Harald Milchrahm

Agile Usability Processes

Dissertation

vorgelegt an der

Technischen Universität Graz

zur Erlangung des akademischen Grades
Doktor der Technischen Wissenschaften
(Dr.techn.)

durchgeführt am Institut für Softwaretechnologie
Technische Universität Graz
Vorstand: O. Univ.-Prof. Dipl.-Ing. Dr. techn. Wolfgang Slany

Graz, im Mai 2010

Deutsche Kurzfassung der Dissertation

Diese Dissertation untersucht die Kombination von agilen Softwareentwicklungsmethoden mit Usability Engineering. Der Prozess der Integration von Extreme Programming und User-Centered Design wird behandelt. Es wird gezeigt, dass agile Softwareentwicklungsmethoden Benutzerzentriertheit, iterative User Interface Entwicklung und die Einbeziehung eines Usability Engineers fördern. Der beschriebene Ansatz basiert auf der Adaption des klassischen Extreme Programming Prozesses und integriert folgende Human-Computer Interaction Instrumente: User Studies, Extreme Personas, Usability Expert Evaluations, Usability Tests, und Automated Usability Evaluations. Die in diesem Forschungsprojekt entwickelte Anwendung ermöglicht einem Benutzer die inhaltsbezogene Suche nach Audio und Video Material, sowie dessen Wiedergabe auf einem Mobiltelefon. Reflexionen über den verwendeten integrierten Prozess, basierend auf Daten gesammelt durch Codeanalysen, Prozessevaluierungstools und Aufzeichnungen aus Review Sitzungen, werden diskutiert. Die Resultate von Usability Tests der, mittels des integrierten Ansatzes entwickelten, Anwendung werden ebenfalls präsentiert.

Von dem agilen, Extreme Programming basierenden, Usability Prozess dieses wissenschaftlichen Projektes wurden drei Agile Usability Prozess Muster abgeleitet. Die Muster bilden die Integration von drei der verwendeten Human-Computer Interaction Instrumente auf iterative Softwareentwicklungsprozesse ab und heissen: Usability Expert Evaluation, Usability Test, und Automated Usability Evaluation. Um die Gültigkeit dieser Muster zu überprüfen wurden sie in einem industriellen, Extreme Programming basierenden, Projekt implementiert. Beide Prozesse wurden mit Hilfe eines Extreme Programming Evaluation Framework evaluiert. Für jeden Prozess wurden Kontextfaktoren aufgezeichnet und quantitative sowie qualitative Prozessmetriken zu zwei Zeitpunkten erhoben. Die Musterimplemtierungsergebnisse zeigen, dass die Usability und die generelle User Experience der entwickelten Systeme signifikant verbessert werden konnte.

Abstract

This thesis examines the integration of Agile Software Development Methodologies with Usability Engineering. The process of integrating Extreme Programming and User-Centered Design is outlined. It is shown that agile development methods facilitate user-orientation, iterative user interface development, and the involvement of usability engineers. The approach described is an adaption to the classical Extreme Programming process and integrates the following Human-Computer Interaction instruments: User Studies, Extreme Personas, Usability Expert Evaluations, Usability Tests, and Automated Usability Evaluations. The application developed within this research project enables a user to perform content based search for audio and video content in large databases and play it on a mobile phone. Reflections on the integrated development process used, based on data collected through code analysis and process evaluation tools, as well as notes of process retrospective review meetings, are discussed. Likewise, the results from usability tests of the application being developed are presented.

From the agile, Extreme Programming based, usability process of this scientific project three Agile Usability Process Patterns were derived. The patterns depict the integration of three of the used Human-Computer Interaction instruments with an iterative software development approach and are named: Usability Expert Evaluation, Usability Test, and Automated Usability Evaluation. In order to prove the validity of these patterns, they were implemented and evaluated in an industrial, Extreme Programming based, project. Both processes were evaluated using an Extreme Programming Evaluation Framework. For each of the processes, context factors were recorded and adherence metrics data (quantitative and qualitative) was collected at two points in time. The pattern implementation results showed that the usability and the overall user experience of the developed systems improved significantly.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter introduces Agile Software Development and Usability Engineering and provides an overview about Agile Usability Process Integration Problems, being the problem statement and the motivation for the choice of the subject of this thesis. Furthermore, it gives a condensed overview about the structure of this thesis.

1.1 Motivation for Agile Software Development

Software development processes are an attempt to structure and standardize development to make the outcome of a project plannable and predictable. The first structured approach was the waterfall model, which gained popularity during the 70s. This traditional approach places its focus on extensive planning and structured processes in order to make development an efficient and predictable activity. The waterfall model consists of a time-ordered list of activities, where one activity can only be executed after the previous one has been completed. The major disadvantage of this approach is its rigidity and inflexibility. For this reason, a shift in software development paradigms has taken place. The newly emerging agile software development paradigms are iterative and light-weight. They place less emphasis on the process and its deliverables, but focus on principles and values. Furthermore, agile approaches are an attempt to address major problems in traditional software development [21]:

- Changeability: In software engineering a number of external and internal changes affect development.
- Software's complex and intangible nature: In software engineering requirements are pervasive, dynamic and rarely well-defined. Consequently, systems are difficult to be specified entirely in advance.

- Heavy processes and lack of feedback: In software engineering changes cause delays and increasing costs. User feedback on what is being developed is often far too slow. Processes should be able to handle changes and allow rapid user feedback.

- Process focus: Software development processes are not truly repeatable. Software development has more to do with individual skill and adaptability than strictly following plans and process descriptions.

1.2 Definition of Agile Software Development

The response to more traditional software processes is the idea of developing software using a light-weight or agile approach. This trend started in the 90s. An important step was taken in 2001. A group of leading software methodologists gathered in Snowbird, Utah, USA, to discuss light-weight development practices. To support their ideas on light-weight software development, the group agreed to describe the lowest common denominator in the form of four core values and twelve other principles. These four values plus twelve principles, which describe the values in greater detail, constitute the definition of agile software development and are known as *Agile Manifesto*. The four values are [31]:

1. Individuals and interactions over processes and tools.

   The first value is attending to the people on the team as opposed to roles in the process chart. Although a process description is needed to get a group of people started, people are not plug-replaceable. The second choice being highlighted there is attending to the interactions between the individuals. New solutions and flaws in old solutions come to life in discussions between people. The quality of the interactions matters.

2. Working software over comprehensive documentation.

   The working system is the only thing that tells what the team has built. Documents showing the requirements, analysis, design, screen flows, object interaction sequence charts, and the like are handy as hints. The team members use them as aids in reflecting on their own experience, to guess what the future will look like. Documents can be very useful but they should be used along with the words "just enough" and "barely sufficient".

3. Customer collaboration over contracted negotiation.

   The third value describes the relationship between the people who want the software built and those who are building the software. The distinction is that in properly formed agile development, there is no "us" and "them", there is only "us". Instead of depending solely upon contracts, the customers work in close collaboration with the development team. Saying "there is only us" refers to the fact that both are needed to produce good software.

4. Responding to change over following a plan.

   Plans are useful in software development, and each of the agile methodologies contains specific planning activities. The agile approach advocates planning for and adapting to changes, as opposed to prescribing strict conformity to a plan in every situation This is necessary because the prerequisites for most systems will evolve during development. Moreover, the initial requirements will be influenced by the fact that communication between people is always more or less incomplete.

Consequently, agile is not a distinct, well-defined process. Instead, it is a generic term and common ground for several different processes or methods, each sharing a set of software development core ideas, values and principles.

In essence, agile methods can be characterized as [99]:

- Iterative: A full system is delivered initially, then the functionality of each subsystem is changed upon each subsequent release.
- Incremental: The system, as specified in the requirements, is partitioned into smaller subsystems by functionality. New functionality is added upon each new release.
- Self-Organizing: The team has the autonomy to organize itself in order to best complete the work items.
- Emergent: Technology and requirements are "allowed" to emerge through the product development cycle.

1.3 Agile Software Development in Projects

According to [22] a project needs to comply with different software project characteristics to be placed in the agile home ground, respectively to be carried out using an agile development methodology.

- Application Characteristics: The primary project goals are delivering rapid value and responding to change. The project employs smaller teams and the project environment is turbulent, highly dynamic and project-focused.

- Management Characteristics: The customers are dedicated on-site customers and are focused on prioritized increments. Planning and control consists of internalized plans and qualitative control. Communication relies on tacit interpersonal knowledge.

- Technical Characteristics: Requirements are prioritized informal stories and test cases and are undergoing unforeseeable change. Development is based on simple design and short increments and refactoring is assumed expensive. Testing comprises executable test cases which define the requirements.

- Personnel Characteristics: Project customers are dedicated, collocated, collaborative, representative, authorized, committed, and knowledgeable performers. The development team of the project consists of at least 30 % full-time Cockburn Level 2 and 3 experts [22]. This means that 30 % of the development team is able to revise a method, breaking its rules to fit an unprecedented new situation, or is able to tailor a method to fit a precedented new situation. The rest of the team is able to perform discretionary method steps such as sizing stories to fit increments, composing patterns, compound refactoring, or complex commercial off-the-shelf product integration with training. The project culture consists of many degrees of freedom to define and work on problems, so that the people feel comfortable and empowered.

1.4 Definition of Usability Engineering

According to ISO 9241, Part 11, usability is 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 [3]. This definition ties the usability of a system to specific conditions, needs, and users. In addition to that, it requires establishing certain levels of usability. Usability engineering defines the target usability level in advance and ensures that the software developed reaches that level. It is a process through which usability characteristics are specified, quantitatively and early in the development process, and measured throughout the process [63]. Traditionally, usability is associated with five usability attributes [118]:

- Learnability: The system should be easy to learn, so that the user can rapidly start getting some work done with the system. It can be assessed by measuring the time a user spends working with the system before that user can complete certain tasks in the time it would take an expert to complete the same tasks. This attribute is crucial for the user experience of novice users.

- Efficiency: The system should be efficient to use, so that once the user has learned the system, a high level of productivity is possible. It refers to the number of tasks per unit of time that the user can perform using the system.
- Memorability: The system should be easy to remember, so that the casual user is able to return to the system after some period of not having used it, without having to learn everything all over again.
- Error rate: The system should have a low error rate, so that users make few errors during the use of the system, and so that if they do make errors, they can easily recover from them. Further, catastrophic errors must not occur.
- Satisfaction: The system should be pleasant to use, so that users are subjectively satisfied when using it.

1.5 Usability Engineering Process

An abstracted generic usability engineering process for building a system with the desired level of usability is outlined in [45]. This process, which most usability practitioners apply with slight variation, is structured around a design - evaluate - redesign cycle. Practitioners initiate the process by analyzing the targeted users and the tasks the users will perform. When applying usability testing late in the development, the discovered issues are costly or even impossible to fix. Therefore, it is crucial to perform usability evaluation during the product development process, which ultimately leads to an iterative development process. A pure waterfall approach to software development makes introducing usability engineering techniques fairly impossible. The suggested process in [45] consists of two different phases.

Usability Analysis Phase. First, the users and their needs, expectations, interests, behaviors, and responsibilities, all of which characterize their relationship with the system, need to be known.

1. User Analysis.

   There are numerous approaches for gathering information about users, e.g., site visits, focus groups, surveys, and derived data. The most important thing about user analysis is to record, structure, and organize the findings.

2. Task Analysis.

   Task analysis describes a set of techniques people use to get things done. Identified tasks are prioritized and serve as a starting point for development.

3. Usability Benchmarks.

   Usability benchmarks are set as quantitative usability goals, which are defined before system design begins. They are based on the five basic usability attributes described in the previous section.

Usability Design Phase. Once the tasks the system will support are analyzed, a first attempt at the conceptual design of the user interfaces is made, which will be evaluated and possibly improved in the next iterations.

1. Conceptual design.

   During the conceptual design phase, the basic user system interaction and the objects in the user interface and the contexts in which interaction takes place are defined. The phase ends with evaluating the results as paper prototypes.

2. Visual design.

   In this phase, the user interface appearance is defined. The deliverables are prototypes that must be tested.

1.6 Usability Engineering Methods

To ensure a software project has the essential usability characteristics, different usability engineering methods are applied during development: usability inspection methods (without end users) [64], usability test methods (with end users) [64], and usability enhancement methods [119].

Usability Inspection Methods. This is a set of methods for identifying usability problems and improving the usability of an interface design by checking it against established standards. These methods include heuristic evaluation, cognitive walkthroughs, and action analysis.

Heuristic evaluation is the most common informal method. It involves having usability specialists judge whether each dialogue or other interactive element follows established usability principles.

A *cognitive walkthrough* is a task-oriented method by which the usability analyst explores the system functionalities. It simulates step-by-step user behavior for a given task.

The *action analysis* method is divided into formal and back-of-the-envelope action analysis. In both, the emphasis is more on what the practitioners do, than on what they say they do. The formal method requires close inspection of the action sequences a user performs to complete a task. Back-of-the-envelope analysis is less detailed and gives less precise results, but it can be performed much faster.

Usability Test Methods. Testing with end users is the most fundamental usability method and is in some sense indispensable. It provides direct information about how people use the systems and their exact problems with a specific interface. There are several methods for testing usability, the most common being thinking aloud, field observation, and questionnaires.

Thinking aloud may be the single most valuable usability engineering method. It involves having an end user continuously thinking out loud while using the system. By verbalizing their thoughts, the test users enable to understand how they view the system, which makes it easier to identify the major misconceptions of end users.

Field observation is the simplest of all methods. It involves visiting one or more users in their workplaces.

Questionnaires are useful for studying how end users use the system and their preferred features, but need some experience to design. They are an indirect method, since this technique does not study the actual user interface: it only collects the opinions of the users about the interface.

Usability Enhancement Methods. One of the most popular usability enhancement methods is the Personas method. It was developed as a tool for raising empathy for the end users in development teams, and as a means for communicating peer group definitions. When developing Personas, archetypical prototypes of end-users are designed. This is done by accumulating knowledge about intended peer groups. One persona represents a typical user group. A persona is an archetypical figure being able to guide decisions about product features, navigation, interactions, and even visual design (among other factors) [119].

1.7 Agile Usability Process Integration Problems

When trying to integrate Usability Engineering into Software Engineering one is faced with several integration problems. This section covers these integration problems, being the problem statement and the motivation for the choice of the subject of this thesis.

1. The Meaning of Usability

   Usability means different things to different people [133]. The following definitions illustrate how the term usability has been perceived differently in three distinct standards [134]:

- The capability of the software product to be understood, learned, used, and attractive to the user, when used under specified conditions [2].
- 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 [3].
- The ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component [1].

2. The Cultural Gap Between Psychologists and Engineers

   Usability Engineering specialists, who are often psychologists, are sometimes regarded as mere nuisances who get in the way of those who, in the end, will really deliver the product, the software engineers [133]. These two groups do not share the same culture and perspective and they do not understand the respective constraints under which each group has to operate [135]. Having a solid understanding of the Human-Computer Interaction and Software Engineering culture and practices can help both software developers and usability experts to update their methodologies, and to learn techniques for improving and mediating their lines of communication [134].

3. The Modularity Fallacy

   Traditional interactive system architectures decompose the system into subsystems that are relatively independent, thereby allowing the design work to be partitioned between the user interface and underlying functionalities. Such architectures extend the independence assumption to usability, approaching the design of the user interface as a subsystem that can be designed and tested independently from the underlying functionality. This Cartesian dichotomy can be dangerous, as functionalities buried in the logic of the application can sometimes affect the usability of the system [133].

4. The Responsibility Gap

   The role of the user interface is often perceived as that of decorating a thin component sitting on top of the software. Software engineers build the software, and afterwards the usability people make the interface layer user-friendly. The usability people, on the other side, view their role as designing the interface first, which is later on implemented and evaluated. These views of each others role are of course in direct opposition and often result in frustrations within one group for not being given sufficient influence on the final product [135].

5. Educational Gap Between Software Professionals and Usability Professionals

A barrier to the wider practice of Usability Engineering methods is that their techniques are still relatively unknown and difficult to master, making them inaccessible to small and medium-sized software development teams and individual developers [133]. This explains why Usability Engineering methods often cannot be fully used [134]. Also, the resulting difficulties of communication between the software engineering team and usability specialists can seriously compromise the integration of usability in the software development lifecycle [135].

6. The Dispensability of Usability Engineering

   Some software managers feel that their project cannot afford to spend so much time on usability. They worry that the Usability Engineering iterations will never end, due to Human-Computer Interaction people trying to get everything perfect. There are two answers to this. First of all, there should be measurable usability objectives set as part of the project plan. And secondly, these managers should consider the longer-term effect of quality work on the productivity of their developers [133].

7. Organizational Shift

   The organizational learning approach asserts that the integration of Usability Engineering into software engineering lifecycles is not primarily a problem of a lack of Usability Engineering methods. The natural inertia of the organization is the obstacle and the solution must be also understood as a problem of organizational learning and software process improvement [133]. In organizational terms, Usability Engineering must be understood not merely as a process improvement to Software Engineering, but as a paradigm shift [135].

8. Empirical Evidence

   To empirically evaluate the value of a specific Usability Engineering method using classical scientific techniques, it would be necessary to compare the same project repeated under conditions employing Usability Engineering techniques versus not employing Usability Engineering techniques, while controlling for skill, motivation, Software Engineering approach and other possible differences between the two project teams [135]. This challenging experiment would need to be repeated many times with different project teams, different software engineering frameworks and on different projects in order for the results to achieve statistical validity. For this reason, there is a lack of empirical evidence [133].

9. The Usability of Usability Engineering Methods

The Usability Engineering methods which are considered to be reasonable for application by engineers are often not used by them for the following interrelated reasons [135]:

- There is no time allocated for Usability Engineering activities: these activities are not integrated in the development process or in the project schedule.
- Knowledge needed for the performance of Usability Engineering tasks is not available within the development team.
- The effort for the application of the Usability Engineering tasks is estimated to be too high because the tasks are regarded as time-consuming.

10. The Lack of Process Support Tools

    Most often software developers working on user interface design and evaluation lack Usability Engineering tools. Yet, as the need for usability becomes recognized by software development organizations, they tend to develop their own in-house tools and sometimes define or reinvent the whole usability engineering toolbox or life cycle [134]. Usability Engineering methods should still be regarded as knowledge-intensive. Thus, tools to provide engineers with the knowledge of how to effectively perform Usability Engineering activities are needed [135].

1.8 Results of this Thesis

- A software development process integrating Extreme Programming and User-Centered Design:

  The integrated approach to application development presented in this thesis (Chapter 2, 4, and 5) focuses on the adoption of Extreme Programming and User-Centered Design, emphasizing iterative user interface development involving usability engineers and non-technical users. The approach described integrates Human-Computer Interaction instruments. The implemented instruments are: User Studies, Extreme Personas, Usability Expert Evaluations, Usability Tests, and Automated Usability Evaluations. Furthermore, it is shown how the integrated process facilitates user-orientation and at the same time preserves the social values of the development team. In addition, experiences in using this integrated process for two years in application development are reported (Chapter 6).

- Agile Usability Process Patterns:

  This thesis presents three Agile Usability Process Patterns (Chapter 7) extending an already existing agile process pattern language [40]. The patterns described are: Usability Expert Evaluation, Usability Test, and Automated Usability Evaluation. These patterns are derived from the agile usability process of this scientific, Extreme Programming based, project. In addition to the pattern definitions, the pattern implementations in this scientific project are outlined. In order to prove the validity of the patterns, they were implemented and evaluated in an industrial, Extreme Programming based, project. Both processes were evaluated using an Extreme Programming Evaluation Framework. The pattern implementation results showed that the usability and the overall user experience of the developed systems improved significantly, as evaluated by the usability expert. A high number of usability issues found in Usability Expert Evaluations was fixed by means of test driven development, namely, writing an automated usability test for each of the discovered usability issues first. Resulting issues of Usability Tests were incorporated and fixed the same way. This, as well as the employment of Automated Usability Evaluation metrics in test driven application development, increased the usability of the developed systems over time to a very high degree.

- A working, usability tested, application developed by means of the integrated agile usability process:

  The application being developed within this research project is a mobile multimedia application that enables a user to perform content based search for audio and video content in large databases and play it on a mobile phone. Features of the application are described (Chapter 2, 5, and 8). The application was developed by means of the integrated agile usability process presented in this thesis, and was usability evaluated on many different levels. Furthermore, results of a usability study are presented (Chapter 5).

1.9 List of Publications

Within the context of this thesis, the following publications have been created:

1. (2007) User Interface Design for a Content-aware Mobile Multimedia Application: An Iterative Approach [69]

2. (2008) User Interface Design for a Mobile Multimedia Application: An Iterative Approach [70]

3. (2008) Optimizing Extreme Programming [68]
4. (2008) Probing an Agile Usability Process [151]
5. (2008) Integrating Extreme Programming and User-Centered Design [72]
6. (2008) Agile User-Centered Design Applied to a Mobile Multimedia Streaming Application [71]
7. (2009) Integration of Extreme Programming and User-Centered Design: Lessons Learned [74]
8. (2009) Concept and Design of a Contextual Mobile Multimedia Content Usability Study [73]
9. (2010) Process Patterns for Agile Usability [116]
10. (2010) Agile Usability Process Patterns [115]

1.10 Structure of this Thesis

This thesis is organized as cumulative thesis and divided into chapters. Each chapter, except Chapter 1 and Chapter 10, is a refined version of one or more publications. The following list contains short descriptions of the contents of the chapters of this thesis.

- This introductory chapter covers Agile Software Development and Usability Engineering. Furthermore, it provides an overview about Agile Usability Integration Problems, being the problem statement and the motivation for the choice of the subject of this thesis.
- Chapter 2 examines the context in which this research took place. Namely, the development of a mobile multimedia application that enables a user to perform content-based search for audio and video content in large databases and play it on a mobile phone. In addition, an approach to application development, focusing on the adoption of agile software development methodologies and user-centered design, is presented. This chapter is a refined version of the publications [69] and [70].
- In Chapter 3, reflections on the used Extreme Programming process, based on the data collected through code analysis and process evaluation tools, as well as notes of process retrospective review meetings, are discussed. This chapter is a refined version of the publication [68].

- Chapter 4 presents adaptations to the classical Extreme Programming process. The approach described integrates Human-Computer Interaction instruments. The implemented instruments are: User Studies, Extreme Personas, Usability Expert Evaluations, Usability Tests, and Automated Usability Evaluations. This chapter is a refined version of the publication [151].

- Chapter 5 describes an adapted development process: the integration of Extreme Programming with User-Centered Design. It is shown how an agile development technique facilitates user-orientation and preserves the social values of the development team. Furthermore, a summary of the results of a usability study of the application developed by means of the adapted development process is outlined. This chapter is a refined version of the publications [71] and [72].

- In Chapter 6, the reflections of a retrospective workshop are examined. The reflections represent the lessons learned after using the integrated process outlined in Chapter 2, 4 and 5 for one and a half years. This chapter is a refined version of the publication [74].

- Chapter 7 discusses three agile usability process patterns: Usability Expert Evaluation, Usability Test, and Automated Usability Evaluation. The patterns described are derived from the agile usability process of this scientific, Extreme Programming based, project. Also, a validation implementation of the patterns in an industrial, Extreme Programming based project, is presented. This chapter is a refined version of the publications [115] and [116].

- In Chapter 8, the concept and design of a contextual mobile multimedia content usability study is described. The features of the developed application used in the study are outlined. This chapter is a refined version of the publication [73].

- Finally, in Chapter 9, general conclusions about the achieved results are drawn and interesting topics for future research are outlined.

Chapter 2

User Interface Design for a Mobile Multimedia Application: An Iterative Approach

Mobile phones have become full-featured mobile computers. Applications providing good user experience and taking full advantage of the increasing capabilities of mobile phones are still rare. One such application is audio and video on mobile phones, which is expected to become a killer application in the near future. A lot of valuable audio and video content is hidden in archives of content providers. We are developing an application that enables a user to perform content-based search for audio and video content in large databases and play it on a mobile phone virtually anywhere, at any time. Our approach to application development focuses on the adoption of agile software development methodologies and user-centered design, emphasizing iterative user interface development involving usability engineers and non-technical users. Thus, the application evolves according to the needs of the end user, providing maximized usability and customer satisfaction.

2.1 Introduction

Mobile computing is leading a revolution. Our lives are changing at a pace never experienced before in human history. A wide variety of applications for mobile phones is available at the moment. Still, there are not so many full-featured applications which utilize the available bandwidth and are accepted by the users.

Studies show that multimedia – Audio and Video (AV) – consumption is on the edge to become one of the next killer applications for mobile devices [38]. User behavior in consuming AV is changing. Traditional broadcasting is losing more and more audience because online and mobile AV intrudes heavily into this area. A recent report states that 43 % of Britons, who watch video regularly from the Internet or on a mobile device, are now watching less TV than before [12]. Clearly, it is in the interest of broadcasting companies to adapt to these changes in user behavior and invest in these new technologies. For these companies, one of the major advantages of mobile phones, compared to other devices, is that they can charge for their services easily and directly, as the existing infrastructure can be reused. Additionally, the possibility to place advertisements for specific user groups is a huge benefit. At the same time, customers are given the flexibility to access rich multimedia content from anywhere, at any time.

The major problem for an average user is the combination of the overwhelming amount of multimedia content available and unsatisfactory user interfaces for accessing it. Usability is the key success factor for such applications.

For this reason, we are developing an application that enables a user to perform content-based search for AV content and play it on a mobile phone. This content includes radio and TV archive material, such as documentaries or other recordings of historical, political and cultural importance, discussion programs, movies, music videos, audio books, and music. The AV content will be stored in a database containing transcribed speech from the clips, as well as additional metadata, such as titles or a summary, where available. The media delivery will be based on standard web technology. This will enable people to use this service with almost any modern mobile phone. Furthermore, the application addresses not only the emerging functional and cognitive needs of the users, but also the objectives of the content providers. The application is designed keeping in mind the social interaction aspects of users. The system provides different community-building features to encourage interaction amongst them. The aim is to build a community platform for mobile phone users, where they can share their views and interests about AV content provided by the system. The feedback from the community will reflect the current trend of multimedia consumption as well.

One of the research goals of this project is to apply usability test procedures for mass-market applications on mobile phones. At present, usability testing for mobile phones is cumbersome and too expensive for small and medium sized enterprises. Another objective of this project is to automate certain parts of the usability testing procedures and provide a testbed for effective and efficient mobile usability testing. Special emphasis is placed on the adoption of agile software development methodologies, in particular Extreme Programming (XP), for mobile phones and their user interfaces.

This chapter presents an overview about related work. The following section presents different usage scenarios. Afterwards, the usability engineering process, which is applied to the iterative UI development cycle, is described. The next section defines various user-based recommendation approaches. Finally, a conclusion is given.

2.2 Related Work

This section provides an overview of related work. Basically, existing work can be divided into the categories applications and iterative user interface design. Related work in each of the categories is examined in the following subsections.

2.2.1 Applications

This section provides an overview of different AV search and streaming applications. They can be categorized by means of features they offer, particularly speech recognition, which is used to enhance the meta data for advanced search and streaming on mobiles.

Mobile YouTube [155] and MobiTV [29] allow to search for content and stream it on mobile phones. Mobile YouTube has the big advantage to have Google as parent organization, but the content is very limited and consists only of user-contributed material. In comparison to that, MobiTV offers different realtime worldwide TV channels streaming without search functionality, but for streaming on a mobile phone, client-side software is required.

In contrast to this, Blinkx [19], TVEyes [145], and EveryZing [43] perform speech recognition in order to enhance the search with additional meta data but do not provide mobile phone streaming. Blinkx provides user-contributed content as well as content from broadcasting companies. The content offered by TVEyes is restricted to news material, and the content of EveryZing is restricted to web based content.

Other applications offering online video search and streaming are JumpCut [81] and Joost [79]. Neither has the feature of speech recognition or streaming on mobiles. Jumpcut has the big community of Yahoo users but is limited to user contributed content. Joost provides more content, but is based on peer2peer technology requiring its own client-side software still in its beta phase.

The comparison shows that there is no application offering both features, speech recognition and streaming on mobiles, at the same time, which our application does.

2.2.2 Iterative User Interface Design

There already exist approaches of combining agile methodologies and Usability Engineering [65][105][34]. However, to the best of our knowledge none applies this specific composition of practices: we combine XP [14], user-centered design, and an iterative user interface design approach utilizing different HCI instruments such as user studies, extreme personas (a variation of the personas approach), usability expert evaluations, usability tests, and automated usability evaluations.

2.3 Usage Scenarios

In daily life, many people are at work or school at day time. They have no time to watch TV or listen to radio programs, because of the schedules set by the broadcasting companies. Time-delayed, played-back, and individually-delivered AV on mobile phones provides a new platform taking radio and TV into the street, car, public transport, waiting room, park, and virtually any other location. This type of application creates additional audiences, eager to access multimedia content during new prime times set by themselves, that is, in their commuting periods or other idle times. Also, a market survey shows that consumers are interested in using this technology and are ready to pay a realistic price for these services [29]. The basic idea of such a system can be illustrated by the following sample usage scenarios.

2.3.1 TV Archive for Subway Riders

- A commuter in the subway searches for “Fernando Alonso”.
- The system matches each word of the textual search query with the positions it occurs in each AV clip.
- The system presents a list of clips in which the name “Fernando Alonso” has been mentioned, sorted by temporal occurrence and relevance based on content, e.g., how often the name was mentioned.
- The user selects one of the presented entries.
- The system’s media server streams the selected clip to the user’s mobile phone.

2.3.2 Radio Archive for Car Drivers

- A user listens to the last sentences of a radio broadcast about the “European Constitution”. The user still has to travel with the car for some time and therefore searches for the keyword “European Constitution”.
- The system lists a number of related news items, interviews, and documentaries.

- The user selects the desired topic.
- The handsfree set of the mobile phone plays back the selected material through the car's stereo.

2.3.3 Media Recommendations for Users

- A user wants to consume AV content but has nothing particular in mind.
- The user asks the system for recommendations.
- The system generates recommendations based on the user's stored preferences and on the recent behavior of other users. A short description of each item is provided as well.
- The user selects an item and plays it on the mobile phone.

