 of the above are partially true, the activity was done with some shortcuts and / or perhaps was not timed well 25 = Only one of 3-5 is true, the activity was done poorly with many shortcomings 0 = None of the above are true, the activity was not done.
Usability evaluation (formative) and refinement of prototype	1. The UI prototype covering all the known use cases was completed before evaluation. 2. The UI prototype was evaluated for potential usability problems by formative usability evaluation with 10 real users per profile. 3. (If 1 was not done) the UI prototype was evaluated for potential usability problems by using discount usability evaluation methods such as Heuristic Evaluation. 4. All major problems found in usability evaluation were addressed by necessary changes in the UI prototype and if necessary in the requirements before coding actually began. 100 = 1, 2 and 4 are true, the activity was performed exceptionally well 75 = 1, 3 and 4 are true, or 1, 2 and 4 are true but not with enough users, the activity was performed reasonably well 50 = At least two of the above (including 2 or 3) are true, the activity was done with some shortcuts and / or perhaps was not timed well 25 = Only one of the above is true, the activity was done poorly with many shortcomings 0 = None of the above are true, the activity was not done.

HCI Activities	Activity Evaluation Guidelines
Phase: Construction	
Development support reviews by usability team	1. The detailed UI prototype was documented and presented by the designers to the developers at least once. The documentation was signed off and then was available for later reference to every developer. 2. The designers of the UI prototype were known to the software developers in the project and were available for discussion on short notice. 3. The UI was reviewed for compliance by the original designers at least once a month during development. 4. Each change to the UI prototype was discussed and approved by the original UI designers. 5. The designers did not foresee any major usability problems creeping in because of the changes. 100 = All of the above are true, the activity was performed exceptionally well 75 = Any four of the above (including 4 and 5) are true, the activity was performed reasonably well 50 = Any three of the above (including 1) are true, the activity was done with some shortcuts and / or perhaps was not timed well 25 = Only one or two of above is true, the activity was done poorly with many shortcomings 0 = None of the above are true, the activity was not done.
Usability evaluation (summative)	1. An early release of the product with at least 50% of the use cases, all the frequent and critical use cases among them, was evaluated for potential usability problems by a summative usability evaluation with 10 real users per profile by independent evaluators (i.e. not connected with the project). 2. (If 1 was not done) the above release was evaluated for potential usability problems by at least using discount usability evaluation methods such as Heuristic Evaluation by independent evaluators (i.e. not connected with the project). 3. All important problems found in usability evaluation were addressed by necessary changes in the UI in a later release. 100 = 1 and 3 are true, the activity was performed exceptionally well 75 = 2 and 3 are true, or 1 and 3 are true, but not with independent evaluators or enough users, the activity was performed reasonably well 50 = 2 and 3 are true, but not with independent evaluators, the activity was performed with some shortcuts 25 = Only 1 or 2 is true, the activity had little impact on the product 0 = None of the above are true, the activity was not done.

Evaluation Guidelines for HCI Activities in Agile

The following table summarizes the Phases, HCI Activities and Activity Evaluation Guidelines for the agile process models. It should be used by the evaluators while computing IoI scores.

