INTEGRATION OF DISCOUNT USABILITY INTO SOFTWARE ENGINEERING TO ENHANCE DEVELOPMENT OF INTERACTIVE MOBILE PLATFORM BASED APPLICATIONS

DENISH OMONDI OTIENO

MASTER OF SCIENCE

(Software Engineering)

JOMO KENYATTA UNIVERSITY OF

AGRICULTURE AND TECHNOLOGY

2015

Integration of discount usability into software engineering to enhance Development of Interactive mobile platform based applications

Denish Omondi Otieno

A thesis submitted in partial fulfillment for the degree of Master of Science in Software Engineering in the Jomo Kenyatta University of Agriculture and Technology

2015

DECLARATION

This thesis is my original work and has not been presented for a degree in any other University

Signature: _____ Date: _____

Denish Omondi Otieno

This thesis has been submitted for examination with our approval as the university supervisors

Signature: _____ Date: _____

Dr. Wilson Cheruiyot

JKUAT, Kenya

Signature: _____ Date: _____

Dr. Michael Kimwele

JKUAT, Kenya

DEDICATION

This thesis is dedicated to my Parents, my loving mother Julian Atieno Ng'iela for the great support, training and upbringing in my life to my Siblings Jacob Ochieng, Beryl Akoth, Nancy Achieng and Sharon Anyango for their support throughout my education, not forgetting all my friends and relatives who encouraged me all the time and above all to Almighty God.

ACKNOWLEDGEMENTS

The undertaking and completion of this research work was made possible by a number of people, to whom I am profoundly grateful. I' am particularly indebted to my supervisors Dr Wilson Cheruiyot and Dr Michael Kimwele for their guidance and encouragement in the course of the research. I also want to express my deep gratitude to the Director School of Computing and Information Technology Dr Stephen Kimani who introduced me to research. Appreciation goes to the lecturers of the School of Computing and Information Technology main Campus Jomo Kenyatta University of Agriculture and Technology (JKUAT), who faithfully imparted their knowledge and skills throughout the course.

TABLE OF CONTENTS

DECLARATION ..... **ii**

DEDICATION ..... **iii**

ACKNOWLEDGEMENTS ..... **iv**

LIST OF TABLES ..... **ix**

LIST OF FIGURES ..... **x**

LIST OF ABBREVIATIONS/ACRONYMS ..... **xi**

ABSTRACT ..... **xiii**

CHAPTER ONE **1**

1.0 INTRODUCTION **1**

1.1 Preliminary Definitions 1

1.1.1 Software Engineering 1

1.1.2 Discount Usability 1

1.1.3 Mobile Devices 2

1.2 Research Background 3

1.3 Statement of the Problem 4

1.4 Justification 4

1.5 Objectives 5

1.5.1 Broad Objective 5

1.5.2 Specific objectives 5

1.6 Research Questions 6

1.7 Scope of Study 6

1.8 Thesis Structure 6

CHAPTER TWO **9**

2.0 LITERATURE REVIEW **9**

2.1 The Need for Usability Engineering 9
2.1.1 Discount usability engineering methods 9
2.1.2 Categories of mobile applications 11
2.1.3 What Makes Mobile Devices Different? 15
2.1.4 Mobile applications development 16
2.1.5 Unique development challenges for mobile devices software engineering 17
2.1.6 Usability Engineering in Software Engineering identifying the gaps in Industry Practices 22
2.2 Agile Process Models 24
2.2.1 HCI Issues with Agile Processes 27
2.2.2 Agile development for mobile applications 30
2.2.3 Review of mobile applications development processes - using an agile approach 32
2.4 Chapter Summary 46
CHAPTER THREE 47
3.0 RESEARCH METHODOLOGY 47
3.1 Research Design 47
3.2 Target population 50
3.3 Sample and Sampling Technique 50
3.4 Data Collection Instruments 51
3.5 Data Processing and Analysis 52
3.6 Chapter Summary 52
CHAPTER FOUR 53
4.0 RESULTS AND DISCUSSIONS 53
4.1 A Framework for Integrating Usability Engineering Into Mobile Platform-Based Devices Software Engineering 53
4.1.1 Planning 58

4.1.1.1 Question 1 What is required? 59
4.1.1.2 Question 2 What will the system do? 63
4.1.2 Analysis 64
4.1.2.1 Question 3 Have we got the requirements right? 65
4.1.3 Design 66
4.1.3.1 Question 4 Have we understood our targeted users? 66
4.1.3.2 Question 5 How should we respond? 67
4.1.3.3 Question 6 How are we doing? 69
4.1.4 Development and Implementation 70
4.1.4.1 Question 7 How should the design be achieved? 70
4.1.5 Testing 71
4.1.5.1 Question 8 How does it perform? 71
4.1.5.2 Question 9 Which areas need more work? 72
4.1.6 Maintenance 73
4.1.6.1 Question 10 How do we compensate for failures? 73
4.1.7 Nature of the framework 74
4.2 Integrating Discount Usability into Mobile Agile Process Model 75
4.2.1 Why integrate 75
4.2.2 Convergence points between agile and usability 77
4.2.2.1 Human-centered development 77
4.2.2.2 Cyclical development 77
4.2.2.3 Continuous testing 78
4.2.3 Divergence points between agile and usability 78
4.2.3.1 Working software vs design documentation 78
4.2.3.2 Phased vs incremental approaches 79
4.2.3.3 Test driven development vs usability evaluations 80

4.2.3.4 Shared understanding vs distinct roles 81
4.2.3.5 Customer focus vs end user focus 81
4.2.4 Approach to Integration 82
4.2.5 The Extended Mobile-D Agile Process Model 83
4.2.5.1 Explore 85
4.2.5.2 Initialize 86
4.2.5.3 Productionize and Stabilize 86
4.2.5.4 System Test & Fix 87
4.2.5.5 Extended Mobile-D with added Evolve phase 87
4.3 Evaluation of Effectiveness of Integration of Discount Usability into Software Engineering 91
4.3.1 Usability Factors for Software Engineering Methodology 91
4.3.2 Findings 93
4.3.2.1 Sample demographics 93
4.3.2.2 Summary of products development information 97
4.3.2.3 Summary of model survey 98
4.3.3 Principle-based analytic evaluation of Extended Mobile-D 111
4.3.3.1 Evaluating whether Extended mobile-D model is an agile process 111
4.4 Chapter Summary 114
CHAPTER FIVE 115
5.0 CONCLUSIONS AND FURTHER WORK 115
5.1 Findings and Contributions 115
5.2 Future work 119
REFERENCES 120
APPENDICES 128

LIST OF TABLES

Table 2: 1 Mobile platform based agile methodologies 32
Table 2: 2 Three layer structure of RaPiD7 38
Table 2: 3 Phases of Hybrid engineering methodology 39
Table 2: 4 Process assets of MASAM 41
Table 2: 5 Phases of MASAM process 42
Table 2: 6 DMAIC 5 Phases of SLeSS approach 45
Table 4: 1 Multi-Disciplinary Framework 54
Table 4: 2 Summary of general Information 95
Table 4: 3 Totals of respondents according to software development 97
Table 4: 4 Summary of Understandability 99
Table 4: 5 Summary of Learnability 101
Table 4: 6 Summary of Applicability 103
Table 4: 7 Summary of Usefulness 104
Table 4: 8 Summary of Satisfaction 106
Table 4: 9 Summary of steps against the scores 108
Table 4: 11 Agile Manifesto 111

LIST OF FIGURES

Figure 2:1 Categories of mobile applications, based on (Oinas – Kukkonen & Kurkela, 2003) (Unhelkar & Murugesan, 2010) and (Kunz & Black, 1999) 14
Figure 2:2 Agile Process 25
Figure 2:3 Phases of Mobile-D software development process 33
Figure 2:4 Mobile-D phases and stages. Adapted from (VTT Electronics, 2006) 34
Figure 2:5 Mobile-D with added Evolve phase. Adapted from (VTT Electronics, 2006) 37
Figure 3:1 Baskerville approach 48
Figure 4:1 Framework's Cyclic Nature 75
Figure 4:2 Mobile-D associate stages 84
Figure 4:3 Extended Mobile-D with Evolve phase 88
Figure 4:4 Extended Mobile-D model in line with Multidisciplinary Framework 90
Figure 4:5 X against Y in Sd 108
Figure 4:6 X against Y in d 109
Figure 4:7 X against Y in n 109
Figure 4:8 X against Y in a 110
Figure 4:9 X against Y in Sa 110

LIST OF ABBREVIATIONS/ACRONYMS

AM	Agile modeling
ASD	Adaptive Software Development
ANSI C	American National Standards Institute for the C programming language
APIs	Application programming interface
BDUF	Big Design Upfront
CSS	Cascading Style Sheet
DSDM	Dynamic Systems Development Method
DMAIC	Define, Measure, Analyze, Improve and Control
GUI	Graphical User Interface
GPS	Global Positioning System
HCI	Human Computer Interaction
JAD	Joint Application Development
ICT	Information Communication Technology
LSS	Lean Six Sigma
LOC	Line of Code
MASAM	Mobile Application Software Agile Methodology
NPD	New Product Development
PC	Personal Computer
QFD	Quality Function Deployment
RaPiD7	Rapid Production of Documentation with 7 Steps
RAD	Rapid Application Development
RUP	Rational Unified Process
SBD	Scenario Based Design
SDKs	Software Development Kits
SE	Software Engineering
SPSS	Statistical Package Software System
SPEM	Software and Systems Process Engineering Meta Model
SRS	software requirements specification
TDD	Test-Driven Development
TRIZ	Theory of Inventors Problem Solving

UI	User interface
UMM	Usability Maturity Model
WAP	Wireless Application Protocol
XP	Extreme Programming

ABSTRACT

Reliability of an interactive mobile computing device or the lack of it is often reflected in user satisfaction. The rapid proliferation and ubiquity of smart devices in the consumer market has forced the Software Engineering (SE) community to quickly adapt development approaches conscious of the novel capabilities of mobile applications. However, the growth of this new computing platform has outpaced the software engineering work tailored to mobile applications development. Designs in Human computer interaction (HCI) aim to create interactive products that are easy and enjoyable to use. However, owing the major gaps between HCI and SE in theory and practice, the multidisciplinary nature of HCI and the different value systems of interface users from various backgrounds and experiences, it is highly challenging for designers to create applications which are usable and affordable to such a heterogeneous set of users. Nowadays, users complain about the bad interaction design of mobile applications. The question is whether this problem is caused by the bad design of products or by the users' ignorance of the logics of HCI design. In this research we focus on integration of discount usability techniques specific to mobile devices into the core values of SE process model without disrupting the same values. We investigate current literature on software development and Usability engineering and propose a process framework. In this framework we identify the essential discount usability techniques, methods, deliverables, and skills relevant to mobile devices software engineering. We further use this framework as a baseline for integrating the essential discount usability techniques and propose an Extended Mobile-D process model. To demonstrate the validity of the Extended integrated process model and framework we assume that it is possible to express numerically the extent to which a team achieves its product goal by following a prescribed process model to the extent X a project could achieve its goals to the extent Y, if we can demonstrate that for every X2 that is greater than X1, Y2 is greater than Y1 in most cases, we can conclude that the process model in question works.

CHAPTER ONE

1.0 INTRODUCTION

1.1 Preliminary Definitions

1.1.1 Software Engineering

Software has become critical to advancement in almost all areas of human endeavors. The art of programming only is no longer sufficient to construct large programs. There are serious problems in the cost, timeliness, maintenance and quality of many software products. Software engineering has the objective of solving these problems by producing good quality, maintainable software, on time and within budget using processes, methods and tools. At the first conference on software engineering in 1968, (Fritz, 1968) defined software engineering as the establishment and use of sound engineering principles in order to obtain economically developed software that is reliable and works efficiently on real machines. (Stephen, 1990) defined the same as a discipline whose aim is the production of quality software, software that is delivered on time, within budget, and that satisfies its requirements. Software engineering is the branch of systems engineering concerned with the development of large and complex software intensive systems. It focuses on,

- The real-world goals for, services provided by, and constraints on such systems.
- The precise specification of system structure and behavior, and the implementation of these specifications.
- The activities required in order to develop an assurance that the specifications and real world goals have been met.
- The evolution of such systems over time and across system families.

1.1.2 Discount Usability

The ISO 9241-11 defines usability as “The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use”. No one explicitly denies the benefits of conducting usability tests earlier to

releasing products, many afraid to adopt it due to the commonly seeming fact that it is expensive and time consuming.

In attempt to correct this perception, (Nielsen, 1992) popularized the term “Discount Usability”. He argues that significant value can be gained by introducing low-cost and easily accessible usability testing methodologies over expensive test labs and sophisticated experimentation. According to Nielsen, usability does not have to slow down your project or be complex or expensive to be effective, in other words Nielson recommends that usability tests do not have to be complex to be effective. Discount methods are deliberately informal and rely less on statistics and more on interface engineer’s ability to observe users and interpret results. In order to find inexpensively usability problems in a system, many lightweight, easy to learn and fast to conduct usability-testing techniques have been proposed by usability experts.

The "discount usability engineering" testing methods are; Scenarios, Simplified thinking aloud, Heuristic evaluation and Card sorting. Scenarios are kind of prototypes for getting quick and frequent feedback from users. It can be implemented as lo-fidelity prototypes or Hi-fidelity prototype. Simplified thinking aloud is an interview based technique where test users are asked to perform a set of tasks using the product or a prototype and explain what they're thinking about while working with the product's interface. If the user expresses that, the sequence of steps to accomplish their task goal is different from what they expected then the interface is complex. A heuristics is set of guidelines given to the evaluators to identify many usability problems. It is best used early in the design phase because it is easier to fix many of the usability problems that arise. Card sorting is helpful in knowing user mental model of an information space. There are two types of card sorts: an open card sort and a closed card sort. In an open card sort, participants are asked to organize the cards into groups that make sense to them and then name each group. In a closed card sort, participants are asked to sort items into pre-defined categories.

1.1.3 Mobile Devices

Mobile applications development is a relatively new phenomenon that is increasing rapidly due to the ubiquity and popularity of smart phones among end-users. Mobile devices can be defined in different ways when they are looked at from different perspectives. They can be defined in terms of the services they offer or based on the level of functionality connected with the devices. According to (Sharpet, et al, 2007) they refer to the devices that are handheld and intended to be

used while on the move. Nowadays, mobile devices are being used by different people for various purposes. A mobile device refers to a pocket-sized computing device, typically having a small display screen, a small keypad with miniature buttons or a touch screen with stylus of input and wireless capability to connect to the Internet.

1.2 Research Background

While applications development for mobile devices goes back at least 10 years. There has been exponential growth in mobile applications development since the iPhone App Store opened in July 2008, from then, device makers have created outlets for other mobile devices, including Android, BlackBerry, Nokia Ovi, Windows Phone, and more. HCI emerged in the 1980s with a focus on usability of computer applications and productivity of users. The spread of computing promoted HCI research expanded interests to include areas such as social computing, ubiquitous computing, creativity, accessibility, and entertainment, (Carroll, 2009). A Usability engineer harmonizes form, content, and behavior of interactive artifacts; both software and hardware, to deliver products that are useful, usable, and desirable. Usability engineers define the structure and behaviors of interactive products and services and user interactions with those products and services IXDA (2009). The practice of usability engineering is grounded in an understanding of real users, their goals, tasks, experiences, needs, and wants, one view of SE is that it strives to develop high-quality software. Usability is an important quality attribute that is strongly related to HCI, are targeted users able to use the product? Do they need to be given a lot of training before they can start using it? Does it take too long for users to complete tasks? Do users make too many errors while doing tasks? The answers to these questions can determine the difference between the success of a product and its failure. This is where the overlap of HCI with SE becomes critical. In the early days, HCI issues were limited as software was deployed in few domains, on limited platforms, clients' requirements were clear, and the software product was often used internally by a few operators. Usability issues did arise, but because the users were internal, in a pinch they could always be trained to work around the problems. Only severest of problems were escalated as enhancements or change requests. Computing has evolved widely over the last decade with the rise of desktop computing in the 1980s, the Internet in the 1990s, and mobile telephony in the 2000s, software products reached beyond the safe group of trained users.