If the user interrupts the media stream, in all scenarios it is possible to resume at the previous location at any time, even weeks later. This feature is unavailable with regular broadcasting or streaming systems. The user of this system has more flexibility for consuming the AV content. This is particularly important because of the short continuous viewing or listening periods. For example, while commuting, interruptions and (possibly much) later resumptions will be the regular case.

Such behavior is rather uncommon for AV consumption so far, especially for viewing video. But it is not so much different from the way a book is read, having breaks between reading periods. Thus, it seems plausible that users will be willing to switch to this new way of listening and viewing AV content with interruptions, as it brings them the convenience of being able to decide what to consume in a just-in-time way, independent of place and time.

2.4 Usability

User interface design determines the success or failure of almost any application. Massive AV consumption on mobile phones will be accepted only, if users can easily find what they are searching for. But search on a mobile phone presents unique challenges as compared to a PC [106]. The inherent interface limitations of mobile phones strongly constrain the choices of user interface and interaction design. Special attention has to be paid to the constraints of small screens [88], possibly unfavorable lighting conditions, and limited text input capabilities.

We propose an iterative and user-centered approach to user interface design and system development in order to solve the stated problems.

2.4.1 Iterative User Interface Design

Usability is evaluated in small iterative steps to gain insight into whether the users' functional and cognitive requirements are met. User interface prototypes of the system are developed and tested throughout the development process. As a result the fidelity of the prototypes increases and evolves.

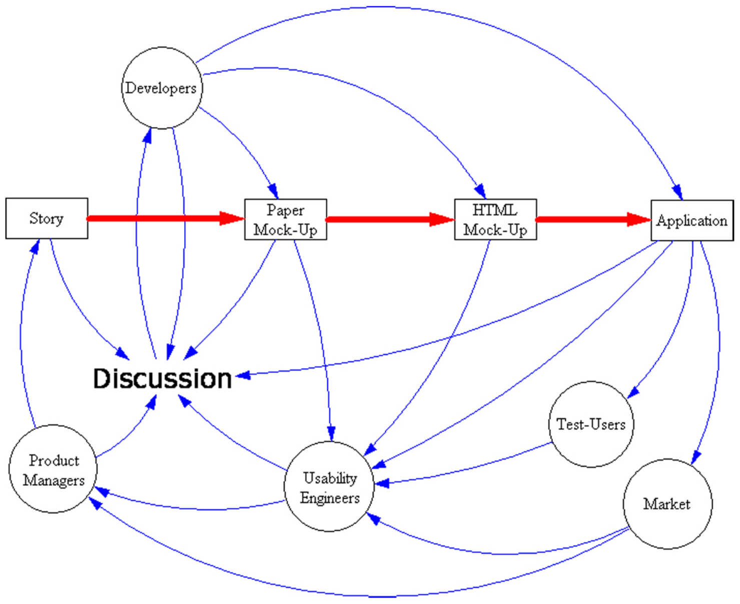
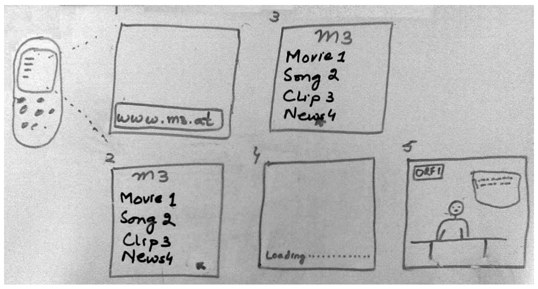


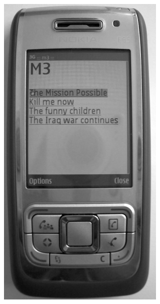
Figure 2.1: Iterative User Interface Design Workflow.

The workflow presented in Figure 2.1 illustrates the iterative design approach. The process starts with the creation of user stories by the customer or the product manager who acts as a representative of the customer. Developers create different paper mock-ups to collect and present ideas. A final mock-up is derived, serving as the basis for further development. The benefit of using paper mock-ups for the interaction design is that they can be designed and modified quickly. Because of that, the feedback given by the usability engineers and the users can be incorporated easily. An additional advantage is that it is easier for users to criticize simple and rough mock-ups compared to ones which look neat and perfect from the graphic design perspective [131]. For simple interaction designs, a paper mock-up suffices as a basis for further discussions and the implementation. In more complex cases, an additional HTML mock-up is created based on the final paper mock-up.

The approach combines the quick feedback-and-change cycle of hand-drawn paper mock-ups with the more time-consuming process of computer-based prototypes. Paper mock-ups are used to get the basic concepts right, while HTML mock-ups are



(a) Paper Mock-Up.



(b) Application on Mobile.

Figure 2.2: From Paper Mock-Up to Mobile: The first Search-Results Screen.

used for a more detailed view.

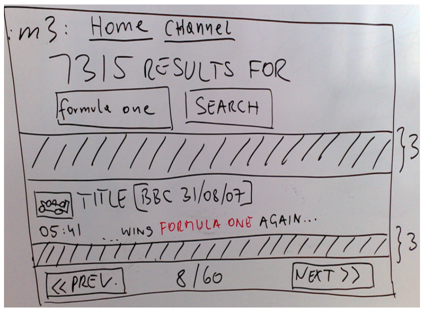
The designs are examined by usability engineers and tested by non-technical users. The feedback from the usability engineers, as well as from the users, is taken as input for further refinements of the design. Also, the results are incorporated into automated tests which are used, by employing test driven development, as an executable specification for the actual implementation. This feedback-and-change cycle provides insights into whether the user interface design is meeting different usability criteria.

For the actual user tests it is important to choose representative test users from different age groups, bearing in mind the targeted customers for the proposed service. These tests are conducted only after incorporating the feedback from the usability engineers on the paper as well as the HMTL mock-up. Therefore, the expensive part of involving real users can be done more effectively.

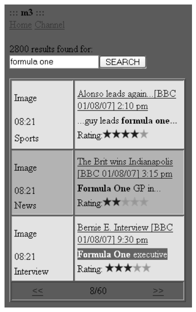
2.4.2 An Iterative Design Example

For the paper mock-up in Figure 2.2(a) the usability engineer raised the following issues:

- Missing strategy for displaying larger result sets (balance between pagination and scrolling).
- Missing feedback mechanism to highlight the pointed-to item (especially needed in unfavorable lighting conditions).
- Undefined application behavior after playback of the clip ends (no return option specified).



(a) Paper Mock-Up.

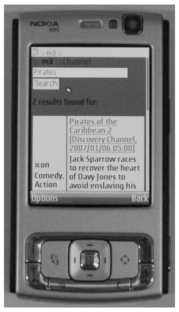

::: m3 :::
Home Channel

2800 results found for:
formula one SEARCH

Image 08:21 Sports	Alonso leads again...[BBC 01/08/07] 2:10 pm ...guy leads **formula one...** Rating:★★★★☆
Image 08:21 News	The Brit wins Indianapolis [BBC 01/08/07] 3:15 pm **Formula One** GP in... Rating:★★☆☆☆
Image 08:21 Interview	Bernie E. Interview [BBC 01/08/07] 9:30 pm **Formula One** executive Rating: ★★★☆☆

<< 8/60 >>

(b) HTML Mock-Up.



(c) Final Application.

Figure 2.3: An additional HTML Mock-Up: A refactored Search-Results Screen.

Figure 2.3 shows the mock-ups of an improved version of the same feature. Here, an HTML mock-up was created after the paper mock-up. The design was derived from the following user scenario (a so-called user story in XP [14]):

Search results presented to the user should contain clip-related information which can aid the user in choosing the clip. Also, the context in which the keyword was found, as well as the number of search results, should be visible. Furthermore, it should be possible to start a new search immediately.

It can be seen that two issues from the previous feedback, namely pagination/scrolling and item highlighting, have been addressed in the refactored design. On the refactored design depicted in Figure 2.3 the usability engineer provided the following feedback:

- Forms: It is common to leave some white space between text-input-field and the button. For graphical user interface solutions there are distances fixed in guidelines for the operating system - for mobiles we recommend to put the button in a second line (this is preferred to putting the button close to the input field).

- Background Colors (Table): The alternating rows should vary decently, and should preferably be coloured in slightly different shades - the selected colors are "eye bending".

2.4.3 User-Centered Application Design

User interface development cannot be separated from the development of the underlying application. Intended user interactions strongly influence the internal structure and functionality of the system [33]. A big issue in mobile user interface practice is that current approaches are not sufficient for mobile phones [139]. Therefore, another focus is placed on the combination of iterative user interface design and user-centered application design.

The design process and user tests provide feedback about the user interface which will be used for the system's functional requirements. It reveals the mental model of the users, how they expect the system to work. The assessment of each feature from the perspectives of the users influences the whole development process of the application and addresses the problem that conversation only with the stakeholders is not enough [77]. As the application development is done in short iterations, the developers are able to refactor the system continuously according to the feedback derived from the parallel, iterative, user interface design process. Hence, the system evolves according to the needs of the end user and the specifications derived from actual usage.

2.5 User-Based Recommendations

Web-based companies already use recommendation systems with great success. Amazon, for example, has millions of customers. Seeing the benefits of recommendations, Amazon has developed its own technique called "item-to-item collaborative filtering". Their customers regularly take advantage of these recommendation facilities when making their purchases [98].

The personalized approach of our system makes it possible to implement user-based recommendations. The unique identification of a user is necessary for accounting purposes, implying that a user profile has to be managed by the system. This profile will be augmented with additional data, which is used for recommendations to the same and to other users. The data is collected by means of two information acquiring models, the interactive model and the behavior-based model.

2.5.1 Interactive Model

The interactive model is based on user ratings. After users finish consuming an item, they are able to rate it according to their liking and preferences. Information about which clips a user consumed is stored in his individual profile. In addition, the corresponding clip ratings are stored as well. The rating of a specific clip in each user profile affects the overall rating of the clip in the database. The individual ratings are still traceable. For more personalized recommendations, ratings of similar user groups can be combined.

2.5.2 Behavior-Based Model

The behavior-based model is applied by collecting usage data. Information about the clips consumed, and the duration of the consumption, is stored in the user profiles. This is used for user-specific recommendations. If many users stop the same clip after a short time, this clip is most likely not very interesting. Of course, this equally depends on the overall playing time. Therefore, a ratio measure is used for clip rating. The system will take into account that users are allowed to stop and resume clips at any time, which can influence the measurements. Alternatively, it is possible to consider ratings of a specific user group only, as described in 2.5.1.

2.5.3 Model Combination

W hen generating recommendations the two models already described are combined. For the system, user ratings are more important than usage data. However, ratings may not be available for every item. In this case, only behavior data is used. Furthermore, the changing preferences of users are taken into account by adding a time-descending weighting factor.

There are different scopes for the rating mechanism. On the one hand, all users are considered, and on the other hand, just a specific user group is considered. This results in different recommendations. The default recommendation setting can be overridden by user preferences.

2.5.4 Implications

An attractive feature of the system is the possibility to target advertisements more precisely. This feature is useful for companies wanting to address specific user groups. Additionally, users benefit, because they receive only advertisements related to their interests. For example, Google's Gmail is using this technique for advertising purposes on its popular mail accounts. The large user base of Gmail is a valuable target for business. The advertising is tailored to users' mail contents. Gmail also offers the possibility to use mobile devices [53]. As Google has purchased YouTube., it is expected that this trend will continue. The advertising and search capabilities are, or will be soon, extended to video content as well [106].

The feature of collecting additional user data provides continuous feedback, enabling constant improvement of the system. By recording this information, valuable data about how the user is interacting with the system is obtained. This allows to react quickly to new usage patterns and needs as they arise.

2.6 Conclusion

The emerging technologies of delivering rich AV content on mobile phones will result in reducing the number of users for traditional TV and radio broadcasting services. This might compel traditional TV and radio broadcasting companies to become partners in this technology by opening their huge collections of AV content to the public. Today's consumers are willing to pay a reasonable price for this service [29]. According to current trends, the community of mobile phone application users will grow rapidly. The standards concerning codecs, formats, and technical infrastructure, required for AV content delivery on mobile phones are already well established. These general trends are in favor of the development of this type of application.

The critical factor for this kind of applications will be user acceptance, which heavily depends on the fact that the system suits the user's needs. To address this issue, in our software engineering process, usability engineers are accompanying the system development team during the whole project life cycle. The engineers provide suggestions that are incorporated continuously into the system. This process is facilitated by the short development iterations and has proven to provide early and valuable feedback. The test-driven development approach allows to convert these findings into a set of automated tests. These tests define the functionality of the system, serve as specifications for the development, and prevent previously discovered usability problems from reappearing. Furthermore, the inclusion of test-users provides additional benefits. This continuous input allows to adjust the system effectively according to the end-user's needs.

Chapter 3

Optimizing Extreme Programming

The vast amount of published literature explaining the "right way" of doing Extreme Programming shows, that in practice there simply is no single right way. Even though Extreme Programming is a simple and slim process, it has to be tailored to the nature of each team and project in order to provide the benefits it promises.

Our team has been working on a project employing the Extreme Programming methodology, experiencing unique issues arising from the distinct project setup and team composition, as well as the additional academic interests in the project. Initially, we aimed at applying "pure Extreme Programming", but it became more and more obvious that for our project some of the practices just cannot be applied in their "pure" form. The concrete interpretation of these practices determines if Extreme Programming can be applied successfully in the context of a team and a project.

In order to reach an optimized process for our project, we continuously evaluate different approaches of applying Extreme Programming practices on short release basis. We have noticed that some practices can be adopted directly, while others need to be tailored according to the unique environment. In this chapter, we reflect on our process based on the data collected through code analysis and process evaluation tools, as well as notes of process retrospective review meetings. The lessons we have learned can also help other teams to lead them to an optimized Extreme Programming process for the success of their projects.

3.1 Introduction

The number of software projects constantly increases, but the overall success rate is still rather low [102]. Many projects fail because of their inability to cope with the changing user requirements. Heavy up-front design without continuous feedback from the customer is another factor. To have a greater probability of success, the developers need a software development process which should be flexible enough to cope with the constantly changing requirements and which is also people-oriented. Agile software development methodologies have emerged in response to these needs, as agile methods give more value to individuals, working software, and change [101].

The intention of large scale research into software engineering techniques has been a formulation of an ideal methodology that can consistently and predictably lead to software development success [109]. A recent survey shows that agile software development has seen far better success rates in comparison to other methodologies [8]. Being an emerging agile methodology, Extreme Programming (XP) offers a number of practices, values, and principles, which are advised to be adopted in order to run a software development project [14]. XP is being experimented in different ways to make it fit to the specific needs of the projects as well as the development teams [143].

This is of interest for many academic, research and development organizations, as there is a room for more explorations in the area of agile development methodologies. Many experience reports in the field of agile research have been presented, helping other teams passing through the same process [143]. However, there is still a need for more experience reports of teams already using XP, giving valuable information for those, who are planning to adopt the XP methodology. In addition, this data also serves the purpose of defining the agility level of software development teams. In this chapter, we present our own experience about the XP methodology which we have gained by applying it to a software development project.

The next section describes our project environment. In Section 3.3 our XP process, focusing on the practices applied in our project, is presented. Section 3.4 describes our reflections. Finally, a conclusion is presented.

3.2 Project Environment

We are developing a multimedia streaming application for mobile devices as a testbed to analyze the XP methodology. XP is being applied in a progressive manner: each practice is consciously applied and constantly evaluated in order to yield process improvement. Hence, each team member is not only taking part in the software development process, but is also making a critical analysis of the way the process is being used. The objective of the project is twofold: on the one hand, having a software product fulfilling the user requirements, and on the other hand, XP process optimization as profound academic research.

We have been working on the project since summer 2007. The project's duration is three years, which is quite appropriate for the development of the product, as well as for the team members to carry out the research for their doctoral studies.

We are a team of six regular members: five developers and a product manager. The product manager plays the role of the "On-Site Customer", enabling the implementation of this XP core practice. Furthermore, he communicates with the partners who come from various domains, including telecommunication, content providing, and hardware infrastructure. Also, developers communicate directly with the engineers of a partner usability research company regarding usability issues.

This project's main scientific and academic goal is the analysis of agile software development methodologies. Another goal is research in the field of mobile application usability. Additionally, the business partners are interested in commercial aspects of the project. Figure 3.1 shows the allocation of the application, research, and business aspects after the first one-month release.

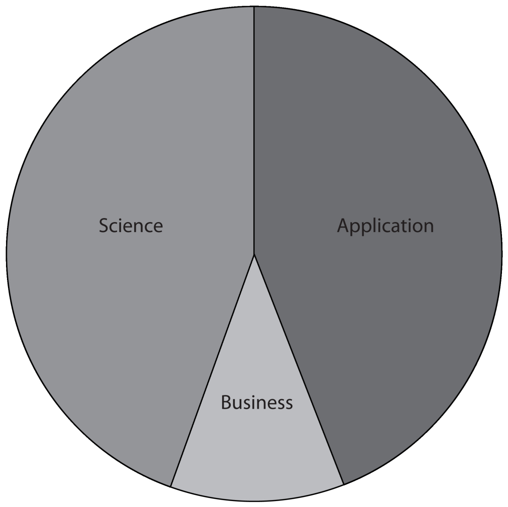


Figure 3.1: Application, Research and Business Aspects in a Release.

3.3 Process

It was pre-decided that XP will be used as a methodology. Therefore, effort was made to establish a basis for its implementation. None of the team members had worked in an agile development environment before, so available literature, especially [13][14], was used for initial guidance and reference. The team tried to apply all those main practices which could be applied at the initial stage of the project. In this way, some of the basic practices were adopted fully, while others were implemented partially, or in modified form. The following subsections outline how the practices have been implemented, as well as the current status.

3.3.1 Fully Implemented Practices

Small Releases

We aim to release a working version of our application to the project partners on a regular basis. In the early stages of the project, the duration of one release cycle was set to one month. This enabled us to quickly get feedback on our work from the partners in order to sharpen our vision of the project goal. As the project took shape, the release size was gradually increased to two and finally to three months. For now, we are satisfied with three-month release cycles, which complies with the quarterly planned business targets of the partners.

For tracking short-term progress, releases are further divided into iterations. Initially, we used a one-week iteration duration, but later we shifted to two week iterations in order to reduce the administrative overhead added by the iteration planning meeting.

The Planning Game

The planning meetings are held on iteration and release basis. Release meetings are attended by all project partners who, as stakeholders of the project, identify and define user requirements. These requirements are then are formulated as XP user stories.

In our project, we distinguish three main types of user stories: application, business, and science.

- Application stories are "traditional" XP user stories, representing features of the application.
- Business stories describe diverse business activities, e.g., collaboration with partners, meetings, presentations, etc.
- Science stories are only relevant for our team, hence they are not specified during the release planning meeting. They depict the academic and research activities of the team, e.g., paper-writing, performing process analysis, etc.

The stories generated during the release planning are written down on story cards and are prioritized by the participating partners. To visually represent the different story types, we use a simple color encoding for the different story types: application story cards are white, business cards are green, and science cards are yellow. Then, the developers estimate the time required for implementing the stories. Also, stories created in former release meetings that have not yet been implemented are re-estimated, if required. The final step of the release planning is to select a subset of the available stories. This is done by prioritizing the available stories and selecting as many top-priority stories as “fit in” the available velocity. The amount of user stories to be scheduled is determined by following calculation: the sum of their estimates is lower than or equal to the sum of the estimates of the user stories finished in the previous release.

The iteration meeting is held at the beginning of each iteration and is attended only by team members including the product manager. The product manager takes the role of the on-site customer and selects and prioritizes stories for the current release. The stories are then broken down into detailed tasks, which are again estimated by the developers. The intended results of the tasks are explicitly defined by writing acceptance criteria.

Figure 3.2 and Figure 3.3 show user story cards of release and iteration plannings stuck on whiteboards.

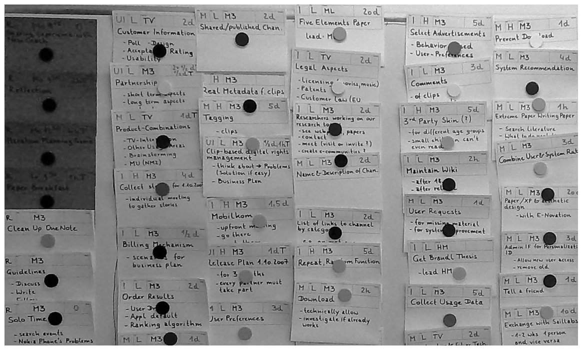


Figure 3.2: Selected Story Cards on the Release-Board (Release Planning).

Pair programming

All production code is written by two developers working together on the same machine with one screen, one mouse and one keyboard [13][14]. This practice has been implemented from the first day. It helped us in sharing the project-specific knowledge and improving the technical skills of the developers.

We also applied working in pairs to non-technical stories. For example, the product manager pairs with a developer when writing customer-acceptance tests

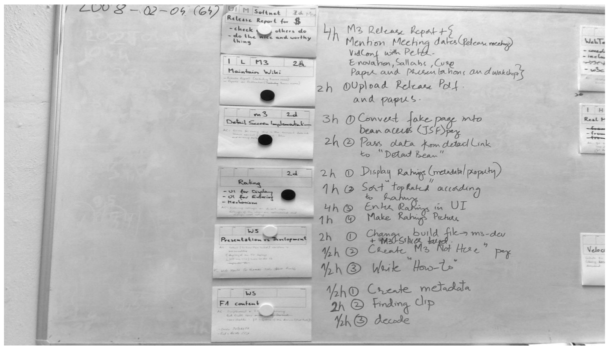

2008-02-05 (64)
4h M3 Release Report + {
Mention Meeting dates (Release meeting)
VidConf with Peter
E-novation, Sallabs, Cure
Paper and presentations and workshops }
2h ① Upload Release Pdf. and papers.
3h ① Convert fake page into bean access (JSF) page
2h ② Pass data from detail Link to "Detail Bean"
2h ① Display Ratings (metadata/property)
1h ② Sort "topRated" according to Rating
4h ③ Enter Ratings in UI
1h ④ Make Ratings Picture
2h ① Change build file → m3-dev
½h ② Create "M3 Not Here" page
½h ③ Write "How-To"
½h ① Create metadata
2h ② Finding clip
½h ③ decode

Figure 3.3: Selected Story Cards on the Iteration-Board (Iteration Planning).

and when creating business assignments requiring technical knowledge. Working in pairs on research stories was not successful. For this kind of stories everyone works solo.

In daily stand–up meetings, the developers sign up for tasks according to their interest. The pair partners are chosen voluntarily [136].

Sit Together

The team members including the product manager sit in one large room at their private workspaces. There are three separate pairing stations in the same room. Due to sitting in the same room, the face–to–face communication has resolved many difficulties which arose within the project, the team, and the process.

Collective Ownership

A Subversion repository is used for managing the code base. The code is shared by all developers. Whenever a chance for code improvement is identified and there is enough time at hand, the required actions are performed on the spot. The changes are communicated in the stand-up meetings, during pair programming, and sometimes through a short ad-hoc discussion involving all developers. One basis for a successful application of collective ownership is the strict adherence to coding standards.

40-hour Week

The purpose of the "40-hour week" practice is that developers should not work overtime, because tired developers make more mistakes during coding [61]. We strictly follow this practice.

3.3.2 Partially Implemented/Modified Practices

On-Site Customer

As the target group of the product being developed within in project is manifold, we cannot directly implement the on-site customer practice. Therefore, initially the product manager, as well as the developers, played the role of the customer. But soon, many shortcomings of this approach became apparent. The absence of common acceptance criteria for the stories resulted in long discussion cycles in the planning meetings and the implementation. We also felt difficulties in the prioritization of the stories. To overcome those problems, we decided that the product manager, who also communicates with our project partners, will play the role of the customer.

Metaphor

As everyone was new to the process and the project, there was no common understanding of the metaphor – the shared terminology about the project and the process [13][14]. This resulted in misunderstandings about the features to be implemented, which eventually led to the delaying of their delivery. The release planning meetings with partners, our internal iteration planning meetings, stand-up meetings, retrospective meetings, and pair programming have contributed to evolvement our metaphor.

Simple Design and Refactoring

From the very beginning the team aimed at keeping the design as simple as possible. The design started with creating paper prototypes to visualize customer requirements illustrating the customer how the requirements will be put into reality. An important factor of being able to keep the design simple is refactoring. For our team, simple design has been beneficial, because it facilitated the incorporation of changes demanded by the partners. Refactoring of code is not a routine practice in our team, but it is done on demand basis, that is, whenever any developer sees the opportunity to improve the code, or when we need to substantially change the application fundamentals, e.g., the usage of a new framework.

Test-first Programming

As all the team members were new to XP, it was difficult to follow the XP style of writing the failing automated test before any code [13][14].

Figure 4 shows a graph comparing lines of executable code, lines of test code and test coverage. For this, the data was collected using Emma, a Java code coverage tool

[42], and LinesOfCodeWichtel [100], and is based on the work performed during the second release of the project. The low amount of test code and test coverage shows that the practice of testing is not well exercised by the team. For the subsequent releases, the implementation of this practice has improved. If not being impossible due to framework restrictions, tests are being written beforehand.

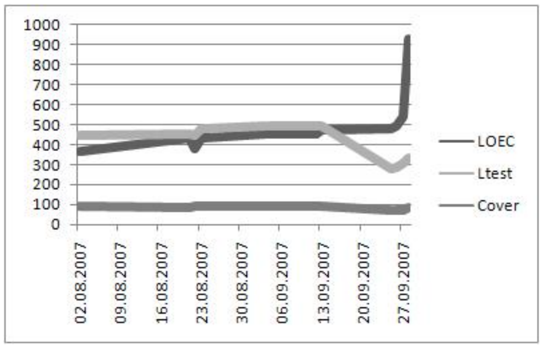


Figure 3.4: Executable Code versus Test Code and Test Coverage.

3.4 Reflection

In order to measure the performance of the team and to resolve human issues, a retrospective review meeting on the process is held after every iteration and release. This retrospective meeting is called “reflection meeting”. It has helped us a lot to find out the reasons for difficulties faced during the process and their remedies. The common decisions that we take after these meetings are noted down and followed by all team members. Almost all XP values – communication, simplicity, feedback, courage, and respect – and XP practices adhere to human aspects [13][14]. The benefits of sitting together, face-to-face communication, feedback, stand-up meetings, the planning meetings, pair programming, and reflection meetings have contributed to improve not only our process, but also increased the overall morale of the team.

To review the development process, we collect empirical data from various sources describing our performance of each applied XP practice.

For a qualitative analysis, we perform the Shodan 2.0 survey [89] on a regular basis (e.g., at the end of each release). Additionally, we use quantitative data generated by the XP tracker tool “XPlanner” [152], and different code analysis tools [42][100].

The data gathered using Shodan 2.0 Input Metric Survey shown in Table 1 gives an overview about the methodology and the extent to which a given XP practice is applied. As there was an explicit effort to apply these practices, low percentages

Table 3.1: Subjective Metric (Shodan 2.0 Input Metric Survey).

Testing metrics	%
Test First Design	44
Automated Unit Tests	68
Customer Acceptance Tests	22
Planning metrics	%
Stand-up meetings	92
Short Releases	86
Customer Access / On-Site Customer	48
Planning Game	96
Coding metrics	%
Pair Programming	98
Refactoring	66
Simple Design	76
Collective Ownership	86
Continuous Integration	100
Coding Standards	84
Sustainable Pace	82
Metaphor	46

indicate that either the team was not fully content with the practice, or the practice needs to be tailored for our project. For example, the team members perceived that pair programming was practiced for almost every development task, while metaphor was given the lowest rating because of the unfamiliarity of the team members with the project. These conclusions are also supported through iteration and release reflection notes and from key discussion points raised in stand-up meetings.

3.5 Conclusion

After working with XP practices for almost one year, our experience shows that most of the practices are helpful for a project with multiple objectives (in our case, research, application development and business). Pair programming helps in spreading knowledge. The benefits of co-location, face-to-face communication, stand-up and planning meetings, and retrospective review meetings have contributed to improve not only our process, but also to increase the overall morale of the team. The low ratings of some practices indicate that our team still needs more experience to apply them in a proper way.

We continuously try to optimize our approach to XP. Future results will help software development teams working under similar environments to improve their development process for the success of their projects.

Chapter 4

Probing an Agile Usability Process

In this chapter, we describe adaptations to the classical Extreme Programming process. The approach described integrates Human-Computer Interaction instruments. The implemented instruments are: User Studies, Extreme Personas (a variation of the Personas approach), Usability Expert Evaluations, Usability Tests, and Automated Usability Evaluations. By combining Extreme Programming and User-Centered Design processes we take advantages of both approaches.

4.1 Introduction

This case study sums up the setup of an adapted Extreme Programming (XP) process. The goal of the adaptation was to examine the applicability of a usability-aware agile software development process. The process was designed to allow applying Human-Computer Interaction (HCI) instruments in XP teams. This should combine the advantages of the XP methodology (on-time delivering, optimized resource investments, short release cycles, working high quality software, tight customer integration) with the advantages of a User-Centered Design (UCD) process (usable, accessible, and accepted products, end-user integration).

The context, where these instruments are utilized, is a project where we develop a mobile multimedia application. The application enables users to create customized mobile multimedia channels based on keywords. The keywords are matched against metadata and transcriptions. Here an example: it is possible to create a news channel with all news-broadcasts mentioning ”formula one”.