HCI Activities	**Activity Evaluation Guidelines**
Phase: Pre-iterations	
Contextual user studies and user modeling, competitive product analysis, ideation	1. Both stakeholder interviews and user studies were done before requirements were finalized 2. User studies were done in the context of the users by the method of contextual inquiry 3. User studies were done with at least 20 users in each profile 4. User studies were done by people with experience in user studies in a similar domain of at least 2 projects 5. The findings including user problems, goals, opportunities and constraints were analyzed, documented and presented in an established user modelling methodology such as personas, work models, affinity diagram or similar 6. Competitive / similar products and earlier versions of the products were evaluated for potential usability problems by at least using discount usability evaluation methods such as Expert Review and were benchmarked 7. User experience goals were explicitly agreed upon before starting iterations 8. Ideation was done explicitly before requirements were finalized 9. Ideation was done by an established method such as brainstorming, participatory design, method cards, thinking hats, synectics, QFD and TRIZ 100 = All the above are true, the activity was performed exceptionally well and was timed well 75 = At least seven of the above are true including point 7, or all are true but the activities were not completed before iterations started or activity 3 was done with fewer than 20 users per profile, the activity was performed reasonably well, though not exceptionally, the activity happened in reasonable time 50 = At least five of the above are true, including point 7, the activity was done with some shortcuts and / or was not timed well 25 = At least two of the above are true, the activity was done poorly with many shortcomings and was poorly timed 0 = None of the above are true, the activity was not done.
Product definition / IA / wireframes with a multidisciplinary team, evaluation	1. Product definition was documented and presented explicitly by an established method such as personas, storyboards, scenarios etc. 2. At least two alternatives for high-level interaction design and information architecture were considered before finalizing functional, non-functional and content requirements 3. At least one experienced representative of each of the following disciplines participated in this and continued through the rest of the project: technology, business, design, user, client 100 = All of the above are true, the activity was performed exceptionally well and in perfect timing 75 = At least points 2 and 3 are true, or all are true but the activities were not completed before iterations started, the activity was performed reasonably well, though not exceptionally 50 = At least one of the above are true, the activity was done with some shortcuts and / or perhaps was not timed well 25 = At least two of the above are true, the activity was done poorly with many shortcomings, the activity was not timed well 0 = None of the above are true, the activity was not done.

HCI Activities	Activity Evaluation Guidelines
Detailed UI prototyping for 1st iteration	1. (If it was not done earlier) the product definition was evaluated for potential usability problems by using discount usability evaluation methods such as Heuristic Evaluation. 2. (If it was not done earlier) the findings usability problems found were addressed by necessary changes in the product definition as well as in the functional, non-functional and content requirements 3. A UI prototype, covering at least the user stories of the first iteration, was developed well before coding began. 4. The UI prototype included interface design, information design and navigation design. 5. The designers of the UI prototype had experience with at least 2 projects in a similar domain and on a similar platform. 100 = All of the above are true, the activity was performed and timed exceptionally well 75 = At least two of points 3-5 are true, or all are true but the activities were not completed before iterations started, the activity was performed reasonably well, though not exceptionally 50 = At least one of points 3-5 is true, the activity was done with some shortcuts and / or perhaps was not timed well 25 = At least one of points 3-5 is partially true, the activity was done poorly with many shortcomings 0 = None of the above are true, the activity was not done.
Usability evaluation (formative) and refinement of prototype for 1st iteration	1. The UI prototype was evaluated for potential usability problems by formative usability evaluation with real users. 2. (If 1 was not done) the UI prototype was evaluated for potential usability problems by using discount usability evaluation methods such as Heuristic Evaluation. 3. All major problems found in usability evaluation were addressed by necessary changes in the UI prototype and if necessary in the product definition before iterations began. 4. The UI prototype covering all user stories for first iteration was completed before coding began. 100 = All of the above are true, the activity was performed and timed exceptionally well 75 = Points 2-4 are true, or all are true but the activities were not completed before iterations started, the activity was performed reasonably well, though not exceptionally 50 = Two of the above are true, the activity was done with some shortcuts and / or perhaps was not timed well 25 = One of the above is true, the activity was done poorly with many shortcomings 0 = None of the above are true, the activity was not done.
During iterations	
Detailed UI prototyping of next iteration	1. A UI prototype, covering at least the user stories of the next iteration, was developed well before coding began. 2. The UI prototype included interface design, information design and navigation design. 3. The designers of the UI prototype had experience with at least 2 projects in a similar domain and on a similar platform. 100 = All of the above are true, the activity was performed and timed exceptionally well 75 = Two of above are true, or all are true but the activities were not completed before the next iteration, the activity was performed reasonably well, though not exceptionally 50 = One of above is true, the activity was done with some shortcuts and / or perhaps was not timed well 25 = One of points 1-3 is partially true, the activity was done poorly with many shortcomings 0 = None of the above are true, the activity was not done.