1.3 Statement of the Problem

Owing to the fast development in the digital technology, the operation of human-computer interface is becoming more complicated. The un-usability of systems, products and services is a tremendous problem for users and consumers all over the world, despite the efforts put in by researchers, usability practitioners and designers.

Using a mobile device is different from working with a desktop or laptop computer. While gestures, sensors, and location data may be used in game consoles and traditional computers, they play a dominant role in many mobile applications. The smaller display and different styles of user interaction also have a major impact on usability design for mobile applications, which in turn has a strong influence on applications development. Therefore, usability still needs to be the main focus of our activities. In practice, usability aspects are usually regarded very late (if at all) in software development.

Software development does not stop with delivery, nor do usability issues. Systems and products are modified and improved in a number of releases over a number of years. Most efforts currently centered on usability matters stop after the initial development process. What do we do after delivery? Furthermore, software development models, such as agile, waterfall, Spiral, Rational Unified Process (RUP) and Dynamic Systems Development Method (DSDM) are widely used in the software development industry but these models are basically not user-centered and most of them provide limited support for usability activities. Thus, it is very important to find ways of integrating usability aspects into such development models. In this research we focus on the enhancement of the Mobile-D agile process model.

1.4 Justification

The relevance of usability as a quality factor is continually increasing for software engineering organizations. Usability and user acceptance are about to become the ultimate measurement for the quality of today's, telematics applications, e-commerce web sites, mobile services and tomorrow's proactive assistance technology. Taking these circumstances into account, human-computer interaction methods for developing interactive systems are changing from a last minute add-on to a crucial part of the software engineering lifecycle.

It is well accepted both among software practitioners and in the human-computer interaction research community that structured approaches are required to build interactive systems with high usability. On the other hand specific knowledge about exactly how to most efficiently and smoothly integrate Usability engineering methods into established software development processes is still missing (Eduard, et al, 2004), while approaches such as the usability maturity model (UMM) provide means to assess an organization's capability to perform usability development processes they lack guidance on how to actually implement process improvement in HCI. It often remains unclear to users of Usability engineering methods why certain tools and methods are better suited in a certain development context than others (Metzker & Reiterer, 2002). We need strategies and tools that support engineering organizations. Little research has been done on integrating methods and tools of Usability engineering in to software engineering development process for the enhancement of interactive mobile devices and on gathering knowledge about HCI activities in a form that can capture relationships between mobile platform development contexts, applicable methods, tools and their impact on the engineering process.

1.5 Objectives

1.5.1 Broad Objective

The broad objective is to propose an Extended Mobile-D agile process model. The approach is to integrate the essential discount usability activities, methods, deliverables, and skills relevant to mobile applications development at a point into the software engineering (SE) process.

1.5.2 Specific objectives

i. To review current trends and concepts of existing discount techniques and how they can be integrated into software engineering to support the development of usable interactive mobile applications.
ii. To identify the essential discount techniques that can be modeled into software engineering practices to develop and improve the safety, utility, effectiveness and usability of mobile applications.
iii. To develop/model a tighter fit between Usability engineering and software engineering practices.

iv. To demonstrate the validity of our Extended Mobile-D agile process model and framework.

1.6 Research Questions

The following are the research questions this research seeks to address,

i. Which current concepts from discount usability techniques and Software Engineering can be harmonized to support the development of usable interactive mobile applications?
ii. What are the essential characteristics of tool-support needed to support the development of usable interactive mobile applications?
iii. How can discount usability techniques and software engineering methods be integrated to support the development of usable interactive mobile applications?
iv. Is our model efficient?

1.7 Scope of Study

HCI and SE have overlapping concerns, and have evolved side-by-side in the last three decades. The two disciplines did not interact until recently. In this research, we focus on integration of essential discount usability techniques relevant to mobile applications software engineering into the well-established Mobile-D agile process model and our target population includes the mobile applications developers. In doing so, we need to deal with two main problems. Firstly, the usability activities described in literature, even the ones that have been established for a long time, have not been integrated into the agile software engineering (SE) processes. Secondly, it is not clear which activities and methods should be integrated. Many HCI methods are defined in literature, but no particular list could be unanimously considered necessary and sufficient for integration with the agile software engineering (SE) process models. The IFIP working group on User Interface Engineering remarks that there are major gaps between HCI and SE in academics, literature, and industrial practice, and the architectures, processes, methods, and vocabulary of one community are often foreign to the other IFIP WG 2.7/13.4 on User Interface Engineering, (2012).

1.8 Thesis Structure

This thesis is broken down into Five Chapters.

Chapter 2, Reviews the current state-of-the-art in the design of usable interactive mobile applications. We dig into agile approaches for mobile applications development; present the unique challenges for the applications development and the gaps in industry practice as we consider the need for integrating discount usability engineering methods into agile process models for better applications development methods

Chapter 3, Describes the methodology used for carrying out the research justifying this thesis. We describe the Research design approach the target population, sample and sample techniques and conclude this chapter by considering data collection instruments and analysis procedures.

Chapter 4, is divide into three Sub Sections 4.1, 4.2 and 4.3. 4.1 Presents a multidisciplinary framework. The framework is proposed to be a flexible way of understanding and communicating the work of Usability engineering practitioners in different contexts. We divide the framework into phases. Each phase consists of one or more activities. Each activity is associated with one or more techniques. Each method requires specific skills and could be associated with a particular discipline to address a specific concern. Each activity results in specific deliverables. We further identify usability engineering activities that, we propose, are essential for integration with the six software engineering process steps. We organize these activities in ten phases, which we describe in terms of ten questions.

Section 4.2 introduces a proposed extension to the select agile approach and building on the framework described in section 4.1; it presents and describes the Extended Mobile-D agile process model and finaly

Section 4.3 measures effectiveness of integration of discount usability into software engineering, having established a process framework and used it as a baseline for integrating the essential discount usability techniques into the Extended Mobile-D model a research question naturally arises. How can one prove that our process model is any good? We propose a set of evaluations to measure how well our contributions are, assume that it is possible to express numerically the extent to which a process model is followed; further, assume that it is possible to express numerically the extent to which a team achieves its product goal by following a prescribed process model to the extent X a project could achieve its goals to the extent Y, if we can demonstrate that for every X2 that is greater than X1, Y2 is greater than Y1 in most cases, we can conclude that the process model in question works.

In **Chapter 5,** we review whether the research questions posed in Chapter One Section 1.6 have been properly answered and conclude this chapter by proposing future work extending the contributions made.

CHAPTER TWO

2.0 LITERATURE REVIEW

2.1 The Need for Usability Engineering

Human-Computer Interaction (HCI) discipline provides the foundations to develop usable interactive applications. “Usability Engineering" is a science that studies how to understand and systematically address the usability demand of a customer (C lee, et al, 2007). The ISO 9241-11 defines usability as "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use".

Usability engineering deals with issues such as system learnability, efficiency, memorability, applicability, errors and user satisfaction. Usability engineering is an approach to product development that is based on customer data and feedback, on direct observation and interactions with customers to provide more reliable data than self-reporting techniques.

Usability engineering begins in the conceptual phase with field studies and contextual inquiries to understand the functionality and design requirements of the product. It is an iterative design and evaluation to provide customer feedback on the usefulness and usability of a product's functionality and design throughout the development cycle. This results in products that are developed to meet the customers' needs. In our work we focus on the "discount usability engineering" methods which are; Scenarios, Heuristic evaluation, Card sorting and simplified Thinking aloud.

2.1.1 Discount usability engineering methods

2.1.1.1 Scenarios

Scenarios are appropriate whenever you need to describe a system interaction from the user’s perspective. A scenario describes a sequence of events when interacting with a system from the users’ perspective and the scenario descriptions can be created before a system is built and its impacts felt. ‘Scenarios’ are similar to ‘Use Cases’, which describe interactions at a technical level. Agile models demonstrate strength in iterative software development, where requirements may change as a system is incrementally put into use, the question of how to devise an initial

design is largely unanswered. In agile, user stories are used to capture requirements. Many of the agile user stories describe legacy features and screens, and often fell short of improvement, such user stories do not often fit into expressing usability requirements.

User stories are short narratives which describe interaction at a technical level while a scenario is a description of a person's interaction with a system where people who do not have any technical background can understand it thus integrating scenarios into agile will make it a tighter fit. Scenarios can be easily understood by anyone regardless of the level of their technical knowledge. Scenarios are especially useful when you need to remove the focus from the technology in order to consider other design possibilities. Scenarios focus in terms of tasks rather than the technology used to support them.

2.1.1.2 Heuristic Evaluation

A heuristic evaluation is an expert evaluation method that uses a set of principles to assess if an interface is user friendly. Heuristic evaluations are suitable at almost any time during a user-centred design cycle. Thus the technique can be applied to prototypes or fully implemented interfaces to retrieve valuable information regarding issues of usability. In agile programming, the customer is to test that the overall system is functioning as specified by Acceptance Tests (also known as Functional Tests). When all the acceptance tests pass for a given user story, that story is considered complete.

A story can have one or many acceptance tests, whatever it takes to ensure the functionality work. However, an acceptance test does not deal with non-functional requirement like usability. Heuristic evaluation is an approach used by the developers to improve the usability of software by applying a small collection of usability principles to the design and development of the software before testable elements are presented to users.

Usability Evaluation solves the problem of ad-hoc input. The simplicity of heuristic evaluation is beneficial as it provides some quick and relatively inexpensive feedback to designers. Usability evaluation with users should be included as part of the acceptance testing process. (Sharp, et al, 2008) also suggest heuristic evaluation can be done in each of the iterations in the agile development methods.

2.1.1.3 Card sorting

Card sorting is a method used to help design or evaluate the information architecture of a software. Card sorting will help you understand your users' expectations. The planning game of agile has two problems, one is that customer needs help to understand, verbalize, visualize and organize their requirements and second that developers have little opportunity to consider how exactly the interface will work, because the conversion of requirement to interface is implicitly assumed to take place within the estimation process.

Card sorts are a well established technique for eliciting knowledge from people by which better external quality can be obtained by involvement of actual end users. Card sorting technique with the help of end users as a part of release planning in agile process can increase the chance for successful usable software.

2.1.1.4 Thinking aloud

In a thinking aloud, you ask test participants to use the system while continuously thinking out loud-that is, simply verbalizing their thoughts as they move through the user interface. The method has a host of advantages. Most important, it serves as a window on the soul**,** letting you discover what users really think about your design. In particular, you hear their misconceptions, which usually turn into actionable redesign recommendations; when users misinterpret design elements, you need to change them. Even better, you usually learn why users guess wrong about some parts of the user interface and why they find others easy to use. User-interface design and usability are largely overlooked by the agile methods.

Thinking aloud allows you to understand how the user approaches the interface and what considerations the user keeps in mind when using the interface. This testing is preferred in design, development and testing phases of the software development where the designer can get the quick feedback about their designer work. Thinking aloud method can be applied too effectively in “Small release” Productionizing phase in agile methods, where decision has to be made if some changes occur.

2.1.2 Categories of mobile applications

There are many ways in which mobile applications can be categorized. Nevertheless, any plausible partition can lead to better results in the development process, due to a higher focus on

issues that are specific to the respective application type. Depending on the experience of the development team, different measures can be taken. For a seasoned team, identifying the application type means experiences from developing similar applications in the past can be used. Teams with less development experience can also benefit from categorization, by obtaining and implementing a specific set of guidelines and principles for the specific type of application.

In (Varshney & Vetter, 2001) the authors identify twelve classes of mobile commerce applications. Example classes include mobile financial applications (banking and micro-payments), product location and shopping (locating and ordering items), and mobile entertainment services (video-on-demand and similar services). However, these classes only apply to mobile commerce applications (mobile applications that involve transactions of goods and services) and do not help to provide guidelines to developing new applications. To this purpose, the findings in (Oinas-Kukkonen & Kurkela, 2003) prove more useful. Citing a report by Ramsay and Nielsen on WAP usability, the authors divide mobile applications into two groups:

- Highly goal-driven and
- Entertainment-focused.

The definition of each group is quite simple: highly goal-driven applications aim to provide fast responses to inquiries, while entertainment-focused applications help users pass the time. The authors move on to provide seven guiding principles for the development of highly goal-driven mobile services:

1. Mobility (provide information while on the move),
2. Usefulness,
3. Relevance (include only relevant information),
4. Ease of use,
5. Fluency of navigation (most important information should be easiest to locate),
6. User-centred (adapt to the users' way of interaction and way of thinking), and
7. Personalization (adapt to users' needs and capabilities).

A taxonomy of mobile applications from an enterprise point of view is established in (Unhelkar & Murugesan, 2010). The authors state that this organization and representation of mobile applications will make the demands placed on the applications more visible, and will help developers focus on the most important aspects of design and implementation for each project.

The lowest level in the taxonomy (organized by application richness and complexity) is represented by

- Mobile broadcast (M-broadcast) applications that are aimed at providing large-scale broadcast of information to mobile platforms.
- Higher-level applications are Mobile information (M-information) applications, which provide information required by mobile users, such as weather conditions.
- The third level of applications is Mobile transactions (M-transaction) facilitating e-transactions and customer relationship management.
- The fourth level, Mobile operation or M-operation deals with operational aspects of the business such as inventory management or supply-chain management.
- Finally, the top level of the taxonomy is represented by Mobile collaboration (M-collaboration), a class of applications that support collaboration within and outside the enterprise.

Even though the authors exclusively analyze mobile applications in an enterprise context, recommendations are provided for each type of application; these can be applied in most similar projects. In M-broadcast applications, content is broadcast to a large number of unregistered users, while in M-information users request and receive information in an individual fashion. Issues associated to this category of applications include usability and privacy, security not being of high relevance. M-transaction applications enable mobile transactions, such as placing and tracking orders and making electronic payments. This category of applications has higher requirements in terms of security, responsiveness and reliability, and requires communication between three parties: user, service provider and financial mediator (such as an online payment gateway). M-operation applications are required to provide real-time information and also integrate back-end systems and databases. The final group of applications, M-collaboration; have associated coding and data-management challenges due to the required support for the interaction between different software modules.

Six different categories of mobile applications are identified in (Kunz & Black, 1999)

- Standalone applications (games or utilities),
- personal productivity software (word processors and office applications),
- Internet applications (e-mail clients, browsers),
- vertically integrated business applications (security),

- location-aware applications (tour planners and interactive guides) and
- Ad-hoc network and groupware application (a group of users establish an ad-hoc network to exchange documents).