4.2 Problem Statement

Experts doubt that the XP process leads to true user-centered design [67]. Following issues can prevent the integration of HCI instruments into XP processes:

- Ad-hoc Input. Because of the short release cycles software engineers would need ad-hoc usability input during the software development. In practice usability input is not given adhoc, but after longer periods (on average between one to two weeks average). XP practitioners cannot accept such time-spans.
- Patchwork Experience. Our work-experience suggests that users often experience software as patchwork when developed from bottom up like done in XP. We attribute this to the missing holistic view at the beginnings of XP projects. The programming activities are not based on a design concept.
- Cultures. Another problem is the difference between cultures: software engineers, on the one hand, and HCI experts, on the other hand, come from different domains with different attitudes, approaches, backgrounds, and even different ways to express themselves. The XP process requires tight cooperation in teams, which reveals differences between engineers and HCI experts very quickly: engineers have a technical approach to software development whereas HCI experts mainly have a psychological background, hence taking a cognitive view on the software development. These differences can lead to problems. Methods to prevent this have to be integrated into the collaboration process.
- Technical Focus. Unit tests in XP environments are designed for technical testing. Hence, the focus is on technical functionality ignoring usability issues. This means that the technical view of testing has to be expanded by HCI approaches and means.
- On-Site Customer Representative. From an HCI point of view the inclusion of customers is a step into the right direction. Nevertheless, the Manifesto for Agile Software Development does not clearly demand end-users as customers [101]. We expect deficits in usability if it is not clearly stated that end-users have to be part of the process. Developers need to have a clear picture of the end-users.

4.3 Background

Creating usable high quality software is not trivial, especially when it comes to mobile applications. Different processes and tools are already in place to create high quality and usable software. They come from different domains (engineering, design, HCI). The processes and tools we chose from are:

- Extreme Programming. The agile software development process.
- Extreme Programming and the User-Centered Design process. The combination of software engineering and HCI skills.
- Personas. A tool to define end-user groups by archetypes to raise empathy for end-users in software development teams. We extend the persona approach to "Extreme Personas" to fit the changing requirements appearing during the project duration.
- Automated Usability Evaluation. A tool and the technical "glue" between HCI knowledge and the XP unit tests. We want to extend known code based usability evaluation by semantics to create (semi-) automated test-scripts which can be included into the set of technical unit tests. These extended unit tests allow HCI experts to directly apply HCI knowledge in the process

4.3.1 Extreme Programming

XP is an agile software development process. The goal of this process is to deliver high-quality software in time. This is done by different means: test-driven development (by using unit tests), short iteration cycles, on-site customers, pair-programming, refactoring (restructuring an existing body of code, altering its internal structure without changing its external behavior), and user stories (where requirements are captured in short narrative stories) are the most important of them [13][14]. Figure 4.1 shows the basic XP process where we will build our adapted approach on (release cycles are 3 months and iteration cycles are 1 week).

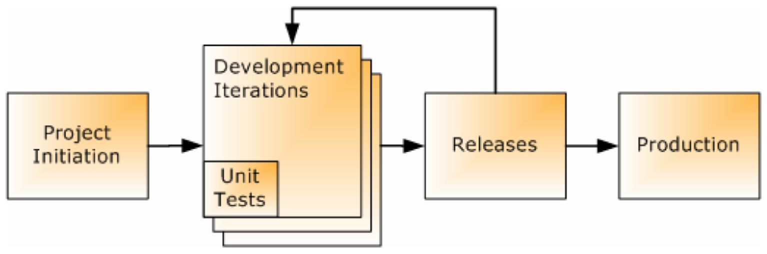


Figure 4.1: Agile Development Process.

4.3.2 Extreme Programming and User-Centered Design

Practitioners already combined UCD and XP by varying approaches [65][120][34]. McInerney, P. and Maurer [105] showed already that a marriage of these two approaches is possible. In our study, we investigated the prerequisites and the process of such an integration.

4.3.3 Personas

The Personas method was developed as a tool for raising empathy for the end users in development teams, and as a means for communicating peer group definitions. When developing Personas, we design archetypical prototypes of end-users. This is done by accumulating knowledge about intended peer groups. One persona represents a typical user group. A persona is an archetypical figure which can guide decisions about product features, navigation, interactions, and even visual design (among other factors) [119].

4.3.4 Automated Usability Evaluation

The idea of Automated Usability Evaluation (AUE) is not new. Basic research goes back to the early nineties [59][10]. In the year 2000 the state of AUE is still described as quite unexplored [76]. Current approaches tend to focus on multimodality [123] and mobile devices [148]. Besides their limited scope (because most tools evaluate on a code basis) for our project the existing tools have a big disadvantage: most of them are isolated solutions for HCI experts. Hence, they hardly integrate seamlessly into the existing development processes.

4.4 Agile Usability Process

The following subsections outline the general approach of our integrated process, the HCI instruments used, as well as their interplay.

4.4.1 General Approach

The novelty of our approach is that we do not rely on one or two selected instruments but took five of them and integrated them into the XP process. Figure 4.2 shows the interplay of the HCI instruments related to the XP process. Applied correctly in different phases of the project, the instruments are designed to reach the goal of maximized software quality (in terms of technical quality but also in terms of usability).

As depicted in Figure 4.2, the modified agile development process includes the following usability instruments: User Studies, Personas, extended unit tests, Usability Tests, and Usability Expert Evaluations. It can be seen that end-users are integrated in two different ways. On the one hand, User Studies inform the development and extension of the Personas, which gives indirect end-user input to the developers. On the other hand, usability tests (as part of the usability evaluations) directly inform development.

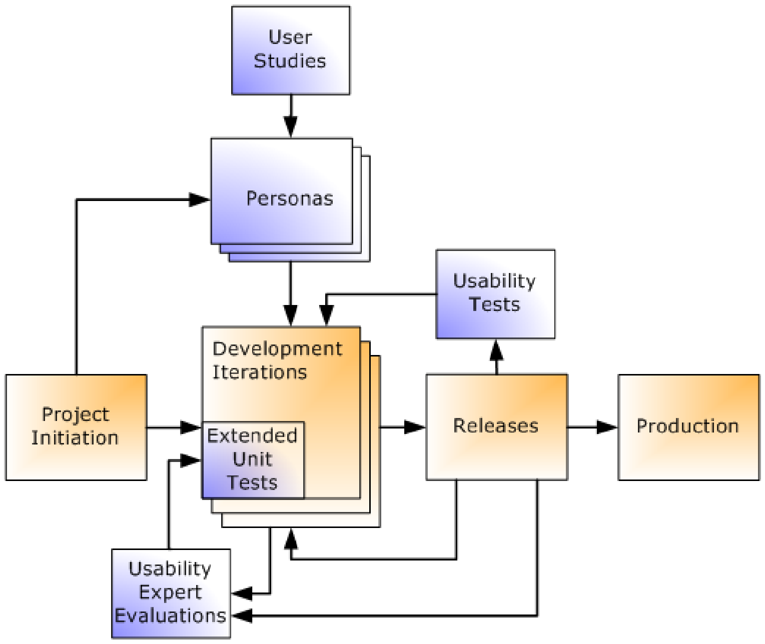


Figure 4.2: Agile Development Process including HCI Instruments.

The idea was to integrate these five instruments into the classical XP process. This multi-instrument approach was developed to solve the problems described in the problem statement above.

The five HCI instruments we rely on are:

- *Extended Unit Tests* for Automated Usability Evaluation.
- *Extreme Personas* (a variation of the classical personas method) extend the typical XP user stories.

- *User Studies* (focus groups, diaries, laddering interviews) are not inherently foreseen in the XP process and are integrated to extend the XP concept of the on-site customer.
- *Usability Expert Evaluations* to solve the ad-hoc input problem.
- *Usability Tests* to solve the problem of the on-site customer representative.

4.4.2 Extended Unit Tests

In XP unit testing is mandatory. Our approach extends the technical unit tests by adding usability-specific test cases. Code based tests are enhanced with semantics to achieve this goal, e.g., code based tests can check against guidelines like the usage of capital letters on buttons. When adding semantics (the correct label of a button) we can include the test into the set of unit tests already used in XP. Test-driven development in XP means to write tests first. The written tests then define the behavior of the application. Adding usability related unit tests with semantics allows us to define the usability of the application. Unit tests by definition test small definable units of the software. The problem of patchwork application suggests using a holistic approach for testing. Therefore, the unit tests are extended by tests which go beyond single units, and test complete interaction flows.

4.4.3 Extreme Personas

This approach starts with the same activities like the classical persona method: preliminary user groups are defined and Personas are modeled for them afterwards. During User Studies new knowledge leads to two distinct actions: when the new knowledge suggests slight changes for a persona the persona will be refactored, if the found knowledge reveals that current Personas do not cover the new insights new Personas will be developed. These actions make the classical Personas extreme by applying the XP paradigm of small iterative steps and refactoring, which is extending the Personas in this case. During the coding phases, the developers pin the Personas beside the user stories. Their first application is in the planning game (the phase where user stories are created), where the Extreme Personas yield as reference representation of the on-site customer.

4.4.4 User Studies

User Studies are the instrument for getting knowledge about the end-users. The outcome of the User Studies informs the design in two ways: on the one hand, knowledge for creating and extending the Personas is created, and on the other hand, direct input for the user stories can be derived.

4.4.5 Usability Expert Evaluations

Usability Expert Evaluations solve the problem of adhoc input. This is done by instant messaging, email, and video-conferencing. Mock-ups (in early phases) and screens (in later phases) are sent to the HCI experts who then give ad-hoc input by using the channels mentioned above.

4.4.6 Usability Tests

Usability laboratory tests will include end-users as demanded by the UCD process but not demanded by the XP process (where it is not mandatory that end-users are part of the on-site customer representative).

4.4.7 Interplay of Usability Instruments

How do the single instruments work in practice? Based on the XP process HCI, experts can intervene in three ways:

- Assist in the creation of user stories (this was done by providing HCI knowledge derived from studies, literature, and tests).
- Contribute by writing automated usability tests.
- Extend XP methods (we brought Extreme Personas to the development team).

The knowledge derived from the used HCI instruments informed all of these activities.

4.5 Conclusion

We use the described process since summer 2007. The project will end in 2010. Then, the final usability tests will prove, if the process has been able to enhance usability of the applications. Until now, we did not experience any cultural problems. The HCI experts in the project are well integrated into the development team.

The tight coupling of different expertise has led to a high motivation among project members. Developers gained insight into the subtleties of UCD, HCI experts learned to understand the origins of some of the usability problems.

Furthermore, we saw that the diverse technical testing frameworks demand technically aware HCI experts. Depending on the used frameworks, the programming expertise required varies. In practice, this could get a problem when the chosen framework is complex and little time for learning is available.

Chapter 5

Agile User-Centered Design Applied to a Mobile Multimedia Streaming Application

Mobile computing is leading a revolution. Multimedia consumption on mobile devices is increasing day by day. The most important factor for the success of such applications is user acceptance. Additionally, the success of a software development project is associated not only with tools and technologies, but also depends on how much the development process is user-centered and developer-oriented. We are working on a project to develop a multimedia streaming application for mobile phones. This chapter describes our adopted development process: the integration of Extreme Programming – one of the popular agile methods – with User-Centered Design. Furthermore, it is shown how the integrated process facilitates user-orientation and at the same time preserves the social values of the development team. This chapter also presents a summary of a recently carried out usability study.

5.1 Introduction

The most important factor for the success of a software application is user acceptance. An inherently usable and technically elegant application cannot be considered a success if it does not satisfy the end-users' needs. End-users are often left out of the development process [107]. Agile development processes involve a customer as a business representative who is responsible to specify the business value of user requirements, but this customer needs not necessarily to be a real end-user.

Agile methods are becoming popular nowadays. Being a lightweight agile method, Extreme Programming (XP) has the advantages of: on-time delivery, co-located team, relying on the team members' knowledge rather than documentation, optimized resource investments, short release cycles, working high quality software, tight customer integration, incremental design, constant communication and coordination, rapid feedback, continuous refactoring, pair programming, and test driven development [13][14][5]. XP is a collection of well-known software engineering practices. XP aims at enabling successful software development despite vague or constantly changing software requirements. The novelty of XP is based on the way the individual practices are collected and lined up to function with each other [5]. It is also a people-oriented process with many social core practices.

Usability measures the quality of a user's experience when interacting with a product or system. User-Centered Design (UCD) is an approach for employing usability [146]. UCD, also called human-centered design, is an approach to user interface design which is based on information about the people who will use the product. UCD processes focus on users throughout planning, design, and development of a product [147]. Holzinger emphasizes that every software practitioner should be aware of different usability methods and apply them according to specific situation of a project [64].

There already exist approaches of integrating agile methodologies and Usability Engineering (UE) / UCD [65][48][105][34][108]. Memmel et al. point out that when UE becomes part of agile software engineering, it helps to reduce the risk of running into wrong design decisions by asking real end users about their needs and activities [108].

The focus of both methodologies, XP and UCD, on users makes it possible to integrate them [54]. The integrated process allows to combine benefits of both methodologies and makes it possible to reduce the shortcomings of each. XP needs to know its true end-users and UCD benefits from a flexible and adaptive development methodology which runs throughout the project life-cycle [144]. We integrate XP and UCD in our project, where we are developing an application that enables a user to perform content-based search for audio and video content and play the streamed content on a mobile phone [72]. The end-users are indirectly involved in the process by our use of different Human-Computer Interaction (HCI) instruments such as user studies, personas, usability expert evaluations, usability tests, and automated usability evaluations [151]. Usability of a mobile application is an important ongoing research issue. Numerous studies address UE / UCD issues for mobile applications [16][66][87]. We conduct various usability studies and in this chapter a summary of one of the studies is presented.

Section 5.2 outlines the similarities between XP and UCD. Section 5.3 examines the project environment. Section 5.5 describes the adopted process. Section 5.6 provides the details of a usability study. Section 5.7 concludes the chapter.

5.2 Similarities between XP and UCD

The core values of XP [14] and UCD [56] are applied to solve different issues. In XP, a simple implementation fulfilling the minimum requirements of the application is created and iteratively extended, while UCD tries to continuously improve the usability of the user interface. However, when comparing some of the core values it seems obvious that the two development processes can benefit from each other's practices.

5.2.1 End-User Involvement

One of the core practices of XP is to have a *Customer on Site* who is co-located with the programmers in order to answer domain-specific questions and give feedback on the system. This practice can be matched well with the testing of prototypes with actual users as proposed by UCD. Especially, if the customer is also the real end-user or if developers have direct access to end-users.

5.2.2 Continuous Testing

Continuous and extensive testing is at the heart of XP. It is mainly embodied by two practices: *Continuous Integration* runs all existing automated tests whenever the code base is changed or extended in order to check if the changes caused any undesired side effects. Most of these tests emerge from *Test-Driven Development*. First, automated tests checking the desired behavior are created. Then, the actual behavior is implemented and can be evaluated right away with the tests. This is usually done only for pure behavioral code, but can be extended to user interfaces. Tests can check the expected behavior of an interface, and these tests can be run whenever the code is changed.

The end-user tests of UCD are a valuable source for test targets. An unexpected user action that caused a problem in the application can be replicated as an automated test. By executing this test in the *Continuous Integration* process it is ensured that the problem, after solving it once, does not reappear.

5.2.3 Iterative Development

Both, XP and UCD, propagate an iterative procedure of design and development [56][13][147]. An XP project yields *Small Releases* (another core XP practice) on a regular and frequent basis (usually a few months). Each release version is based on the previous one, incorporating new features and fixing bugs of the predecessor. Inside a release time frame, work is organized in "iterations" (usually taking one to four weeks). On an even smaller scope, many feedback-and-change cycles take place, especially in conjunction with *Test-Driven Development* and *Refactoring* (the practice of changing source code in order to improve its quality without changing its functionality).

UCD also proposes a design–test–modify circle for developing user interfaces. The scope of iterative development in XP and UCD differs. Releases and iterations in XP are mainly organizational units and *Refactoring* is considered to be a development tool. Inc contrast to this, UCD's iterative user interface refinement is a more explicit process as its involvement of external persons (the test users) makes it more complex. Nonetheless, iterative interface development of UCD fits well into the iteration principle of XP, because both approaches are aware of the value (and necessity) of evolutionary development.

5.3 Project and Team Setup

We are working in a project where we develop an application that enables a user to perform content-based search for audio and video content and play it on a mobile phone. The project started in summer 2007 and will end in 2010. The application enables a user to search not only in the metadata but also in the spoken words of the AV clips. This content includes radio and TV archive material, such as documentaries or other recordings of historical, political and cultural importance, discussion programs, movies, music videos, audio books, etc. The application is being designed keeping in mind the social interaction of users and provides some Web 2.0 features [70].

In addition, one goal of the project is the analysis of agile software development methods, particularly XP, and to devise a usability test procedure for mass applications on mobile devices with emphasis on UCD and iterative user-interface design.

The team consists of six full-time regular members having different social and cultural backgrounds, five developers and a product manager who plays the role of the *On-Site Customer* of XP.

The customer communicates with the project partners who come from various domains, including user interface design, usability research, telecommunication, content providing, and software-hardware infrastructure. Also, developers communicate directly with the engineers of a partner usability research center regarding usability issues. The usability engineers working for our project are also active in UCD research with the team.

5.4 Application Features

The user interface of the current prototype comprises the following main features. The application's main screen provides the features: "Search", "Top rated", and "Most recent" clips. It provides links to the "Channel" and "Media Feeds" pages, as well as a link to the "Clip Detail" page when one clicks on the title of a clip. The application also implements a few other Web 2.0 features like "Recommended" and "Most viewed" clips.

Search:

"Search" allows to search the whole AV content by entering keywords. It displays the search results ordered by broadcast date (if any). For each result item, the clip's title, link, description, duration, originating channel (if any), and a representative thumbnail image are shown. The user can play a clip by clicking on the respective link.

Channel:

"Channel " allows to browse the schedule of TV and radio channels. A channel lists the original program schedule of the current day, but users can browse the schedule of previous days or can search within the channels. Users can select the date and the time (either Morning, Afternoon, Prime time, or Night), where the system then displays the list of clips in the selected time period, ordered by broadcast time. The resulting items are shown in a similar format as in the "Simple Search" result list with rating stars and a channel icon for each clip in the schedule. The top of the page also provides a dropdown list for selecting channels.

Media Feeds:

The "Media Feeds" feature is intended to provide the users a facility to create and consume a constantly updated stream of clips based on the users' search criteria. This media feed can be sent to a friend by SMS or email.

Clip Detail:

The system shows the "Clip Detail" page when users click on the title of the clip. Users can rate a clip, add a comment, or view all comments. With the "Tell a friend" feature, users can send a clip to their friend by SMS or by email.

5.5 Agile Usability Process

The following subsections describe the process which is followed in application development.

5.5.1 Approach to User-Centered Design

User interface design plays an important role in the acceptance of a web based application. The overall process of our approach to UCD is based on evaluating the usability of the application in small iterative steps. This helps us to gain insights into the functional and cognitive requirements of real users. We design prototypes of the user interface of the system and test them throughout the development process. As a result the fidelity of the prototypes increases and evolves.

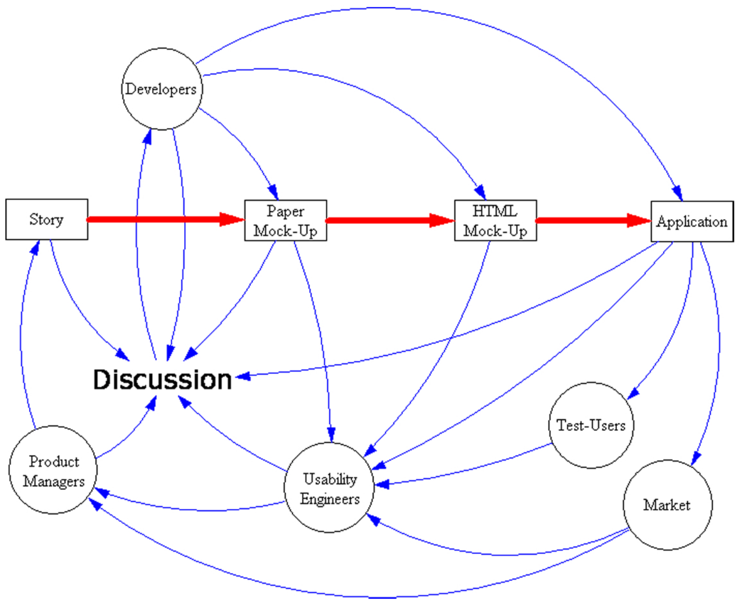


Figure 5.1: Iterative User Interface Design Workflow [70].

The work flow presented in Figure 5.1 illustrates the iterative design approach incorporating UCD into our XP process. From a broad perspective, the application development cycle starts with defining user stories, then comes to mock-up designing, and at the end to the actual implementation. The process is executed as follows:

- Different feature-related user stories of the application are created by the customer along with partners.

- Developers create different paper mock-ups for each of the required features to collect ideas and to present them to the customer.
- The customer decides which of the mock-ups best suits his needs or suggests modifications to the mock-ups.
- A final mock-up is derived according to customer's wishes, which then serves as the basis for the actual implementation.
- Once the implementation mock-up of a feature or a group of related features is finished, the usability engineers are asked to give feedback on it.
- After incorporating the feedback given by the usability engineers into the application, end-user tests are conducted by the usability engineering team.
- The feedback on the application from the usability engineers, as well as from the test-users, is taken as input for further refinements of the user interface design of the application.
- The results are then incorporated into automated tests which serve as an executable specification for the actual implementation.

This feedback-and-change cycle provides insights into whether the user-interface design is meeting different usability criteria. As the application development is done in short iterations, the developers are able to refactor the system continuously according to the feedback derived from the parallel, as well as iterative, UI design process. Hence, the system evolves according to the needs of the end user and the specifications derived from actual usage.

5.5.2 Choosing the Type of Mock-Up

We make use of two different types of mock-ups; low fidelity paper mock-ups and high fidelity implementation mock-ups. The benefit of using paper mock-ups for the interaction design is that they can be designed and modified quickly. For simple interaction designs, a low fidelity paper mock-up suffices as a basis for further discussions and the implementation. An additional advantage is that it is easier to criticize simple and rough mock-ups compared to ones, which look neat and perfect from the graphic design perspective [131]. But for some features a high fidelity mock-up is required to clearly visualize the interface. As we have the benefit of an on-site customer co-located with the development team all the time, for those tasks a quick implementation mock-up is designed and immediately presented to the customer. This implementation mock-up is then modified based on the immediate feedback of the customer. If our customer would not have been co-located with us all the time, it would have been difficult to have the benefit of this quick feedback-and-change cycle.

5.5.3 Frequency of End-User Tests

The end-user tests are made on an on-demand basis. That is, when the customer says that now is the appropriate time, from the business point of view, to run a usability test with test-users. Also, when there is enough amount of new functionality added to the application, it becomes effective to perform usability tests and then proceed with development. It would have been good if user tests could have been made on regular basis, e.g., at the end of each release, but considering the expenses and resources required for, it we have kept it only on an on-demand basis. So, the expensive part of involving real users is done more effectively.

5.5.4 Integration of HCI Instruments

Figure 5.2 describes our model of integrating HCI instruments (user studies, personas, extended unit tests, usability tests, and usability expert evaluations) into the XP process [151]. It shows the interplay of the HCI instruments with the XP process. When applied correctly in various phases of the project, the instruments are designed to reach the goal of improved software quality not only in terms of technical quality, but also in terms of usability. End-users are integrated in two different ways. On the one hand, user studies are taken into account to develop personas [82]. The personas specify the direction of development by guiding the customer in identifying user stories and are extended at the end of the iteration when the vision about the user has broadened. This serves as an indirect end-user input for the development process. On the other hand, feedback from usability tests performed by test-users as part of the usability evaluations serves as a direct input for further enhancement and development of the application [151].

5.5.5 Testing Issues

A big issue in mobile user-interface design practice is that current approaches are not sufficient for mobile phones [139]. For designing any software, use of UCD practices ensures that the product is accepted by the users [84]. This further supports the use of the UCD approach for user interface design. To enhance it further, we provide high fidelity implementation prototypes to our usability engineers for user testing. Paper prototypes are good and sufficient for verifying non mobile-based product requirements. But in case of applications for mobile phones, they are not sufficient for

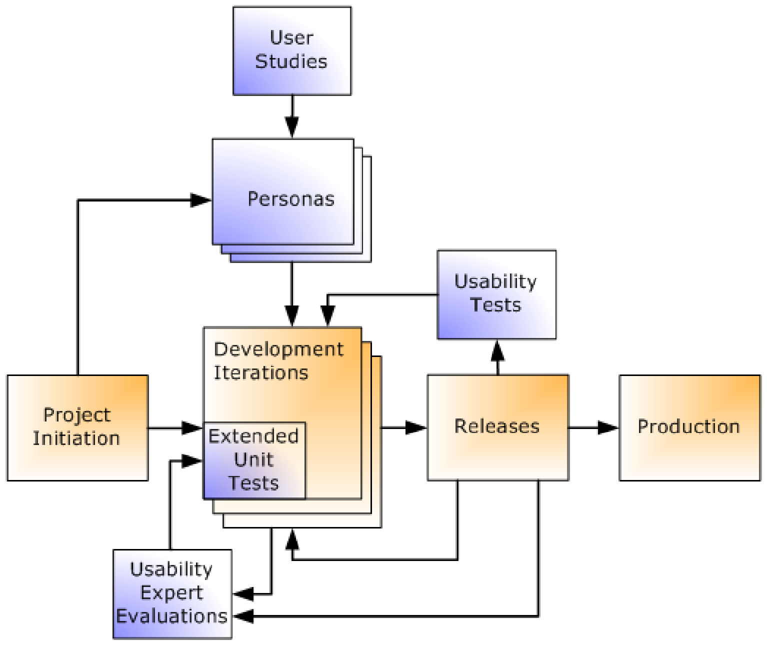


Figure 5.2: The Integration of HCI Instruments into XP [151].

finding and solving usability issues related to detailed interaction on the small device with its limited user input capabilities [84]. Therefore, in our case the application is tested on mobile phones and not on any web based simulator in order to understand issues concerning the use of mobile phone interface [84].

5.5.6 Communication and Collaboration

Communication between stakeholders is an important characteristic of software development. Communication and collaboration between customers, business partners, developers, and other stakeholders enhance the overall team efficiency [107]. The value of communication is expressed by the XP practices of pair programming, metaphor, informative workspace, simple design, on-site customer, the planning game, and coding standards [61]. Other factors in communication are the use of whiteboards, positioning and sharing of desk facilities to ease pair programming, stand–up meetings, developers buying-in to the concepts of the rules and practices of XP, and collective code ownership [52]. We sit side by side in a spacious room having enough space for private workplaces, as well as for three separate pairing stations. This seating arrangement has promoted effective interaction in the team and has helped in resolving technical issues on the spot [92]. The teams' XP room is equipped with six whiteboards which are used to record the XP stories agreed at release and iteration planning meetings. Story cards are physically stuck to the whiteboards in prioritized order with adjacent notes written on the board. Various graphs showing architecture and velocity of the project are also drawn on the whiteboards. By looking at the whiteboards, anyone can see the current status of the project.

Email, phone calls, and video conferencing are the tools used in routine communication with the usability engineers and other partners. Personal visits to and by project partners are also made by the product manager and by other team members whenever necessary.

5.5.7 The Planning Game

We hold two types of planning meetings: release based meetings and iteration based meetings. A release lasts for three months, where within a release, an iteration lasts for two weeks. Project partners attend release meetings where through discussions user requirements are identified and defined in the form of so-called XP user stories [68]. The parallel with the UCD approach is visible in the understanding and appreciation of the users and their requirements [107]. The user stories are written down on story cards and are prioritized by the project partners. Developers then estimate the time required for implementing the stories.

At the beginning of each iteration, an iteration meeting is held which is attended only by the team members including the product manager. The product manager selects and prioritizes stories which fit in the current iteration depending on the available velocity. Then, developers break down the stories into detailed tasks and estimate them. Finally, the product manager defines the acceptance criteria for each story and task.

Before and after implementing the user stories, continuous feedback is obtained from the usability engineers. Then, these stories are modified according to the feedback of end-users and the usability engineers. Once again, this is a common step with UCD approaches; an understanding of the user goal and the tasks to achieve that goal. Addressing a requirement in terms of the user and their goals focuses development upon what is needed [107].

5.5.8 Pair Programming

This practice has helped us in spreading and sharing the project-specific knowledge and improving the technical skills of the developers. We also applied the practice of working in pairs with the product manager [68]. The product manager pairs with a developer when writing customer-acceptance tests, thus exposing the customer to the process and the internal status of the application, which helps in better understanding and implementing the end-users' requirements. This also has enhanced the enthusiasm of the team members to work in a collective and collaborative team environment.

5.5.9 On-Site Customer

In UCD, all activities are focused on providing business value through ensuring a useful, usable and engaging product. The customer is not only defined as the project stakeholder, but the end user as well [107]. The Manifesto for Agile Software Development [101] does not clearly demand end-users as customers. In our process, the product manager plays the role of an "on-site customer" and communicates with the various stakeholders. The end-users are indirectly involved by the usability engineers.

5.6 Usability Study

Usability tests are carried out to evaluate the running prototype. One of the usability studies was executed in January 2008 with 10 respondents using a mobile phone. The classical task-based usability test method was used [128]. Each respondent was asked to execute 5 different tasks. Tasks were carried out on a Nokia N95 mobile phone. To gather general feedback and general opinions, two interviews were carried out: One before and one after the task session (pre- and post-interview). Each task was accompanied by task specific post-questionnaires. Interview sessions lasted about 1 hour. For the tests the device's standard browser, as well as opera-mini 3.0, were used (the first is incorporating a web-like mouse pointer, the latter a link marker to navigate through the interface).

After the test, respondents had to judge three different visual design paper-prototypes. We used the AttrakDiff questionnaire [60] to capture the attitudes of the users towards the application in terms of graphical design, enjoyment, and aesthetics. The AttrakDiff questionnaire was filled-out after the task.

The following two subsections outline the results of the test in the form of improvement suggestions.

5.6.1 Improvements of Layout and Design

Main improvements should be made concerning the visual appearance of the site:

- The actual site, menu, and navigation layout is not ideal. Through the use of the color blue as text color and background color at the same time, equal text sizes throughout the interface and different alignments, the site's hierarchy is not visible for users.

- The current layout does not incorporate visually attractive design elements and is rated as pragmatic and monotone with a lack of stimulating elements (Attrakdiff questionnaire).

Figure 5.3 shows the prototype of the home page presented in the usability study.