HCI Activities	Activity Evaluation Guidelines
Usability evaluation (formative) and refinement of next iteration	1. The UI prototype covering all user stories for the next iteration was ready before evaluation began. 2. The UI prototype for the next iteration was evaluated for potential usability problems by formative usability evaluation with real users. 3. (If 1 was not done) the UI prototype was evaluated for potential usability problems by using discount usability evaluation methods such as Heuristic Evaluation. 4. All major problems found in usability evaluation were addressed by necessary changes in the UI prototype and if necessary in the product definition before the next iteration. 100 = Points 1, 2, and 4 are true, the activity was performed and timed exceptionally well 75 = Points 1, 3 and 4 are true, or 1, 2, and 4 are true but the activities were not completed before the next iteration started, the activity was performed reasonably well, though not exceptionally 50 = Two of the above are true, the activity was done with some shortcuts and / or perhaps was not timed well 25 = One of the above is true, the activity was done poorly with many shortcomings 0 = None of the above are true, the activity was not done.
Development support reviews by HCI design team during the iterations	1. The detailed UI prototype of all the user stories for the current iteration was presented to the developers before the current iteration began. The prototype was signed off and then was available for later reference to every developer. 2. The designers of the UI prototype were personally known to all the software developers in the project and were available for discussions. 3. The UI being developed was reviewed for compliance by the designers of the UI prototype at the end of each day. The UI team members participated in daily team meetings. 4. Each change to the documented UI prototype was discussed and approved by the original UI designers. 5. The designers did not foresee any major usability problems creeping in because of the changes. 100 = All of the above are true, the activity was performed exceptionally well 75 = Four of the above (including 1, 4 and 5) are true, the activity was performed reasonably well 50 = Three of the above (including 1) are true, the activity was done with some shortcuts and / or perhaps was not timed well 25 = Point 1 is true, the activity was done poorly with many shortcomings 0 = None of the above are true, the activity was not done.
Usability evaluation (summative) of the output from the earlier iteration	1. The working product from the earlier iteration was evaluated for potential usability problems by a summative usability evaluation with real users by independent evaluators (i.e. not connected with the project). 2. (If 1 was not done) the above release was evaluated for potential usability problems by using discount usability evaluation methods such as Expert Review. 3. All problems found in usability evaluation were addressed by necessary changes in the UI. 100 = Points 1 and 3 are true for at least 40% of the iterations, and 1 is true for 50% of the early iterations, the activity was performed exceptionally well 75 = 2 and 3 are true for 40% of iterations, or 1 and 3 are true for 20% of the iterations, the activity was performed reasonably well 50 = 2 and 3 are true for 20% of the iterations (particularly the early iterations), the activity was performed with some shortcuts 50 = 2 and 3 are true for at least one intermediate iteration, the activity was performed with many shortcuts 0 = None of the above are true, the activity was not done.
Phase: Pre-Delivery	

HCI Activities	Activity Evaluation Guidelines
Usability evaluation (summative) of the final delivery	1. After all iterations (or preferably after all but two iterations), the product was evaluated for potential usability problems by a summative usability evaluation with 10 real users per profile by independent evaluators (i.e. not connected with the project). 2. (If 1 was not done) the above release was evaluated for potential usability problems by at least using discount usability evaluation methods such as Expert Review. 3. All major problems found in usability evaluation were addressed by necessary changes in the UI in the last iteration. 100 = Points 1 and 3 are true, the activity was performed exceptionally well 75 = Points 2 and 3 are true, or points 1 and 3 are true but not with independent evaluators or with fewer users per profile, the activity was performed reasonably well 50 = 2 and 3 are true, but not with independent evaluators, the activity was performed with some shortcuts 25 = Only 1 or 2 is true, the activity had little impact on the product 0 = None of the above are true, the activity was not done.

References

Adams R, Bass L, and John B Experience with Using General Usability Scenarios on Software Architecture of a Collaborative System, in Human Centred Software Engineering, Seffah A, Gulliksen J, and Desmarais M (eds.), Springer, 2005.