The authors point out some important requirements associated to the identified groups of mobile applications. For personal productivity software, synchronization between the mobile and desktop versions of the software is indicated as an important requirement. For the third category, Internet applications, the issue of client application performance and resource requirements is emphasized. The authors state that a mobile client application cannot "borrow" from non-mobile client applications, as these have completely different underlying assumptions in terms of performance requirements and availability of resources. These issues also apply to vertically integrated business applications, as the servers should remain unaware of the type of client they are communicating with (mobile or non-mobile), in order to ease the deployment of mobile applications.

The works described above serve as a basis for establishing a way to categorize mobile applications as shown in Figure 2:1, and to integrate the categorization task into Mobile-software engineering.

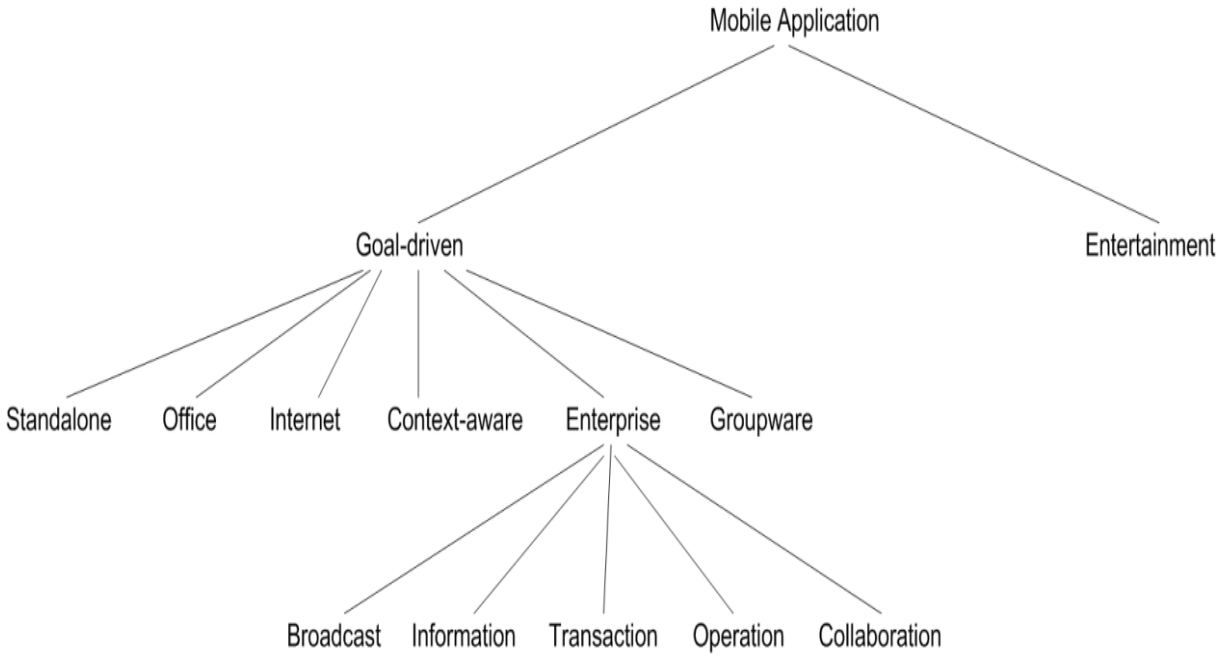

Figure 2:1 Categories of mobile applications, based on (Oinas – Kukkonen & Kurkela, 2003) (Unhelkar & Murugesan, 2010) and (Kunz & Black, 1999)

The categories are not exhaustive or exclusive and if the team identifies the category of application they are developing, they can establish project goals that respect the specific guidelines, and can shape the initial schedule according to data gathered from previous or similar projects.

2.1.3 What Makes Mobile Devices Different?

In many respects, developing mobile applications is similar to traditional software engineering for other embedded applications. Common issues include integration with device hardware, as well as traditional issues of security, performance, reliability, and storage limitations. However, mobile applications present some additional requirements that are less commonly found with traditional software applications, including:

2.1.3.1 Sensor handling

Modern mobile devices, e.g., “smart phones”, include an accelerometer that responds to device movement, a touch screen that responds to numerous gestures, along with real and/or virtual keyboards, a global positioning system, a microphone usable by applications other than voice calls, one or more cameras, and multiple networking protocols.

2.1.3.2 Native and hybrid (mobile web) applications

Mobile devices often include applications that invoke services over the telephone network or the Internet via a web browser and affect data and displays on the device but embedded devices use only software installed directly on the device.

2.1.3.3 Families of hardware and software platforms

Most embedded devices execute code that is custom-built for the properties of that device, but mobile devices may have to support applications that were written for all of the varied devices supporting the operating system, and also for different versions of the operating system.

2.1.3.4 Security

Currently embedded devices are “closed”, in the sense that there is no straight forward way to attack the embedded software and affect its operation, but mobile platforms are open, allowing the installation of new “malware” applications that can affect the overall operation of the device.

2.1.3.5 User interfaces

Using a custom-built embedded application, the developer can control all aspects of the user experience, but a mobile application must share common elements of the user interface with other applications and must adhere to externally developed user interface guidelines, many of which are implemented in the software development kits (SDKs) that are part of the platform.

2.1.4 Mobile applications development

The mobile applications market is currently undergoing rapid expansion as mobile platforms continue to improve in performance and as the users’ need for a wide variety of mobile applications increases. The latest mobile platforms allow for extensive utilization of network resources and thus offer a strong alternative to workstations and associated software. Software development for mobile platforms comes with unique features and constraints that apply to most of the lifecycle stages. The development environment and the technologies that support the software are different compared to “traditional” settings. Traditional development and quality frameworks offer comprehensive criteria for conducting general purpose software projects but none of them has been developed considering the context of

- Mobile users
- Mobile execution environments and
- Mobile application markets

Mobile software development teams must face the challenge of a dynamic environment, with frequent modifications in customer needs and expectations. Abrahamsson (2007) document that technological constraints apply to mobile platform-based devices software engineering in the form of limited physical resources and rapidly changing specifications. There is also a great variety of devices, each with particular hardware characteristics, firmware and operating

systems. The unique technological constraints to mobile platform-based devices software engineering are fully discussed below.

2.1.5 Unique development challenges for mobile devices software engineering

The creation of applications intended to execute on newer mobile devices such as smart phones and tablets involves unique requirements and challenges. Containing global positioning sensors, wireless connectivity, photo/video capabilities, built-in web browsers, voice recognition, among other sensors, mobile devices have enabled the development of mobile applications that can provide rich, highly-localized, context-aware content to users in handheld devices equipped with similar computational power as a standard Personal Computer (PC) (Oulasvirta, et al, 2011). Yet, these same novel features/sensors found in mobile devices present new challenges and requirements to application developers that are not found in traditional software applications, (Wassermann, 2010).

Traditional software engineering approaches may not directly apply in a mobile device context. First, mobile device user interfaces (UI) provide a new paradigm for new human-computer interaction sequences (e.g., multi-touch interfaces, QR code scanning, image recognition, augmented reality, etc.) that have not been previously explored in research and of which no established UI guidelines exist (Oulasvirta, et al, 2011).

Second, the divergent mobile platforms (e.g., iOS, Android, Windows 7, etc.), differing hardware makers for platforms (e.g., Android versions found on HTC, Google, Samsung) and mobile phone and tablet platforms (e.g., Apple's iPhone and iPad) have necessitated developers to make a series of the same application tailored for each type of device (Wassermann, 2010). Third, the novelty of a truly mobile computing platform provides both unique opportunities and challenges below we outline the fundamental, unique challenges to the state-of-practice in mobile applications software engineering:

2.1.5.1 Form factors

The first and most obvious unique aspect of mobile applications is that the form factor for display and user interaction is significantly different from prior forms of software. Smart phones usually provide only a four-inch area in which to display the application content and offer lower screen resolution pixel density compared to personal computer (PC) displays, which are trending

toward greater display sizes and number of screen pixels. Even tablet devices have generally lower display sizes than PCs, especially when compared to the large flat-screen displays in use for newer desktop PCs.

A smaller form factor means that the amount of data displayed to the end user, and layout of that data, needs to be different for these applications than for applications expected to run on PC devices. Significantly less data can be displayed at one time and therefore it must be exactly the "right" data, most relevant to what the user needs at that point in the application.

2.1.5.2 Usability and user interaction design

Several factors motivate the need for more attention to usability and user interaction design for mobile applications. One is the difference in form factors and user input methods. It is much more difficult and time consuming to plan how to display only the data that is precisely necessary than it is to simply display all possible data and let the end users visually sift through it for what they want. The mobile application designer has to consider the screens real estate.

2.1.5.3 Creating Universal User Interfaces

There has been some preliminary research in creating a universal user interface for mobile devices (Oulasvirta, et al, 2011), (Balagtas, et al, 2009). Each mobile platform has a unique guide to address developer user interface requirements. The user interface guidelines have several overlapping themes.

A significant consideration for mobile UI development relates to screen size and resolution. For example, Apple devices are limited to two sizes based on the size of the iPhone and the iPad whereas Windows 7, Android, and Blackberry provide screens of varying sizes and screen resolutions.

As a result, UI design is difficult and mobile platform-based devices application developers must anticipate the targeted device(s). Seffah et al. list a set of obstacles in integrating usability in software engineering (SE) (Seffah, et al, 2005).

- One obstacle is the deep-rooted myth that usability is not a central topic of SE. Usability activities are considered easily dispensable by a software project manager when the project is short on budget or time.

- Another obstacle is the ambiguity associated with usability, the different meanings it presents to different people. Claims about usability methods are hard to prove using classical scientific techniques because of the difficulty in collecting statistically valid empirical evidence.

2.1.5.4 User input technology

Another obvious physical difference for mobile applications is that the mechanisms for user input are different. Mobile devices have pioneered the use of non-keyboard "gestures" as an effective and popular method of user input. Touch, swipe, and pinch gestures must be planned for and be supported in a satisfying mobile application user experience.

These tactile end user input mechanisms have proven to be so popular that they are now being retrofitted into traditional desktop PC systems such as the Apple "Lion" OS X release and Windows 8 "Metro" OS. In addition to tactile user input, mobile devices are a natural target for voice-based user input. Besides input directly from the end user, mobile devices have the capability to receive input from other sources, such as geo-location input from the GPS component of the device and image information from the camera typically built into the device. These unique forms of input must be considered during mobile applications design and development. They offer new and valuable mechanisms to make mobile applications more powerful and useful than applications with a more limited array of input possibilities.

2.1.5.5 Enabling Software Reuse across Mobile Platforms

Mobile applications currently span several different operating system platforms (e.g., iOS, Android, Windows 7, etc.), different hardware makers (Apple, HTC, Samsung, Google, etc.), delivery methods (i.e., native application, mobile web application) and computing platforms (i.e., Smartphone, tablet). Each of these options must be considered during mobile applications development as they have a direct influence on the software requirements. Companies currently need to make a business decision to target a single mobile device platform with rich features, multiple platforms through a mobile website with less rich features or spend the resources necessary to broadly target the gamut of mobile devices with rich, native applications.

2.1.5.6 Choice of implementation technology

There is a spectrum of implementation choices for mobile applications in the market. There is no one perfect answer for the choice of implementation for a mobile application, and all of the choices across the spectrum have their advantages and disadvantages. Therefore, the challenge for mobile devices software development teams is to understand the trade-offs between the technologies and make a choice based on the specific application requirements.

The choice of implementation technology for a mobile project will have an impact on other decisions related to the application's development. It may limit the choices for development tools. The implementation choice will likely have an impact on the teams' roles and structure. It may have an impact on how the application is tested and verified, and how it is distributed and delivered to the end user. So, the choice of implementation approach for a mobile application is a crucial, early-stage decision to be made very carefully.

2.1.5.7 Designing Context-Aware Mobile Applications

Mobile devices represent a dramatic departure from traditional computing platforms as they no longer represent a "static notion of context, where changes are absent, small or predictable" (Roman, et al, 2000). Rather, mobile devices are highly personalized and must continuously monitor its environment, thereby making mobile applications inherently context aware (collectively time-aware, location-aware, device-aware, etc.) (Hofer, et al, 2003), (Dey, et al, 2008).

Mobile applications are now contextualizing proximity, location, weather, time, etc. To deliver hyper-specialized, dynamic, rich content to users through context-aware applications. Previously, web applications would often provide contextualized content based on time, detected location and language.

However, the extent of context-awareness currently possible in mobile applications is beyond what software engineering approaches have encountered outside of agent-oriented software engineering. The consideration of context-awareness as a first-class feature in mobile applications software engineering is needed so that the requisite attention is paid by developers when analyzing these requirements resulting in better designed context-aware applications.

2.1.5.8 Behavioral Consistency versus Specific HCI Guidelines

Ideally, a given mobile application should provide the same functionality and behavior regardless of the target platform it is running on. However, due to the internal differences in various mobile devices and operating systems, “a generic design for all platforms does not exist”. “An Android design cannot work all the way for the iPhone.” This is mainly due to the fact that HCI guidelines are quite different across platforms, since no standards exist for the mobile world, as they do for the Web for instance. On the other hand, developers would like their applications to behave similarly across platforms, e.g., user interaction with a certain feature on Blackberry should be the same as on iPhone and Android thus, creating a reusable basic design that will translate easily to all platforms while preserving the behavioral consistency is challenging.

2.1.5.9 Balancing Agility and Uncertainty in Requirements

While most mobile application developers utilize an agile approach or a nearly ad hoc approach, the growing demand for context-aware applications, competition amongst mobile applications and low tolerance by users for unstable and/or unresponsive mobile applications (even if free) necessitates a more semi-formal approach. This should be integrated into agile process engineering to specify and analyze mobile applications requirements.

2.1.5.10 Mobile applications build and delivery

Because of the strong business motivations to deliver mobile applications into the market quickly, mobile development projects typically have extremely aggressive time lines. Inception-to-delivery periods of a few months are common. The pressure to deliver mobile applications quickly results in the adoption of agile development methods for most mobile projects.

An important element in agile development practices is continuous integration and builds. Application changes delivered by developers need to be processed immediately for all of the mobile operating systems on which the application is required to execute. If the mobile application is a hybrid or native implementation, several different builds of the application need to be triggered each time a change set for the application is delivered by a developer. The build setup and configuration for each supported mobile environment will be different from the others,

and it is most likely that a small “farm” of build servers will need to be provisioned and available to handle these builds of the mobile applications for multiple operating systems.

2.1.5.11 Testing of mobile applications

Another area where mobile applications development poses a huge challenge is testing. Testing for mobile applications represents a quantum leap in complexity and cost over more traditional applications. Unlike traditional PC and web applications, the range of potentially supported mobile devices and release levels is staggering. It is quite common to see test matrices for mobile projects that contain hundreds, and even thousands, of permutations of device, mobile OS level, network carrier, locale, and device orientation combinations.

2.1.6 Usability Engineering in Software Engineering identifying the gaps in Industry Practices

In a survey of 63 HCI and 33 software engineers by (Jerome &Kazman (2005)) to analyze the gaps between SE and usability engineering practices; they found that the state of practice is not very encouraging. They report that there is substantial lack of mutual understanding among software engineers and HCI practitioners and the two disciplines hardly follow each other. They also do not collaborate much in projects.