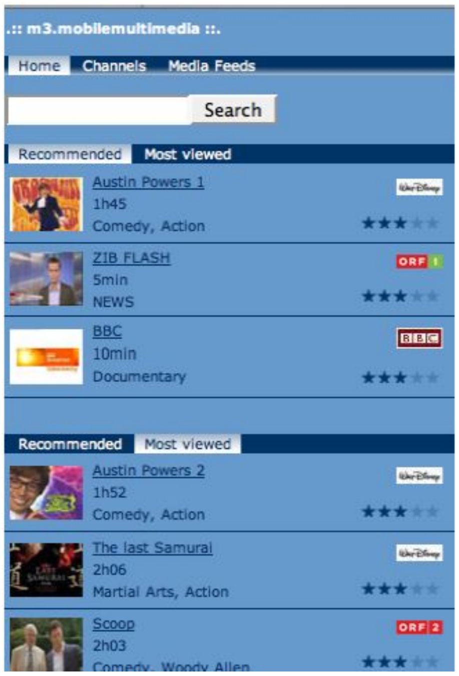
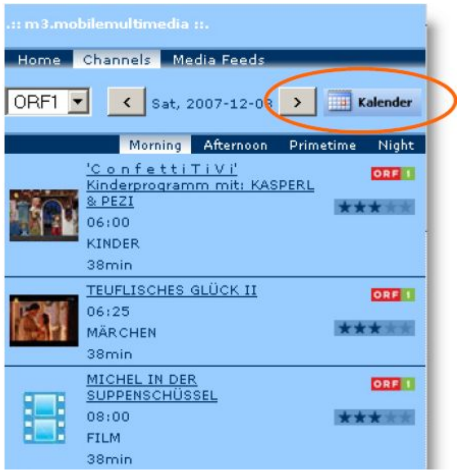


Figure 5.3: The Prototype of the Home Page [72].

5.6.2 Improvements of the Usability of the Prototype

On the "Channel" web page, a web-like calendar function to select dates should be integrated (the current function will not be usable for greater amounts of data). All navigation menu elements should be separated from content menu elements ("Home" vs. "Watch"). Furthermore, interactive elements ("Rate", "Comment" etc.) should be placed on a separate page and not on the bottom of a description page. Figure 5.4 shows the recommended prototype of the "Channel" page showing the calender. Figure 5.5 shows the menu entries without any visual separation.



'C o n f e t t i T i V i' Kinderprogramm mit: KASPERL & PEZI
06:00
KINDER
38min

TEUFLISCHES GLÜCK II
06:25
MÄRCHEN
38min

MICHEL IN DER SUPPENSCHÜSSEL
08:00
FILM
38min

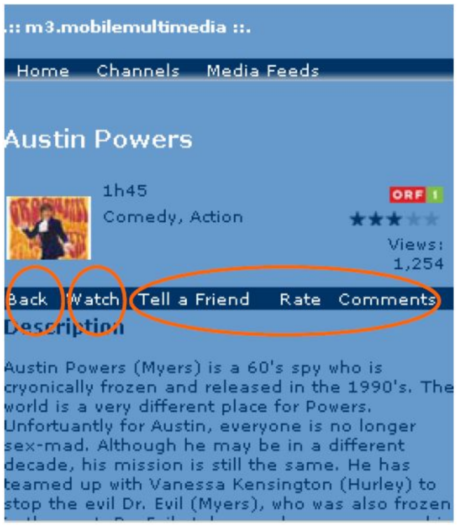
Figure 5.4: The Prototype of the Channel Page showing the Calender.

For the further development of the prototype the sub-site “Media Feeds” should be separated into two categories introducing the sites “create Media Feed” and “watch Media Feed”. Special attention should be given to feedback mechanisms, which at the moment do not support the user (feedback of search queries, display of media feed search results).

From the mock-ups of three different designs, the AttrakDiff results suggest that a yellow design was most liked by the respondents. It was also suggested that the blue design may be used in the future, but the following improvements should be made:

- Accentuate contrast on whole site.
- Avoid light blue text on darker blue backgrounds.
- Introduce visually attractive design elements that increase the attractiveness of the site.
- Eliminate monotony by introducing more colors.

Two of the developers also observed the usability study session which gave them a chance to realize the impressions of actual end-users and their feelings. This helped in guiding the development according to the wishes of end-users.



Austin Powers (Myers) is a 60's spy who is cryonically frozen and released in the 1990's. The world is a very different place for Powers. Unfortuantly for Austin, everyone is no longer sex-mad. Although he may be in a different decade, his mission is still the same. He has teamed up with Vanessa Kensington (Hurley) to stop the evil Dr. Evil (Myers), who was also frozen

Figure 5.5: The Menu Entries without any visual Separation.

5.6.3 A Task Example

In this subsection, a task example is presented. The task is: "Find the detailed description of a given movie, write a comment and rate it".

Facts on Task:

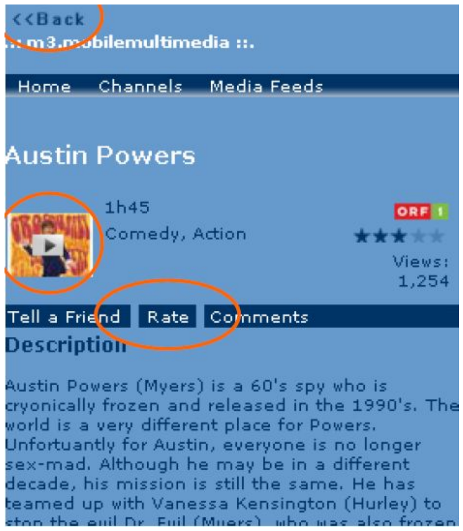
The task was completed without any greater difficulties by all respondents. On the "Home" page and on the "Channel" page respondents used the heading to find the detailed description and the video's thumbnail to watch the video. Respondents did not encounter much problems on the "Clip Detail" page. The prominent position of the links "Comments", "Rate" and "Tell a friend" – Figure 5.5 – on top of the description page helped respondents to understand which possibilities are offered. On the "Clip Detail" page there are two interaction paradigms that were both understood: Clicking on the link "Tell a fried" opens a new page. This did not cause any problems for users. The functions "Rate" and "Comment" are placed at the bottom of a "Clip Detail" page and users had to scroll down or use a link to jump down. In reference to both described paradigms, user comments indicate that the longer the list of comments is, the more uncomfortable the site is to browse. Further, the task uncovered that on the mobile interface respondents did not recognize that they were scrolling down the page when using the anchor-links "Comment" and "Rate". To get back to the top of the site they pushed the "back" button. This did confuse some of the respondents as they jumped back to "Home" although their intention was to get to the top of the "Clip Detail" page. Of course, this depends on how the browser implements the "back" functionality.

A solution that incorporates interactive functions ("Rate", "Comments", "Tell a Friend") on a separate page is recommended.

Suggested improvements resulting fro this concrete tasks:

- Back Button: A dedicated back button should be integrated on top of the page. This is the place, where basic navigation elements are expected.
- Watch Button: A watch button should be designed and integrated consistently. An additional watch button – if necessary – should be placed on a particular spot on the site and not be integrated in the navigation menu. The watch button should be visually highlighted.
- Tell a friend, Rate and Comments: These elements describe interactive functions on the site and therefore should be kept together and aligned to the left side of the page.

Figure 5.6 shows the recommended menu layout and arrangement.



Austin Powers (Myers) is a 60's spy who is cryonically frozen and released in the 1990's. The world is a very different place for Powers. Unfortuantly for Austin, everyone is no longer sex-mad. Although he may be in a different decade, his mission is still the same. He has teamed up with Vanessa Kensington (Hurley) to stop the evil Dr. Evil (Myers), who was also frozen

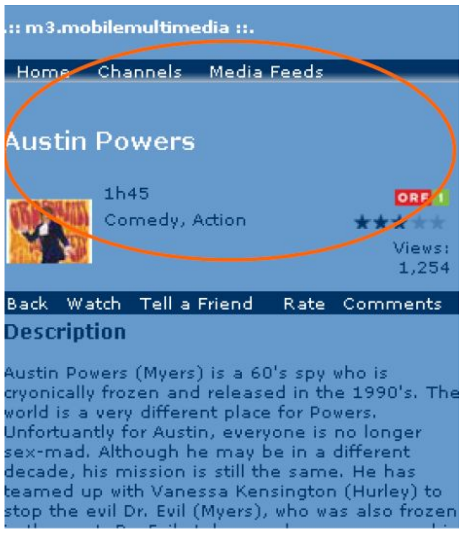
Figure 5.6: Improvements of Menu Layout and Arrangement.

Respondents' Feedback/Comments:

- All respondents indicated that in their opinion the "Clip Detail" page provides a good overview.

- The design of the "Comments" and "Rating" section is good and intuitive. Too many comments on one page should be avoided as the page would get too long (1 respondent).
- Comments should be ordered in chronological sequence, beginning with the most recent entry (1 respondent).
- It should be possible to select which information is sent to another person via the "Tell a friend" function (the video's description, the video itself, etc.). Radio buttons should be used to specify one out of different possibilities (1 respondent).
- The space on top of the "Clip Detail" page (heading) should be used in a better way. This would provide more space for description texts (1 respondent).

Figure 5.7 shows the space on top of the "Clip Detail" page which should be used more efficiently.



Austin Powers (Myers) is a 60's spy who is cryonically frozen and released in the 1990's. The world is a very different place for Powers. Unfortuantly for Austin, everyone is no longer sex-mad. Although he may be in a different decade, his mission is still the same. He has teamed up with Vanessa Kensington (Hurley) to stop the evil Dr. Evil (Myers), who was also frozen

Figure 5.7: Use the Space on top of the Clip Detail Page more efficiently.

5.7 Conclusion

Agility is an invitation to adapt, to mold, and to reshape the software development methodology according to the requirements of a project. Being a lightweight agile process, it is easy to extend the XP process with additional practices. Our XP process fits well into the UCD approach, because of the many overlapping principles (iterative development, end-user incorporation, testing) of both methodologies. The usability engineers in the project are well integrated into the development team. It would have been even better if one of the usability engineers had been present physically with the team all the time, as face-to-face communication is more helpful in quickly resolving design issues. We could not conduct usability tests frequently with end-users due to time and budget constraints, but mitigated it with usability expert evaluations. The whole development process is influenced when each feature of the application is assessed from users' perspective. This addresses the problems which arise when the system requirements are gathered only by discussions with stakeholders [77]. The involvement of all stakeholders, particularly end-users, in the process, can increase the chance of the success of a project.

We continuously try to optimize our process as long as the project lasts and will provide further insights whether the process has been able to enhance the usability of the application. In October 2008, we will conduct a contextual mobile multimedia content usability study which will give insights into mobile HCI concerning the coherence of content types, consumption times, and consumption contexts. Integrating of end-users indirectly in the form of HCI instruments, co-location, communication, and planning meetings has not only contributed to improving our process, but also has led to increasing the overall morale of the team.

Chapter 6

Integration of Extreme Programming and User-Centered Design: Lessons Learned

One of the most important factors for the success of a software application is user acceptance by having a usable user interface. Since summer 2007 in our project regarding mobile phone application, we have combined Extreme Programming and User-Centered Design methodologies aiming to deliver usable and useful software. The Human-Computer Interaction instruments we have integrated are: user studies, personas, usability expert evaluations, usability tests, automated usability evaluations in the form of extended unit tests, as well as lightweight prototypes. After one and half years we conducted a retrospective workshop with our off-site usability engineer to reflect on the adopted process regarding the Human-Computer Interaction instruments. This chapter presents those reflections - the lessons that we learned.

6.1 Introduction

One of the most important factors for the success of a software application is user acceptance by having a usable user interface. Extreme Programming (XP) - one of the mostly adopted processes of agile methods in the industry - aims to continuously deliver quality software by satisfying the customer. The Agile Manifesto does not clearly mention that the customer should be an end-user and rarely end-user take the customer role [30]. There is also the evidence that coordination only with the customer does not ensure good usability, which can result in lowering the user acceptance rate [77]. User-Centered Design (UCD) is an approach to user interface design focusing on end-users throughout the planning, design, and development stages of a product [147].

Recently, there has been an increasing interest in integrating agile and user experience/UCD methodologies, both, in the agile community and the Human-Computer Interaction (HCI) community. Being integrated into agile methods, UCD/usability engineering helps to reduce the risk of running into wrong design decisions by involving real users, and results in increasing the acceptance of software applications [108]. We have integrated XP and UCD in our project regarding a multimedia streaming application for mobile phones since summer 2007. The end-users are indirectly involved in the process by our use of different HCI instruments like user studies, personas, usability expert evaluations, usability tests, extended unit-tests, and lightweight prototypes [151][71]. Recently, a retrospective workshop was conducted. It was attended by all the team members and the usability engineer to reflect on the integrated process, as well as on the HCI instruments after introducing them at the start of our project. This chapter mainly describes those lessons learned. The next section describes the related work of combining Agile/XP with UCD methodologies. Section 6.3 describes the project context. Section 6.4 gives details about the retrospective reflection workshop regarding the HCI instruments used and the adopted process. Section 6.5 concludes the chapter by presenting future work.

6.2 Related Work

The integration of agile methods with HCI practices was discussed by Kent Beck and Alan Cooper in 2002, concluding that both interaction design and XP have strengths to be combined [117]. There are several studies examining various aspects of the integration of both methodologies. Patton [124] has described the way of incorporating interaction design in an agile process. Chamberlain et al. [30] have conducted an ethnographic field study to explore a framework for integrating agile methods with UCD. In their case study, McInerney and Maurer [105] interviewed three UCD specialists for integrating UCD within agile methods and reported positive feedback. Ferreira et al. [48] investigated several projects for the relation of user interface design and agile methods. Fox et al. [51] also conducted a qualitative study that describes how the agile methods and UCD are integrated in industry. Obendorf and Finck [121] report their experience of combining XP and scenario-based usability engineering.

6.3 The Project Context

Since summer 2007, we are working on a project to develop a multimedia streaming application for mobile phones. The project is based in Austria and will end in 2010. The team consists of six full-time regular members, five developers and a product manager who plays the role of the "on-site customer". One dedicated off-site usability engineer of a partner usability research center is also included in the team regarding usability guidance. The application enables a user to perform content-based search for audio and video content in large digital archives and play it on a mobile phone [71].

6.4 Retrospective Workshop

The following subsections describe the HCI instruments we have used in our process and the reflections about them, discussed in the retrospective workshop.

6.4.1 User Studies

User studies are the instrument for getting knowledge about end-users. The purpose of user studies is to uncover user needs, desires and contexts of use. In an agile UCD process they can be used for developing new or refactoring existing personas, as well as for the user-story creation process. In our process we employed user studies in the form of laddering interviews and field studies. The laddering interviews were conducted in autumn 2007 and the results were published in [96]. Currently, a large field trial study is being conducted with 150 real end-users spread throughout Austria. The trial study includes field trials with diary studies, contextual interviews, laboratory usability tests, questionnaires, and focus group.

6.4.2 Personas

Personas are archetypical figures - fictitious characters created as a tool to represent a typical user group. In our process, initial personas were created based on initial user studies and were iteratively refactored when new user studies suggested some changes. The personas helped to gear the project towards the on-site customer and end-users. However, the initially developed personas were not satisfactorily distributed by the usability engineer to the the development team and the customer-on-site. In addition to that, the development team and the on-site customer did not give much credit to the two personas which were provided. Nevertheless, they got in touch with them instead of neglecting them fully. It was concluded that personas should be properly introduced to the team, so that they will be present consciously or unconsciously in the minds of the team members during planning, developing or undertaking any decision process.

6.4.3 Lightweight Prototypes

We make use of two different types of mock-ups; low fidelity paper mock-ups and high fidelity mock-ups. Both are getting evaluated by the customer. As the developers and the customer have been increasingly gaining knowledge about usability engineering, evaluating paper prototypes with the customer is good on the one hand, but on the other hand, the customer has now become an expert user instead of a casual or novice user. So there is always a chance to ignore the actual needs of real casual end-users. We have mitigated this risk by having more ad hoc input from the usability engineer and suggested to conduct a formal usability test with at least 10 end-users after every release, e.g., quarterly. For special "ad hoc" questions instant messaging is used to gather HCI feedback within a short time frame from the usability engineer regarding stories or mock-ups. It would have been more beneficial if the usability engineer would have been present on-site to quickly give his feedback. In addition, he, instead of the developers, should do the prototyping not only with the customer, but also with at least a few end-users.

6.4.4 Usability Expert Evaluations

Expert usability evaluations are reviews conducted by experts. In our project, usability expert evaluations by the off-site usability engineer are given by instant messaging, email, and video-conferencing, usually in the form of ad hoc input. In our project, usability input is needed at different points in time: when writing user interface related stories and before or during implementation of the stories, as well as after implementation. As the customer writes stories with the help of a developer, it is decided that when a user interface related story is written, it should be sent along with the refined paper prototype to the usability engineer at least three days before the iteration planning. The advantage is that during the iteration planning the user interface stories are already usability tested story. When technical questions arise during the implementation - for instance that a certain demand from the usability side would cost too much - it is advised to call the usability engineer or have a short video-conference. As a result, one should get far less usability fixes to make. For us, two hours to one day duration is perfect for this quick feedback of the usability engineer. After the implementation (when the application is deployed), usability feedback can be delayed for days. It was suggested that the usability engineer should give his feedback in the form of stories along with wireframes when he thinks they are needed. Consequently, he should be trained in writing user stories with the help of one of the developers.

6.4.5 Usability Tests

Usability tests involve real users testing an application. We conducted a formal usability test with 10 users in January 2008, which was also attended by two developers as observers [71]. It was noticed that the mindset of the developers changed dramatically when seeing real users handling the application. The developers who observed the users during the test got more biased towards user-centered thinking than the others. When it comes to the results of the test, there was an agreement that the test was too early in the project to tell us a lot about the usability problems of the application. At that point in time the system was very fast moving and because of new demands from the stakeholders, changes in features were high. Furthermore, the reporting period was too long. Until the report arrived, the application had changed so much that the recommendations were partially obsolete. Therefore, smaller tests after every 3 iterations were recommended. Since the system is a very fast moving target, not always the entire system should be tested. Big formal tests were recommended after every release, when major changes are expected.

6.4.6 Extended Unit Tests

Extended unit tests root in automated usability evaluation. Our intended approach extends the XP unit tests by adding usability-specific test cases. Code based tests are enhanced with semantics to achieve this goal. For example, code based tests can check against guidelines such as the usage of capital letters on buttons or the correct label of a button. So far we have not focused on extended unit tests because of priorities in areas, but we intend to work in this direction from the second quarter of this year.

6.5 Conclusion

Since summer 2007, the mentioned HCI instruments have been used in our project for enhancing the usability of the product. Until now, we have learned lessons that are summarized here. In our project the XP process fits well into the UCD approach because of the many overlapping principles (focus on delivering value, iterative development, end-user incorporation, continuous testing) of both methodologies [71]. The usability engineer is well integrated into the development team. We found no cultural difference until now. Developers have gained insights into the UCD practices, while the usability engineer learned the origin of usability problems [151].

Furthermore it was found out that especially ad hoc input can be given sufficiently via mail, since most of the time no synchronous communication between the project members is needed. It is also fitting for various response times since, e.g., for usability input in the story-writing process it is sufficient to get results within 3 - 4 days. When quick fixes are needed or other input during an urgent re-planning, the usability engineer should be easy to contact for a quick advice via cell-phone or chat. The interaction with the usability engineer early in the story creation-process results in saving time, increasing motivation, and gaining better realization of needed usability input early in the development. Furthermore, instead of a big report of a formal usability test, the usability engineer should give the report in the form of checkpoints, which then will be converted into stories quickly. The usability engineer should be trained in XP-story writing to be able to deliver the user-stories in a technical-aware manner. Proper customer and usability engineer coordination is necessary for enabling a good usability process in the development.

The field trial study is underway these days. We have implemented user-tracking and feedback mechanisms already in the basic architecture. The results of the interviews, diary studies, usability tests, focus groups, and log file analysis will provide feedback on not only how the end-users perceive the product. In addition, statistical data will be collected from the actual usage behavior of the users. In this way, we will be able to provide more insights into the integration of HCI instruments into our adopted process. We will also gain insights into the context of mobile multimedia usage. We aim to continuously optimize our process till the project ends and will share the knowledge gained to the agile as well as the HCI communities.

Chapter 7

Agile Process Patterns

Today, the state of the art in web application development is the usage of agile web development processes. The main benefit of agile process definitions is to be lightweight and flexible. Nevertheless, often the opposite is true and they can be exhaustive and verbose, particularly when integrated with usability. One way to alleviate this problem is to capture processes in the form of patterns.

This chapter presents three Agile Usability Process patterns extending an already existing agile process pattern collection. The patterns described are: Usability Expert Evaluation, Usability Test, and Automated Usability Evaluation. In pattern literature, it is stated that patterns are not invented but found. The patterns described in this chapter are derived from the agile usability process of a scientific, Extreme Programming based, project. In addition to the pattern definitions, the pattern implementations in this scientific project are outlined. In order to prove the validity of the patterns, they were implemented and evaluated in an industrial, Extreme Programming based project. For the purpose of comparing the two process pattern implementations, both processes were evaluated using an Extreme Programming Evaluation Framework. For each of the processes, context factors were recorded and adherence metrics data (quantitative and qualitative) was collected at two points in time.

The pattern implementation results showed that the usability and the overall user experience of the developed systems improved significantly. A high number of usability issues found in Usability Expert Evaluations was fixed by means of test driven development, namely, writing an automated usability test for each of the discovered usability issues first. Resulting issues of Usability Tests were incorporated and fixed the same way. This, as well as the employment of Automated Usability Evaluation metrics in test driven application development, increased the usability of the developed systems over time to a very high degree.

7.1 Introduction

Agile software development techniques are applied by many software development organizations with great success [75]. However, none of the major agile development methods explicitly incorporates usability engineering practices [83]. Nonetheless, usability is a critical quality factor for the successful adoption of a software product. In [91] it is stated that there is probably no other technique with greater disproportion between its importance for the success of software development and the lack of attention and formal education as usability engineering and the design of the user interface.

The first activity in a Usability Process deals with specifying how user-centered activities fit into the whole system life cycle process, and to select usability methods and techniques [47]. Often, these techniques, aiming at increasing the usability level of the software product, are applied following development processes particular to the Human-Computer Interaction (HCI) field, and these processes are not formalized from the point of view of software engineering. Therefore, they are not easy to transfer to the formalized software engineering processes [46]. Nevertheless, much work already has been done in the field of integrating HCI techniques into software development processes [64][65]. All these proposals have the same disadvantage. The descriptions of such integrated processes are always exhaustive and verbose and lack fast and easy applicability in the sense of flexible and lightweight agile process definitions. One way to alleviate this problem is to capture processes in the form of patterns. This chapter presents three Agile Usability Process Patterns (AUPPs), which are intended to be an extension to an already existing agile process pattern language named Agile Adoption Patterns [40].

It is common knowledge in the scientific community that patterns are not invented, but found. The AUPPs presented in this chapter are derived from an agile usability process of a scientific project started in summer 2007. The basis for this agile usability process is the Extreme Programming (XP) methodology. The application being developed within this research project enables a user to perform a content-based search for audio and video content and to play it via streaming on a mobile phone. The basic research goal is the analysis of agile software development methodologies, in particular XP, with special emphasis on usability. This is obtained by two means. On the one hand a development process was established, where the quality focus is not only placed on technical excellence, but also on delivering a usability-tested high-quality end-product. On the other hand, a testbed for effective and efficient mobile usability testing automating certain parts of the usability testing procedures was created.

Within this research project, a lot of topics already have been covered. An iterative and user-centered approach to user-interface design is published in [69][70]. Measures for optimizing XP processes are examined in [68]. The integration of XP with User-Centered Design is outlined in [72]. The results of a usability study are made public in [71]. Lessons learned from the integration of XP and User-Centered Design are presented in [74]. The concept and design of a contextual mobile multimedia content usability study is treated in [73]. An agile usability process is described in [151].

The AUPPs, Usability Expert Evaluation, Usability Test, and Automated Evaluation, presented in this chapter are derived from our agile usability process published in [151]. The approach integrates HCI instruments into the classical XP process. The goal of the adaption was the examination of the applicability of a usability-aware agile software development process. The process was designed to allow applying HCI instruments in XP teams. The novelty of the approach lies in the fact that not only one or two HCI instruments were used for integration, but in fact five instruments were selected to enhance the existing XP process. Applied correctly in different phases of the project, these instruments are designed to reach the goal of maximized software quality in technical terms as well as increased usability quality. The five HCI instruments are: Usability Expert Evaluations, Usability Tests, Automated Usability Evaluation, Extreme Personas, and User Studies. The integration of these HCI instruments into the XP process is shown in Figure 7.1.

The basic concept of using patterns and pattern languages in the field of software development is derived from work done in building architecture to describe qualities for good architectural designs. The architect Christopher Alexander started in the seventies to use pattern languages to describe the events and forms that appeared in cities, towns, and buildings in the world at large. The main purpose of a pattern is to describe a solution to a problem in a certain context. Most importantly, such a solution can be applied as often as one wishes without ever applying it in exactly the same way twice.

The three AUPPs presented in this chapter are intended to be an extension to an already existing agile process pattern collection named Agile Adoption Patterns (described in [40]). The patterns in this clear and concise pattern collection are generated by distilling the knowledge of the state of the art agile development methods and transferring it into pattern form. All available agile practices are broken down into a pattern of adoption resulting in a very clean and simple agile process definition. The pattern catalog comprises feedback practices, technical practices and supporting practices. Moreover, special emphasis is placed on business values like Reduce Time to Market, Increase Product Utility (Value to Market), Increase Quality to

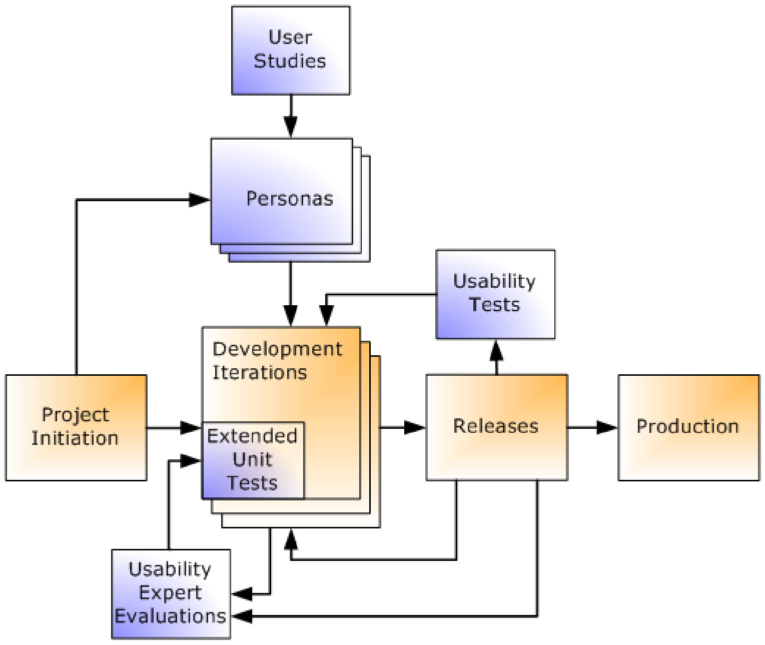


Figure 7.1: Agile Development Process including HCI Instruments [151].

Market, Increase Flexibility, Increase Visibility, Reduce Costs, and Increase Product Lifetime. All patterns of this collection address one or more of these business values. Despite the fact that this pattern catalog is very comprehensive in terms of incorporating all currently available agile knowledge, it still lacks the usability aspect. The three AUPPs presented are the first attempt to close this gap. Naturally, each of the AUPPs directly adheres to one or more of the mentioned business values. What is more, the AUPPs have been written according to the pattern language of pattern writing [113].

The AUPPs are derived out of an agile usability XP based process [151] of a scientific project and validated in an industrial XP project. For the purpose of being able to compare the results of these two process pattern implementations, both processes were evaluated by means of using an Extreme Programming Evaluation Framework (XP-EF) [150]. The XP-EF provides a benchmark measurement framework for researchers and practitioners to assess concretely the extent to which an organization has adopted XP practices and the result of this adoption [150]. The framework provides informative feedback utilizing streamlined process and project metrics appropriate for a lightweight software process [149]. The XP-EF is a compilation of validated and proposed metrics and was designed for use throughout development by agile teams. Small software development teams require a smaller, more manageable metrics set that provides constructive feedback about their development process [149]. For this reason the XP-EF metrics are focused, concise, and can be collected by a small team without a dedicated metrics specialist [150].

The framework enables the necessary meta-analysis for combining families of case studies. Two published case studies, one at IBM [149] and one at Sabre Airline Solutions [93], have already been completed with XP-EF. The primary focus of the XP-EF is to assess the extent to which an organization has adopted XP practices and the result of this adoption. What is more, the framework is an excellent tool to describe a given agile process. Within the context of this work the XP-EF was used exclusively to compare two different agile software development processes.

The framework consists of three parts:

1. XP Context Factors (XP-CF)
2. XP Adherence Metrics (XP-AM)
3. XP Outcome Metrics (XP-OM)

In the XP-EF, researchers and practitioners record essential context information about a project via the XP Context Factors (XP-CF). Drawing general conclusions from empirical studies in software engineering is difficult because the results of any process largely depend upon the specifics of the study and relevant context factors. Therefore, recording an experiment's context factors is essential for comparison purposes and for fully understanding the similarities and differences between different environments [150].

The second part of the XP-EF is the XP Adherence Metrics (XP-AM). Many software development teams do not exercise all the XP practices to their full extent, some employ only a few practices. The XP-AM contains subjective and objective measures as well as qualitative analysis about the team's use of XP practices to triangulate the extent to which a team uses each of the XP practices [150].

Part three of the XP-EF is the XP Outcome Measures (XP-OM), which are business-oriented metrics. It provides researchers and practitioners a means to assess and report a team's project outcome from using a full or partial set of XP practices. The XP-OM consists of traditional external software development measures, such as productivity and quality [149]. As stated before, within this work the XP-EF is used exclusively to compare two different agile software development processes. Therefore, this third part of the framework was omitted.