Agile Manifesto 2001, Accessed June 1, 2009, http://agilemanifesto.org/.

Apple Inc. Apple Computer Knowledge Navigator, 1987, Accessed August 8, 2009, http://video.google.com/videoplay?docid=-5144094928842683632.

Archer B Systematic Method for Designers, Council of Industrial Design, 1965.

Baker F Chief Programmer Team Management of Production Programming, IBM Systms Journal, 1972, 1 : Vol, 11.

Battle L Patterns of Integration: Bringing User Centred Design into Software Development Lifecycle, in Human Centred Software Engineering, Seffah A, Gulliksen J, and Desmarais M (eds.), Springer, 2005.

Beck K Extreme Programming Explained: Embrace Change, 2nd Edition, Addison-Wesley Professional, 2004.

Bevan N Classifying and Selecting UX and Usability Measures, International Workshop on Meaningful Measures: Valid Useful User Experience Measurement, 2008.

Beyer H, Holtzblatt K, and Baker L, An Agile Customer-Centered Method: Rapid Contextual Design, XP / Agile Universe, 2004.

Beyer H and Holtzblatt K Contextual Design, Morgan Kaufmann, 1998.

Bias R and Mayhew D (eds.), Cost-Justifying Usability, Second Edition: An Update for the Internet Age, Morgan Kaufmann, 2005.

Blomkvist S Towards a Model for Bridging Agile Development and User-Centred Design, in Human Centred Software Engineering, Seffah A, Gulliksen J, and Desmarais M (eds.), Springer, 2005.

Brace N, Kemp R, and Snelgar R SPSS for Psychologists, 2nd edition, Palgrave Macmillan, 2003.

Brooke J System Usability Scale, 1996, Accessed January 1, 2009, http://www.hostserver150.com/usabilit/trump/documents/Suschapt.doc.

Brooks F The Mythical Man-month, 20th Anniversary Edition, Addison-Wesley, 1995.

Buxton B Sketching User Experiences, Morgan Kaufmann, 2007.

Carroll J Human Computer Interaction, Interaction-Design.org, 2009, Accessed June 12, 2010, http://www.interaction-design.org/encyclopedia/human_computer_interaction_hci.html.

Carter J [et al.] Transforming Usability Engineering Requirements into Software Engineering Specifications: From PUF to UML, in Human Centred Software Development, Seffah A, Gullksen J, and Desmarais M (eds.), Springer, 2005.

Chamberlain S, Sharp H, and Maiden N Towards a Framework for Integrating Agile Development and User-Centred Design, XP, 2006.

Chin J, Diehl V, and Norman K Development of an Instrument Measuring User Satisfaction of the Human-Computer Interface, ACM CHI, 1988.

Clements P Constructing Superior Software, Sams, 1999.

Constantine L Process Agility and Software Usability: Toward Lightweight Usage Centered Design, Information Age, 2002.

Constantine L, Biddle R, and Noble J Usage-Centered Design and Software Engineering: Models for Integration, Bridging the Gaps Between SE and HCI, ICSE, 2003.

Constantine L and Lockwood L Software for Use, Addison-Wesley, 1999.

Cooper A and Reimann R About Face 2.0, Wiley, 2003.

Cooper A Allen's Keynote at Agile 2008, Accessed August 10, 2009, http://www.cooper.com/journal/2008/08/alans_keynote_at_agile_2008.html.

Costabile M Usability in the Software Life Cycle, Handbook of Software Engineering and Knowledge Engineering, in Chang S (ed.), World Scientific, 2002, Vol. 1.

Cox N in Mandel, T The Elements of User Interface Design, John Wiley & Sons, 1997.

Cross N Engineering Design Methods 3rd ed., John Wiley & Sons, 2000.

da Silva P and Norman W The Unified Modeling Language for Interactive Applications, Proc. of UML 2000, Springer, 2000.

Dix A [et al.] Human-Computer Interaction, Second Edition, Prentice Hall, 1998.