68% software engineers report that they made key software design decisions that affect the user interface without consulting HCI practitioners. Even greater proportion of HCI practitioners (91%) believe that software engineers were making key design decisions without consulting any HCI practitioners. When collaboration does occur, it usually happens too late. Only 1 out of 21 software engineers and 2 out of 60 HCI practitioners reported that they collaborated during the specifications phase below we explore the challenges

2.1.6.1 Usability engineering inputs are not taken during requirements specifications

HCI inputs are needed early in the process before requirements are finalized. Use cases in requirements documents routinely over-specified the HCI design, including details such as the sequence, the contents of dialog boxes in the applications, navigating and browsing for mobile devices that generally have small screens etc. This over-specification happened possibly because there is a physical and cultural distance between the developers and users, the development

teams are less familiar with the context of users, and the requirements specifiers want to have a control on the user interface.

2.1.6.2 Porting projects get minimal HCI inputs

Every software project represents an opportunity to improve the user experience. Conversely, every project also represents a risk of degrading the user experience. This applies even to porting and migration projects. Less importance is normally given to requirements gathering in general and usability requirements.

It is assumed that most requirements are well-understood and had to be "copied over" from earlier version. However, projects often involve a change of delivery platform, a change of context, or a change of users and coping over can have a big impact on HCI design and the corresponding requirements.

2.1.6.3 Client representatives take design decisions

Client representative routinely drives many HCI design and usability considerations. Such a person may have never been a user himself or may have moved out of that role a long time ago. His / her sign-off may not imply that the product is usable. This can be revealed only by usability evaluations with real users.

2.1.6.4 Usability engineering skills do not have process support

Software engineering (SE) projects have some involvement of HCI practitioners, though they still ended with unresolved usability issues that they knew could be solved (Jerome & Kazman, 2005). A multi-disciplinary team needs to work together. The team needs to be armed with appropriate user inputs and needs a common set of work products and a common process to approach the product development holistically and add value. Role of each discipline needs to be mutually understood and respected, first within the team and then across the organizations.

2.1.6.5 Too little and too late is not good enough

In projects, HCI practitioners are pulled in towards the end when too many obvious usability problems surfaced (Jerome & Kazman, 2005). In these situations, HCI practitioners work under severe constraints. They have no time to understand the scope of the project and no budget to do

usability activities they would have done earlier. Even if some HCI activities were done, most of the recommendations they come up with to improve the User Interface seemed too impractical to implement in the given situation. Few cosmetic changes would be made, mainly to satisfy the client representative, and the project would be pushed through.

2.2 Agile Process Models

Agile process models have come to represent the iterative nature of software development as shown in figure 2:2 below. Agile process methods are incremental (multiple releases), cooperative (a strong cooperation between developer and client), straightforward (easy to understand and modify) and adaptive (allowing for frequent changes). The ideas behind these methods originate from the principles of Lean Manufacturing (in the 1940s) and Agile Manufacturing (1990s), which emphasized the adaptability of enterprises to a dynamic environment (Salo, 2006).

Agile methodologies have been developed in the last decade as a way to address constantly changing requirements and other key problems including the increasing cost and complexity of software development, communication breakdowns among stakeholders, and missed schedules and budget overruns.

Agile methodologies purport to address these software development problems by focusing heavily on quick delivery of working software, incremental releases, team communication, collaboration and the ability to respond to change. Agile methods in one form or another have become increasingly popular in practice. A significant majority of practitioners who have been on agile teams indicate that they produce higher quality software, greater productivity and higher stakeholder satisfaction, (Ambler, 2008).

Several process models have emerged and Pressman summarizes seven agile process models: Extreme Programming, Adaptive Software Development, Dynamic Systems Development Method, Scrum, Crystal, Feature Driven Development, and Agile Modeling Pressman (2005 pp. 103-124). These process models may vary in their details, but they have several common elements best captured by the agile manifesto (2001).

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation

- Responding to change over following a plan

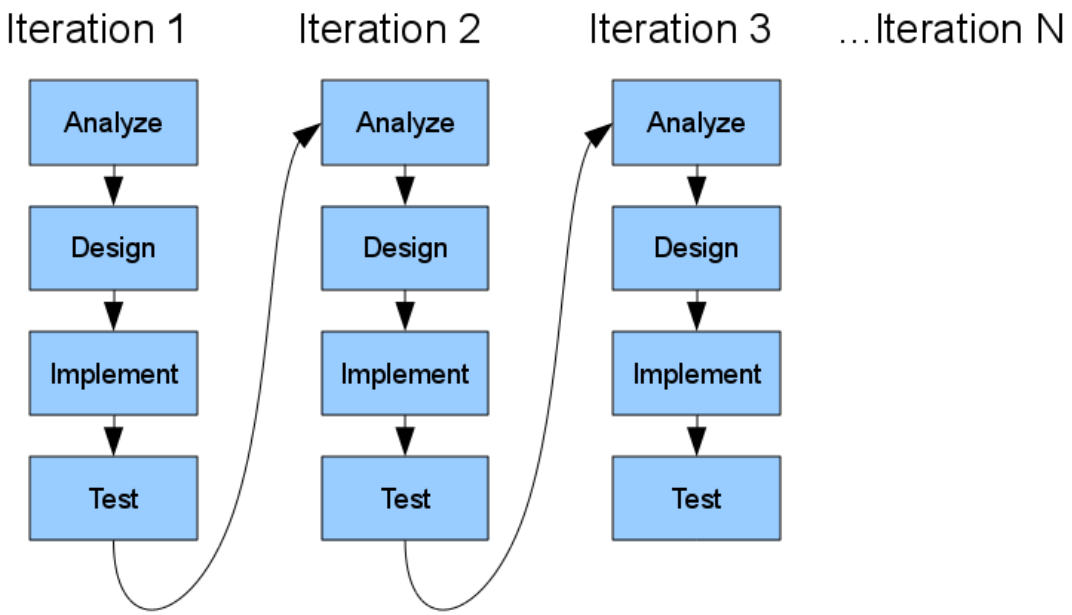

Figure 2:2 Agile Process

The last point is particularly important. In agile processes, it is typical to solve a small part of the problem to begin with and to grow the solution in iterations. Agile processes believe that changes in software requirements will necessarily happen. Agile processes are designed to accommodate changes even late in the process to harness change for the customer's competitive advantage Agile Manifesto (2001).

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

The fundamental business forces in today's economy are so dynamic that every six months, new requirements are likely to emerge. In agile processes, the main measure of progress is working software agile methods deliver working software in small pieces frequently and sometimes as frequently as once a week. This length of time forms a heartbeat for the project and helps maintain pace. Agile methods also insist that development needs to happen smoothly, without the developers working overtime.

Each iteration of an agile process follows a mini-waterfall within itself. Sufficient requirements are expressed, analyzed, the software architecture is re-factored if necessary, the code is written or re-written, tested and released. If some requirements could not be completed in the current iteration, they are carried over to the next iteration. Agile methods do not plan a timeline for the whole project. Because new versions of the software are constantly being released, it makes it easier for everyone (including the customer) to see momentum in the project. This makes it easier to estimate the time needed to achieve the overall vision of the project and to make course corrections.

While testing is important in all software process models, agile methods emphasize on testing. Agile methods suggest not only testing the current version of the product, but setting up of automated testing procedures so that testing is frequent and when changes happen, during iterations the automated regression testing detects the breaks soon. Automated regression testing is particularly important because it saves on time compared to manual testing. Agile methods depend a lot on teamwork and internal communication. It is believed that best architectures, requirements, and designs emerge from self organizing teams. Developers work alongside customers during the development. There is usually little documentation, but there is a lot of emphasis on face-to-face communication between team members. Pair-programming (programming done by two developers together) and daily stand-up meetings (that last no more than 15 minutes) help in maintaining communication going among team members.

HCI processes share several qualities with agile processes. HCI design is intrinsically an iterative process consisting of analysis, design, and usability evaluation. The problems found during the evaluation are fixed in the next iteration, such iterations continue until no problems are found and user experience goals are met. Given this preference for iterations, agile methods seem a good fit for integrating usability engineering activities within the agile processes. The emphasis

on people and deliverable products rather than documentation and planning are also common qualities, just like agile programmers HCI designers are more of doers.

The informality of the agile methods gels well with the informal culture of design. Designers are more at ease in face-to-face communication and visual presentation of ideas than with wading through long documents. Most critiques agree that there is potential to integrate usability engineering activities with agile development. Nielsen acknowledges that agile methods hold promise for addressing the many ways in which traditional development methodologies erected systematic barriers to good usability practice (Nielsen, 2008). However, despite the similarities, several HCI issues still emerge with agile methods.

2.2.1 HCI Issues with Agile Processes

Design in the HCI world involves working with the user to understand the problem and come up with a user interface – typically on paper - of the entire system before turning it over, in Big Design Upfront (BDUF) manner, to the rest of the development team to build. Following our surveys the following were found to be a challenge in the current agile development paradigm.

2.2.1.1 Software Engineers Are Asked to Design

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

Blomkvist comments that though agile processes value people, skills, and teamwork in other areas, they do not regard that usability and interaction design skills as important, (Blomkvist, 2005). Nielsen identifies threats of agile methods, (Nielsen, 2008). The biggest threat, according to Nielsen, is that agile methodologies are developed by programmers to address the implementation side of software development, overlooking HCI design. While Nielsen is not against HCI design being performed by the same people who do the coding, he feels it must be recognized as a separate activity rather than leaving it to happen as a “side effect of coding”.

Constantine concludes that agile methods seem to be at their best in applications that are not GUI intensive, (Constantine, 2002).

2.2.1.2 Users Are Asked to Design

To help design a new system, agile methods put representative customers or users in the team. This may give a feeling to the development team that the voice of users is being heard, this may not be true critics. Bayer et al. argue that there is no such thing as representative users. At best, they are a sub-set of users and often, they only represent themselves (Beyer, et al, 2004). Further, even real users are unable to articulate what they do and how, particularly when they are not in the context of that work, and certainly if they have not been doing the work for a while. Finally, users are not able to make design decisions for a new system. Users may not have the appropriate skills required to create visions of future systems.

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

2.2.1.3 Change is Managed Well, But Anticipated Poorly

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

- Changes that arise because a new user need or user problem is discovered after requirements are frozen.
- Changes that arise because someone thinks of a new idea after the requirements were frozen.

- Changes that arise because something that was thought to be technically feasible turns out not to be so and a workaround is required.
- Changes that arise because late usability evaluations of early releases throw up unanticipated usability problems that were not captured on early prototypes and
- Finally, changes that could not have been anticipated.

Agile methods seem to give a license to do a poor job at anticipating and containing change. Proponents of agile methods seem to do little introspection about the reasons for intra-lifecycle changes, which are the most common type of changes in projects. HCI activities can help in anticipating many of the intra-lifecycle changes that arise out of human needs and business processes.

2.2.1.4 Agile User Stories Are Not Interaction Design Scenarios

Agile teams use user stories to define, manage, and test features of a product. It is tempting to think of these as parallel to scenarios in interaction design and to think of stories as a direct link between HCI and agile methods. However, a closer look at tells a different story. Agile user stories are written by customers, focus on the user interface of one feature, and are supposed to be about three sentences long, (Wells, 2009). The length of the story is determined by the time it takes to implement it in code.

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

2.2.1.5 Short Iterations

An important HCI issue is that breaking down product development into small parts and constant change can potentially undermine the totality of the user experience. While some HCI researchers have no issues with this, a few have critiqued this of agile methods, (Constantine, 2002), (Nielsen, 2008). Piecemeal design could lead to lack of cohesiveness and allow

inconsistencies to creep in. Maintaining a comprehensible and consistent user interface as new features are added becomes increasingly difficult. Short iterations cause further problems as the usability team tries to maintain the project.

2.2.2 Agile development for mobile applications

Agile methods represent a relatively new approach to mobile software development. The use of agile methods in mobile software development has received both supporting and opposing arguments. The main argument against mobile agile methods is the asserted lack of scientific validation for associated activities and practices, as well as the difficulty of integrating plan-based practices with mobile agile ones. There is also some amount of uncertainty in distinguishing mobile agile methods from ad-hoc programming. However, as stated in (Salo, 2006) agile methods do provide an organized development approach. When trying to compare mobile applications characteristics to those of an agile method, (Dyba & Dingsoyr, 2009) noted that difficulty comes partly from the fact that boundaries of agile methodologies are not clearly established. Findings of their research indicate that the introduction of agile methods to software projects yields benefits, especially if agile practices do not completely replace traditional ones, but work in conjunction with them.

In (Abrahamsson, 2005) the author performs a direct comparison between agile method characteristics and mobile applications features, focusing on environment volatility, amount of documentation produced, amount of planning involved, size of the development team, scale of the application in-development, customer identification, and object orientation. Except customer identification, all other agile characteristics render the methods suitable for mobile applications development. The customer may be identified as the software distributor. However, especially in the case of mobile applications, the customer identification problem is much more complex.

2.2.2.1 Is agile – a natural fit for mobile application development?

The mobile telecommunications industry comprises a highly competitive, dynamic and uncertain environment. Mobile applications should be developed quickly while keeping a low price in a competitive market of millions of potential users and products.

The agile approach is seen as a natural fit for mobile applications development and studies carried out on the application of the agile development approach to mobile applications

development indicates the need for software development processes tailored to suite the mobile applications requirements, (Holler, 2011). It has been recommended that agile practices are the best choice because they assure different phases of software development life cycle and work to solve the mobile applications development issues more efficiently (Abrahamsson & Wartso, 2003). It is believed that agile innovations may offer a variety of solutions for mobile applications and assist service developers in need of high quality development processes (Wasserman, 2010).

(Abrahamsson, 2005) has demonstrated the traits and reasons why agile technologies best fits in mobile devices software development. The various issues include, high environment volatility, small development teams, identifiable customer, object oriented development environment, non-safety critical software, application level software, small systems and short development cycles.

(Kannan, 2011) has also highlighted the suitability of agile software development in mobile applications development linked to small teams, short deadlines, putting importance on usability, fast delivery and less complexity. The authors have suggested seven methods in which Agile development practices enhance the development of mobile applications that includes:

1. Experimentation and adaption nature of mobile applications;
2. Reliability that leads to continued use of applications;
3. Extension of Agile sprints into mobile application model,
4. Responsiveness to technology changes;
5. Rapidly accommodating customer feedback;
6. A more thoughtful user experience; and
7. Phased roll out of feature sets.

(Holler, 2011) suggested that agile software development offers tremendous opportunities and value, for mobile applications development teams working into introducing a lightweight development process or scale back bureaucratic processes.

The author has emphasized the progress in mobile computer technology and the rapid escalation of wireless networks in quality and quantity that has brought in new applications and concerns in this dynamic environment. He has also underlined the promptness with which the industry needs to adapt and change itself from conventional systems development techniques fulfilling the special needs of this field. Agile methods allow adapting processes and practices to the unsteady needs of the mobile domain while providing flexibility.

2.2.3 Review of mobile applications development processes - using an agile approach

The following Agile methodologies in Table 2:1 have been proposed by various scientists and they use combination of agile and non-agile techniques for the development of mobile applications.

Table 2: 1 Mobile platform based agile methodologies

Mobile Process	**Mobile Development Process Description**	**Techniques**
Mobile D	An Agile Approach for Mobile Applications Development	XP, Crystal, RUP
RaPiD 7	Rapid Production of Documentation - 7 steps	AM
Hybrid Methodology Design	An Agile Methodology for Mobile Software Development - A Hybrid Engineering Method Approach	ASD, NPD
MASAM	Development Process of Mobile Applications SW Based on Agile Methodology	XP, RUP, SPEM
SleSS	A Scrum and Lean Six Sigma Integration Approach for the Development of Software Customization for Mobile Phones	Scrum, Lean Six Sigma

2.2.3.1 Mobile-D

One of the pioneering studies in agile approach is by (Abrahamsson, et al, 2004), where it was assessed that agile development solutions provide a good fit for mobile applications development environment and proposed a new approach called Mobile D.