This chapter is organized as follows. Section 7.2 presents related work. The process description of the scientific process according to the XP-EF can be found in Section 7.3. Section 7.4 contains the AUPPs. The scientific process implementation of the AUPPs is outlined in Section 7.5. Section 7.6 comprises the industrial process XP-EF process description. How the AUPPs were implemented in the industrial XP process is outlined in Section 7.7. A conclusion is drawn in Section 7.8.

7.2 Related Work

This section provides an overview of related work. All publications treated in this section are collected by means of a systematic literature review [86]. In total, eight different online libraries were searched using a query with different connections of the following keywords: agile pattern, process pattern, agile process pattern, extreme programming pattern, usability process pattern, agile usability process pattern, agile usability, and agile usability process. The outcome of the systematic literature research was that no work on agile usability process patterns exists. Basically, existing work can be divided into the categories agile process patterns, individual agile practice patterns, agile usability processes, and usability patterns. The publications on usability patterns are not reported in full, but exemplary, because they are not process-oriented. Related work in each of the categories is examined in the following subsections.

7.2.1 Agile Process Patterns

The work in [35] introduces forty-three software development process patterns in the form of a generative pattern language. The basic assumption is that most highly productive organizations exhibit the same patterns of organization, process, and introspection. For this reason, data on a wide spectrum of software development organizations has been gathered over two years, and patterns capturing organizational structures and practices were extracted.

An agile process pattern language named Agile Adoption Patterns is described in [40]. The patterns in this clear and concise pattern collection are generated by distilling the knowledge of the state of the art agile development methods and transferring it into pattern form. All available agile practices are broken down into a pattern of adoption resulting in a very clean and simple agile process definition.

Patterns describing the SCRUM development method are presented as an extension pattern language to already existing organizational pattern languages in [15]. The SCRUM development method has been proven to be an effective tool for productive software development. In this work, SCRUM is decomposed into patterns which are combined with other existing organizational patterns. The intention is to lead to highly adaptive, well-structured software development organizations.

In [17], an agile development pattern collection is presented. In particular, XP is considered to be a pattern language in which the practices are the basis for the patterns. It is claimed that the practices have the characteristics of a true pattern language in that they are synergistic and generative. Based on this assumption, the practices are transformed to patterns and additional patterns are generated.

Organizational patterns for agile software development are presented in [36]. The patterns are divided into four interrelated pattern languages: Project Management (the organizational aspects of managing projects), Piecemeal Growth of the Organization (how an organization grows and develops over time), Organizational Style (the general approach to the way the organization works), and People and Code (how different people produce different code). The four pattern languages provide different views on the complex multiple structures of organizations.

In [110], the concept of patterns is introduced into software processes, and experiences proven to be effective in agile methods are organized as a group of patterns. Based on this group of process patterns, a process pattern language for agile methods is proposed. Any agile method can be mapped to this generic process pattern language. Process patterns from different agile practices can be combined and appropriate agile process models for different projects in different organizations can be generated.

The agile patterns described in [26] are based on the principles and practices of the state of the art agile methodologies. While individual practices included in any of these methods vary, they all have particular objectives and related activities. Therefore, every pattern is described so as to show the core solution to a particular problem. The patterns are organized in a framework of agile patterns.

An approach to acquiring and defining knowledge about agile software development in terms of patterns is presented in [27]. The agile patterns discussed are derived from six different agile methodologies. The definition of agile patterns, deriving and recovering best practices from agile methods, is examined and the topic of how the usage of the agile patterns contributes to organizing and delivering organizational knowledge is discussed.

In [62], it is outlined how a pattern language can be derived from existing agile development processes, and how the result can be used for process adaptation. The principle behind it lies in rewriting software processes as pattern languages and integrating them into a single comprehensive pattern language. Software processes can be constructed from this single pattern language by using concrete, project specific constraints as input.

A set of agile-specific process patterns that can be used for method engineering purposes is proposed in [142]. Pattern extraction is based on a detailed inspection of seven prominent agile methodologies, whereby a generic agile software process is identified which is used as the starting point for the extraction process. Each of the studied agile methodologies can be realized using the proposed process patterns.

A process pattern description language which provides concepts for the semiformal description of process patterns and relationships between process patterns is presented in [57]. By using this pattern language, single process patterns can be modeled and, by definition of relationships, be composed to more complex processes. The pattern language is applied to the Rational Unified Process illustrating how a process pattern catalog and the contained process patterns are modeled.

7.2.2 Individual Agile Practice Patterns

In [127], a set of twelve customer interaction patterns is introduced. The patterns are designed to cope with the increasing emphasis on business awareness and are intended to improve the effectiveness in customer interaction. The patterns target developers and service providers having direct interaction with customers. Different levels of customer interaction such as building relationships, doing negotiations, allocation of responsibilities, and contact culture, are addressed.

The roles and practices that will increase the effectiveness of the customer on an XP project are outlined in [103]. Since customers have one of the most complex and difficult roles on a project and XP includes very few practices that support the customer in their role this paper presents patterns serving the purpose of outlining the customer role comprehensively.

The problem if a customer is actually a provider for another customer is treated in [125]. The work describes a situation involving a complex customer relationship with competing goals and a lack of centralized ownership for the product and the consequences of failing to identify the correct customer. Out of this a behavioral pattern that is relatively common in the software world is derived.

Patterns of stand up meetings, intended to help new practitioners as well as remind experienced practitioners of what they might already know, are presented in [153]. Despite the simplicity of daily stand ups, when the whole team meets every day for a quick status update, stand ups often do not work. The patterns outlined are written as an attempt to communicate tacit knowledge on the benefits and consequences of common practices for daily stand ups.

In [41] functional testing in pattern format, aggregating experiences with functional testing over several agile development projects, is presented. Functional tests are automated, business process tests co-owned by customers and developers. They help elucidate requirements, make project progress visible, and improve code quality. However, functional testing can become more costly than its benefits, therefore, symptoms of potentially costly problems are outlined and solutions to those problems are proposed.

The results of an observational study identifying patterns in the use of the FIT acceptance testing framework are presented in [126]. Executable acceptance testing allows both the specification of customer expectations in the form of tests and to compare those to actual results that the software produces. The usage patterns outlined are intended to lead to a better understanding of the strengths and weaknesses of acceptance testing.

Concrete examples of applying test automation patterns to user acceptance testing are provided in [9]. Furthermore, a description of various extensions to the WebTest acceptance testing framework that facilitate developing automated acceptance tests according to these established best practices is given. The customizations made to the Canoo WebTest acceptance-testing framework were implemented during the course of an agile project to develop automated acceptance tests.

An xUnit test pattern language containing test automation patterns specifically for the xUnit test framework is presented in [111]. The language comprises patterns of different abstraction levels and scopes as well as a set of test smells, which indicate potential test failures, not only restricted to the code level. Ten different types of patterns on the abstraction levels test strategy, test design, and test coding idioms are treated.

A pattern specific to the agile release planning practice is published in [154]. The basic concept of the pattern is to use poker chips when estimating and scheduling user stories. The motivation behind this is the fact that physical models emotionally convey the limits of resources and time much better than words or numbers, respectively a boring release planning process will tend to receive less engagement and commitment from stakeholders.

In [85], an approach to applying patterns that combine the top-down utility of design patterns with the bottom-up discovery of iterative development and continuous refactoring is examined. Based on the assumption that design patterns are targets for refactorings, the work introduces the theory and practice of pattern-directed refactorings: sequences of low-level refactorings that allow designers to safely move designs to, towards, or away from pattern implementations.

Simple, elegant and proven solutions to the specific problems of writing use cases on real projects described by means of three-dozen patterns can be found in [6]. These patterns are based on observable signs of quality that successful projects tend to exhibit. The goals are providing a vocabulary for discussing, advice for writing and organizing use cases effectively, and diagnostics for evaluating use cases.

An agile, lightweight and sufficient, approach to documentation is outlined in [129]. The work presents a collection of patterns guidelines that offer solutions to the recurring yet multi-faced problems of documentation. The patterns are governed by the principles of producing lightweight documentation, having a high-quality documentation standard, using tools and techniques facilitating documentation generation, and an efficient and straightforward documentation process.

7.2.3 Agile Usability Processes

The work in [47] offers developers who have the objective of integrating usability practices into their software process a framework that characterizes 35 selected HCI techniques in relation to six relevant criteria from a software engineering viewpoint, and organizes them according to the kind of activities in the development process where they may be applied, and to the best moment of application in an iterative life cycle.

An approach for bridging the gap between software engineering and HCI, by offering orientation to software practitioners on the application of HCI techniques and activities is described in [46]. For this purpose, a survey of the HCI literature has been carried out to define the activities in a user-centered development process, and to select the HCI techniques that are more appropriate for integration into software engineering practice.

An experience-based, human-centered design life cycle, an interdisciplinary effort of experts in the fields of software engineering, HCI, and process improvement is presented in [114]. The approach aims at supporting the introduction, establishment and continuous improvement of Human Centered Design (HCD) processes and comprises a process model, tools, and organizational measures that promote the utilization of HCD methods and facilitate organizational learning in HCD.

In [49], a report on a qualitative grounded theory study of agile projects involving user interface design can be found. Within the study, semi-structured interviews with software team members from different companies in different countries were conducted. Key results are that agile iterations facilitate usability testing, incorporation of results of usability tests into subsequent iterations, and improve the relationship between user interface designers and software developers.

A development process and toolset that draws on XP and Scenario-Based Design is examined in [94]. The approach makes the contribution of allowing developers who use agile software development processes to efficiently address usability issues, supporting collaboration between software engineers and usability specialists by facilitating communication of design intent and rationale, and supporting efficient design representation-based development by leveraging techniques from agile software development.

The work in [50] addresses how interaction design and agile development work together, with a focus on the issue of interaction design being done up-front. The study method used interviews with interaction designers and software developers on several agile teams. The interpretation includes benefits seen for a certain amount of up-front interaction design, and benefits seen for continuing interaction design with the iterations of software development.

An experience report which proposes U-SCRUM as a variant of the SCRUM methodology can be found in [137]. Unlike typical SCRUM, where at best a team member is responsible for usability, U-SCRUM is based on the experience that the role of product owner is assigned to two peers. One is focused largely on traditional functions and the other is focused on usability and user experience.

In [112], usability testing based on paper prototypes and early versions of the software was added in the second release to an agile development process. Compared to the first release, in the second release a significant reduction of usability related rework was noted. The paper prototype became a tangible representation of the project vision that was used in many ways that contributed to the resounding success of the project.

A customer-centered systems definition method named Contextual Design is introduced in [18]. The approach is adapted to XP teams developing quick-turnaround, short-development-lifecycle projects. It is shown how integrating the two methods fills the gaps in agile methods for both fast-turnaround iterative projects as well as large scale, high-impact, enterprise projects. The resulting process incorporates the customer voice and provides room for user interaction design.

The publication [121] reports on experiences made in both academia and industry in putting an agile development process pattern to the test. The pattern combines XP and Scenario-Based Usability Engineering, based on a blend of perspectives on equal terms. The central aspect is the use of scenarios as a focal point to connect a design vision with the more technical tasks of the programmer.

The work in [83] tries to connect discount usability engineering with agile development. The expression discount usability engineering describes a collection of simple, low-cost techniques for designing and testing systems for improved usability. There are a number of similarities between discount usability engineering and agile development found and the overall strategy presented is to fashion a discount usability engineering approach for use with Scrum.

The usage of Interaction Design in an agile development process is examined in [124]. Interaction Design as a day-to-day practice throughout an iterative development process helps in delivering high quality software. Recommendations are provided on how to practice an agile form of Usage-Centered Design and how to incorporate bits of Interaction Design thinking into everyday development and product planning decisions.

The paper [30] reports a field study designed to investigate the use of agile methods alongside Usage-Centered Design in one particular organization. The aim of the study was to develop a framework for use by project teams wishing to integrate User-Centered Design practices with agile development. The study, its findings and five principles for integrating User-Centered Design and agile development arising from this work are discussed.

A study of a particular XP based interaction design process which is extended with the personas approach is presented in [58]. A model is constructed by analyzing the principles of personas and agile software development. Empirical evaluation of the model is performed in a case project. The results provide viewpoints on the applications of interaction design activities in different stages of agile processes.

The aim of [51] is to show that agile methodologies and User-Centered Design can co-exist effectively by conducting a study with participants that have previously combined these two methodologies. The findings connected with existing work outline that the existing model used for integration of Agility and User-Centered Design can be broadened into a more common model. Three different approaches for achieving this goal are presented.

The results of a case study on UCD in real life environments are examined in [105]. Three UCD specialists were interviewed, and their experiences are presented. These specialists, all having degrees with a UCD/HCI specialization and prior UCD experience, were working on their first agile project. The reported experiences are structured according to the categories: basic user understanding, user interface design processes, and usability evaluation.

In [134] five major obstacles of usability and software engineering and their related myths are discussed from the perspective of both the usability and software engineering communities. Tools and techniques for avoiding usability pitfalls in managing the software development life cycle are presented. Furthermore, traditional practices in software development and best practices in human-centered development are aligned to synchronize usability and software engineering.

A streamlined and simplified variant of the usage-centered process that is readily integrated with lightweight methods is outlined in [32]. The process is based on the typical agile characteristics of successive release cycles and iterations. In essence, the process consists of ten consecutive activities intended to be executed in an iterative way. The activities are based on card-based modeling and decision-making.

In the publication [140], a usability study on how a User Experience Team adapted their user-centered design practices in an agile environment is reported. In essence, ways to conduct usability tests, interviews, and contextual inquiries, both in the lab and the field, within an agile framework are described. In particular, adjustments of the timing and granularity of these investigations, and the way usability findings are reported, are outlined.

The role of the interaction designer in an agile software development process is examined in [97]. Observations of a contrast in thinking styles between a user-interface design team and a software engineering team developing a new software product are described. Based on a case study, key roles for the interaction designer working in a SCRUM environment are identified and elaborated with a temporal view on the development process.

The work in [95] presents a development approach that draws from XP and Scenario-Based Design. Tensions between agile software development and usability engineering resulting in difficulties to integrate both methods are outlined. Three key questions that need to be addressed for agile software development methods and usability engineering practices to work together effectively are described. Furthermore, interface architectures and design representations that can address these questions are introduced.

A coherent strategy for bringing usability practices into agile projects is presented in [7]. The publication examines both user experience (UEX) and agile software development (ASD) approaches, comparing and contrasting the underlying philosophies and practices of each. Then, using agile model-driven development as the foundation, strategies for tailoring UEX into agile methods are described. It is stated that the key factor is flexibility of both UEX and ASD practitioners.

In [65] ideas on how to combine XP and Usability Engineering in a software development method called Extreme Usability are reported. Extreme Usability is based on the fact that the two combined development approaches have different goals but, at the same time, employ similar methods to achieve them. Additionally, the authors have embedded their ideas into Software Engineering education.

The definition of a software development process that integrates practices from Software Engineering and HCI is outlined in [138]. The process aims at helping professionals in the development of interactive systems with usability, making HCI an essential part of software engineering, and describing the basis for developing user interfaces by integrating HCI concepts rather than on depending only on the experience of user interface designers.

In the book [33], the models and methods of an approach to software engineering to deliver more usable software are elaborated. Recognizing usability as the key to successful software, concrete tools and techniques that programmers can employ are provided. The systematic software development process outlined is called usage-centered design. In this process two major threads in software development methods, use cases and essential modeling, are weaved together.

The most relevant frameworks for integrating HCI and usability techniques into the software development life cycle are reviewed in [132]. Their strengths and weaknesses as well as how far the objective of the integration been reached are assessed. In addition, conclusions about research directions towards the development of a generic framework for the integration of usability engineering methods in software development practices are drawn.

The paper [44] presents a framework for the integration of usability techniques and activities. The framework characterizes selected usability techniques and activities using software engineering terminology and concepts, according to what kind of activity they belong to and at what development stage their application contributes most to the usability of the final software product. The proposed framework targets at enhancing iterative development processes with usability aspects.

The major challenges in integrating usability and user-centered design techniques in the software engineering life cycle are discussed in [133]. Issues covered are: user and user interface design specialist involvement, practical experiences of using usability engineering techniques and artefacts in the analysis, design and evaluation processes, organizational obstacles to user-centered design, role usability professionals in the development processes, and communication problems of usability experts with computer scientists.

The publication [21] describes the core principles of agile development and investigates to what extent usability-enhancing activities can be supported within the agile approaches. Although agile approaches and user-centered design come from different fields, they share common targets. Therefore, user-centered design qualities in the agile software development approach are analyzed. Further, a model for integrating agile development and user-centered design is outlined.

7.2.4 Usability Patterns

The work in [37] presents the results of an empirical study on browsing strategies and user goals when using a web system that designed using a web design patterns catalogue. Also, the relation between the user goal and the browsing strategy as well as the impact on the quality of the use of the patterns is analyzed. The overall intention is to design patterns catalogues that take into account the goals and expectations of their end-users.

A comprehensive web usability pattern language comprising 79 individual web usability patterns is described in [55]. The patterns address the four aspects of website design that can affect the success of a site: usability, content, navigation, and aesthetics. Furthermore, the pattern language is organized in a way that reflects the structure of a typical web design project and provides knowledge ranging from requirements understanding to detailed design guidelines.

The publication [25] examines a pattern approach to interaction design. A comprehensive pattern language for the interface design of interactive exhibits is presented. The patterns of the language capture and structure user interface design knowledge. Also, a number of evaluations of different aspects of the pattern language and an evaluation of the systems that have resulted from using this approach are outlined.

7.3 m3 XP-EF

In this section, the results of the evaluation by means of the XP-EF of a scientific project are presented. The product being developed is an application that enables a user to perform a content based search for audio and video content and play it via streaming on a mobile phone. From this point on, the scientific project will be referred to as the m3 project (mobile multimedia). One of the basic research goals of the m3 project is to examine different aspects of XP, especially XP in combination with usability. Since the m3 project is an XP research project, employment of the XP-EF was self-evident. The characteristics of the m3 project placed it in the agile home ground [22] and XP was implemented nearly in pure form.

The XP-EF data collected is the outcome of the evaluation and comparison of two releases of the product being developed. More precisely, the third and fourth release, from this point forth referred to as the old release and the new release respectively, were evaluated and compared. The m3 project was initiated at the beginning of 2007. Development for the old release began in July 2007 and lasted for 3 months. Work on the new release commenced in October 2007 and lasted for 3 months as well. At the beginning of the old release, the development team was already familiar with XP and the team exercised the practices to its full extent. For each of the releases, detailed data was collected and the result in the form of XP-CF and XP-AM is outlined in the following two subsections.

7.3.1 XP Context Factors (XP-CF)

The XP-CF utilize six categories of context factors outlined by Jones [78]: **software classification**, **sociological**, **geographical**, **project-specific**, **technological**, **ergonomic**, and an additional category, **developmental** factors, based upon work by [22].

Software Classification. In the XP-EF, projects are classified as one of six software types: *systems* (used to control physical devices); *commercial* (leased or marketed to external clients); *information systems* (for business information); *outsourced* (developed under contract); *military*; or *end user* (private, for personal use). The m3 project is a scientific research project which is funded by the government. Additionally, there are several academic and industry partners. Nevertheless, the product being developed is intended for the mass market, and commercialisation efforts exist. For this reason, the customer role is taken by a product manager. Since the product is built and marketed to appeal to many customers, this project is classified as *commercial* software.

Sociological Factors. Team conditions for both releases are shown in Table 7.1. For both releases, some factors were the same and for this reason no comparison was made. Personnel is often considered one of the most prominent risk factors in software development [24]. Therefore it is important to capture relevant information about team makeup. Sociological factors capture the development experience of the team members as well as their knowledge of the problem domain [149].

As shown in Table 7.1, in the old release the turnover rate was 20 %. The turnover rate was calculated by adding the number of people who joined or left the team and dividing by the team size at the end of the release. The reason why this is not reflected in the team size is that the new team member was not a developer, but a business person taking the role of a product manager. The team size parameter only counts full-time developers and full-time testers. Furthermore, the morale factors indicate that in the old release the team was experiencing delays caused by external partners. A public trial of the software was planned but the external partners did not make the deadlines. Therefore work needed to be replanned and rescheduled. In the new release, the morale factors indicate reduced process. This happened for scientific reasons in order to experiment with XP and its practices. Some of the XP practices were dropped in the process implementation, because the initial introduction of all XP practices at the same time did not prove to be an efficient adoption strategy.

Table 7.1: m3 Sociological Factors.

Context Factor	**Old**	**New**
Team Size	5	
Team Education Level	Masters: 5	
Experience Level of Team	$>$ 5 years: 3 $<$ 5 years: 2	
Domain Expertise	Low	
Language Expertise	Moderate	
Experience of Project Manager	Low	
Specialist Available	Usability Engineer	
Personnel Turnover	20 %	0 %
Morale Factors	Delays	Reduced process

Project-Specific Factors. Projects of varying size and scope are subject to differing risk factors that may substantially affect development quality and schedule, making it necessary to record this context information [149]. Table 7.2 compares the project-specific factors for the two releases. The numbers of new and changed classes, new and changed methods, new and changed lines of code, and system and component kilo lines of executable code (KLOEC) are quite low for a development team of that size. The reason for this is that on the one hand the development team was not developing full-time but was also doing a lot of research tasks. And on the other hand, all coding specific XP practices such as simple design, test-driven development, refactoring, pair programming, collective code ownership, coding standards, and spiking were applied in an exaggerated way for scientific purposes.

Table 7.2: m3 Project-Specific Factors.

Context Factor	**Old**	**New**
Delivered User Stories	66	103
Domain	Mobile multimedia	
Person Months	12	18
Elapsed Months	2	3
Nature of Project	Enhancement	
Relative Feature Complexity	Low	
Product Age	5 Months	8 Months
Constraints	Date constrained Scope constrained Science constrained	
New and Changed Classes Total Classes	13 13	14 18
New and Changed Methods Total Methods	79 79	45 104
New and Changed Lines of Code Delta Set	x x	x x
Component KLOEC	0.930	1.428
System KLOEC	0.930	1.428

Ergonomic Factors. The physical working environment can have a direct impact on the communication flow and overhead. This is particularly important to the XP core values of communication and feedback [149]. Table 7.3 documents the ergonomic factors of the m3 project. Because both the old and new releases had the same conditions, no comparison was made. Ideally, an XP team has an open space office environment. Since the m3 project is an XP research project, the physical working environment was set up exactly according to the methodology. Each of the five developers, as well as the product manager, had their own individual desk. Additionally, three pair programming stations were available. Everything was located in an open space office environment. Additionally, six whiteboards were available where release planning, iteration planning, backlog, and process metrics were recorded.

Table 7.3: m3 Ergonomic Factors.

Context Factor	**Old**	**New**
Physical Layout	Open space office environment	
Distraction level of office space	Low	
Customer Communication	On-site, constant interaction	

Technological Factors. General software development tools and practices, such as code inspections or project management, can have a dramatic effect on a project. Therefore, it is important to document these technological influences on a project as well [149]. The technological factors of the m3 project are summarized in Table 7.4. For both releases the same factors were recorded and for this reason no comparison was made. The planning game was used to establish release and iteration plans. User stories and task estimates were used to forecast release points and iterations based on the velocity of the project team. Concerning defect prevention and removal practices, the XP practices of pair programming, unit tests (test-driven development), continuous integration, and collective code ownership were employed. Testing by developers served as the primary means to identify potential problems in the code during development of both releases.

Geographic Factors. Team location and customer location may greatly impact the feedback cycle length during software development [149]. Table 7.5 documents

Table 7.4: m3 Technological Factors.

Context Factor	**Old**	**New**
Software Development Methodology	XP	
Project Management	Planning Game	
Defect Prevention and Removal Practices	Pair Programming Unit Tests Continuous Integration Collective Code Ownership	
External/System Test	Done by developers	
Language	Java	
Reusable Materials	Demodata and democlips	

the geographical factors. Because both the old and new releases had the same conditions, no comparison was made. The team worked collocated with one product manager always on site.

Table 7.5: m3 Geographic Factors.

Context Factor	**Old**	**New**
Team Location	Collocated	
Customer cardinality and location	1, on-site	
Supplier cardinality and location		

Developmental Factors. [23, 22] acknowledge that agile and plan-driven methodologies each have a role in software development and suggest a risk-based method for selecting the appropriate methodology. In the context of their work, plan-driven methodologies are defined as classical waterfall development methodologies. Their five project factors (*team size*, *criticality*, *personnel understanding*, *dynamism*, and *culture*) aid in selecting an agile, plan-driven, or hybrid process. Criticality indicates the magnitude of loss due to a defect, ranging from loss of comfort to loss of lives. Personnel understanding indicates the team's ability, ranging from the ability to perform procedural methods (Level 1B) to the ability to revise a method in an unprecedented situation (Level 2 & 3). Dynamism is a measure of requirements volatility, the average amount of changed requirements per month, and culture indicates the attitude of the team towards change [149].

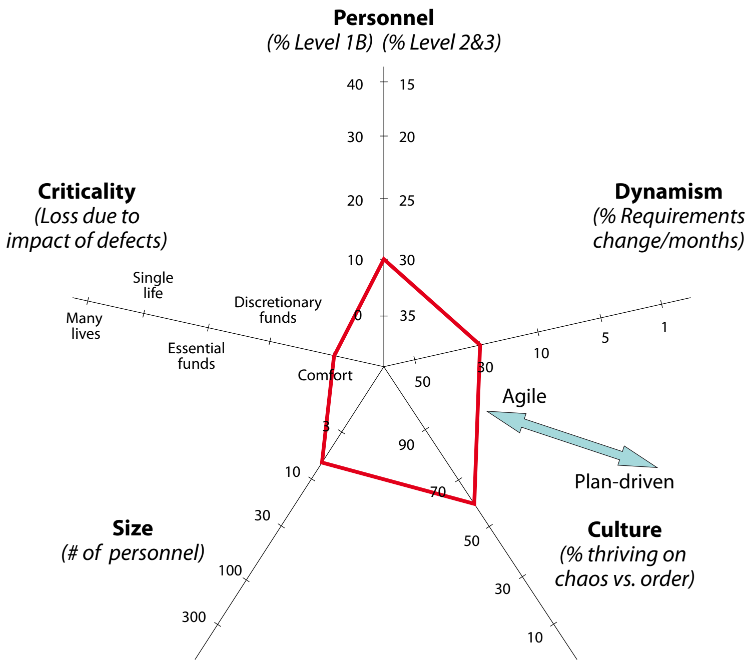


Figure 7.2: m3 Developmental Factors.

The developmental factors of the m3 development team are graphed on a polar chart's five axes, as shown in Figure 7.2. When a project's data points for each factor are joined, shapes distinctly toward the center of the graph suggest using an agile method. Shapes distinctly toward the periphery suggest using a plan-driven methodology. More varied shapes suggest a hybrid method of both agile and plan-driven practices [150]. The shape indicates that an agile method is appropriate. The evaluation of the developmental factors for both releases resulted in the same parameter values and therefore no comparison was made.

7.3.2 XP Adherence Metrics (XP-AM)

Most companies that use XP adopt the practices selectively and develop customized approaches to operate within their particular contexts [39]. For comparison purposes it is therefore essential to determine and record the subset of practices employed by a team. The XP-AM enables one to express concretely and comparatively the degree to which a team applies XP practices.

The adherence metrics contain subjective and objective measures as well as a qualitative analysis about the team's use of XP practices to triangulate the extent to which a team uses each of the XP practices [150]. The Shodan Adherence Survey is an in-process, subjective means of gathering XP adherence information from team members [90]. Survey respondents report the extent to which they use each practice on a scale from 0% (never) to 100% (always). For the old as well as for the new release, five team members took the survey at the reflection meeting at the end of the release. The objective measures describe the quantifiable adherence to XP practices, e.g., test coverage.

The combined results of the adherence metrics of the m3 project presented in this subsection are based upon three categories: Planning Adherence Metrics (Table 7.6), Coding Adherence Metrics (Table 7.7), and Testing Adherence Metrics (Table 7.8). For each category, first the results from objective measures followed by results from subjective measures are outlined.

Planning Adherence Metrics. The planning adherence metrics of the m3 project are summarized in Table 7.6. The release length as well as the iteration length of the new release increased in comparison to the old one. This resulted from experimenting with different release and iteration lengths. One outcome was that a 2 week iteration reduces the administrative overhead in terms of iteration planning and reflection meetings. But the progress of 1 week iterations is more stable due to the shorter development cycle permitting the feasibility of more accurate estimates for smaller stories. Requirements dynamism, representing the average amount of changed requirements per month, stayed approximately the same.

In the old release, the team held a mandatory stand up meeting every morning. The stand up meeting adherence dropped in the new release because constant knowledge distribution by means of pair programming was considered the main knowledge management tool. The software being developed was tested and deployed to a test system at the end of every iteration in both releases. Customer availability was quite low in the old release (roughly half the time) but accordingly, a business person playing the role of the product manager joined the team full-time for the new release. The subjective measures of the planning game reflect the fact that in the old release the team implemented this practice in a too structured and heavy-weight fashion and therefore, to a certain degree, it was neglected in the new release.

Coding Adherence Metrics. Table 7.7 depicts the coding adherence metrics of the m3 project. The pairing frequency for the old release was 100 %. This unusually high number resulted from the scientific background of the project, namely to examine XP in all its aspects. All production code as well as all spiking was done in pairs. Process data such as user stories, tasks, acceptance criteria, implementation

Table 7.6: m3 Planning Adherence Metrics.

Planning Metric	**Old**	**New**
Objective Metrics		
Release Length	2 months	3 months
Iteration Length	1 week	1-2 weeks
Requirements Dynamism	30 %	30 %
Subjective Metrics (Shodan)	**Mean**	
Stand up meetings	92 %	56 %
Short Releases	86 %	74 %
Customer Access / On-site Customer	48 %	72 %
Planning Game	96 %	68 %

time, and names of developers who paired, was recorded on whiteboards, and pictures were taken at the beginning and the end of each iteration. In the new release, pair programming frequency dropped because only production code was written in pairs and spikes were done by individual developers. Classical code inspection [150] frequency for both releases was 0 % since the XP methodology, interpreted in a strict sense, does not contain this activity. Continuous code inspection was done by means of constant pairing.