Dumas J and Redish J A Practical Guide to Usability Testing, Intellect Ltd, 1999.

Eames C Charles Eames Quotes, BrainyQuote.com, Accessed August 15, 2009, http://www.brainyquote.com/quotes/quotes/c/charleseam169187.html.

Earthy J Usability Maturity Model: Processes Version 2.2, Lloyd's Register of Shipping, August 19, 1999, Accessed August 16, 2009, http://www.idemployee.id.tue.nl/g.w.m.rauterberg/lecturenotes/Usability-Maturity-Model[2].PDF.

Fenton N and Pfleeger S Software Metrics – A Rigorous and Practical Approach, Thomsan Brooks / Cole, 2002.

Ferre X Integration of Usability Techniques into the Software Development Process, Workshop on Bridging the Gaps Between SE and HCI, ICSE 2003, 2003.

Ferre X, Juristo N, and Moreno A Which, When and How Usability Techniques and Activities Should be Integrated, in Human Centred Software Engineering, Seffah A, Gulliksen J, and Desmarais M (eds.), 2005.

Fowler M The New Methodology, December 13, 2005, Accessed June 1, 2009, http://martinfowler.com/articles/newMethodology.html.

Garrett J The Elements of User Experience, New Riders, 2003.

Google Inc. Google Maps for Mobile with My Location (beta), December 12, 2007, Accessed August 8, 2009, http://mediakey.dk/~cc/google-maps-for-mobile-with-my-location-gsm-positioning/.

Göransson B, Lif M, and Gulliksen J Usability Design – Extending Rational Unified Process with a New Discipline, Interactive Systems, Design, Specification, and Verification, 10th International Workshop, 2003.

Gould J and Lewis C Designing for Usability: Key Principles and What Designers Think, Communications of the ACM, 1985, 3 : Vol. 28.

Gulliksen J [et al.] Key Principles for User-centred System Design, Behaviour and Information Technology, 2003, 6 : Vol. 22.

Gulliksen J, Cajander A, and Eriksson E Only Figures Matter? – If Measuring Usability and User Experience in Practice is Insanity or a Necessity, International Workshop on Meaningful Measures: Valid Useful User Experience Measurement, 2008.

Hackos J and Redish J User and Task Analysis for Interface Design, Wiley, 1998.

Hanna M Farewell to waterfalls?, Software Magazine, 1995, 5 : Vol, 15, pp, 38-46.

Hatcher L A Step-by-Step Approach to Using the SAS System for Factor Analysis and Structural Equation Modeling, SAS Publishing, 1994.

Heller D Why I'm Not Calling Myself an Information Architect Anymore, 2002, Accessed August 13, 2009, http://www.boxesandarrows.com/view/why_im_not_calling_myself_an_information_architect_anymore.

Heumann J User experience storyboards: Building better UIs with RUP, UML, and use cases, The Rational Edge, The Rational Software, 2003, Accessed March 7, 2009, http://www.ibm.com/developerworks/rational/library/content/RationalEdge/nov03/f_usability_jh.pdf.

Hornbæk K and Law E Meta-analysis of Correlations Among Usability Measures, ACM CHI, 2007.

Humphrey W A Discipline for Software Engineering, Addison-Wesley, 1995.

Humphrey W Introduction to the Team Software Process, Addison-Wesley Professional, 1999.

IEEE IEEE Standard Glossary of Software Engineering Terminology, 1993.

IEEE Software Engineering Coordinating Committee Guide to the Software Engineering Body of Knowledge Trial Version 1.00, Los Alamitos : IEEE Computer Society, 2001.

IFIP WG 2.7/13.4 on User Interface Engineering Bridging the SE & HCI Communities, 2004, Accessed September 17, 2007, http://www.se-hci.org/bridging/index.html.

International Organization for Standardization ISO 9241-1:1997 Ergonomic Requirements for Office Work with Visual Display Terminals, 1997.

International Organization for Standardization ISO/IEC 9126-1:2001 Software Engineering Product Quality, 2001.