Mobile-D comprises five phases: Explore, Initialize, Productionize, Stabilize, and System Test & Fix.

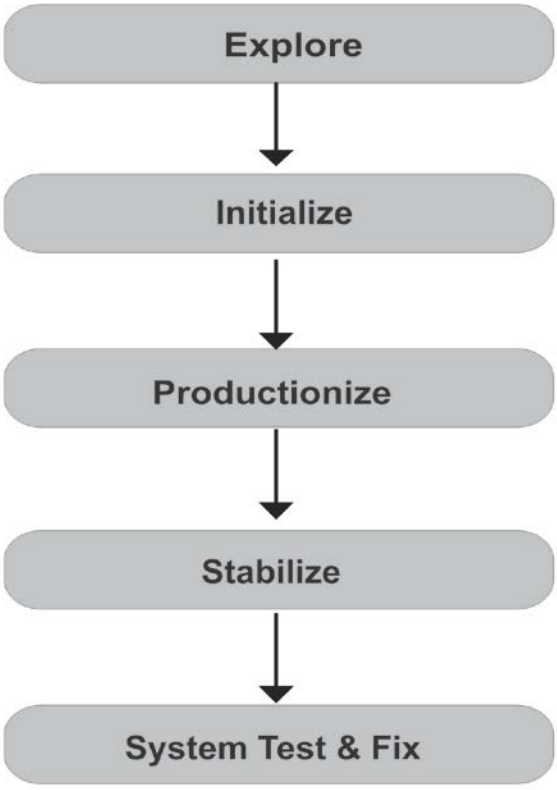

Figure 2:3 Phases of Mobile-D software development process

Their research provides an overview on to the mobile applications software development process. A diagrammatic representation of mobile D with it five phases is shown in Figure 2:3.

I. Mobile-D overview

Mobile-D approach is based on Rational Unified Process RUP (life-cycle coverage), EXtreme Programming XP (development practices) and Crystal methodologies (scalability). According to (Abrahamsson, et al, 2004), it is recommended to use Mobile-D by a small co-located team of at most ten co-located developers, working in a short development cycle towards a product delivery within 8 to 10 weeks of calendar time.

There are nine main elements involved in the different practices throughout the development cycle:

1. Phasing and Placing
2. Architecture Line
3. Mobile Test-Driven Development
4. Continuous Integration
5. Pair Programming
6. Metrics
7. Agile Software Process Improvement
8. Off-Site Customer
9. User-Centred Focus

The Architecture Line in the methodology is a new addition to the already established agile practices. An architecture line is used to capture an organization's knowledge of architectural solutions, from both internal and external sources, and to use these solutions when needed it aims at producing an application framework, which guides the development of future mobile applications.

The phases: Explore, Initialize, Productionize, Stabilize, and System Test & Fix. Each have a number of associated stages, tasks and practices as shown in Figure 2:4.

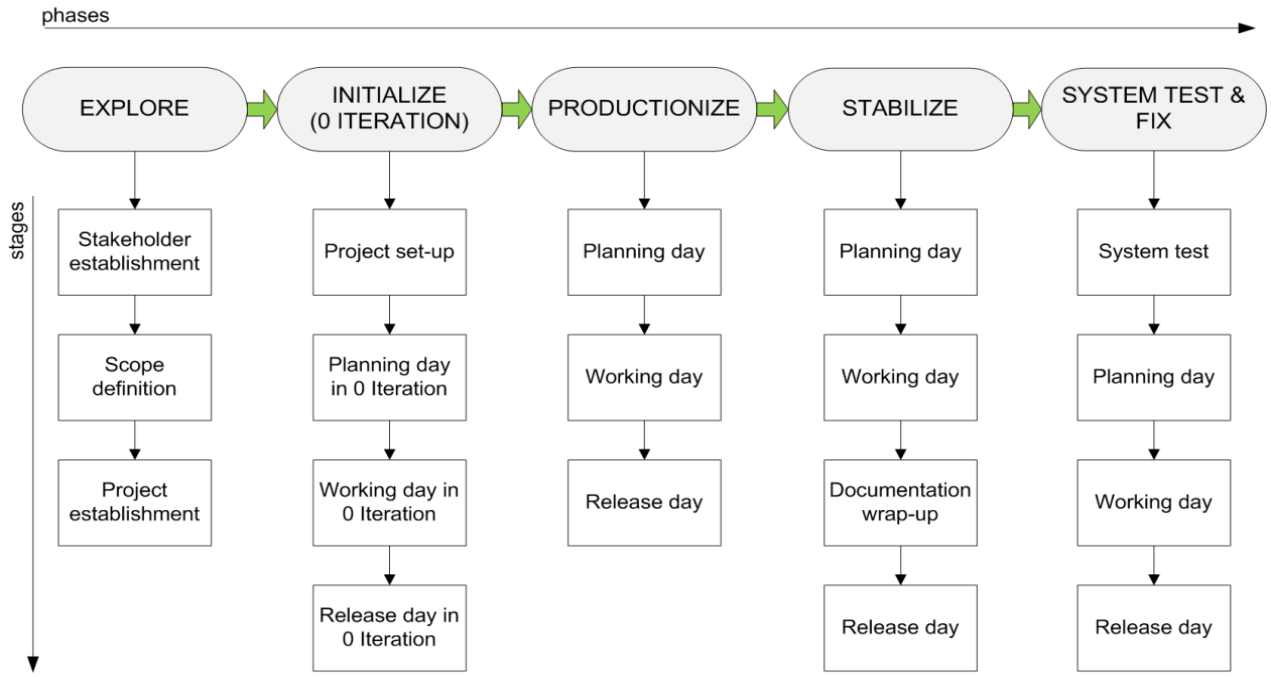

Figure 2:4 Mobile-D phases and stages. Adapted from (VTT Electronics, 2006)

a) Explore

Explore means to setup initial characteristics version of the project requirements and establishing the project plan. In Explore, the development team must generate a plan and establish project characteristics. This is done in three stages: stakeholder establishment, scope definition and project establishment. Tasks associated to this phase include customer establishment (those customers that take active part in the development process), initial project planning and requirements collection, and process establishment. The main purpose of explore phase is to highlight the scopes and requirements within the project.

b) Initialize

In Initialize, the development team and all active stakeholders understand the product in development and prepare the key resources necessary for production activities, such as physical, technological, and communications resources. When the initial requirements and plans of the project are well-organized and established, then, the Initialize phase begins. This phase is divided into three stages: project set-up, initial planning and trial day. Identifying the resources within the project technically and physically is one of the key points of this phase providing the communication channel between the developer and stakeholders is another important point.

c) Productionize

The Productionize phase mainly means the implementation of functionalities that are collected within the Explore and Initialize phases of the project. This phase is divided three stages

- Planning days,
- Working days, and
- Release days.

Planning days are aimed at enhancing the development process, analyze the gathered requirements and prioritize them to identify the core functionalities within the project (prioritizing and analyzing requirements), planning the iteration contents for implementation of the application development process, and creating acceptance tests that will be run later in release days.

In working days, the Test-Driven Development (TDD) practice is used to implement functionalities, according to the pre-established plan for the current iteration. Using TDD along

with Continuous Integration, developers create unit tests, write code that passes the tests, and integrate new code with the existing version of the product, addressing any errors that may arise in the integration process.

Finally, in release days a working version of the system is produced and validated through acceptance testing.

d) Stabilize

It means to collect and combine iterations together to finalize the product (product finalization). To stabilize the application, one of the vital stages is to integrate all the parts and put them together as a system.

e) System Test & Fix

System Test & Fix is the final phase of Mobile-D agile methodology which is based on the application testing frequently and fixing errors while completing the documentation of the application.

II. Mobile-D with added Evolve phase.

The new Evolve phase deals with continuously integrating end-user feedback on the delivered product into future releases. Feedback can come from multiple sources, such as consumer and peer reviews, or data generated by the application itself (usage statistics and crash reports). The first task, Data analysis, as in figure 2:5 requires the team to obtain and analyze feedback data. By analyzing usage statistics, conclusions can be drawn on whether a particular component of the software is used enough to justify further maintenance and updates, while error and crash reports trigger a sequence of stages, similar to those in the System Test & Fix phase. When a defect is reported, the team locates and documents it. Then, a Fix iteration comprising a Planning day, Working day and Release day is performed. In the Planning day, the developers attempt to reproduce reported defects, in order to fix them and create a new release of the product.

Mobile-D is organized into a framework that conjoins the main processes (plan, design, implement, test and release) with the support processes (project management, software configuration management, software process improvement). Mobile-D has already been applied in development projects, and some advantages have been observed, such as increased progress visibility, earlier discovery and repair of technical issues, low defect density in the final product,

and a constant progress in development (Abrahamsson, et al, 2004). Other applications of the method are presented in (Pikkarainen, et al, 2005) and (Hulkko & Abrahamsson, 2005).

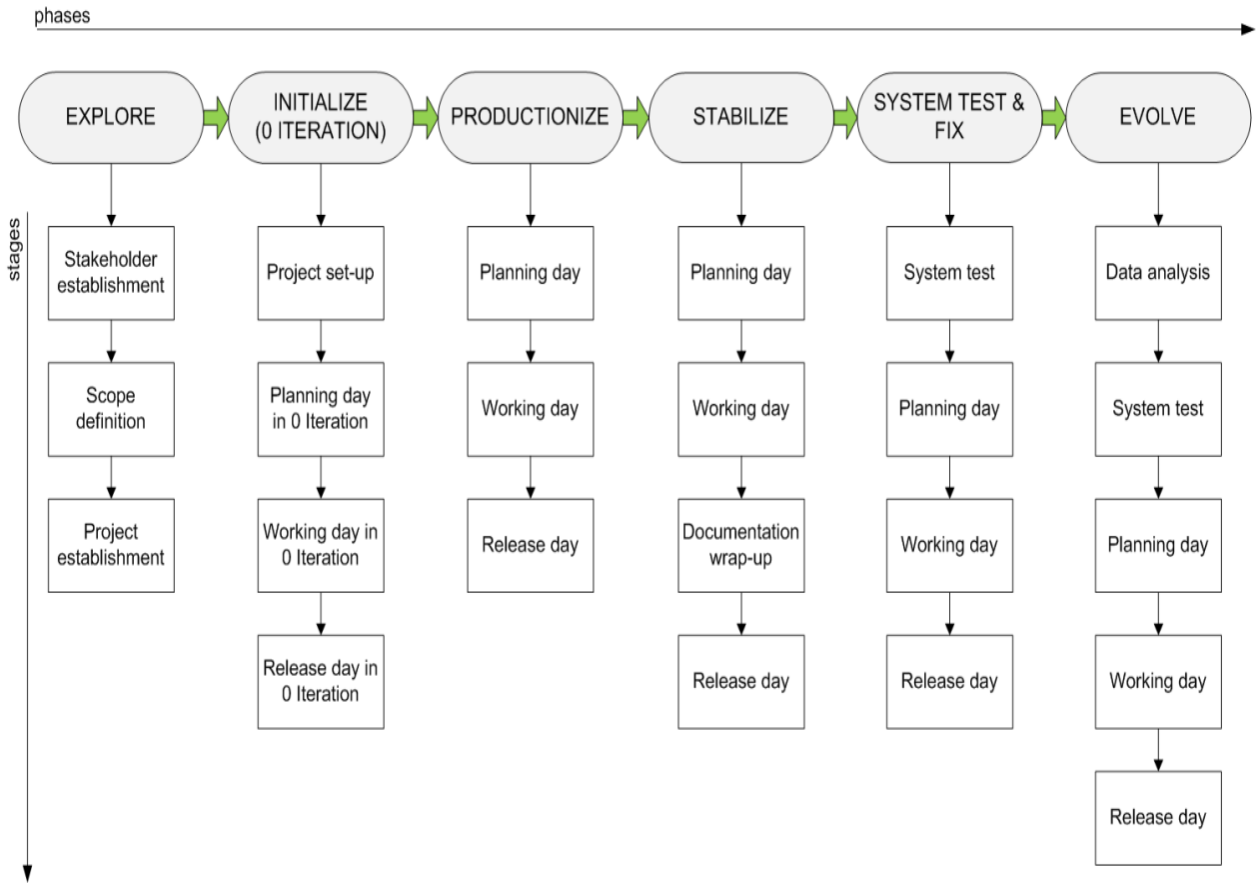

Figure 2:5 Mobile-D with added Evolve phase. Adapted from (VTT Electronics, 2006)

The approach has also been successfully assessed against the CMMI level 2 certification. Although this methodology being a pioneering study in the field seems very promising and plays an important role in theory, it is important to mention that this approach is cursory and not completely defined in order to be literally used in practice. Also, further improvements on it, have been suggested by other authors and the model could further be improved using hybrid agile techniques and user centered design approaches.

2.2.3.2. RaPiD7

Working software in software development has always been the focus over comprehensive documentation; however the required documents should be identified and be documented too. Dooms et al. has proposed a method called ‘RaPiD7’ (rapid production of documentation with 7

steps) that improves the documentation work without scarifying the quantity and quality of documentation (Beck, 1999). RaPiD7 describes how human interaction is planned in software projects and how documents are to be created in facilitated workshops. (Dooms & Roope, 2005) state that RaPiD7 provides a three-layer structure: Project, Case and Workshop layers as shown in Table 2:2.

Table 2: 2 Three layer structure of RaPiD7

Layers	**Description**
Project Layer	Describes how human interaction and joint decision-making is planned for software projects.
Case Layer	Describes how the selected cases such as documents are to be created in consecutive workshops.
Workshop Layer	Describes how the actual work is carried out in form of facilitated workshop, using seven steps of method.

The seven steps of RaPiD7 Workshop Layer are

- Preparation phase
- Kick off phase
- Idea gathering phase
- Analyzing idea phase
- Detailed design phase
- Closing phase

RaPiD7 supports all software development projects, whether related or unrelated to mobile applications development, this method was tested at Philips Digital Systems Laboratory and it was developed within Nokia in the 2002-2003 timeframe.

The motivation is to make documents that match reality and create them with as little effort as possible. RaPiD7 approach embraces two agile practices: Team work & Do the Simplest Thing That Will Work. RaPiD7 improves the traditional approach for specification work by offering a

way to plan the human interaction in the early phases of software projects and by providing means to make decisions and to document.

2.2.3.3 Hybrid Methodology Design

(Rahimian & Ramsin, 2008) present a different approach. They propose a hybrid Agile and risk-based methodology that generates a method suitable for mobile applications development designed from Methodology Engineering techniques.

Table 2: 3 Phases of Hybrid engineering methodology

Hybrid Engineering Methodology Phases	
Idea Generation	
Project Initiation	Preliminary Analysis
	Business Analysis
Analysis	Detailed Analysis
	Creation of Functional Prototype
Design	Architectural Design
	Detailed Design
Implementation (Development Engine)	Adaptive Cycle Planning
	Concurrent Component Engineering
	Updates to Component Library
Test	Quality Review
	Market Testing
Commercialization	

It is concerned with creating methodologies suitable for different development scenarios, motivated by the belief that no single process fits all situations. Hybrid Methodology Design is

built on a combination between agile methodologies, Adaptive Software Development (ASD) and New Product Development (NPD).