The subjective metric for pair programming for both releases reflects exactly the objective metric. Due to the fact that pair programming is a very controversial practice, there were a lot of discussions among the team members and a lot of experimenting took place. Some developers noted their dislikes for pairing because of differences in expertise. What is more, the team structure was heterogeneous in the sense of different nationalities. Naturally, cultural differences were an issue when it came to pairing. Another point often discussed was the value of this practice when implementing trivial tasks. Miscellaneous concepts for switching of pairs were examined. A natural and effective concept found was to create user stories with an estimated effort of approximately one day, so switching pairs was possible on a user story basis. The data for the old and the new release for refactoring are approximately the same and are strongly connected to the "Do the simplest thing that could possibly work" principle. Simple design was a source for discussion as well, and here different levels of expertise and differing knowledge about frameworks were obstructive. The strong interdependency between collective ownership and the concept of switching pairs was noted as well. For the old as well as the new release a dedicated continuous integration machine was available. Coding standards were adhered to a certain degree, but a checkstyle implementation in the continuous integration process was missing. The value of the sustainable pace metric decreased in the new release because the development process per se was changed and redefined a few times. The value for the metaphor in the old release was quite low because the basic requirements of the system were still not fixed at that point in time. In the new release, also because of the new team member acting as permanent customer on site, the overall system concept was refined resulting in a much clearer metaphor.

Table 7.7: m3 Coding Adherence Metrics.

Coding Metric	**Old**	**New**
Objective Metrics		
Pairing Frequency	100 %	84 %
Inspection Frequency	0 %	0 %
Subjective Metrics (Shodan)	**Mean**	
Pair Programming	98 %	68 %
Refactoring	66 %	62 %
Simple Design	76 %	74 %
Collective Ownership	86 %	86 %
Continuous Integration	100 %	98 %
Coding Standards	84 %	76 %
Sustainable Pace	82 %	70 %
Metaphor	46 %	68 %

Testing Adherence Metrics. The testing adherence metrics of the m3 project are illustrated in Table 7.8. The team's test coverage for both releases denotes that special emphasis was placed on testing. The team wrote automated unit tests before adding or changing functionality and before refactoring code. Test Run Frequency measures how often the automated tests are run. Ideally, the measure should be automated, and the value should be at least 1.0, indicating that each team member runs the test suite at least once per day [93]. The data shown was manually calculated and partially estimated and indicates that on average each member of the team ran the test suite three times a day. As can be read out of the ratio between Test LOC and Source LOC for the new release, a lot more test code was written. The percentage for changed classes with test class and new classes with test class for both releases stayed approximately the same.

Table 7.8: m3 Testing Adherence Metrics.

Testing Metric	**Old**	**New**
Objective Metrics		
Test Coverage	84,8 %	89,6 %
Test Run Frequency	3.0	3.0
Test LOC / Source LOC	0,355913978	0,816793893
New and Changed Classes w/ Test Class	38 %	37 %
New Classes w/ Test Class	27 %	25 %
Subjective Metrics (Shodan)	**Mean**	
Test First Design	44 %	38 %
Automated Unit Tests	68 %	48 %
Customer Acceptance Test	22 %	32 %

The numbers for the subjective metric for Test First Design are relative low for both releases because the evaluation was done in a strict way. Although, when implementing user stories, test-driven development was exercised, architectural decisions were made upfront and not while coding. All team members agreed that it was difficult to write automated unit tests for the user interface because of limitations of the used web component library. The number for the new release for Automated Unit Tests suggests that less unit tests were written. But essentially this resulted from different approaches of how to learn test-driven development in a proper way, namely with this practice a lot of experiments were done. For both releases the numbers of the Customer Acceptance Test metric are very low due to the unavailability of an appropriate testing framework adequate for a technically unskilled person, the customer on site, to write tests easily.

7.4 AUPPs

In this section, the three AUPPs are presented. The patterns are intended to be an extension of the already existing pattern collection Agile Adoption Patterns [40] and therefore adhere to the same pattern format with a few modifications. The pattern sections dependency diagram, sketch, and references have been left out for the sake of brevity.

7.4.1 Usability Expert Evaluation

Description. The Usability Expert Evaluation practice aims at having a usability engineer as part of the development team and frequently evaluating the system being developed. The evaluations are carried out at different development stages yielding continuous usability input and ensuring maximized usability for the software being created.

Business Value. The Usability Expert Evaluation practice helps to increase product utility, value to market, and quality to market in delivering usability expert evaluated software with maximized usability quality and a minimum of usability defects. Furthermore, costs are reduced and product lifetime is increased by lowering the maintenance effort of fixing usability defects after the software has been deployed.

Context. You are on a project creating a system having a graphical user interface. A usability engineer is part of the development team and the usability quality should be assured through all stages of development.

Forces. Usability is one of the most critical success factors in the adoption of a software system. Often the usability of a developed system is evaluated after completion and a usability engineer has not participated from inception. Continuous and iterative evaluation of the usability of a developed system by a usability engineer and incorporating the issues found assures high usability quality.

Therefore. Usability expert evaluations of a feature are performed in the planning process, the implementation process, and after the feature has been completed. User interface stories as well as corresponding low-fidelity paper prototypes are prepared before the actual planning game. The usability engineer evaluates and refines the prototypes used for the planning game of the next iteration. If technical obstacles arise during the implementation, such as usability demands that could be too expensive to implement, a solution in cooperation with the usability engineer is developed. At the end of the iteration, all new features are deployed and the usability is evaluated. The found usability issues of the usability engineer are reported in the form of user stories. This stories are used for the planning game of the next iteration.

But. Usability expert evaluations should be an integral part of the development process and evaluations should not be shifted to "when there is time". Furthermore, short release cycles require quick usability input during the development process. Therefore, for an agile project the usual way usability expert evaluations are done is not ideal. Instead of big, long lasting, application wide evaluations, smaller and faster evaluations on story level are necessary. Concerning cultural problems, it should be stated that developers have a technical approach and usability engineers have a mainly psychological focus. These differences can lead to problems and for this reason methods to prevent this have to be integrated in the collaboration process.

Variations. In agile projects, usability expert evaluations solve the problem of adhoc usability input. A special situation occurs when the usability engineer is not on site. Namely, different communication issues arise. To overcome these problems, communication channels such as instant messaging, email, and video-conferencing should be used. Since most of the time no synchronous communication is needed, exchanging of low-fidelity paper prototypes and deploying completed features for evaluation on a public server is sufficient. For quick feedback on usability issues during development the usability engineer should be available via phone or instant messaging.

7.4.2 Usability Test

Description. The Usability Test practice is used to capture usability issues experienced by real end users. After completing a significant set of features the system is put under test employing real end users in a real setting. This gives direct feedback of the usability quality of the software being created.

Business Value. The Usability Test practice helps to increase product utility, value to market, and quality to market in delivering usability tested software with maximized usability quality and a minimum of usability defects. The project visibility is improved by having end user tested features sets. Furthermore, costs are reduced and product lifetime is increased by lowering the maintenance effort of fixing usability defects after the software has been deployed.

Context. You are on a project creating a system having a graphical user interface. Real end users of the system are available and the usability quality should be assured through all stages of development.

Forces. Usability is one of the most critical success factors in the adoption of a software system. Often the usability of a developed system is evaluated after completion and real end user tests do not accompany the development from inception. Continuous and iterative evaluation of the usability of a developed system by real end users and incorporating the issues found assures high usability quality.

Therefore. Usability tests are conducted after a few iterations and after a significant set of features has been completed. A representative subset of real end users in a usability laboratory is introduced to the application together with a list of tasks intended to be accomplished. These empirical studies, the most fundamental usability evaluation method, are conducted to measure accuracy, user performance, recall-value, and the emotional response of the user. During the usability tests the users are observed using the application by a usability engineer and are asked to think aloud and verbalize their thoughts to get better insights into their mental model. Additionally, other methods such as interviews can be combined with usability tests. The encountered usability issues are reported by the usability engineer in the form of user stories intended to be fixed in the next iterations.

But. Usability tests require preparation of the system, appropriate test data, hardware, and organization of usability laboratory rooms and participants. Therefore, usability tests should be planned in advance and cannot be executed spontaneously. The usability issues arising from usability tests should quickly be inserted back into the development process in the form of user stories, because otherwise priorities could have changed and fixing of the usability issues found is delayed or even dropped. Furthermore, usability tests should be conducted only after a significant set of features has been completed.

Variations. In agile projects, usability tests are intended to give feedback on the usability of the developed system by real end users. If there is no usability laboratory and or representative subset of real end users available, usability tests should be conducted in a different setup. The physical execution of the usability test should be done at a different place, e.g., a meeting room, and the representative subset of real end users should be replaced by either the real customer on site, the real customer not on site who should be invited for testing, or the customer proxy on site. One shortcoming of this variation is that the real end user replacement may be already familiar with the system and therefore is not unbiased. Nevertheless, the resulting data in terms of discovered usability issues is valuable.

7.4.3 Automated Usability Evaluation

Description. The Automated Usability Evaluation practice targets having a set of automated tests testing the usability of the system being developed. Usability tests are written before writing production code and when fixing discovered usability issues. These automated usability tests are included in the continuous integration and the nightly build process. This ensures constant usability quality by automatically testing the usability of the software being created.

Business Value. The Automated Usability Evaluation practice helps to increase product utility, value to market, and quality to market in delivering automatically usability tested software with maximized usability quality and a minimum of usability defects. The project visibility is improved by having a set of usability tests reflecting progress. Furthermore, costs are reduced and product lifetime is increased by lowering the maintenance effort of fixing usability defects after the software has been deployed.

Context. You are on a project creating a system having a graphical user interface and the usability quality should be assured through all stages of development. A continuous integration environment is in place and an appropriate usability testing environment has been set up.

Forces. Usability is one of the most critical success factors in the adoption of a software system. Often the usability of a developed system is evaluated after completion and automated usability evaluation is not considered an integral part of the development. When implementing automated usability evaluation, knowledge about usability issues, both found in the project as well as from best industry practice, is effectively remembered and applied. Implementing automated usability tests assures high usability quality.

Therefore. Automated Usability Evaluation aims at having a set of automated usability tests which are executed in the continuous integration process, the nightly build process, and during development by each developer. As test-driven development allows to define the behavior of the application by writing automated tests first, automated usability evaluation allows one to define the usability of the application by writing automated usability tests first. When developing any user interface feature, automated usability tests for the user interface are written before any functionality is implemented. When fixing any discovered usability issue, automated usability tests for that usability issue are written before fixing it. The metrics according to which the automated usability tests are written are three different kind of user interface design metrics. Structural metrics, which are based on surface properties, e.g., number of visual components on the screen. Semantic metrics, which are content sensitive, e.g., all links to the same page have the same label. Procedural metrics, which are task sensitive, e.g., how to create a user account.

But. The quality and frequency of automated usability tests written depends essentially on the available infrastructure. Substantial factors are the usage of an appropriate testing framework and the existence of a continuous integration environment. In addition to this, the system should be based on automated (unit and usability) tests. Necessary test data should be in place and the execution time of the automated tests, or subparts of it, should be low in order to enable each developer to execute the automated tests, or subparts of it, locally.

Variations. In this context, automated usability tests are written to evaluate directly the user interface according to structural, semantic, and procedural user interface design metrics. Nevertheless, other types of automated usability tests are available as a variation to applying user interface design metrics. Automated log-file analysis in order to identify paths and execution time of user interactions, automated graph based navigation structure analysis for measuring complexity and consistency of navigation trees, and automated code based analysis according to design guidelines can be considered as a variation.

7.5 m3 AUPPs Implementation

The following three subsections present the AUPPs implementation of the scientific process.

7.5.1 Usability Expert Evaluation

The usability expert evaluation practice was implemented in the scientific process in the form of a variation of the pattern described in Section 7.4.1. Since there was no on-site usability engineer available, the cooperation between the development team and the off-site usability engineer was based on communication channels such as instant messaging, email, phone, and video-conferencing, as well as low-fidelity paper prototypes and a publicly accessible, at the end of each iteration deployed, application comprising completed features. Usability input by the usability engineer was given at different points in time: When writing user interface related stories, before and during the implementation, as well as after the implementation.

Before the planning game of the next iteration, a technician of the team paired with the customer in order to create simple and exact stories for discussion and estimation in the upcoming planning game. Together, they ensured that the story was written in a way that was unambiguous and understood by both sides. When writing user interface related stories, a low-fidelity paper prototype was created and sent to the usability engineer. The feedback from the usability engineer was then incorporated in the paper prototype. This resulted in the fact that when the user interface story was introduced in the actual planning game, the user interface story was already usability tested. During the implementation it was often the case that user interface questions arose which had to be clarified. For example, switching to a different web framework required clarification of user interface parts in terms of used components having a different look and feel. In these cases, ad hoc usability input was ensured through the usage of communication channels such as instant messaging, email, phone, and video-conferencing. After an iteration was completed, the application, including the implemented features, was deployed to a publicly accessible server. The usability engineer evaluated the completed features and any usability issues discovered were dealt with in the form of user stories for the next iteration [74].

The experience made was that sufficient ad hoc usability input can be given by the channels mentioned above, because most of the time synchronous communication between the project members and the usability engineer is not needed. Therefore, the geographical distance between the developers and the usability engineer was not a major obstacle in the cooperation process. Furthermore, used communication artifacts were not only low-fidelity paper prototypes, but occasionally hi-fidelity html prototypes for clarifying task based questions. The usability engineer was trained in story writing and the discovered usability issues were reported directly as user stories. This proved to be a very efficient cooperation process. In summary, the continuous evaluation of the user interface by the usability engineer and immediate incorporation of the usability issues found yielded a system with a very high usability quality [74].

7.5.2 Usability Test

The usability test practice was implemented in the scientific process following the pattern as well as the variations described in Section 7.4.2. A formal usability test with a representative subset of real end users, which was also attended by two developers, was conducted in a usability laboratory. The application being developed was usability tested in the open office environment on iteration basis by the customer on site, and frequently usability tested in the open office environment by the customer not on site. The discovered usability issues of the usability tests were transformed into user stories and scheduled for discussion for the next planning game.

One of the usability tests was carried out with 10 respondents using a mobile phone utilizing a classical task-based usability test method. Each respondent was asked to execute 5 different tasks. To gather general feedback and general opinions two interviews were carried out, one before and one after the task session (pre and post interviews). In addition to that, each task was accompanied by task specific post-questionnaires. After the test, the respondents had to judge three different visual design prototypes to capture the attitudes of the users towards the application in terms of graphical design, enjoyment, and aesthetics. The results of the tests suggested two main areas for improvement (example issues) [71]:

1. Usability: For choosing a date, a web-like calendar function should be integrated instead of a text based input field. All navigation menu elements should be separated from content menu elements. Interactive elements should be placed on a separate page and not on the bottom of a description page. Special attention should be given to feedback mechanisms.

2. Layout and Design: The contrast on the whole site should be more accentuated. Light text color on dark backgrounds should be avoided. Visually attractive design elements should be introduced in order to increase the attractiveness of the whole site. More colors should be used to eliminate monotony.

In contrast to the usability tests conducted with a representative subset of real end users, the usability issues resulting from the usability tests with the customer on site as well as the customer not site were fixed almost immediately. Since the usability tests with the customer on site and the customer not on site were conducted in the open office environment, the feedback cycle was much faster and more verbal than user story based. Small usability issues were fixed right away and usability issues which required more implementation effort were scheduled as user stories.

The experience made was that the amount of resulting user stories of the usability tests was quite high in relation to the overall amount of user stories. The outcome was that in almost every iteration at least a few user stories dealing with usability issues had to be implemented. Furthermore, it was noticed that the mindset of the developers who attended the usability test changed dramatically when seeing real users handling the application: They got more biased towards user-centered thinking. In summary, the amount of usability issues discovered and fixed resulted in a system with a very high usability quality [74].

7.5.3 Automated Usability Evaluation

The automated usability evaluation practice was implemented in the scientific process following the pattern described in Section 7.4.3. Automated usability tests have been written prior to the implementation of any user interface related functionality. Also, automated usability tests have been written prior to fixing any user interface related usability issue. The metrics applied comply with the metrics described in the pattern, namely structural, semantic, and procedural metrics. In addition to that, the set of automated usability test cases has been included in the continuous integration and the nightly build process.

For the purpose of being able to write automated usability tests, different frameworks have been used. All of the frameworks had their advantages and disadvantages in terms of how fast they can be learned, how easy they are to use, and how easy they are to integrate. The employment of the metrics yielded different kind of tests to be implemented (example tests):

1. Structural Metrics: Tests concerning the existence of buttons, input fields, headlines, labels, descriptions, FAQs, etc.

2. Semantic Metrics: Tests concerning the usage of capital letters on buttons, consistence of label names of links to the same page, existence of three dots on a button opening a popup window, etc.

3. Procedural Metrics: Tests concerning the execution of user initiated search queries, playback of videos, navigation on the whole site, browsing through sub pages, submission of forms, etc.

The experience made was that the integration of the automated usability tests into the continuous integration and the nightly build process led to a high transparency concerning the usability quality of the application. In the beginning, the usability engineer participated in writing usability tests, but at the point in time when the development team switched to a different testing framework, the complexity for a non-programmer was too high and the usability engineer quit to write usability tests. Also, a potential problem in executing automated usability tests is the necessity (of almost all frameworks) to have a running web server locally. Deployment and execution of the tests may take too long and obstructs the development. Another issue is the imperative of a different development style. Test-driven development by itself is already a challenge, but developing a user interface by means of test-driven development requires an even more radical approach to programming. In summary, automated usability tests and their integration into the automated build processes created a very high transparency concerning the usability quality of the system.

7.6 Other XP-EF

In this section, the results of the evaluation by means of the XP-EF of an industrial project at a company located in Vienna are presented. The component being customized is part of a speech-to-text system. More precisely, the component is the media mining server of the system, enabling users to access extracted speech in textual representation and their corresponding audio or video source. From this point on, the industrial project will be referred to as the other project. The software development process of the other project is based on agile principles and tailored to the needs and the environment of the company. Since the XP-EF allows the capture of any given agile software development process, the agile custom process of the other project was analyzed according to the metrics contained in the XP-EF. The characteristics of the other project placed it in the agile home ground [22] and XP was implemented in a modified form.

The XP-EF data collected is the outcome of the evaluation of a customization project. The project started in May 2009 and lasted for 5 months. Process evaluation points were project initiation and project termination. Therefore, captured data at project initiation is referred to as the old release and contains data based on the last customization project of the same system. Data captured at project termination is referred to as the new release and comprises a 5 month period of development. In the beginning of the project, several workshops on XP were held and it was agreed that within this project special emphasis should be placed on a few selected XP practices. The detailed data collected for each of the releases and the result in the form of XP-CF and XP-AM is outlined in the following two subsections.

7.6.1 XP Context Factors (XP-CF)

As already described in Section 7.3.1 the XP-EF utilizes six different categories of context factors. The context factors for the other project are presented in this subsection.

Software Classification. In the XP-EF, projects are classified as one of six software types: *systems* (used to control physical devices); *commercial* (leased or marketed to external clients); *information systems* (for business information); *outsourced* (developed under contract); *military*; or *end user* (private, for personal use). The software built within this customization project is sold to one single customer, and therefore the other project is classified as *commercial* software.

Sociological Factors. Table 7.9 illustrates the team conditions for both releases. The overall experience level of the team is quite high and almost all developers were already familiar with the code base. In contrast to this, the experience of the project manager was low. The fact that in the beginning of the project almost no requirements were existing, is reflected in the morale factors of the old release. This resulted in the fact that in the first month not much coding but a lot of requirements engineering took place. The morale factors in the new release refer to holidays. Since the project started in May and was planned to end in October, in the summer every member of the development team went on vacation. The consequence was increased communication overhead.

Table 7.9: Other Process Sociological Factors.

Context Factor	**Old**	**New**
Team Size	5	
Team Education Level	Masters: 5	
Experience Level of Team	$>$ 5 years: 2 $>$ 10 years: 3	
Domain Expertise	Moderate	
Language Expertise	High	
Experience of Project Manager	Low	
Specialist Available	Usability Engineer	
Personnel Turnover	0 %	
Morale Factors	No requirements	Holidays

Project-Specific Factors. The project-specific factors for the two releases are compared in Table 7.10. The product being customized was already 7 years old at project initiation and the complexity of the already existing software was quite high. This, as well as the relative feature complexity, which was also high, resulting in relatively few new and changed classes. In addition to that, the requirements had to be elaborated with a customer not on site, in a different country. The number of delivered user stories for the new release also reflects these circumstances.

Ergonomic Factors. The ergonomic factors of the other project are documented in Table 7.11. For both releases the ergonomic factors were the same and therefore no comparison was made. In contrast to the recommendation of agile methodologies that a development team should share an open space office environment, the physical layout of the other project was private offices with doors. To ensure a good information flow, regular meetings were held. The unavailability of a customer on site complicated requirements engineering a lot. Based on prototypes which had to be built, negotiations on the scope and size of each feature were made via mail and a few face-to-face meetings with the real customer.

Table 7.10: Other Process Project-Specific Factors.

Context Factor	**Old**	**New**
Delivered User Stories	N/A	15
Domain	Web Database development	
Person Months	0	25
Elapsed Months	0	5
Nature of Project	Enhancement	
Relative Feature Complexity	High	
Product Age	7 Years	
Constraints	Date constrained Scope constrained Resource constrained	
New and Changed Classes Total Classes	N/A 3085	54 3098
New and Changed Methods Total Methods	N/A 19960	93 20022
New and Changed Lines of Code Delta Set	N/A N/A	794 3087
Component KLOEC	176.5	177.1
System KLOEC	176.5	177.1

Table 7.11: Other Process Ergonomic Factors.

Context Factor	**Old**	**New**
Physical Layout	Private office with door	
Distraction level of office space	Low	
Customer Communication	Customer proxy available Real customer not on site	

Technological Factors. Table 7.12 summarizes the technological factors of the other project. For both releases, the same factors were recorded and for this reason no comparison was made. The software development methodology used was not pure XP but agile with XP practices. Concerning project management, the planning game and an XP based project management tool called XPlanner were used. For defect prevention, one dedicated tester was allocated, respectively automated testing in a nightly build environment was executed after the implementation.

Table 7.12: Other Process Technological Factors.

Context Factor	**Old**	**New**
Software Development Methodology	Agile with XP practices	
Project Management	Planning Game, XPlanner	
Defect Prevention and Removal Practices	1 dedicated tester	
External/System Test	Test after implementation	
Language	Java	
Reusable Materials	Code libs, 3rd party libs	

Geographic Factors. Table 7.13 documents the geographical factors. Because both the old and new releases had the same conditions, no comparison was made. What made things a little bit complicated was that the project had a customer who was located in a different country, in a different timezone, having a different culture and language. Also a supplier located in a different country in a different timezone but with the same language and same culture was involved. This hindered quick feedback cycles to a considerable degree.

Table 7.13: Other Process Geographic Factors.

Context Factor	**Old**	**New**
Team Location	Collocated	
Customer cardinality and location	1 Other country Other timezone Different culture Different language	
Supplier cardinality and location	1 Other country Other timezone	

Developmental Factors. A description of the parameters of the developmental factors of the XP-EF has already been given and can be found in Section 7.3.1. Figure 7.3 depicts the developmental factors of the other project for the old release respectively Figure 7.4 illustrates the developmental factors for the new release. The difference between the old and the new release is that the project's dynamism was reduced from 75 % to 25 % due to the fact that at project initiation the requirements were not completely defined. Nevertheless, both shapes indicate that an agile method is appropriate.

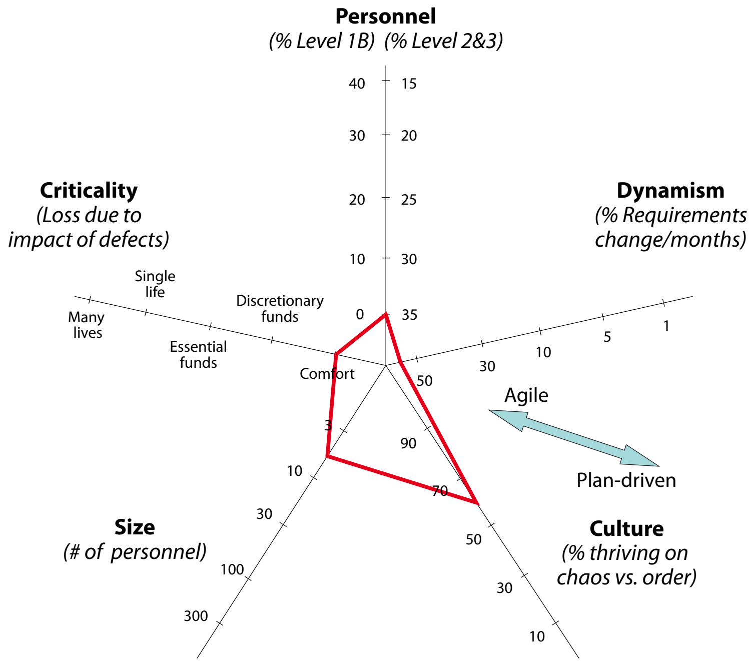


Figure 7.3: Other Process Old Release Developmental Factors.

7.6.2 XP Adherence Metrics (XP-AM)

Most companies that use XP adopt the practices selectively and develop customized approaches to operate within their particular contexts [39]. The software development process of the other project is exactly such a customized approach. The XP-AM enables one to express, concretely and comparatively, the degree to which a team applies XP practices. A description of the individual metrics, as well as the data collection process, can be found in Section 7.3.2.

The combined results of the adherence metrics of the other project presented in this subsection are based upon three categories: Planning Adherence Metrics (Table 7.14), Coding Adherence Metrics (Table 7.15), and Testing Adherence Metrics (Table 7.16). For each category, first the results from objective measures followed by results from subjective measures are described.

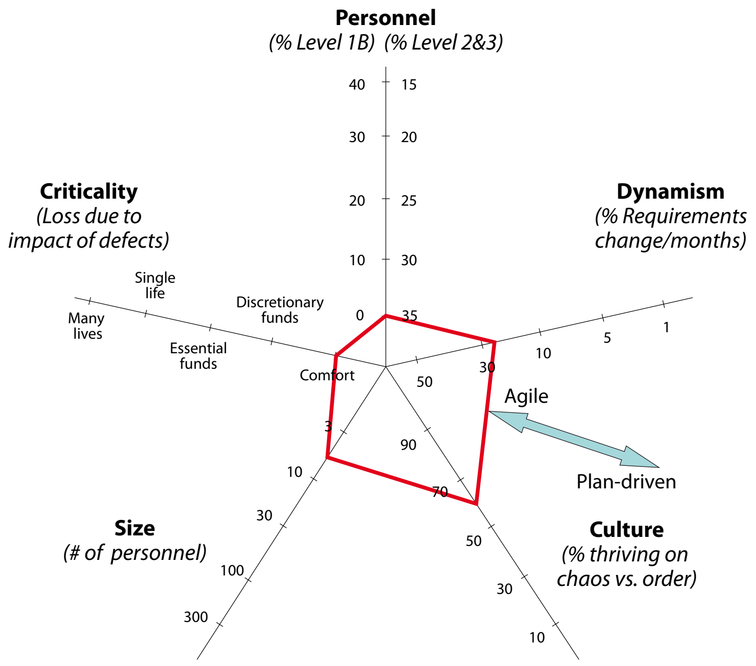


Figure 7.4: Other Process New Release Developmental Factors.

Planning Adherence Metrics. Table 7.14 illustrates the planning adherence metrics of the other project. The release length of the new release, respectively the length of the project itself, was 5 months with an iteration length of 1 week. The significant value of the objective planning metrics is contained in the requirements dynamism, representing the average amount of changed requirements per month. For the old release, capturing the point in time of project initiation, the requirements dynamism was 75 %. The reason for this high number was that at project initiation the requirements were very high level and had to be elaborated in detail during development. For the new release, after the requirements were defined, the requirements dynamism decreased to 25 %.

In the old release, the number of stand up meetings resulted from the fact that the team did not implement this XP practice at all. However, in the new release this practice was enforced. Although there were no official releases of the software, after completion of a feature the system was presented to the customer as a demo version. Customer access was one of the major challenges of this project because the customer was located in a different country, in a different timezone, having a different culture and language. Within the project there were efforts to establish the planning game practice. In the beginning, a couple of guided planning game sessions were held with the whole development team. But after a few months, the team tacitly dropped this practice and switched back to their old planning method being more a coordination meeting in contrast to the very interactive nature of an XP planning game.

Table 7.14: Other Process Planning Adherence Metrics.

Planning Metric	**Old**	**New**
Objective Metrics		
Release Length	N/A	5 Months
Iteration Length	N/A	1 Week
Requirements Dynamism	75 %	25 %
Subjective Metrics (Shodan)	**Mean**	
Stand up meetings	4 %	20 %
Short Releases	31 %	24 %
Customer Access / On-site Customer	30 %	36 %
Planning Game	34 %	20 %

Coding Adherence Metrics. The coding adherence metrics of the other project are summarized in Table 7.15. The pairing frequency was approximately the same in both releases. Even though the pair programming practice was not explicitly implemented in the software development process, the frequency for both releases was quite high due to the fact that pair programming happened in a natural way. Namely, the developers paired when doing complex tasks like building architecture or fixing bugs. Code inspection frequency for both releases was 0 %, since there was no classical designated review activity included in the process definition of the other project.

The subjective metric for pair programming for both releases reflects exactly the objective metric. For the old as well as for the new release the number for the refactoring practice is the same because during the customization of the system, necessary adjustments of the already existing infrastructure were made. The simple design adherence slightly increased in the new release as, in contrast to the last customization project, different developers were customizing the system. The number for collective ownership for both releases is quite low. For historical reasons, parts of the system were maintained by individual developers. Continuous integration was implemented in both releases. It needs to be noted that one nightly system test had a duration of a few hours. This, and used frameworks in the system hindered executing unit tests at individual work stations. Coding standard adherence dropped slightly in the new release towards the end of the project. In correlation to the coding standard adherence, the sustainable pace also dropped in the new release towards the end of the project. Due to the customization nature of the project and the relatively small amount of code written, in comparison to the whole system, the metaphor did not change much in the new release.