IXDA About Interaction Design, Interaction Design Association, 2009, Accessed May 30, 2009, http://www.ixda.org/about_interaction.php.

Jalote P An Integrated Approach to Software Engineering, Narosa Publishing House, 1997.

Jerome B and Kazman R Surveying the Solitudes: An Invetigation into the Relationships between HCI and SE in Practice, in Human Centred Software Engineering, Seffah A, Gulliksen J, and Desmarais M (eds.), Springer, 2005.

John B [et al.] Avoiding "We can't change that!": Software Architecture & Usability, CHI 2004, Vienna, 2004.

Jokela T Guiding Designers to the World of Usability: Determining Usability Requirements Through Teamwork, in Human Centred Software Engineering, Seffah A, Gulliksen J, and Desmarais M (eds.), 2009.

Jones C Design Methods, Seeds of Human Futures, Wiley Interscience, 1970.

Jordan P Designing Pleasurable Products, Taylor & Francis, 2000.

Joshi A Index of Integration, 2009, Accessed June 25, 2010, http://www.idc.iitb.ac.in/~anirudha/ioi.htm.

Joshi A Institutionalizing HCI - the Challenges in India, CSI Communications, 2005.

Joshi A Usability Goals Setting Tool Online Version, IIT Bombay, 2009, Accessed September 22, 2009, http://www.idc.iitb.ac.in/~anirudha/ugt.htm.

Juristo N [et al.] Improving Software Usability through Architectural Patterns, Workshop on Bridging the Gaps Between SE and HCI, ICSE, 2003.

Kirakowski J and Corbett M SUMI: The Software Usability Measurement Inventory, British Journal of Educational Technology, 1993, 3 : Vol, 24.

Kreitzberg Charles Managing for Usability, Multimedia: A Management Perspective, in Alber Antone (ed.), Wadsworth, 1996.

Kroll P and Kruchten P The Rational Unified Process Made Easy, Pearson Education, 2003.

Kujala S Linking User Needs and Use Case-Driven Requirements Engineering, in Human Centered Software Engineering, Seffah A, Gulliksen J, and Desmarais M (eds.), Springer, 2005.

Kuniavsky M Observing the User Experience, Morgan Kaufman, 2003.

Kurchten P The Rational Unified Process An Introduction 3rd ed., Pearson Education, 2004.

Kurtz N Introduction to Social Statistics, McGraw Hill Book Company, 1983.

Laseau P Graphic Thinking for Architects and Designers, Van Nostrand Reinhold Company, 1980.

Law E [et al.] Understanding, Scoping and Defining User eXperience: A Survey Approach, CHI 2009, 2009.

Lawson B The Design Process Demystified, Butterworth Architecture, 1980.

Lee J and McCrickard D Towards Extreme(ly) Usable Software: Exploring Tensions Between Usability and Agile Software Development, Agile, 2007.

Lewis J A Rank-Based Method for the Usability Comparison of Competing Products, Human Factors and Ergonomics Society 35th Annual Meeting, 1991.

Lewis J IBM Computer Usability Satisfaction Questionnaire: Psychometric Evalution and Instructions for Use, International Journal of Human-Computer Interaction, 1995, 1 : Vol. 7.

Lin H, Choong Y, and Salvendy G A Proposed Index of Usability: A Method for Comparing the Relative Usability of Different Software Systems, Behaviour and Information Technology, 1997.

Mahlke S Understanding Users' Experience of Interaction, EACE, 2005.

Mayhew D Usability Engineering Lifecycle, Morgan Kaufmann, 1999.

McCarthy J and Wright P Technology as Experience, The MIT Press, 2004.

McGee M Master Usability Scaling: Magnitude Estimation and Master Scaling Applied to Usability Measurement, ACM CHI, 2004.

McKim R Experiences in Visual Thinking, Brooks / Cole Publishing Company, 1980.

Myers B and Rosson M Survey on User Interface Programming, Proc. of the SIGCHI Conference on Human Factors in Computing Systems, 1992.

NASSCOM Extending India's Leadership of the Global IT and BPO Industries Executive Summary, NASSCOM, 2005.