The ideal mobile software development characteristics that the hybrid engineering methodology is based on are: agility, market consciousness, software product line support, architecture-based development, support for reusability, inclusion of review and learning sessions, early specification of physical architecture as detailed in Table 2:3.

The Hybrid Methodology Design process has been developed as a top-down, iterative-incremental process comprised of the following tasks:

- Prioritization of requirements,
- Selection of the design approaches to be used in the current iteration,
- Application of the selected design approaches,
- Revision,
- Refinement and restructuring of the methodology built so far,
- Defining the abstraction level for the next iteration, and
- Finally the revision and refinement of the requirements, prioritizing them for the next iteration.

The proposed mobile development methodology was created in four iterations, starting from a generic software development lifecycle. In the first iteration, the methodology was detailed by adding practices commonly found in agile methods. Taking into account market considerations, the second iteration included activities from New Product Development, a process concerned with introducing a new product or service to the market. In the third iteration, Adaptive Software Development (ASD) ideas were integrated into the methodology, while in the fourth & final iteration, prototyping was added to mitigate likely technology-related risks.

Though HME is a mobile application development focused it is a high-level abstraction method and does not provide its phases with the details needed to apply it to customized development for mobile platform-based devices (phones). The published material on Hybrid Engineering Methodology does not include any case study or shows that the methodology has been empirically tested on developing an actual mobile software product.

2.2.3.4 MASAM

(Jeong, et al, 2008) proposed the Mobile Application Software Agile Methodology (MASAM) that provides the process for developing the mobile applications swiftly using an agile approach.

Table 2: 4 Process assets of MASAM

Kind	**Description**	**Name**
Role	It defines a set of related skills, competencies and responsibilities of an individual(s).	Planner, Manager, UI designer, Developer, Development team, Initial development team, Tester, User
Task	It is an assignable unit of work assigned to a specific role. The granularity of a task is generally a few hours to a few days and Usually affects one or only a small number of work products.	Product Summary, Initial Planning, User Definition, Initial Analysis, Select Resource, Select Process, Establish Environment, Write Story Card, UI Design, Define Architecture, Planning, Iteration plan, Face-to-face Meeting, Incremental Design, TDD, Refactoring, Release Plan, Feedback, Pattern Manage, Pair Programming, Integration, Acceptance Test, User Test Figure
Work Product	It is a general term for task inputs and outputs. There are three types of work product.	Product Summary, Project Planner, UI Sample, UI Model, UI pattern, Architecture Pattern, Application Pattern, Story Card, Task Card, Architecture Model, Component Model, Test Case.

It is based on Extreme Programming (XP), Agile Unified Process, RUP and the Software and Systems Process Engineering Meta model (SPEM). It is a GUI based architecture-centered that uses Agile approaches for rapid development and utilizes domain knowledge.

MASAM shows a strong tie with the Mobile-D methodology and introduces slight variations, such as project management and follow up tool coupled with Eclipse Process Framework.

MASAM is described according to Software and Systems Process Engineering Meta-model (SPEM) by the following three kinds of process assets described by (Goldsbury, 2012) and presented in Table 2:4.

MASAM proposes a mobile application development cycle comprised of four phase, Table 2:5.

1. The Preparation Phase defines a summary and a first notion of the product, and assigns roles and responsibilities.
2. The Embodiment Phase focuses on understanding user's needs and defining the architecture of the software product.
3. Developing Phase, that benefits from traditional agile principles to furnish an iterative Extreme Programming development sequence. The implementation of the software product is carried out through Test-Driven Development, Pair Programming, Refactoring and Continuous Integration, with a close relationship with iterative testing activities.
4. Finally, a Commercialization Phase concentrates on product launching and product selling activities.

It is recommended to use MASAM methodology for small companies that are focused on the development of mobile software applications. However the, authors have not presented a case study of an actual implementation of this methodology in a real-world environment.

Table 2: 5 Phases of MASAM process

Phase	**Activity**	**Task**
Preparation Phase	Grasping Product	Product summary
		Pre-planning
	Product Concept Sharing	User Definition
		Initial product analysis

	Project Set-up	Development process coordination
		Project resource coordination
		Pre study
Embodiment Phase	User Need Understanding	Story card workshop
		UI design
	Architecting	Non-functional requirement analysis
		Architecture definition
		Pattern management
Development Phase	Implementation & Preparation	Environment setup
		Development Planning
	Release Cycle	Release Planning
		Iteration Cycle
		Release
Commercialization Phase	System Test	Acceptance Test
		User Test
	Product Selling	Launching Test
		Product Launching

2.2.3.5 SLeSS

(Cunha, et al, 2011) proposed SLeSS, an integration approach of Scrum and Lean Six Sigma used in real projects for developing embedded software customized for mobile phones. SLeSS enables the achievement of performance and quality targets, progressively improving the development process and the outcome of projects. The approach uses two types of product

backlogs, Customization Product Backlog (for customizing development projects) and LSS Product Backlog (for process improvements). The use of SLeSS assists in easy adaptation to requirement changes in the later stages of the project and with less overall impact than the traditional approach, helps in meeting deadlines, reduces overtime hours, and delivers more rapidly versions while shortening the development cycle.

Besides this, the use of the approach enables the achievement of performance and quality targets of real software development project, increases productivity, improves process quality, helps in cost reduction, progressively improves the development process, management process and the outcome of projects with fewer defects and failures.Scrum is an agile methodology for project management and software development that adopts an empirical approach rather than prescriptive one and therefore it may be used in complex projects. On the other hand, Lean Six Sigma (LSS) is a methodology for defining and improving products, processes and services with a focus on reduction of defects and failures, on variation and waste elimination, prioritizing, in a planned and objective way, the achievement of quality and financial results.

1. Scrum in SLeSS

Scrum is widely used in software development, and it has been observed and documented in the scope of mobile software development (Scharff & Verma, 2010)**.** The execution of SLeSS assumes an incremental approach by first implementing the Scrum alone and once the Scrum is well settled in the organization, LSS should be implemented as a quality framework.

2. Lean Six Sigma in SLeSS

Once Scrum is well settled in the organization, Lean Six Sigma (LSS) is applied as a quality framework that complements Scrum as a development methodology. The model is represented by the 5-phase DMAIC phases Table 2:6 (Define, Measure, Analyze, Improve and Control).

The SLeSS approach has been used in real embedded software customization development projects for mobile phones. The approach was experimented in P&D&I laboratory, with a mobile phone manufacturer as a client with a team of 7-12 developers, in duration from 4-6 months, with an average size of 529 LOC (Line of Code) developed in ANSI C programming language.

SLeSS results in obtaining higher outcomes, such as better adaptation to changes in requirements, fulfillment of the deadlines, decrease in number of unplanned overtime and delivering more versions rapidly with fewer defects or failures. It demonstrates increase in

productivity, improvement in process quality and reduction in cost. Besides this, the approach has allowed improvement in development and management processes.

Table 2: 6 DMAIC 5 Phases of SLeSS approach

Phases	Backlog Items
Definition Phase	LSS Project Contract
	Initial Analysis
Measurement Phase	SIPOC Diagram (Supplier, Inputs, Process, Outputs, Customers)
	Process Map
	Cause and Effect Diagram
	Cause and Effect Matrix
	Impact Effort Matrix
	Initial Capability
	Measurement and Inspection Systems
Analysis Phase	FTA (Fault Tree Analysis)
	Analysis of critical inputs of the processes
Improvement Phase	Action Plan
	SIPOC
	Process Map
	Piloted Solution
	Final capability of the processes
Control Phase	Control Plan

2.4 Chapter Summary

In this Chapter we discussed agile development for mobile applications a relatively new approach to mobile platform-based devices applications development, presented a comprehensive review of the current state-of-the-art in the design of usable mobile platform based devices. Further we highlighted Usability engineering Issues with Agile Processes, the unique development challenges for mobile platform based devices and the gaps in industry practice leading us to consider the best way possible to address the challenges for a better mobile platform based devices applications development environment.

CHAPTER THREE

3.0 RESEARCH METHODOLOGY

In this chapter, we elaborate the research design and specific approach that was adopted in this study to put forward a process framework and used the framework later as a baseline for proposing an Extended Mobile-D agile process model. The chapter focuses on data collection, processing and analysis methods. Data collection instruments and procedures are also discussed as well as the target population and study sample.

(Zikmund, et al, 2010) describe a research methodology as a part that must explain technical procedures in a manner appropriate for the audience. It achieves this by addressing the research and sample designs used for the study, the data collection and fieldwork conducted for the study and the analysis done to the collected data. (Dawson, 2009) states that research methodology is the philosophy or general principle which guides the research. (Kombo & Tromp, 2009) concur with (Zikmund, et al, 2010) that research methodology deals with the description of the methods applied in carrying out the research study.

3.1 Research Design

(Dawson, 2002) describes the purpose of this section as to set out a description of, and justification for, the chosen methodology and research methods. (Polit & Beck, 2003) describe a research design as the overall plan for obtaining answers to the questions being studied and for handling some of the difficulties encountered during the research process. (Miller & Yang, 2008) and (Kothari, 2004) describe a research design as the arrangement of conditions for collection and analysis of data in a manner that aims to combine relevance to the research purpose with economy in procedure. (Kombo & Tromp, 2009) describe a research design as the review of the overall research aim, the literature and chosen research methods. (Kothari, 2004)) states that research design facilitates the smooth sailing of the various research operations, thereby making research as efficient as possible, yielding maximal information with minimal expenditure of effort, time and money.

(Lavrakas, 2008) asserts that choosing an appropriate research design depends on;

a) The nature of the research questions and/or hypotheses
b) The variables
c) The sample of participants

d) The research settings
e) The data collection methods and
f) The data analysis methods.

Thus, a research design is the structure, or the blueprint, of research that guides the process of research from the formulation of the research questions and/or hypotheses to reporting the research findings.

In designing any research study, the researcher should be familiar with the basic steps of the research process that guide all types of research designs. Also, the researcher should be familiar with a wide range of research designs in order to choose the most appropriate design to answer the research questions and/or hypotheses of interest.

This study used Action research design the term Action Research was coined by (Lewin, 1946) who describes it as "comparative research on the conditions and effects of various forms of social action, and research leading to social action." As an interventionist approach, Action Research is a self-reflective form of inquiry in that the researcher gleans knowledge about his or her role in the action, as well as knowledge about how valid the action might be. Action Research is an iterative research approach involving the planning of an intervention, carrying it out, analyzing the results of the intervention, and reflecting on the lessons learned; these lessons contribute to the re-design of the action and the planning of a new intervention. The Action Research methodology used in this dissertation is based on that proposed by (Baskerville, 1999), who breaks an intervention into the research cycle illustrated in figure 3:1

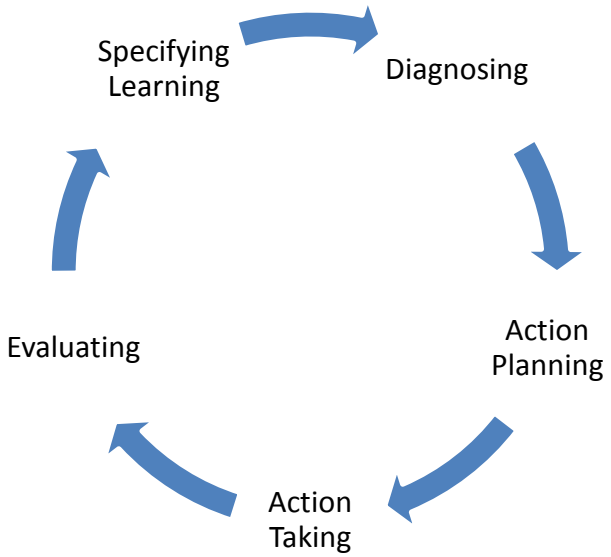

Figure 3:1 Baskerville approach

The Action Research intervention is situated within a specified and agreed Research Environment and for our study the research environment happened to be mobile platform-based devices computing solutions development organizations.

The Diagnosing phase involves identifying the problems motivating the intervention. The client may stipulate their interpretation of these problems, but the researcher also needs to undertake some form of empirical or conceptual investigation to develop his or her own assumptions. This phase commanded us to identify crucial, essential and important discount usability activities that must be integrated in software engineering (SE) processes through literature review, case studies and empirical findings.

The Action Planning phase involves devising the nature of the intervention that will relieve or improve the identified problems. This involves agreeing on the desired future state, and the changes necessary to achieve this.

The planned intervention takes place during the Action Taking phase, according to some form of intervention strategy. This strategy may involve the researcher being an active participant in the intervention; alternatively, the researcher may provide explicit or implicit guidance to other participants and observe the outcome.

Irrespective of the strategy, the researcher collects data about the intervention for subsequent analysis. As Action planning phase involves devising the nature of the intervention that will relieve or improve the identified problems and the identified issues in the diagnostic phase it helped us to consider the best way to develop a framework which went on to work as a baseline for integrating discount usability activities into software engineering (SE) and further propose an Extended Mobile-D agile process model.

Once the intervention is complete, the researchers and practitioners evaluate the outcome as part of the Evaluating phase; this involves questioning whether the intervention was the sole cause of success (or failure) and the validation of our framework and model was necessary to ensure their soundness. A questionnaire was administered to software and HCI practitioners as a means of getting their industry view of our improvement in the software engineering community.

Cyclically the final phase, the Specifying Learning phase takes place on an on-going basis, and involves re-applying lessons learned during the intervention. This may involve recommending organizational changes, using the knowledge to inform the approach to take for future interventions, and reporting any general insights to the scientific community.

3.2 Target population

(Lavrakas, 2008) defines a population as any finite or infinite collection of individual elements. (Hyndman, 2008) describes a population as the entire collection of 'things' in which we are interested. According to (Zikmund, et al, 2010) and (Kothari, 2004)), a population refers to all items in any field of inquiry and is also known as the 'universe'. (Polit & Beck, 2003) refer to population as the aggregate or totality of those conforming to a set of specifications. The target populations for this study were the mobile devices software developers from the industries and research institution. The main reason for choosing the particular group of software developers was because they are responsible for using the software development models to transform a scanty idea into a mobile application.

3.3 Sample and Sampling Technique

(Lavrakas, 2008) describes a sample in a survey research context as a subset of elements drawn from a larger population. (Kombo & Tromp, 2009) and (Kothari, 2004) also describe a sample as a collection of units chosen from the universe to represent it, before collecting data, it is essential to determine the sample size requirements of a study. (Polit & Beck, 2003), strongly recommend that it is more practical and less costly to collect data from a sample than from an entire population. The risk, however, is that the sample might not adequately reflect the population's behaviours, traits, symptoms, or beliefs. Various methods of obtaining samples are available. These methods vary in cost, effort, and skills required, but their adequacy is assessed by the same criterion of the representativeness of the selected sample.

The study used a purposive sampling procedure to identify the sample units. (Lavrakaz, 2008) states that a purposive sample, also referred to as a judgmental or expert sample, is a type of non-probability sample. The main objective of a purposive sample is to produce a sample that can be logically assumed to be representative of the population. This is often accomplished by applying expert knowledge of the population to select in a non-random manner a sample of elements that represents a cross-section of the population.

(Miller & Yang, 2008) and (Kothari, 2004) define purposive sampling as involving deliberate selection of particular units of the universe for constituting a sample which represents the universe. Purposeful sampling method enables the researcher to select specific subjects who will provide the most extensive information about the phenomenon being studied.