Table 7.15: Other Process Coding Adherence Metrics.

Coding Metric	**Old**	**New**
Objective Metrics		
Pairing Frequency	38 %	40 %
Inspection Frequency	0 %	0 %
Subjective Metrics (Shodan)	**Mean**	
Pair Programming	38 %	40 %
Refactoring	57 %	50 %
Simple Design	51 %	68 %
Collective Ownership	30 %	28 %
Continuous Integration	83 %	84 %
Coding Standards	68 %	52 %
Sustainable Pace	53 %	30 %
Metaphor	57 %	56 %

Testing Adherence Metrics. Table 7.16 depicts the testing adherence metrics of the other project. The test coverage for both releases is quite low because 7 years prior to project initiation, when the first version of the system was written, no special emphasis was placed on testing and appropriate test frameworks. Although efforts were undertaken to remedy this drawback, there never was enough time to do this. Test Run Frequency measures how often each team member runs the test suite once a day. A value of 1.0 would indicate that one developer executes the test suite once a day. Because of the unavailability of an appropriate test framework to execute the test suite locally, the value is 0 for both releases. The ratio between Test LOC and Source LOC, the percentage of changed classes with test class, and new classes with test class reflect this as well.

The numbers for the subjective metric for Test First Design are relative low for both releases for the same reason. Even though functional tests were implemented

Table 7.16: Other Process Testing Adherence Metrics.

Testing Metric	**Old**	**New**
Objective Metrics		
Test Coverage	5 %	5 %
Test Run Frequency	0	0
Test LOC / Source LOC	0,016974504	0,016915659
New and Changed Classes w/ Test Class	N/A	0 %
New Classes w/ Test Class	N/A	0 %
Subjective Metrics (Shodan)	**Mean**	
Test First Design	8 %	12 %
Automated Unit Tests	43 %	46 %
Customer Acceptance Test	26 %	48 %

as automated unit tests and executed in the nightly build process, there was no opportunity to run these tests locally since they required a setup of the whole system. Essentially, the numbers for the Automated Unit Test metric result from this circumstance. In the course of the project, communication with the customer not on site was established and several visits took place. Therefore, the adherence to the Customer Acceptance Test practice increased in the new release.

7.7 Other Process AUPPs Implementation

The following three subsections present the AUPPs implementation of the industrial process.

7.7.1 Usability Expert Evaluation

The usability expert evaluation practice was implemented in the industrial process in the form of a variation of the pattern described in Section 7.4.1. Since there was no on-site usability engineer available, the cooperation between the development team and the off-site usability engineer was based on communication channels such as email and phone. In addition, several usability expert evaluation meetings were held. Usability input by the usability engineer was given on different parts of the system being in different development stages. The whole system was usability expert evaluated and new features were developed in cooperation. Usability expert evaluations were executed iteratively for the existing as well as the newly developed features.

In the first usability expert evaluation meeting, which was attended by the usability engineer and the developers, the first activity was an evaluation of the already existing part of the system. The outcome of this first activity was a comprehensive list of usability issues, and as a side effect, an introduction to usability concepts and basics was given to the developers. The usability issues discovered were transferred into user stories and scheduled for subsequent iterations. Additionally, guidelines for further development were defined and already used user interface concepts were discussed and revised. The second activity was a presentation of the planned features and a joint development of the corresponding user interface parts by means of paper and screen prototyping. This resulted in already usability tested user interface stories. In the following usability expert evaluation meeting, attended by the usability engineer and the developers, all usability issues fixed in the previously existing part of the system were usability expert evaluated. Due to framework related issues, some usability bugs could not be implemented in the suggested way. For this, new solutions were developed and recorded in the form of user interface stories. Moreover, the user interface parts of the new implemented features were evaluated as well. Completed functionality and corresponding user interface parts were evaluated. User interface parts for not completed functionality were revised, and additional user interface stories were generated.

The experience made was that the initial usability expert evaluation of the system together with the developers consequently raised the usability awareness among them. Furthermore, a lot of knowledge transfer in terms of usability concepts took place, e.g., the usage of consistent navigation structures or the usage of common user interface metaphors. Obstacles appeared through the unavailability of a customer on site. Proposed and usability evaluated user interface parts were implemented and afterwards thrown away because they did not meet the expectations of the real customer in the sense of the underlying functionality. Nevertheless, common user interface development of the usability engineer and the developers for new features was considered very important, because of the sometimes high visualization complexity of the user interface parts. In summary, all fixed usability issues found in the initial evaluation of the already existing parts of the system and iterative evaluations of newly developed features increased the overall usability quality of the system according to the usability expert's evaluation to a very high degree.

7.7.2 Usability Test

The usability test practice was implemented in the industrial process in the form of a variation of the pattern described in Section 7.4.2. The software under development was usability tested in the office on an iteration basis by the customer proxy on site, respectively the product manager and other developers not working on this project.

In addition to that, the real customer not on site usability tested the application in several coordination meetings which were held on a regular basis in the office. The usability issues discovered by the usability tests were transformed into user stories and scheduled for the next iterations.

All usability tests were executed utilizing a classical task-based usability test method meaning that each tester was asked to accomplish several tasks. Usability tests with the customer proxy on site and other developers not working on the project were carried out each iteration. At the end of an iteration, each completed feature was usability tested ensuring continuous usability feedback accompanying the development. This fine-grained usability test cycle resulted in the fact that the usability of the developed features evolved in parallel to the developed functionality. Usability tests with the real customer not on site were executed in regular intervals in coordination meetings. Logically, the amount of features tested in these meetings was much higher. This coarse-grained usability test cycle also served the purpose of usability acceptance testing of completed feature sets. In contrast to the usability tests conducted with the real customer not on site, the usability issues resulting from the usability tests with the customer proxy on site as well as the other developers not working on the project were fixed and again usability tested almost immediately. Naturally, the availability of the customer proxy on site and the other developers not working on the project shortened the feedback cycle substantially. Small usability issues were fixed right away and usability issues which required more implementation effort were scheduled as user stories.

The experience made was that the employment of two different usability test cycles, fine- and coarse-grained, proved to be a very efficient process. On the one hand, the confidence of the customer proxy on site and the developers in building a system which has a high usability quality according to the usability issues found and fixed was raised. On the other hand, the customer not on site was presented feature sets which were already usability tested. In summary, the usage of usability test cycles of different granularity and the resulting and fixed usability issues ensured a high usability quality of the developed features.

7.7.3 Automated Usability Evaluation

The automated usability evaluation practice was implemented in the industrial process following the pattern described in Section 7.4.3 in conceptual form. At project initiation no usability test framework was existent in the system to be customized. For this reason, several frameworks were evaluated. In the course of the project it became clear that there is too little time available to accomplish the integration of a usability test framework in the system, the continuous integration, and the nightly build process. Therefore, the user stories corresponding to the usability issues found in the usability expert evaluations and the usability tests as well as the user stories for the new features were collected, intended to be transferred into a set of automated usability tests. It was planned that after project completion a framework would be integrated and the set of tests would be implemented.

The metrics applied comply with the metrics described in the pattern, namely structural, semantic, and procedural metrics. Schematically, the resulting tests correspond to the tests already described in Section 7.5.3 with a few modifications.

The experience made was that the integration of an appropriate usability test framework should happen as early as possible in product development. The effort of integrating such a framework in a later stage is increases proportionally. Also, the initial choice of the web framework used in the application should depend on the availability of corresponding usability test frameworks. In summary, the proper setup of an environment enabling the execution of automated usability tests is essential. This was not the case in the other project, and therefore the integration of a framework and the implementation of the tests was shifted to a later point in time.

7.8 Conclusion

This chapter presented three AUPPs, extending an already existing agile process pattern collection. The patterns described are: Usability Expert Evaluation, Usability Test, and Automated Usability Evaluation. The AUPPs are derived from the agile usability process of a scientific XP based project and are validated in an industrial agile based project. Both pattern implementations have been outlined. For the purpose of comparing the two process pattern implementations, both processes were evaluated using the XP-EF. For each of the processes, context factors were recorded and adherence metrics data (quantitative and qualitative) was collected at two points in time. The AUPP implementation results showed that the usability and the overall user experience of the developed systems improved significantly, as evaluated by the usability expert.

1. Scientific AUPPs Implementation: In the scientific project a high number of usability issues was found in Usability Expert Evaluations. The issues were fixed by means of test driven development, namely writing a test first for each of the discovered usability issues. Results of User Tests were incorporated and fixed in the same way. Also, new features were developed employing test driven development. The application of Automated Usability Evaluation metrics as well as the integration of automated tests for usability issues increased the usability of the developed application over time to a very high degree.

2. Industrial AUPPs Implementation: In the industrial project a high number of usability issues was found in Usability Expert Evaluations as well. In contrast to the scientific project, in the industrial project, development did not start from scratch. Therefore, the already existing part of the system was already usability expert evaluated. Resulting usability issues were fixed and corresponding usability tests have been recorded. Results of User Tests were incorporated and usability tests were recorded likewise. Concerning the employment of automated usability evaluation, framework and temporal limitations prevented immediate implementation of the tests. For this reason, the integration of a usability test framework and the implementation of the automated usability tests was scheduled for a later point in time. All usability issues found and fixed in the already existing part of the system as well as the continuous and iterative evaluation of the new features improved the overall usability substantially, as evaluated by the usability expert.

Within the context of this chapter, a systematic literature review on agile process patterns was presented. Results from this study showed that relatively little work exists in this area. For future work, the next logical step will be the transformation and adaption of different HCI instruments into a coherent AUPP language similar to the agile process pattern collection described in [40]. Of course, emerging AUPP needs to be implemented and validated in different agile processes in order to gain detailed knowledge on best practices in application.

Chapter 8

Concept and Design of a Contextual Mobile Multimedia Content Usability Study

The popularity of consuming multimedia content on mobile phones is increasing more and more, not only because of the availability of the technical infrastructure, but also because of the mobility in modern society. We are developing a mobile multimedia streaming application. The crucial factor for such applications in order to be adopted and successful is user acceptance. This chapter presents the preliminary concept and design of a contextual mobile multimedia content usability study. The study is conducted within a research project on agile software development methodologies with special emphasis on Extreme Programming and continuous usability evaluation. Past work included satisfaction of the needs of end users by means of focusing on user-experience in all steps of the development process. To gain scientific relevant data, the careful design of a study is considered most important. The study which will be conducted in October 2008 will give insights into mobile Human-Computer Interaction concerning the coherence of content types, consumption times, and consumption contexts.

8.1 Introduction

We are working on a research project where we are developing an application that enables a user to perform content-based search for audio and video content and play it via streaming on a mobile phone. The basic research goal is to examine agile software development methodologies, in particular Extreme Programming, with special emphasis on User-Centered Design. This is obtained by two means. On the one hand, we have established a development process where the quality focus is not only placed on technical excellence, but also on delivering a usability-tested high-quality end-product. On the other hand, we have created a testbed for effective and efficient mobile usability testing automating certain parts of the usability testing procedures.

Within this research project, a lot of topics already have been covered. Our adopted development process, the integration of Extreme Programming with User-Centered Design, examined in [72], facilitates user-orientation and at the same time preserves the social values of the development team. The techniques of enhancing Extreme Programming by integrating Human-Computer Interaction (HCI) instruments (user studies, personas, extended unit tests, usability tests, and usability expert evaluations) are treated in [151]. An iterative and user-centered approach to user interface design, where usability is evaluated in small iterative steps to gain insight into whether the users' functional and cognitive requirements are met, is published in [70]. Also, the results of a usability study applying a classical task-based usability test method [128], executed in January 2008, was made public in [71]. The outcome of a study on the relation of basic human values to behavior patterns of the usage and production of mobile multimedia content, conducted with a technique referring to the means-end theory, can be found in [96].

For us, the next logical step in the realm of mobile HCI is the evaluation of the coherence of consumption behavior of different content type categories and the consumption behavior in different contexts. This chapter presents the preliminary study setup of a usability evaluation study to be conducted in October 2008. The goal of the study is to examine mass-market capability of a mobile multimedia streaming application. More precisely, the aim is to analyze adoption and consumption behavior of such applications by means of a field trial, where diary studies and contextual interviews will be used. The resulting data is intended to give insights into what needs to be in place from the content type setup perspective in order to make such applications successful.

This chapter presents an overview about related work. The following section briefly outlines the application being developed. Then, the selection of the respondents is described. The next section examines the study setup concerning the media content, the diary study, and the contextual interview. Finally, a conclusion is given.

8.2 Related Work

This section provides an overview of related work in the field of usability evaluation of contextual mobile multimedia content. Basically, existing work in this area can be divided into the categories contextual studies, studies on mobile video consumption, and studies on technical issues in mobile multimedia consumption.

In [4] the importance of context in interactive mobile applications is stated, and definitions and categories of context, in order to create a framework for the development of context-aware applications, are presented. The work in [141] aims at understanding different mobile contexts and provides design implications for mobile and context-aware human-computer interaction. The necessity of considering the mobile user's context in conjunction with the user's cultural context is shown in [20].

In the realm of mobile video consumption [122] presents a study identifying the social motivations and values of people when using mobile video technologies. Also, [130] focuses on human behavior, human needs, and interaction design concerning the creation, management, and consumption of moving images using mobile devices.

In [80] the effects of codecs and combinations of audio and video streams with low bitrates and different contents on the perceived video quality of mobile devices are described. A methodology to evaluate the perceived quality of mobile video with variable physical quality is introduced in [104]. Interestingly, the outcome of the studies conducted with this methodology was that when watching high motion videos users prefer high-resolution images to high frame rate.

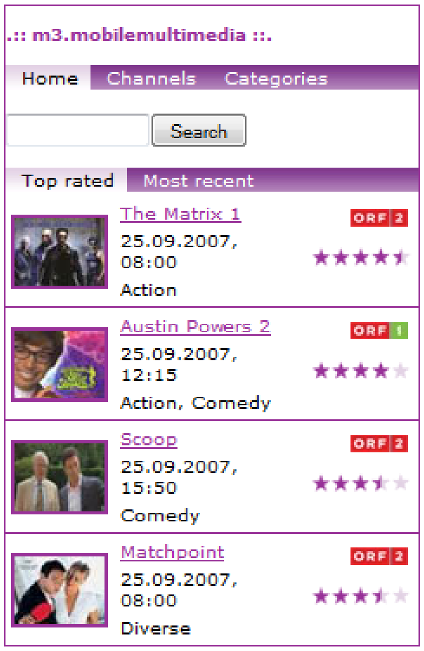
However, to the best of our knowledge, none of the existing work examines a contextual mobile multimedia study aiming at relating user context, consumption behavior, and content type categories.

8.3 Application

The application being developed enables a user to perform content-based search for audio and video content and play it on a mobile phone via streaming. The user is able to search not only in the meta-data, but also in the spoken words of the audio and video clips. The content includes radio and TV archive material, such as documentaries or other recordings of historical, political and cultural importance, discussion programs, movies, music videos, audio books, etc. The content setup for the study is especially tailored and described in Section 8.5.1. Moreover, the application is being designed keeping in mind the social interaction of users and provides several Web 2.0 features [70].

8.3.1 Features

The user interface of the application comprises several main features. The main screen (see Figure 8.1) provides the features "Search", "Top rated", and "Most recent" clips. Links to the "Channels" and "Categories" pages are presented. When one clicks on the title of a clip the "Clip Detail " page is shown. The application also implements other Web 2.0 features like “Rate” and “Comment” clips respectively “Others also watched” and “Tell a Friend” the link to a clip [71].



The Matrix 1
25.09.2007,
08:00
Action
ORF 2

Austin Powers 2
25.09.2007,
12:15
Action, Comedy
ORF 1

Scoop
25.09.2007,
15:50
Comedy
ORF 2

Matchpoint
25.09.2007,
08:00
Diverse
ORF 2

Figure 8.1: Home Page.

Search

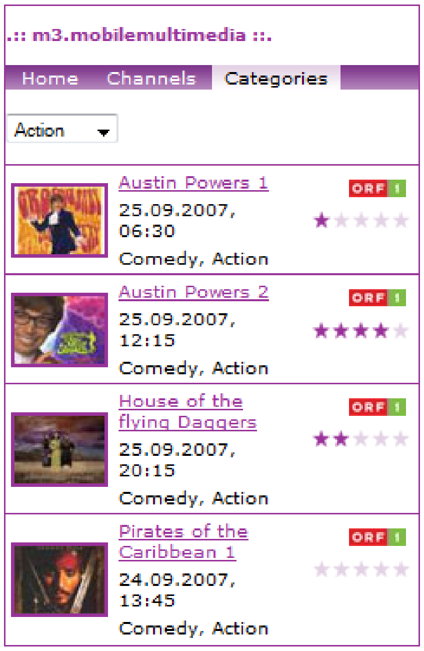
“Search” allows to search the whole content by entering keywords (see Figure 8.1). The returned search results are ordered by broadcast date (if any). For each result item, the clip’s title, link, rating, duration, originating channel (if any), content type, and a representative thumbnail image is shown. The user can play a clip by clicking on the image respectively go the “Clip Detail” page when clicking on the title.

Channels

“Channels” allows to browse the schedule of TV and radio channels. One channel lists the program schedule of a particular day. Different channels can be selected by using a drop-down box, listing all available channels, at the top of the page. The user can browse to previous or future days and is able to search within a channel. The search result items are presented in the same format as the search result items from the “Search” feature of the main screen.

Categories

The "Categories" page (see Figure 8.2) displays the whole content filtered by content type. By means of a drop-down box, containing all content type categories, the user can select to have only clips of a specific content type presented.



Austin Powers 1
25.09.2007, 06:30
Comedy, Action

Austin Powers 2
25.09.2007, 12:15
Comedy, Action

House of the flying Daggers
25.09.2007, 20:15
Comedy, Action

Pirates of the Caribbean 1
24.09.2007, 13:45
Comedy, Action

Figure 8.2: Categories Page.

Clip Detail

The system responds with the "Clip Detail" page when a user clicks on the title of a clip. On this page the user is given the possibility to "Rate" a clip, add a "Comment", or view comments. In addition to that, the "Others also watched" feature displays clips which have been watched by users with similar interests. The "Tell a friend" functionality enables a user to send the link to a clip to a friend by SMS or by email.

8.4 Selection of Respondents

For the study 16 respondents in the age group between 18 and 35 are chosen. From the chosen respondents the percentage of men and woman is balanced. Ideally, the respondents are interested in the following topics:

- They are interested in politics, economics, and other classical news themes. Respondents have to indicate to watch news on television on a regular basis (at least three times a week).
- The respondents are interested in technical content, especially in the field of computer science and information technologies. They are interested in products and news from Internet and telecommunication branches.
- They are interested in television series from the entertainment sector. Respondents have to indicate to watch pre-evening series on television on a regular basis (at least three times a week).
- The respondents are interested in music and music television. They are interested in particular bands in the pop and rock scene.

All respondents need to have at least some experience in using mobile multimedia, no matter if it is listening to music on an mp3-player or using the photo-gallery of their mobile.

8.5 Study Setup

The following section describes the study setup. First, the choice of the media content is examined. Then, the setup and execution of the individual methods of the study is outlined.

Fundamentally, the study is composed of two methods which are not depending on each other and therefore may be executed independently:

- Diary Study.
- Contextual Interview.

Both methods will be executed with the same respondents. Although there are no dependencies between the two, ideally the diary study is executed shortly before the contextual interview. This allows to examine user-experience issues when they are still in the minds of the respondents. As a prerequisite, each respondent fills out a questionnaire covering basic and demographic data as well as data concerning current mobile multimedia consumption.

8.5.1 Media Content

In order to be able to gather relevant data, the basic parameter for the study is the choice of specific content type categories. Four types of video content, each representing different information and entertainment levels, are chosen. Moreover, in the four media content types, the importance of the audio component (spoken text, audio) and the video component (visual information, graphics and pictures) is different. On the one hand, the video sequences for the diary study have a length between 15 and 20 minutes. On the other hand, the video sequences for the contextual interview have a shorter duration of approximately 10 minutes for the purpose of minimizing the overall interview time.

1. Music Video Content: When watching a music video, the audio as well as the video information is important. In practice, the audio content can exist without the additional video content. Consequently, the user does not have to pay primarily attention to the screen.
2. News Content: News content is a mixture of audio and video material. In addition to that, text is an important factor of news content as well. Nevertheless, video material strongly supports the given information. Often, news content is separated in different themes and topics.
3. Documentaries/Scientific and Technical Content: This type of content is comparable to news content, hence both, visual and auditive, information is important. In contrast to news content, scientific content forces the viewer to keep up with the video, since all information in the video sequence is important.
4. Television Series/Entertainment Content: Series have a continuous plot over the whole video sequence. In comparison to scientific and news content, the visual and auditive information is equally important to keep up with the story. But, even if missing seconds of both information, one is still able to follow the story.

8.5.2 Diary Study

In the diary study, each respondent is given a mobile device with which she or he is told to use the application for one week. The web interface, as well as the content presented to the respondents, is especially tailored to the diary study. The setup fulfills the following characteristics:

- The web interface for the diary study has limited possibilities in comparison to the actual application developed.
- The content type categories are limited to four different types (see Section 8.5.1).
- In each content type category approximately 40 to 50 clips are presented (except the news category, where there are less but only current clips).

For the sake of gathering relevant data for the diary study, two different groups of respondents are created:

- One group is told that they should watch at least X videos per day.
- The other group is told that they should watch less than X videos per day. This group is created in order to provide more realistic data during the one week study.

All respondents are told to choose time and place of consumption, as well as content types, on their own. After each video session, the respondents fill out a questionnaire and take a photo of the current context with the mobile device.

For the purpose of recording activities of the respondents, different log files (web server logs and streaming server logs) of the application are used. The user tracking of the application, respectively the activities recorded in the log files, give information about:

- When are videos watched.
- How long is each video watched.
- How many and which videos are watched during one video session.

With the overall setup of the diary study it is possible to get the following results:

- The average length of video sessions.
- Information on the content selection according to the day time and context.
- Overall information on the content selection (mix up of content types during sessions).
- Information on the context (when and where videos are watched and correlations to content types).

8.5.3 Contextual Interview

For the contextual interview, each respondent is given a mobile device. She or he is supposed to use the application in four controlled video sessions, each in a different context. The web interface presented is tailored as described in Section 8.5.2. In each video session, the respondent watches four different videos from four different content type categories. The content setup for the contextual interview is described in

Section 8.5.1. In the contextual interview, sixteen clips are shown to each respondent on the whole.

During one contextual interview (one video session in a specific context) the respondent is accompanied but not interrupted. The procedure is as follows:

- The interview starts with a pre-questionnaire covering basic and demographic data as well as data concerning mobile multimedia consumption.
- The respondent is explained the device.
- The respondent watches four videos from different content type categories.
- After each video, the respondent gives qualitative feedback to the interviewer and fills out post-video questionnaires (see Section 8.5.3).

After each contextual interview the respondent gives qualitative feedback to the interviewer and fills out post-session questionnaires (see Section 8.5.3).

For the purpose of getting significant results from the individual contextual interviews, different locations are necessary. The interviews are conducted in two indoor locations and two outdoor locations. The locations themselves, as well the order of the locations the interviews are conducted in, are as follows:

1. Indoor Location: Usability Lab.
2. Indoor Location: Cafe.
3. Outdoor Location: Tube and Bus: Respondents watch a video traveling a predefined distance in the public transport network. Respondents will have to change three times the means of transport having to cross train and bus stations and moving between people.
4. Outdoor Location: Public Spaces: Respondents watch a video while standing on and walking across a public space.

After all four sessions the respondent fills out an overall questionnaire.

Post-Video Questionnaires

The post-video questionnaires the respondent fills out after each video cover three different types of feedback:

- Emotion after Watching the Video: To capture the post-video-watching emotion the "SAM - Self Assessment Manikin" will be used [28].

- User Experience and Sensation of Usability: Similar to relevant concepts out of the Appeal Measurement questionnaire by Hoonhout [11].
- Qualitative Feedback concerning the Video.

Post-Session Questionnaires

The post-session questionnaires the respondent fills out after each contextual interview cover three different types of feedback:

- Context specific Feedback.
- User Experience and Sensation of Usability: Similar to relevant concepts out of the Appeal Measurement questionnaire by Hoonhout [11].
- Qualitative Feedback concerning the Session.

8.6 Expected Results

The basic concept of the study is that we use a method-mix of diary studies and contextual interviews. Therefore, from each individual method different results are expected. We also expect that the use of these two methods will give insight into the two approaches, hence backing up and assisting each other.

The expected findings from the diary study are:

- Coherence between daytime of consumption and content type.
- Coherence between context of consumption and content type.
- Average number of watched clips per session.
- Average time how long a clip is watched.

From the contextual interview, the following results are expected:

- Availability of different contexts: Ability to define “Contextas” like “Personas” [82] for mobile multimedia consumption.
- Availability of qualitative user feedback on content types in different contexts.
- Availability of Manikin [28] ratings according to content type and context.
- Availability of user experience data concerning context variables like light, noise, people, and others.

All findings, from the diary study, as well as from the contextual interview, are correlated to sex and age of the respondents.

8.7 Conclusion

The current trend in mobile multimedia consumption is more than obvious: it is becoming more and more popular. For this reason, we are developing a mobile multimedia streaming application with special focus on agile software development methodologies and usability. The most critical factor for this kind of application will be user acceptance. While developing this application we took all efforts to satisfy the needs of the end user. We combined Extreme Programming with User-Centered Design, integrated HCI instruments in our development process, developed an iterative UI development process, conducted usability studies and did studies on the relation of basic human values to behavior patterns of the usage and production of mobile multimedia content.

With the study whose setup is described in this chapter and which will be conducted in October 2008, we expect new insights in the field of mobile user-experience. In order to gain scientific relevant data, we consider the careful design of the study as being crucial. The data to be gathered promises the ability to draw conclusions on coherence of content types, consumption times, and consumption contexts. Furthermore, deriving different contexts, as well as the availability of qualitative and experience data, is an important goal. Future work includes publishing the results of this study.

Chapter 9

Epilogue

In this chapter general conclusions about the achieved results are drawn and interesting topics and open problems for future research are presented.

9.1 General Conclusions

In Chapter 2, the context in which this research took place was examined. The mobile multimedia application being developed was presented and an approach to application development, focusing on the adoption of agile software development methodologies and user-centered design, was outlined.

The reflections on the used Extreme Programming process with respect to fully implemented and partly implemented or modified Extreme Programming practices were discussed in Chapter 3.

In Chapter 4, an enhanced and fleshed out integrated agile usability process utilizing Human-Computer Interaction instruments was examined. Furthermore, a description of the experiences made in applying this integrated agile usability process was outlined.

An approach to integrating Extreme Programming and User-Centered Design is treated in Chapter 5. In addition, the results of a usability test conducted on the application being developed by means of this integrated approach are presented.

In Chapter 6, the lessons learned after utilizing an agile usability process for one and a half years were discussed. Each of the used Human-Computer Interaction instruments and their integration was reviewed. Especially the problem of having a usability engineer not on-site was identified as not severe, because most of the time no synchronous communication is needed.

Three Agile Usability Process Patterns, derived from the agile usability process of this scientific, Extreme Programming based, project, were examined in Chapter

7. The patterns have been validated in an industrial, Extreme Programming based, project. Both processes were evaluated with an Extreme Programming Evaluation Framework. The Agile Usability Process Pattern implementation results showed that the usability and the overall user experience of the developed systems improved significantly, as evaluated by the usability expert.

In Chapter 8, the setup of a contextual mobile multimedia study was presented. Also, the features of the developed application for the study were outlined.

9.2 Open Problems and Future Perspectives

One important observation is that although there has been a lot of effort on the Software Engineering side, as well as on the Usability Engineering side, to create integrated Agile Usability Processes, *The Agile Usability Process*, such as Extreme Programming or Scrum for agile software development, is still not available.

This has different reasons. On the one hand, industrial software development is still lacking in the awareness of the importance of usability and for this reason the need for integrated processes is not pushed by the industry. On the other hand, integrated agile usability process definitions are often very verbose because of their inherent complexity and are therefore not adopted. One way to alleviate this problem is to capture agile usability processes in the compressed form of Agile Usability Process Patterns clearly identifying the business value of applying such a pattern.

Therefore, an open issue for future research is the transformation and adaption of different Human-Computer Interaction instruments into a coherent Agile Usability Process Pattern language following the format and extending the agile process pattern language described in [40]. Of course, emerging Agile Usability Process Patterns need to be implemented and validated in different agile processes in order to gain detailed knowledge on best practices in application.

Bibliography

[1] IEEE Std. 1061. Software quality metrics methodology. IEEE Std. 1061, 1998.

[2] ISO 9126. Software product evaluation: Quality characteristics and guidelines for their use. ISO 9126, 1991.

[3] ISO 9241-11. Ergonomic requirements for office work with visual display terminals. ISO 9241-11, ISO, Geneva, 1998.

[4] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *HUC '99: Proceedings of the 1st international Symposium on Handheld and Ubiquitous Computing*, pages 304–307, London, UK, 1999. Springer-Verlag.

[5] Pekka Abrahamsson, Juhani Warsta, Mikko T. Siponen, and Jussi Ronkainen. New directions on agile methods: A comparative analysis. In *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, pages 244–254, Washington, DC, USA, 2003. IEEE Computer Society.

[6] Steve Adolph, Paul Bramble, Alistair Cockburn, and Andy Pols. *Patterns for Effective Use Cases (Agile Software Development)*. Addison Wesley, September 2002.

[7] Scott Ambler. Tailoring usability into agile software development projects. In *Maturing Usability*, volume 1 of *Human-Computer Interaction Series*, pages 75–95. Springer London, 2008.

[8] Ambysoft. It project success rates survey results. http://www.ambysoft.com/surveys/success2007.html. Last Visit: 2008.01.15.

[9] Jennitta Andrea. Putting a motor on the canoo webtest acceptance testing framework. In *XP*, pages 20–28, 2004.

[10] Sandrine Balbo, Joelle Coutaz, and Daniel Salber. Towards automatic evaluation of multimodal user interfaces. In *IUI '93: Proceedings of the 1st International Conference on Intelligent User Interfaces*, pages 201–208, New York, NY, USA, 1993. ACM Press.