NASSCOM Knowledge Professionals in India, NASSCOM, 2006, Accessed July 9, 2009, http://www.nasscom.in/upload/5216/NASSCOM%20Knowledge%20Professionals%20Factsheet%202006.pdf.

NASSCOM Strategic Review 2007 Executive Summary, Accessed August 5, 2009, http://www.nasscom.in/upload/51054/Executive%20Summary.pdf.

NASSCOM Strategic Review 2009, Accessed January 24, 2010, http://www.nasscom.org/Nasscom/templates/NormalPage.aspx?id=55772.

Neill C and Laplante P Requirements Engineering: The State of the Practice, IEEE Software, 2003, 6 : Vol. 20.

Nelson E Extreme Programming vs. Interaction Design, 2002, Accessed August 8, 2009, http://web.archive.org/web/20070313205440/http://www.fawcette.com/interviews/beck_cooper/.

Nielsen J Agile Development Projects and Usability, November 17, 2008, Accessed March 7, 2009, http://www.useit.com/alertbox/agile-methods.html.

Nielsen J Usability Engineering, Morgan Kaufmann, 1993.

Norman D Emotional Design: Why We Love (or Hate) Everyday Things, Basic Books, 2004.

Nunes N and e Cunha J Towards a UML profile for Interaction Design: The WISDOM approach, Proc. of UML, Springer, 2000.

Parnas D Software engineering or methods for the multi-person construction of multi-version programs, Lecture Notes In Computer Science, Programming Methodology, 4th Informatik Symposium, Springer-Verlag, 1974, Vol. 23.

Patton J Hitting the Target: Adding Interaction Design to Agile Software Development, OOPSLA, 2002.

Preece J, Rogers Y, and Sharp H Interaction Design Beyond Human-Computer Interaction, 1st edition, John Wiley & Sons, 2002.

Pressman R Software Engineering – a Practitioner's Approach (6th Edition), McGraw Hill, 2005.

Pyla P [et al.] Ripple: An event driven Design Representation Framework for Integrating Usability and Software Engineering Lifecycles, in Human Centred Software Engineering, Seffah A, Gulliksen J, and Desmarais M (eds.), Springer, 2005.

Royce W Managing the Development of Large Software Systems, IEEE WESCON, TRW, 1970.

Rumbaugh J [et al.] Object Oriented Modelling and Design, Pearson Education, 1991.

Sauro J and Kindlund E A Method to Standardize Usability Metrics into a Single Score, ACM CHI, 2005.

Sauro J and Lewis J Correlations among Prototypical Usability Metrics: Evidence for the Construct of Usability, ACM CHI, 2009.

Shneiderman B Designing the User Interface: Strategies for Effective Human-Computer Interaction (4th Edition), Addison Wesley, 2004.

Schön D The Reflective Practitioner How Professionals Think in Action, Basic Books, 1983.

Seffah A, Desmarais M, and Metzker E HCI, Usability and SE Integration: Present and Future, in Human Centred Software Engineering, Seffah A, Gulliksen J, and Desmarais M (eds.), Springer, 2005.

Seffah A, Gulliksen J, and Desmarais M (eds.), Human Centred Software Engineering, Springer, 2005.

Sharp H, Rogers Y, and Preece J Interaction Design Beyond Human-Computer Interaction, 2nd Edition, Wiley India, 2007.

Snyder C Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces, Morgan Kaufmann, 2003.

Software Engineering Institute CMMI for Development Version 1.2, August 2006, Accessed August 15, 2009, http://www.sei.cmu.edu/publications/documents/06.reports/06tr008.html.

Sousa K and Furtado E RUPi A Unified Process that Integrates Human-Computer Interaction and Software Engineering, ICSE 2003 Workshop Bridging the Gaps Between Software Engineering and Human-Computer Interaction, 2003.

Suitcliffe A Convergence or Competition between SE and HCI, in Human Centered Software Engineering, Seffah A, Gulliksen J, and Desmarais M (eds.), Springer, 2005.