The sample units in this study were chosen due to ease of access to information and this study used simple random sampling for distribution of the questionnaires.

3.4 Data Collection Instruments

The study used a combination of both open and closed questions questionnaires containing a design likert scale that allowed us to assign numbers 1-5 to collect both qualitative and quantitative data about the the proposed Extended Mobile-D model and framework. Likert scale was chosen because of its straight forward nature and ease of analysis of data.

(Mugenda & Mugenda, 2003) and (Kothari, 2004) define a questionnaire as a document that consists of a number of questions printed or typed in a definite order on a form or set of forms. According to (Dawson, 2002), there are three basic types of questionnaires;

- Closed ended,
- Open-ended or a
- Combination of both.

Closed-ended questionnaires are used to generate statistics in quantitative research as these questionnaires follow a set format, and as most can be scanned straight into a computer for ease of analysis and greater numbers can be produced.

Open-ended questionnaires are used in qualitative research, although some researchers will quantify the answers during the analysis stage. The questionnaire does not contain boxes to tick, but instead leaves a blank section for the respondent to write in an answer. Whereas closed-ended questionnaires might be used to find out how many people use a service, open-ended questionnaires might be used to find out what people think about a service, as there are no standard answers to these questions, data analysis is more complex. Also, as it is, opinions which are sought rather than numbers, fewer questionnaires need to be distributed. However, many researchers tend to use a combination of both open and closed questions. That way, it is possible to find out how many people use a service and what they think about that service on the same form.

(Mugenda & Mugenda, 2003) and (Kothari, 2004) agree that questionnaires have various merits, like; there is low cost even when the universe is large and is widely spread geographically; it is free from the bias of the interviewer; answers are in respondents' own words; respondents have adequate time to give well thought out answers; respondents who are not easily approachable can

also be reached conveniently; large samples can be made use of and thus the results can be made more dependable and reliable.

They also concur that the main demerits of questionnaires are; low rate of return of the duly filled in questionnaires; bias due to no-response is often indeterminate; it can be used only when respondents are educated and cooperating; the control over questionnaire may be lost once it is sent; there is inbuilt inflexibility because of the difficulty of amending the approach once questionnaires have been dispatched; there is also the possibility of ambiguous replies or omission of replies altogether to certain questions .In view of the advantages and the need to gather more information, questionnaires were administered to mobile devices software developers to solicit their views concerning our framework and the Extended Mobile-D model. The steps followed to design and administer the questionnaires include:

a) Defining the objectives of the survey
b) Determining the sampling group
c) Writing the questionnaire
d) Administering the questionnaire and
e) Interpretation of the results

3.5 Data Processing and Analysis

Statistical analysis was carried out in this project. Statistical analyses cover a broad range of techniques, from simple procedures that we all use regularly like computing an average to complex and sophisticated methods. Although some methods are computationally formidable, the underlying logic of statistical tests is relatively easy to grasp, and computers have eliminated the need to get bogged down with detailed mathematical operations (Polit & Beck, 2003).The data collected was analyzed using SPSS to get information about quality of the framework and the Extended Mobile-D model, important factors about the two and their suitability to the current mobile devices software development environments plus more findings were presented inform of graphs and tables since they are visual and can be easily interpreted.

3.6 Chapter Summary

In this Chapter, we have reviewed the research approach discussed our target population, the sample and sampling techniques and concluded this chapter by considering data collection instruments and analysis procedure.

CHAPTER FOUR

4.0 RESULTS AND DISCUSSIONS

4.1 A Framework for Integrating Usability Engineering Into Mobile Platform-Based Devices Software Engineering

Our framework is proposed to be a flexible way of understanding and communicating the work of Usability Engineers practitioners in different contexts. Our objective is not to come up with another prescriptive “one-size-fits-all” Usability design process model, but rather to articulate the typical Usability Engineering activities within which several activities, techniques and deliverables can be assimilated.

A usability engineering project involves people guided by common goals and strategies working with a collection of tools to produce documents and code. The tools include compilers, debuggers, environments, change management, source control, project management, document processors, and domain modeling tools.

The documents produced include requirements that define the problem, customer manuals, test plans, scenarios, a design that defines the architecture, and implementation plans. The code may deal with objects, data structures, algorithms, methods, modules, protocols, and interface definitions.

The strategies are materialized through the collection of the architecture, methods, paradigms, risk analyses, conventions, and a mission statement. The materialization of the strategies addresses what is to be built, how it will be built, building it, and making it high quality. By combining the essential characteristics of the various strategies we propose a multi-disciplinary framework for integrating usability engineering design into mobile applications software engineering. We divide our framework into phases as in Table 4:1.

Each phase consists of one or more activities. Each activity is associated with one or more techniques. Each method requires specific skills and could be associated with a particular discipline to address a specific concern in mobile applications development. Each activity results in specific deliverables.

A deliverable may be an end in itself, or may be an input for another activity in the usability engineering design process or the software development process. Not all activities or techniques may be essential in each instance of the process.

Table 4: 1 Multi-Disciplinary Framework

Software development stage	Mobile applications and devices concern	Phase	Question	Disciplines Involved	Usability Activities	Usability Techniques	Impact
Planning	**Identify** Your business needs A problem to be solved by your application Application target users Mobile platforms and devices to be supported	Requirements elicitation and analysis	1. What is required?	Users, Designers, Ethnographers, Business analysts, Client/business stakeholders, Domain experts, Usability experts	User analysis Task analysis Market analysis	Conversational Observational Analytic Synthetic	Complete coverage of the current problem All relevant requirements are captured Definition of each problem
		Requirements specifications	2. What will the system do?	Domain experts, Ethnographers, client/business stakeholders, Designers, users, Business analysts, Usability experts	Essential use cases	Usability specification goals	Highlight any inconsistence and conflicting requirements
Analysis	Validate appropriate needs in accordance with the stakeholders wants	Requirements validation	3. Have we got the requirements right?	Domain experts, Client/business stakeholders, Designers, Ethnographers, Usability experts	user modeling	Collaborative inspections Cognitive walkthroughs Usability	Establishes confidence that requirements are fit for purpose

						evaluation (walkthroughs)	
Design	**Understand** The UI design Multi-touch gestures for touch enabled devices	Pre-design	4. Have we understood our targeted users	Usability experts, Designers, Domain experts, Users	Ideation Specific targeted user analysis	Contextual design techniques Heuristic evaluations QFD TRIZ	Design ideas Concrete set of usability goals
	Consider Approach to develop the application (Native, Web or Hybrid) Platform design standards	Design	5. How should we respond	Usability experts, Designers and Domain experts.	Formative usability evaluation and refinement	Heuristic evaluations Thinking aloud	Prototype to gather requirements and usability principles
	Evaluate Prototype	Post-design	6. How are we doing?	Usability experts, Designers and Users	Prototype usability inspection	Impact analysis Interviews and surveys	Data on prototype performance
Development and	**Build**	Transformation	7. How should the	Designers and Usability	Design	Interface	Early version of the

Implementation	Exactly what is needed **Integrate** An appropriate analytics tool		design be achieved	experts	detailing	development Visual development Information development Navigation design	product becomes visible
				Designers and Usability experts	Development support	Reviews during development	A clear picture of the product
Testing	Listen to application feedback and integrate relevant ones	Expert Evaluation	8. How does it perform?	Users, Usability experts and Designers	Detailed usability evaluation	Field observations Attitude questionnaires User auctions logging Field trials Heuristic Evaluation Collaborative	Usability problems

						inspections Thinking aloud	
		Usability Testing	9. Which areas need more work?	Users, Usability experts, Designers and Client/business stakeholders,	Refinement	Reviewing development and implementation step	Improvement on stated usability problems
Maintenance	Upgrade your application with improvements and new features	Evolution phase	10. How do we compensate for failures	Usability experts, users and Designers.	Summative usability evaluation	Usability test Heuristic evaluations	Smooth running of the system

In our framework, we identify usability engineering activities that, we propose, are essential for integration with the six core software engineering process activities , I e;

a. Planning
b. Analysis
c. Design
d. Development and Implementation
e. Testing
f. Maintenance

We organize these activities in ten phases, which we describe in terms of ten questions.

1. What is required?
2. What will the system do?
3. Have we got the requirements right?
4. Have we understood our targeted users?
5. How should we respond?
6. How are we doing?
7. How should the design be achieved?

8. How does it perform?
9. Which areas need more work?
10. How do we compensate for failures?

The following sections describe the details of the software development stages the concerns to be addressed, disciplines involved, usability activities including the techniques that could be used to perform the usability engineering activities, and expected deliverables/impact or outcomes.

4.1.1 Planning

Understanding why your organization needs mobile applications, and what business processes they will support, is key to a successful mobile application strategy. An effective mobile application strategy involves knowing the reasons behind the mobile application drive. To create a successful mobile application, the first things you need to keep in mind are:

- Identify a problem which can be resolved by your application
- Decide the features of your application

The application should provide customers with tangible benefits including reducing costs via productivity enhancements, new revenue or improving the customers experience.

You need to take a deep dive into the goal of your mobile application and determine exactly

- What it is that the mobile application will do are you looking to serve your customers, employees, vendors, channels, or all of the above?
- Will mobile applications enhance or replace current technologies? Think of the business processes you want to enable with mobile applications.

You need to identify or be clear about:

- **Application target users**

An application should always be developed keeping in mind the target users of the application. Having a clear vision regarding the target group enhances the success ratio of the application.

- **Mobile platforms and devices to be supported**

Mobile platforms and devices should be selected regarding the hardware performance, battery life, ruggedness and required peripherals. Certain factors that needs to be considered while selecting mobile platforms and devices includes coverage, device support, performance and other features.

Question 1 and 2 successfully navigate us through the planning stage.

4.1.1.1 Question 1 What is required?

This is a broader question and asks the design team to look at the problem at hand as holistically as possible. It is not only about what is "required" by someone. This question is answered through divergent thinking, user analysis, task analysis and market analysis while looking beyond what had been specified in the design brief, and trying to set the problem before solving it the phase here is known as requirement elicitation.

Requirements elicitation is the practice of collecting the requirements of the application from users, customers and other stakeholders its major goal is to avoid the confusions between stakeholders and the design team. It is non-trivial because the team can never be sure to get all requirements from the user and customer by just asking them what the application should do. Before requirements can be analyzed, modeled, or specified they must be gathered through an elicitation process. This phase involves intensive interaction between stakeholders and the team.

The usability techniques associated with this phase are

- **a.** Conversational
- **b.** Observational
- **c.** Analytic and
- **d.** Synthetic.

a. Conversational methods

The conversational method provides a means of verbal communication between stakeholders and the team it's an effective means of expressing needs and ideas, and the methods are used massively to understand the problems and to elicit generic product requirements. The Conversational Methods are also known as verbal methods, such as Interviews, Questionnaire, and Brainstorming.

Conversation is one of the most prevalent yet invisible forms of social interaction. People are usually happy to describe their work and difficulties they face. They verbally expressive demands, needs and constraints

- **Interviews:**

An Interview is generally conducted by experienced analysts design teams, who have some generic knowledge about the application domain as well. The team discusses the desired product

with different stakeholders and develops an understanding of their requirements. Our framework advocates for closed and open Interviews.

Closed Interview: In this the team prepares some predefined questions and tries to get the answers for these questions for the stakeholder. Open-ended Interview: here the team does not need to prepare any predefined questions, and information is collected through open discussions.

- **Questionnaire:**

The team might also employ questionnaires it's one of the methods of gathering requirements in less cost and reach a large number of people, not only in less time but also in a lesser cost. The framework suggests the general factor to consider in the usage of the questionnaire is that the type of requirements that has to be gathered depends on the level of the respondent's knowledge and background.

- **Brainstorming :**

In brainstorming stakeholders gather together for a period but in this time period they develop a large and broad list of ideas. Here "out -of-the-box" thinking approach is encouraged. The brainstorming involves both usability idea generation and idea reduction.

b. bservational methods:

The observational method provides the team with a means to develop a better understanding about the domain of application, the observer team must be accepted by the people being studied and the people being studied should carry on with their normal activities as if the team is not there.

The design team observes usability activities at environments where the application is expected to be deployed. The observation methods come into play where verbal communication becomes helpless for collecting tacit requirements.

Therefore, by the team observing how people carry out their routine work forms a means of acquisition of information which are hard to verbalize. The observational methods are well suited when stakeholders find it difficult to state their needs and when the team is looking for a better understanding of the context in which the desired product is expected to be used.

Observational methods include Social analysis, Observation, Ethnographic study, and Protocol analysis.

- **Social analysis, Observation, Ethnographic study**

The team or a member spends some time in a society or culture for making detailed observation of all their practices. This practice gives the initial understanding of the applications, work flow and organizational culture.

- **Protocol analysis**

In protocol analysis the team observes a stakeholder when s/he is engaged in some task, and concurrently speaks out loud and explains his/her thought, with the protocol analysis it is easy to identify interaction problems in existing applications and it gives better and closer understanding of work context and work flow.

c. Analytic methods

The team uses conversational or observational methods to directly extract requirements from people's behavior and their verbalized thought but still there is a lot of knowledge that is not directly expressed, for example expert's knowledge, information about regulation and legacy products are some examples of such sources. All the stated sources provide usability engineers with rich information in relation to the product. Analytic methods provide the team ways to explore the existing documentation or knowledge and acquire requirements from a series of deductions. It includes requirements reuse, documentation studies, laddering, and repertory grid

- **Requirement reuse**

In this technique, the team uses glossaries and specification of legacy systems or systems within the same product family to identify requirements of the desired application. It has been observed that many requirements in a new system are more or less same as they were in a legacy system's requirement. So it is not a bad idea to reuse the details of requirements of an earlier system in a new system.

- **Documentation studies**

Here the team read and study different available documents (e.g. organizational policies, standards, legislation, market information, specification of legacy systems) to find the content that can prove to be relevant and useful for the requirements elicitation tasks.

- **Laddering**

This technique can be divided in three parts: creation, reviewing and modification.

Laddering method is a form of structured interview that is widely used in the field of knowledge elicitation activities to elicit stakeholder's goals, aims and values.

The team uses laddering method to create, review and modify the hierarchical contents of expert's knowledge in the form of tree diagram. It was first introduced by the clinical psychologists in 1960 to understand the people "score values and beliefs". Its success in the fields of psychology allows other researchers in the industries to adapt it in their fields. Specifically software developers have adapted the laddering techniques to gather the complex user tacit requirements.

- **Repertory grid**

The stakeholders are asked for attributes applicable to a set of entities and values for cells in entity-attribute matrix. The analytic methods are a complementary way to improve the efficiency and effectiveness of requirements elicitation, especially when the information from legacy or related products is reusable.

d. Synthetic methods

It is apparent that no single method is sufficient enough to develop all the requirement of an application. All these methods are good and very handy in some certain context and circumstances. It is often a good idea to combine different elicitation methods for developing requirement. The combination helps the usability engineering team to uncover the basic aspects and gain a generic knowledge of the application domain.

The synthetic coherently combine conversation, observation, and analysis into single methods. The team and stakeholder representatives communicate and coordinate in different ways to reach a common understanding of the desired product.

Synthetic methods are also known as collaborative methods as they are collaboration of multiple requirement elicitation methods. Synthetic techniques include scenarios, passive storyboards, prototyping, interactive storyboards, JAD/RAD sessions, and contextual inquiry.

- **Scenarios, passive storyboards**

This is the teams' interaction session. In this session a sequence of actions and events for executing some generic task which the applications are intended to accomplish are described. Here the team establishes clear requirements related to the procedure and how data flow can be achieved.