[11] Christoph Bartneck. Interacting with an embodied emotional character. In *DPPI '03: Proceedings of the 2003 International Conference on Designing Pleasurable Products and Interfaces*, pages 55–60, New York, NY, USA, 2003. ACM.

[12] BBC. Online video eroding tv viewing. http://news.bbc.co.uk/2/hi/entertainment/6168950.stm. Last Visit: 2007.05.31.

[13] Kent Beck. *Extreme Programming Explained: Embrace Change (1st Edition)*. Addison-Wesley Professional, 1999.

[14] Kent Beck and Cynthia Andres. *Extreme Programming Explained: Embrace Change (2nd Edition)*. Addison-Wesley Professional, November 2004.

[15] Mike Beedle, Martine Devos, Yonat Sharon, Ken Schwaber, and Jeff Sutherland. Scrum: A pattern language for hyperproductive software development. In Neil Harrison, Brian Foote, and Hans Rohnert, editors, *Pattern Languages of Program Design 4*, pages 637–652. Addison Wesley, 2000.

[16] Raquel Benbunan-Fich and Alberto Benbunan. Understanding user behavior with new mobile applications. *Journal of Strategic Information Systems*, 16(4):393–412, 2007.

[17] Joseph Bergin. Patterns for agile development practice part 3 (version 4). In *PLoP '06: Proceedings of the 2006 Conference on Pattern Languages of Programs*, pages 1–14, New York, NY, USA, 2006. ACM.

[18] Hugh Beyer, Karen Holtzblatt, and Lisa Baker. An agile customer-centered method: Rapid contextual design. In *Extreme Programming and Agile Methods - XP/Agile Universe 2004*, volume 3134, pages 50–59. Springer Berlin/Heidelberg, 2004.

[19] Blinkx. Video search engine. http://www.blinkx.com. Last Visit: 2007.11.01.

[20] Jan Blom, Jan Chipchase, and Jaakko Lehikoinen. Contextual and cultural challenges for user mobility research. *Communications of the ACM*, 48(7):37–41, 2005.

[21] Stefan Blomkvist. Towards a model for bridging agile development and user-centered design. In *Human-Centered Software Engineering Integrating Usability in the Software Development Lifecycle*, pages 219–244. Springer Netherlands, 2005.

[22] Barry Boehm and Richard Turner. *Balancing Agility and Discipline: A Guide for the Perplexed.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[23] Barry Boehm and Richard Turner. Using risk to balance agile and plan-driven methods. *IEEE Computer*, 36(6):57–66, 2003.

[24] Barry W. Boehm. Software risk management: Principles and practices. *IEEE Software*, 8(1):32–41, 1991.

[25] Jan Borchers. *A Pattern Approach to Interaction Design.* John Wiley & Sons, May 2001.

[26] Teodora Bozheva and Maria Elisa Gallo. Framework of agile patterns. In Ita Richardson, Pekka Abrahamsson, and Richard Messnarz, editors, *EuroSPI*, volume 3792 of *Lecture Notes in Computer Science*, pages 4–15. Springer, 2005.

[27] Teodora Bozheva and Maria Elisa Gallo. Defining agile patterns. In *Rationale Management in Software Engineering*, volume 4, pages 373–390. Springer, 2006.

[28] M.M. Bradley and P.J. Lang. Measuring emotion: the self-assessment manikin and the semantic differential. *Journal of Behavioral Therapy and Experimental Psychiatry*, 25(1):49–59, March 1994.

[29] Christer Carlsson and Pirkko Walden. Mobile tv - to live or die by content. In *HICSS '07: Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, page 51, Washington, DC, USA, 2007. IEEE Computer Society.

[30] Stephanie Chamberlain, Helen Sharp, and Neil A. M. Maiden. Towards a framework for integrating agile development and user-centred design. In *7th International Conference on Extreme Programming and Agile Processes in Software Engineering*, volume 4044 of *LNCS*, pages 143–153, Heidelberg, Germany, 2006. Springer Verlag.

[31] Alistair Cockburn. *Agile Software Development: The Cooperative Game (2nd Edition) (Agile Software Development Series).* Addison-Wesley Professional, 2006.

[32] Larry L. Constantine. Process agility and software usability: Toward lightweight usage-centered design. Technical Report 110, Constantine & Lockwood, Ltd., 2001.

[33] Larry L. Constantine and Lucy A. D. Lockwood. *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1999.

[34] Larry L. Constantine and Lucy A. D. Lockwood. Usage-centered software engineering: An agile approach to integrating users, user interfaces, and usability into software engineering practice. In *ICSE '03*, pages 746–747. IEEE Computer Society, 2003.

[35] James O. Coplien. A generative development-process pattern language. In *Pattern Languages of Program Design*, volume 1, pages 183–237. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1995.

[36] James O. Coplien and Neil B. Harrison. *Organizational Patterns of Agile Software Development*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2004.

[37] Paloma Díaz, Mary Beth Rosson, Ignacio Aedo, and John M. Carroll. Web design patterns: Investigating user goals and browsing strategies. In *IS-EUD '09: Proceedings of the 2nd International Symposium on End-User Development*, pages 186–204, Berlin, Heidelberg, 2009. Springer-Verlag.

[38] Jen-Wen Ding, Chin-Tsai Lin, and Kai-Hsiang Huang. Ars: An adaptive reception scheme for handheld devices supporting mobile video streaming services. In *International Conference on Consumer Electronics. ICCE '06*, volume 1, pages 141– 142, 2006.

[39] Khaled El-Emam. Finding success in small software projects. *Agile Project Management*, 4, 2003.

[40] Amr Elssamadisy. *Agile Adoption Patterns: A Roadmap to Organizational Success*. Addison-Wesley Professional, 2008.

[41] Amr Elssamadisy and Jean Whitmore. Functional testing: A pattern to follow and the smells to avoid. In *PLoP '06: Proceedings of the 2006 Conference on Pattern Languages of Programs*, pages 1–13, New York, NY, USA, 2006. ACM.

[42] EMMA. Emma: Java code coverage tool. http://emma.sourceforge.net/. Last Visit: 2008.03.26.

[43] Everyzing. Everyzing. http://www.everyzing.com. Last Visit: 2007.11.01.

[44] Xavier Ferré, Natalia Juristo, and Ana Moreno. Which, when and how usability techniques and activities should be integrated. In *Human-Centered Software Engineering Integrating Usability in the Software Development Lifecycle*, pages 173–200. Springer Netherlands, 2005.

[45] Xavier Ferré, Natalia Juristo, Helmut Windl, and Larry Constantine. Usability basics for software developers. *IEEE Software*, 18(1):22–29, 2001.

[46] Xavier Ferré, Natalia Juristo Juzgado, and Ana Maria Moreno. Improving software engineering practice with hci aspects. In *SERA*, pages 349–363, 2003.

[47] Xavier Ferré, Natalia Juristo Juzgado, and Ana María Moreno. Framework for integrating usability practices into the software process. In *PROFES*, pages 202–215, 2005.

[48] Jennifer Ferreira, James Noble, and Robert Biddle. Agile development iterations and ui design. In *Agile 2007*, pages 50–58. IEEE Computer Society, 2007.

[49] Jennifer Ferreira, James Noble, and Robert Biddle. Agile development iterations and ui design. In *AGILE '07: Proceedings of the AGILE 2007*, pages 50–58, Washington, DC, USA, 2007. IEEE Computer Society.

[50] Jennifer Ferreira, James Noble, and Robert Biddle. Up-front interaction design in agile development. In *Agile Processes in Software Engineering and Extreme Programming*, pages 9–16. Springer Berlin, 2007.

[51] David Fox, Jonathan Sillito, and Frank Maurer. Agile methods and user-centered design: How these two methodologies are being successfully integrated in industry. In *AGILE '08: Proceedings of the Agile 2008*, pages 63–72, Washington, DC, USA, 2008. IEEE Computer Society.

[52] Robert Gittins and Sian Hope. A study of human solutions in extreme programming. In *PPIG 2001, The 13th Annual Psychology of Programming Interest Group Conference, Bournemouth, UK. 10th - 12th September 2008*, pages 41–51, 2001.

[53] Google. About gmail. http://mail.google.com/mail/help/intl/en/about.html. Last Visit: 2007.05.25.

[54] Bengt Göransson, Jan Gulliksen, and Inger Boivie. The usability design process - integrating user-centered systems design in the software development process. *Software Process: Improvement and Practice*, 8(2):111–131, 2003.

[55] Ian Graham. *A Pattern Language for Web Usability.* Pearson Education, January 2003.

[56] Jan Gulliksen, Bengt Göransson, Inger Boivie, Stefan Blomkvist, Jenny Persson, and Äsa Cajander. Key principles for user-centred systems design. *Behaviour & Information Technology, Special Section on Designing IT for Healthy Work*, Vol. 22 No. 6:397–409, 2003.

[57] Mariele Hagen and Volker Gruhn. Towards flexible software processes by using process patterns. In *IASTED Conference on Software Engineering and Applications*, pages 436–441, 2004.

[58] Jukka Haikara. Usability in agile software development: Extending the interaction design process with personas approach. In *XP*, pages 153–156, 2007.

[59] Monty L. Hammontree, Jeffrey J. Hendrickson, and Billy W. Hensley. Integrated data capture and analysis tools for research and testing on graphical user interfaces. In *CHI '92: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 431–432, New York, NY, USA, 1992. ACM.

[60] Marc Hassenzahl, Michael Burmester, and Franz Koller. Attrakdiff: Ein fragebogen zur messung wahrgenommener hedonischer und pragmatischer qualität. In G. Szwillus and J. Ziegler, editors, *Mensch and Computer 2003: Interaktion in Bewegung*, pages 187–196, Stuttgart, 2003. B. G. Teubner.

[61] Orit Hazzan and James E. Tomayko. Human aspects of software engineering: The case of extreme programming. In *XP*, pages 303–311, 2004.

[62] Hasko Heinecke, Christian Noack, and Daniel Schweizer. Constructing agile software processes. In *Third International Conference on eXtreme Programming and Agile Processes in Software Engineering (XP2002), 2002*, 2002.

[63] Deborah Hix and H. Rex Hartson. *Developing User Interfaces: Ensuring Usability through Product & Process.* John Wiley & Sons, Inc., New York, NY, USA, 1993.

[64] Andreas Holzinger. Usability engineering for software developers. *Communications of the ACM*, 48:71–74, 2005.

[65] Andreas Holzinger, Maximilian Errath, Gig Searle, Bettina Thurnher, and Wolfgang Slany. From extreme programming and usability engineering to extreme usability in software engineering education (xp+ue$\rightarrow$xu). In *COMPSAC*

'05: Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC'05) Volume 2, pages 169–172, Washington, DC, USA, 2005. IEEE Computer Society.

[66] Andreas Holzinger, Gig Searle, and Alexander K. Nischelwitzer. On some aspects of improving mobile applications for the elderly. In Constantine Stephanidis, editor, *Universal Access in HCI*, volume 4554 of *Lecture Notes in Computer Science*, pages 923–932. Springer, 2007.

[67] Williams Hudson. A tale of two tutorials: A cognitive approach to interactive system design and interaction design meets agility. *Interactions*, 12(1):49–51, 2005.

[68] Zahid Hussain, Martin Lechner, Harald Milchrahm, Sara Shahzad, Wolfgang Slany, Martin Umgeher, and Thomas Vlk. Optimizing extreme programming. In *ICCCE 2008: Proceedings of the International Conference on Computer and Communication Engineering, Kuala Lumpur, Malaysia*, pages 1052–1056. IEEE, 2008.

[69] Zahid Hussain, Martin Lechner, Harald Milchrahm, Sara Shahzad, Wolfgang Slany, Martin Umgeher, Thomas Vlk, and Peter Wolkerstorfer. User interface design for a content-aware mobile multimedia application: An iterative approach. In *Frontiers in Mobile and Web Computing: Proceedings of MoMM2007 & iiWAS2007 Workshops*, volume 231, pages 115–120, Jakarta, Indonesia, 2007.

[70] Zahid Hussain, Martin Lechner, Harald Milchrahm, Sara Shahzad, Wolfgang Slany, Martin Umgeher, Thomas Vlk, and Peter Wolkerstorfer. User interface design for a mobile multimedia application: An iterative approach. In *ACHI '08: Proceedings of the International Conference on Advances in Computer-Human Interaction 2008*, pages 189–194, 2008. Published 1st International Conference on Advances in Computer-Human Interaction (ACHI 2008) February 10-15, 2008 - Sainte Luce, Martinique.

[71] Zahid Hussain, Martin Lechner, Harald Milchrahm, Sara Shahzad, Wolfgang Slany, Martin Umgeher, and Peter Wolkerstorfer. Agile user-centered design applied to a mobile multimedia streaming application. In *USAB 2008*, volume 5298/2008 of *Lecture Notes in Computer Science*, pages 313–330. Springer Berlin / Heidelberg, November 2008.

[72] Zahid Hussain, Martin Lechner, Harald Milchrahm, Sara Shahzad, Wolfgang Slany, Martin Umgeher, and Peter Wolkerstorfer. Integrating extreme pro-

gramming and user-centered design. In *PPIG 2008, The 20th Annual Psychology of Programming Interest Group Conference, Lancaster University, UK. 10th - 12th September 2008*, 2008.

[73] Zahid Hussain, Martin Lechner, Harald Milchrahm, Sara Shahzad, Wolfgang Slany, Martin Umgeher, and Peter Wolkerstorfer. Concept and design of a contextual mobile multimedia content usability study. In *ACHI*, pages 277–282. IEEE, 2009.

[74] Zahid Hussain, Harald Milchrahm, Sara Shahzad, Wolfgang Slany, Manfred Tscheligi, and Peter Wolkerstorfer. Integration of extreme programming and user-centered design: Lessons learned. In *XP*, volume 31 of *LNBIP*, pages 174–179. Springer, 2009.

[75] Zahid Hussain, Wolfgang Slany, and Andreas Holzinger. Investigating agile user-centered design in practice: A grounded theory perspective. In A. Holzinger and K. Miesenberger, editors, *HCI and Usability for e-Inclusion. 5th Symposium of theWorkgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society*, volume 5889 of *Lecture Notes in Computer Science*, pages 279–289, Berlin, Heidelberg, New York, 2009. Springer.

[76] Melody Y. Ivory and Marti A. Hearst. The state of the art in automating usability evaluation of user interfaces. *ACM Computing Surveys*, 33(4):470–516, 2001.

[77] Timo Jokela and Pekka Abrahamsson. Usability assessment of an extreme programming project: Close co-operation with the customer does not equal to good usability. In *5th International Conference, PROFES '04*, pages 393–407, 2004.

[78] Capers Jones. *Software Assessments, Benchmarks, and Best Practices.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.

[79] Joost. Free online tv. http://www.joost.com. Last Visit: 2007.11.01.

[80] Satu Jumisko-Pyykkö and Jukka Häkkinen. Evaluation of subjective video quality of mobile devices. In *MULTIMEDIA '05: Proceedings of the 13th Annual ACM International Conference on Multimedia*, pages 535–538, New York, NY, USA, 2005. ACM.

[81] Jumpcut. Be good to your video. http://www.jumpcut.com. Last Visit: 2007.11.01.

[82] Plinio Thomaz Aquino Junior and Lucia Vilela Leite Filgueiras. User modeling with personas. In *CLIHC '05: Proceedings of the 2005 Latin American Conference on Human-Computer Interaction*, pages 277–282, New York, NY, USA, 2005. ACM.

[83] David Kane. Finding a place for discount usability engineering in agile development: Throwing down the gauntlet. In *ADC '03: Proceedings of the Conference on Agile Development*, pages 40–46, Washington, DC, USA, 2003. IEEE Computer Society.

[84] Eeva Kangas and Timo Kinnunen. Applying user-centered design to mobile application development. *Communications of the ACM*, 48(7):55–59, 2005.

[85] Joshua Kerievsky. *Refactoring to Patterns (Addison-Wesley Signature Series)*. Addison-Wesley Professional, August 2004.

[86] Barbara Kitchenham and Stuart Charters. Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 2007.

[87] Jesper Kjeldskov and Jan Stage. New techniques for usability evaluation of mobile systems. *International Journal of Human-Computer Studies*, 60(5-6):599–620, May 2004.

[88] Hendrik Knoche, John D. McCarthy, and M. Angela Sasse. Can small be beautiful?: Assessing image resolution requirements for mobile tv. In *MULTIMEDIA '05: Proceedings of the 13th Annual ACM International Conference on Multimedia*, pages 829–838, New York, NY, USA, 2005. ACM Press.

[89] Bill Krebs. Shodan 2.0 input metric survey. http://agile.csc.ncsu.edu/survey/shodan_survey.html. Last Visit: 2008.01.04.

[90] William Krebs. Turning the knobs: A coaching pattern for xp through agile metrics. In *Proceedings of the Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods - XP/Agile Universe 2002*, pages 60–69, London, UK, 2002. Springer-Verlag.

[91] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[92] Amy Law and Raylene Charron. Effects of agile practices on social factors. In *HSSE '05: Proceedings of the 2005 Workshop on Human and Social Factors of*

Software Engineering, volume 30, pages 1–5, New York, NY, USA, July 2005. ACM Press.

[93] Lucas Layman, Laurie Williams, and Lynn Cunningham. Exploring extreme programming in context: An industrial case study. In *ADC '04: Proceedings of the Agile Development Conference*, pages 32–41, Washington, DC, USA, 2004. IEEE Computer Society.

[94] Jason Chong Lee. Embracing agile development of usable software systems. In *CHI '06: CHI '06 Extended Abstracts on Human Factors in Computing Systems*, pages 1767–1770, New York, NY, USA, 2006. ACM.

[95] J.C. Lee and D.S. McCrickard. Towards extreme(ly) usable software: Exploring tensions between usability and agile software development. In *AGILE 2007*, pages 59–71, August 2007.

[96] Michael Leitner, Peter Wolkerstorfer, Reinhard Sefelin, and Manfred Tscheligi. Mobile multimedia: Identifying user values using the means-end theory. In *Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services*, pages 167–175, Amsterdam, The Netherlands, 2008. ACM.

[97] Matthew A. Lievesley and Joyce S. R. Yee. The role of the interaction designer in an agile software development process. In *CHI '06: Extended Abstracts on Human Factors in Computing Systems*, pages 1025–1030, New York, NY, USA, 2006. ACM.

[98] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, January 2003.

[99] Mikael Lindvall, Victor R. Basili, Barry W. Boehm, Patricia Costaand Kathleen Dangle, Forrest Shull, Roseanne Tesoriero, Lauri A. Williams, and Marvin V. Zelkowitz. Empirical findings in agile methods. In *Proceedings of the Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods - XP/Agile Universe 2002*, pages 197–207, London, UK, 2002. Springer-Verlag.

[100] LinesOfCodeWichtel. Linesofcodewichtel. http://www.andreas-berl.de/linesofcodewichtel/en/index.html. Last Visit: 2008.03.25.

[101] Manifesto. Manifesto for agile software development. http://www.agilemanifesto.org/, 2001. Last Visit: 2008.03.26.

[102] Joe Marasco. Software development productivity and project success rates: Are we attacking the right problem? http://www.ibm.com/developerworks/rational/library/feb06/marasco/index.html, Feb 2006. Last Visit: 2008.01.15.

[103] Angela Martin, James Noble, and Robert Biddle. Programmers are from mars, customers are from venus: A practical guide for customers on xp projects. In *PLoP '06: Proceedings of the 2006 Conference on Pattern Languages of Programs*, pages 1–9, New York, NY, USA, 2006. ACM.

[104] John D. McCarthy, M. Angela Sasse, and Dimitrios Miras. Sharp or smooth?: Comparing the effects of quantization vs. frame rate for streamed video. In *CHI '04: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 535–542, New York, NY, USA, 2004. ACM.

[105] Paul McInerney and Frank Maurer. Ucd in agile projects: Dream team or odd couple? *Interactions*, 12(6):19–23, 2005.

[106] Laurianne McLaughlin. Next-generation entertainment: Video goes mobile. *IEEE Pervasive Computing*, 06(1):7–10, 2007.

[107] Marc McNeill. User centred design in agile application development. http://www.thoughtworks.com/pdfs/agile_and_UCD_MM.pdf. Last Visit: 2008.03.30.

[108] Thomas Memmel, Harald Reiterer, and Andreas Holzinger. Agile methods and visual specification in software development: A chance to ensure universal access. In Constantine Stephanidis, editor, *Universal Access in HCI*, volume 4554 of *LNCS*, pages 453–462. Springer, 2007.

[109] John Mendonca and Jeff Brewer. *Lean, Light, Adaptive, Agile and Appropriate Software Development: The Case for a less methodical Methodology*, pages 42–52. IGI Publishing, Hershey, PA, USA, 2003.

[110] Xiang-xi Meng, Ya-sha Wang, Lei Shi, and Feng-jian Wang. A process pattern language for agile methods. In *Software Engineering Conference, 2007. APSEC 2007. 14th Asia-Pacific*, pages 374–381, Dec. 2007.

[111] Gerard Meszaros. *XUnit Test Patterns: Refactoring Test Code.* Addison-Wesley, 2007.

[112] Gerard Meszaros and Janice Aston. Adding usability testing to an agile project. In *AGILE '06: Proceedings of the Conference on AGILE 2006*, pages 289–294, Washington, DC, USA, 2006. IEEE Computer Society.

[113] Gerard Meszaros and Jim Doble. Metapatterns: A pattern language for pattern writing. In *3rd Pattern Languages of Programming Conference*, 1996.

[114] Eduard Metzker and Michael Offergeld. An interdisciplinary approach for successfully integrating human-centered design methods into development processes practiced by industrial software development organizations. In *EHCI '01: Proceedings of the 8th IFIP International Conference on Engineering for Human-Computer Interaction*, pages 19–34, London, UK, 2001. Springer-Verlag.

[115] Harald Milchrahm, Wolfgang Slany, and Andreas Holzinger. Agile usability process patterns. *Journal of Systems and Software*. Under review.

[116] Harald Milchrahm, Wolfgang Slany, and Andreas Holzinger. Process patterns for agile usability. In *ACHI*. IEEE, 2010. Accepted.

[117] Elden Nelson. Extreme programming vs. interaction design, 2002. FTP Online.

[118] Jakob Nielsen. *Usability Engineering (Interactive Technologies)*. Academic Press, 1993.

[119] James E. Nieters, Subbarao Ivaturi, and Iftikhar Ahmed. Making personas memorable. In *CHI '07: CHI '07 Extended Abstracts on Human Factors in Computing Systems*, pages 1817–1824, New York, NY, USA, 2007. ACM.

[120] Donald A. Norman. Logic versus usage: The case for activity-centered design. *Interactions*, 13(6):45–ff, 2006.

[121] Hartmut Obendorf and Matthias Finck. Scenario-based usability engineering techniques in agile development processes. In *CHI '08: CHI '08 Extended Abstracts on Human Factors in Computing Systems*, pages 2159–2166, New York, NY, USA, 2008. ACM.

[122] Kenton O'Hara, April Slayden Mitchell, and Alex Vorbau. Consuming video on mobile devices. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 857–866, New York, NY, USA, 2007. ACM.

[123] Fabio Patern, Angela Piruzza, and Carmen Santoro. Remote web usability evaluation exploiting multimodal information on user behaviour. In *Proceedings of 6th International Conference on Computer-Aided Design of User Interfaces CADUI'2006*. Springer Verlag, 2006.

[124] Jeff Patton. Hitting the target: Adding interaction design to agile software development. In *OOPSLA '02: OOPSLA 2002 Practitioners Reports*, pages 1–ff, New York, NY, USA, 2002. ACM.

[125] Tom Perry. The intermediate customer anti-pattern. In *AGILE '08: Proceedings of the Agile 2008*, pages 280–283, Washington, DC, USA, 2008. IEEE Computer Society.

[126] Kris Read, Grigori Melnik, and Frank Maurer. Examining usage patterns of the fit acceptance testing framework. In *XP*, pages 127–136, 2005.

[127] Linda Rising. Customer interaction patterns. In *Pattern Languages of Program Design 4*, pages 585–609. Addison-Wesley, 1999.

[128] Jeffrey Rubin and Theresa Hudson. *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests.* John Wiley & Sons, Inc., New York, NY, USA, 1994.

[129] Andreas Ruping. *Agile Documentation: A Pattern Guide to Producing Lightweight Documents for Software Projects.* John Wiley & Sons, Inc., New York, NY, USA, 2003.

[130] Barbara Schmidt-Belz and Matt Jones. Mobile usage of video and tv. In *MobileHCI '06: Proceedings of the 8th Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 291–292, New York, NY, USA, 2006. ACM.

[131] Reinhard Sefelin, Manfred Tscheligi, and Verena Giller. Paper prototyping - what is it good for?: A comparison of paper- and computer-based low-fidelity prototyping. In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, pages 778–779, New York, NY, USA, 2003. ACM Press.

[132] Ahmed Seffah, Michel Desmarais, and Eduard Metzker. Hci, usability and software engineering integration: Present and future. In *Human-Centered Software Engineering Integrating Usability in the Software Development Lifecycle*, pages 37–57. Springer Netherlands, 2005.

[133] Ahmed Seffah, Jan Gulliksen, and Michel Desmarais. An introduction to human-centered software engineering: Integrating usability in the development process. In *Human-Centered Software Engineering Integrating Usability in the Software Development Lifecycle*, pages 3–14. Springer Netherlands, 2005.

[134] Ahmed Seffah and Eduard Metzker. The obstacles and myths of usability and software engineering. *Communications of the ACM*, 47(12):71–76, 2004.

[135] Ahmed Seffah and Eduard Metzker. *Adoption-centric Usability Engineering: Systematic Deployment, Assessment and Improvement of Usability Methods in Software Engineering.* Springer Publishing Company, Incorporated, 2008.

[136] Helen Sharp and Hugh Robinson. An ethnographic study of xp practice. *Empirical Software Engineering*, 9(4):353–375, 2004.

[137] Mona Singh. U-scrum: An agile methodology for promoting usability. In *Agile, 2008. AGILE '08. Conference*, pages 555–560, Aug. 2008.

[138] Kenia Sousa, Elizabeth Furtado, and Hildeberto Mendonça. Upi: A software development process aiming at usability, productivity and integration. In *CLIHC '05: Proceedings of the 2005 Latin American Conference on Human-computer Interaction*, pages 76–87, New York, NY, USA, 2005. ACM.

[139] S.R. Subramanya and Byung K. Yi. User interfaces for mobile content. *IEEE Computer*, 39(4):85–87, April 2006.

[140] Desiree Sy. Adapting usability investigations for agile user-centered design. *Journal of Usability Studies*, 2(3):112–132, May 2007.

[141] Sakari Tamminen, Antti Oulasvirta, Kalle Toiskallio, and Anu Kankainen. Understanding mobile contexts. *Personal Ubiquitous Computing*, 8(2):135–143, 2004.

[142] Samira Tasharofi and Raman Ramsin. Process patterns for agile methodologies. In Jolita Ralyt, Sjaak Brinkkemper, and Brian Henderson-Sellers, editors, *Situational Method Engineering*, volume 244 of *IFIP*, pages 222–237. Springer, 2007.

[143] Bjornar Tessem. Experiences in learning xp practices: A qualitative study. In *XP*, pages 131–137, 2003.

[144] Anders Toxboe. Introducing user-centered design to extreme programming. http://blog.anderstoxboe.com/uploads/16082005_XP_and_UCD.pdf, May 2005. Last Visit: 2008.03.30.

[145] TVEyes. Tveyes. http://www.tveyes.com. Last Visit: 2007.11.01.

[146] Usability.gov. Step-by-step usability guide. http://www.usability.gov/. Last Visited: 2008.08.18.

[147] W3C. Notes on user centred design process (ucd). http://www.w3.org/WAI/EO/2003/ucd, April 2004. Last Visit: 2009.01.19.

[148] Sarah J. Waterson, Jason I. Hong, Tim Sohn, James A. Landay, Jeffrey Heer, and Tara Matthews. What did they do? understanding clickstreams with the webquilt visualization system. In *AVI '02: Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 94–102, New York, NY, USA, 2002. ACM.

[149] Laurie Williams, William Krebs, Lucas Layman, Annie Anton, and Pekka Abrahamsson. Toward a framework for evaluating extreme programming. In *8th International Conference on Empirical Assessment in Software Engineering (EASE 04)*, pages 11–20, May 2004.

[150] Laurie Williams, Lucas Layman, and William Krebs. Extreme programming evaluation framework for object-oriented languages version 1.4. Technical report, North Carolina State University, Department of Computer Science, June 2004.

[151] Peter Wolkerstorfer, Manfred Tscheligi, Reinhard Sefelin, Harald Milchrahm, Zahid Hussain, Martin Lechner, and Sara Shahzad. Probing an agile usability process. In *CHI '08: CHI '08 Extended Abstracts on Human Factors in Computing Systems*, pages 2151–2158, New York, NY, USA, 2008. ACM.

[152] XPlanner. Xplanner: (xp) project planning and tracking tool. http://www.xplanner.org/. Last Visit: 2008.01.04.

[153] Jason Yip. It is not just standing up: Patterns for daily stand-up meetings. In *PLoP '06: Proceedings of the 2006 Conference on Pattern Languages of Programs*, 2006.

[154] Jason Yip. Hands-on release planning with poker chips. In *PLoP '07: Proceedings of the 2007 Conference on Pattern Languages of Programs*, 2007.

[155] Mobile YouTube. Mobile youtube. http://m.youtube.com. Last Visit: 2007.11.01.

Acknowledgments

I would like to thank my parents for their support over the last years.

I owe gratitude to my supervisor Professor Dr. Wolfgang Slany for his constant encouragement, help, and invaluable supervision of my research.

Lots of thanks to my external supervisor Professor Dr. Andreas Holzinger for his guidance, support, and encouragement.

Many thanks to my colleagues at the Institute for Software Technology whose inspiration and help has contributed to the success of my research.

Harald Milchrahm

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am ……………………………   ………………………………………………..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

……………………………   ………………………………………………..
date   (signature)