Swallow D, Blyth M, and Peter W Grounding Experience Relating Theory and Method to Evaluate the User Experience of Smart-phones, Annual Conference on European Association of Cognitive Ergonomics, 2005.

The Standish Group Report CHAOS, 1995, Accessed January 31, 2010, http://www.projectsmart.co.uk/docs/chaos-report.pdf.

Tractinsky N, Katz A, and Ikar D What is Beautiful is Usable, Interacting with Computers, 2000, Vol, 13.

Tullis T and Stetson J A Comparison of Questionnaires for Assesing Website Usability, Usability Professional Association Conference, 2004.

Tullis T and Albert B Measuring the User Experience, Morgan Kaufmann, 2008.

Usability Professionals Association, Usability Body of Knowledge, 2004, January 24, 2010.

Walpole R [et al.] Probability & Statistics for Engineers & Scientists, Pearson Education, 2007.

Wells D User Stories, Extreme Programming, 1999, Accessed January 28, 2010, http://www.extremeprogramming.org/rules/userstories.html.

Publications Arising From This Research

Joshi Anirudha HCI in SE Process Literature, Indo-Dan HCI Research Symposium, Guwahati, 2006.

Joshi Anirudha HCI + SE Integration – Case Studies from Offshore Development Projects, Workshop on Increasing the impact of usability work in software development, ACM CHI, 2007.

Joshi Anirudha HCI and SE – The Cultures of the Professions, HCI International, 2007.

Joshi Anirudha and Sarda NL HCI and SE: Towards a 'Truly' Unified Waterfall Process, HCI International, 2007.

Joshi Anirudha and Tripathi Sanjay User Experience Metric and Index of Integration: Measuring Impact of HCI Activities on User Experience, International Workshop on the Interplay between Usability Evaluation and Software Development, 2008.

Joshi Anirudha Usability Goals Setting Tool, 4th Workshop on Software and Usability Engineering Cross-Pollination: Usability Evaluation of Advanced Interfaces, INTERACT 2009.

Joshi Anirudha, Sarda NL and Tripathi Sanjay Measuring Effectiveness of HCI Integration in Software Development Processes, Journal of Software Systems, 2010. DOI: 10.1016/j.jss.2010.03.078.

Joshi Anirudha and Sarda NL Evaluating Relative Contributions of Various HCI Activities to Usability, Human-Centred Software Engineering, 2010.

Joshi Anirudha and Sarda NL Do Teams Achieve Usability Goals? Evaluating Goal Achievement with Usability Goals Setting Tool, INTERACT, 2011.

Acknowledgements

I thank my guide Prof. N.L. Sarda for his support, time, patience, feedback, insights, and encouraging guidance. He gave me the freedom to explore my areas of interest and gave me nudges in the right direction when I needed them. He was critical where necessary and taught me to pay attention to details. Special thanks are due to Prof. Umesh Bellur, Prof. S Sudarshan, and Prof. UA Athavankar for their constant feedback and comments.

I thank all the participants in my studies. Without their contributions, it would not have been possible to gather data my research. Special acknowledgements are due to Pramod Khambete and his colleagues from Tech Mahindra for their cooperation and support in trying out pilots. Acknowledgements are also due to Atul Manohar and his colleagues from Satyam for their cooperation and help, and for allowing me to use Satyam premises for conducting my tutorials in Pune, Bangalore, Hyderabad, and Chennai.

Before this, most of my research was based on qualitative techniques. In this research, I had to take a major leap into statistical techniques. This leap became possible because of the help and guidance of Dr. Sanjay Tripathi, Aniruddha Puranik, Girish Dalvi, Smita Wangikar, Devanuj, and Prof. SV Sabnis.

I thank my colleagues and students in IDC for their support and help during my doctoral thesis.

Finally, I thank my wife Dr. Madhavi, sons Advait and Aditya, and my parents Nana and Mani, who continue to allow me the opportunity and the luxury to pursue an academic goal at the cost of our time together.

Anirudha Joshi
March 29, 2011