- **Prototyping, Interactive storyboards**

In this technique, the team discusses a concrete but partial application with stakeholders. The full version is expected to be delivered at the end of the project. The purpose of showing this concrete but partial application to stakeholders is to elicit and validate functional requirements.

- **JAD/RAD sessions**

Joint Application Development/Rapid Application Development emphasizes user involvement through group sessions with unbiased facilitator. JAD is conducted in the same manner as brainstorming, except that the stakeholders and the users are also allowed to participate and discuss on the design of the proposed application. The discussion with the stakeholders and the users continues until the final requirements are gathered.

- **Contextual inquiry**

The team gets this technique as a combination of open-ended interview, workplace observation, and prototyping. This method is preferred for interactive applications design where the user interface design is critical. It is very essential for requirements engineers to study how people perceive, understand, and express the problem domain, how they interact with the desired product, and how the physical and cultural environments affect their actions.

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

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

4.1.1.2 Question 2 What will the system do?

The requirements specification focuses on what the application will do not how it will be implemented. A requirements specification is a description of a software system to be developed, laying out functional and non-functional requirements detailing the essential behavior of a software product from a user's point of view.

The team must understand the objects the application will manipulate (information domain), the services (functions) the application will perform, the constraints on the project (time, money and

technical) and the performance expected (timing). Essential use cases are of primary importance in this phase. Their purpose is to document the usability process that the application must support without bias to technology and implementation. The team is to express the narrative in the essential use case in the language of the applications domain and of users'.

Essential use cases should help achieve the following goals:

- Serve as an effective communication tool between users and the team.
- Identify and document the applications logics and actions.

The requirement specification is used for verifying whether all the functional and non functional requirements specified in the software requirements specification (SRS) are implemented in the product. The complete description of the functions to be performed by the product specified in the (SRS) will assist the potential users to determine if the product specified meets their needs or how the product must be modified to meet their needs. The basic issues that the team must address here include

- Functionality
- External interfaces
- Performance
- Attributes and
- Design constrains imposed on an implementation.

As the phase progresses the team might encounter inconsistence and conflicting requirements to be addressed.

4.1.2 Analysis

In analysis the team validates the applications appropriate needs in accordance with the stakeholders' views, without approving that you have got the requirements right as described in question 3 below and clearly understanding who your audience is/are and what they want, as well as their preferred applications. Companies may end up building the greatest applications no one ever uses.

Mobile devices have some key differentiators that set them apart as widely discussed in Chapter Two.

- Mobile devices are more personal and mobile applications are often aimed at a particular user and not the general mass market.

- Mobile development is much more complex and there is a huge number of various mobile devices running on different platforms.
- People spend more time using mobile applications than even surfing the internet – more than six times as much per month (based on recent analysis by Business Insider).

This means getting the applications requirements right and what business processes they will support, is key to a successful mobile application development.

4.1.2.1 Question 3 Have we got the requirements right?

Validation works with a final draft of the requirements document with negotiated and agreed requirements "Have we got the usability requirements right" is the key question to be answered at this stage, these features called requirements, must be quantifiable, relevant and detailed.

Requirements verification includes a broader reviewer pool and occurs in stages. First, the common product line requirements must be verified. Later, as each product comes on the scene (or is updated), its product-specific requirements must be verified. But the product-line-wide requirements must also be verified to make sure that they make sense for this product.

Cognitive walkthrough a usability activity we prefer in this phase is an expert based evaluation technique that steps through a scenario/task by focusing on the users' knowledge and goals requirements.

The expert team evaluation first starts with descriptions of the applications requirements, the task(s) from the users' perspective while validating the correct types of requirements needed. Then the team walks through the tasks reviewing the actions that are necessary and attempting to predict how the users' will behave just trying to make sure that only the right and most important requirements sail through.

The team should ensure energy is directed towards ensuring that the final product conforms to client usability needs rather than attempting to mold user expectations to fit the requirements.

Requirements validation is a team effort that demands a combination of hardware, software and human factors engineering expertise as well as skills in dealing with people.

The teams objectives is to certifies that the requirements document is an acceptable description of the system to be implemented and checks a requirements document for

- Completeness and consistency
- Conformance to standards
- Requirements conflicts

- Technical errors
- Ambiguous requirements

4.1.3 Design

This stage represents the ``how" stage. Here the architecture is established. This stage starts with the requirements document delivered by the requirements phase and maps the requirements into architecture. The architecture defines the components, their interfaces and behaviors.

Software design is both a process and a model. The design process is a sequence of steps that enable the team to describe all aspects of the software to be built. Any design problems must be tackled in three stages;

- Study and understand the problem
- Identify gross features of at least one possible solution
- Describe each abstraction used in the solution

The important phases of the software development lifecycle in which the usability activities will be integrated are before, during and after product design and implementation. Nielsen recommends these activities be applied in an iterative fashion similar to Boehm’s (1988) spiral model. Software designers do not arrive at a finished design immediately. They develop design iteratively through number of different versions. The starting point is

- Pre-design
- Design and lastly
- Post-design

4.1.3.1 Question 4 Have we understood our targeted users?

The purpose of activities in this pre-design phase would be to understand target users, their tasks and their work environment. Designing your application is yet another significant factor responsible for the success of the application in the market. An application developer should concentrate on the UI design, multi-touch gestures for touch-enabled devices and consider platform design standards as well. Today, emphasis is on the UI design of an application as it plays a crucial role in the success of the application. Designing applications is becoming increasingly popular as it creates an instant impact on the mind of the user while ensuring usability of the application.

In pre-design the teams' objective is to not rush into the design phase, until clear usability goals have been set transforming the problem space so that one or a few solutions become evident. The team begins with ideation using a range of ideation techniques such as brainstorming, synectics, participatory design, quality function deployment (QFD), and theory of inventor's problem solving (TRIZ) the team comes up with a range of design ideas that solve the problems and realize the opportunities Buxton (2007). The ideas are typically wild and divergent to begin with, and early on, the focus is on generating more ideas rather than evaluating them. Individual user differences and variability in tasks are two factors that have a large impact on usability, thus users' must be observed in their natural work environment. By knowing the target demography of the user population, personas can be developed that will allow the engineering team to anticipate learning difficulties and other factors. This will allow the team to design the UI such that it caters for these factors, with regards to the tasks that users perform, the overall goals of users must be examined. Personas are a usability technique designed to direct the focus of the development process towards the goals of the people who actually use the product.

Information flow and the sequence of events of normal tasks and exceptional tasks should be studied. Apart from this, functional analysis must be performed to find out what is it that users' are really trying to do. Contextual design techniques and its artifacts can be used to find and model all this information. Competition or existing products that the software will replace must be taken into account. These will influence factors such as the UI of the new software must not conflict with skills users' have already learnt. It is a good idea for the teams' to perform heuristic analysis of existing competition, and later use that information to create prototypes, which can be used to further understand and refine requirements.

The end result of usability activities in this phase should be to come up with a concrete set of usability goals. Usability goals should be based on the different defining parameters of usability (understandability, applicability, efficiency etc.). These different parameters should be prioritized and findings should be documented in a concrete way, this will allow teams' to better gauge the scope and required quality of the software.

4.1.3.2 Question 5 How should we respond?

This is a holistic question and the teams' main goal at the design phase is to define an interaction specification that is usable and implementable. This should be achieved by creating a prototype

based on the gathered requirements and usability principles and performing test iterations with target users.

Selecting the right approach for developing the application is highly important. Ideally, applications development approach must be in accordance with the time and budget. The common development approaches are Native, Web and Hybrid

- **Native:**

Native applications enable delivery of the best user experience but require significant time and skill to be developed. These applications are basically platform specific and require expertise along with knowledge. Native applications are costly as well as time consuming to be developed and deliver the highest user experience amongst all the approaches.

- **Web:**

Web applications are quick and the cheap ones to develop and can run on multiple platforms. These applications are developed using HTML5, CSS and JavaScript code. These web applications are less powerful than native applications.

- **Hybrid:**

Hybrid approach is the latest approach to develop any application.

This approach combines prebuilt native containers with on-the-fly web coding in order to achieve the best of both worlds. In this approach, the developer augments the web code with native language to create unique features and access native APIs which are not yet available through JavaScript.

Experimental prototyping and heuristic evaluation should be done early on in the design phase so that the design can be refined quickly.

Heuristic Evaluation is usually conducted in a series of four steps:

- Prepare: create a prototype to evaluate; select evaluators; prepare coding sheets to record problems
- Determine approach: either set typical user tasks (probably the most useful approach) allow evaluators to establish their own tasks or conduct an exhaustive inspection of entire interface
- Conduct the evaluation: evaluators inspect interface individually to identify all violations of heuristics (the usability problems); record the problem (feature and location), severity (based on frequency, impact, criticality/cost) and heuristic violated

- Aggregate and analyze results: group similar problems; reassess severity; determine possible fixes

In early stages, prototypes can be low fidelity paper based ones. As they are tested out with users' and refined, they could be implemented using software. Also, the advantage of heuristic evaluation is that it can be applied without having a running system. This allows the design teams' to test their interfaces when they cannot be tested with users'.

Additionally, it is also important to have the technical tools in place to ensure a consistent and high quality development of the applications. A centralized authority can be setup that will coordinate different aspects of the interface design to ensure consistency. Other Software frameworks, libraries and standards should be used to ensure a consistent look of the product. A number of different empirical testing techniques can also be applied in this phase to verify the designs, such as thinking aloud, attitude questionnaires, automatic logging of user actions and usability testing. It is important to prioritize the problems found during design testing, because it might not be possible to solve all of them.

It is important to perform tests in an iterative fashion, and be ready to change and retest (refine) the interface when problems are uncovered. It is recommended that at this stage, elaborate tests of single design ideas should be avoided. Instead, the team should focus on different design ideas and be tested in small tests, so as not to wear out test users. It is also important to realize that if your test users become too accustomed to the prototype being tested repeatedly, they stop being the representative novices you should be performing tests with.

4.1.3.3 Question 6 How are we doing?

The teams main goal at post-design phase is to gather data for future versions of the software. Follow-up field observations should be performed to gather data on how the application is performing. Features (analytics tools) built into the application, such as automated collection of usage information and allowing users to easily send feedback to vendors can be considered. The teams' simply performs an analysis of the application and surveys its performs so carefully.

(Dillon (2001)) emphasizes the importance of these activities because typical usability tests with prototypes fail to capture how the relationship between the users' and application evolves over time, since it is not a holistic evaluation approach. Field observations allow teams' to overcome this shortcoming, and be prepared to anticipate changes in usability requirements.

4.1.4 Development and Implementation

Every mobile application platform has different characteristics and so developers must be familiar with the specifics for their platform if they are going to begin developing applications. Also, development of mobile applications typically requires applications to be initially developed and run on an emulator before being tested on actual mobile devices.

Mobile and smartphone applications have very different threat models than their web-based counterparts. Developers need to both understand the capabilities of their chosen development platform(s) as well as understand how to design and build applications to securely take advantage of mobile capabilities without exposing their organizations or application users to unnecessary risks.Developers building mobile applications need to understand the threat model for the system they are building as well as understanding that the mobile application itself is only part of the system that attackers will attempt to compromise.

Input that crosses a trust boundary should be positively validated and should not be used to make critical security decisions. Also, developers must be careful about what data is stored on the device because devices may be stolen or otherwise fall into unauthorized hands. Access permissions for local files and databases are also important because device owners might unwittingly install other applications on the device that are malicious.

Secure architecture and design principles can be useful when beginning the development of a new application so that possible concerns are known up-front. The recommendations drawn from the design exercises in design stage above must then be implemented during development and implementation as stated in Question 7.

4.1.4.1 Question 7 How should the design be achieved?

In the implementation phase, the team builds the components either from scratch or by composition. Once a feasible product definition is agreed upon, the detailed user interface is designed. Designers explore the details of the user interfaces such as labels, icons, and behavior of widgets. Given the architecture document from the design phase and the requirement document from the analysis phase, the team should build exactly what has been requested, though there is still room for innovation and flexibility.

The code is written the process of writing source code often requires expertise in many different subjects, including knowledge of the application domain, specialized algorithms and formal

logic. The phase deals with issues of quality, performance, baselines, libraries, and debugging. The end deliverable is the product itself.

4.1.5 Testing

Testing involves listening to feedback and integrating the relevant ones. Identifying and using beta-testers is a good idea in this stage. Beta testing is the first opportunity to get feedback from the target customers. It is especially important as it enhances the teams' visibility in the application or system store. It not only reduces product risk but gets the team that initial push in the application or system store. To identify beta testers is another important task to ensure success of an application.

It is highly important to identify and clearly define the target audience. This will enable the teams' to identify the right testers during the beta tester recruiting. Early market research helps in understanding market analysis which eases the process of beta testing, before beta testing the application on different platforms the team needs to take into account majority of the devices which eliminate device specific bugs. Alpha testing with a small number of users enables to clear out maximum bugs. At the same time, device coverage plan is significant for quality assurance of mobile applications.

Testing is the best opportunity to get real feedback from target customers. It provides a great opportunity to further understand target market and their requirements as discussed in question 8 and 9 helping to reduce the product launch risk.

4.1.5.1 Question 8 How does it perform?

Simply stated, usability quality is very important**.** It is much easier to explain to a customer why there is a missing feature than to explain to a customer why the product lacks quality. Testing also identifies important defects, flaws, or errors in the applications code that must be fixed. The programmer(s) who wrote the application must have a reduced role in the testing if possible. The concern here is that they're already so intimately involved with the product and "know" that it works that they may not be able to take an unbiased look at the results of their labors.

Testers' team must be cautious, curious, critical but non-judgmental, and good communicators. One part of their job is to ask questions that the developers might find not be able to ask themselves or are awkward, irritating, insulting or even threatening to the developers. After the code is developed the team tests it against the requirements to make sure that the product is

actually solving the needs addressed and gathered during the requirements phase. During test planning the team decide what an important defect is by reviewing the requirements and design documents with an eye towards answering the question “Important to whom?”

Generally speaking, an important defect is one that from the customer’s perspective affects the usability or functionality of the application. First, the team tests what’s important. Focusing on the core functionality the parts that are critical or popular before looking at the ‘nice to have’ features. Concentrate on the applications capabilities in common usage situations before going on to unlikely situations. In our framework Testing has three main purposes: verification, validation, and defect finding.

- The verification process confirms that the application meets its technical specifications. A “specification” is a description of a function in terms of a measurable output value given a specific input value under specific preconditions.
- The validation process confirms that the application meets the business requirements.
- A defect is a variance between the expected and actual result. The defect’s ultimate source may be traced to a fault introduced in the specification, design, or development (coding) phases.

The team does unit testing, integration testing, system testing and acceptance testing and checks

- If the application meets the requirements that guided its design and development,
- Responds correctly to all kinds of inputs,
- Performs its functions within an acceptable time,
- Is sufficiently usable,
- Can be installed and run in its intended environments, and
- Achieves the general result its stakeholder’s desire.

A primary purpose of testing is to detect failures so that defects may be discovered and corrected. Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions.

4.1.5.2 Question 9 Which areas need more work?

The main concern here is to understand which areas need more adjustments to make the application better, after implementing question 8 it becomes important to identify and upgrade the application with suggested improvements and innovative features before release.

A mobile application without innovative features loses its usability in the long run. Upgrading your application with innovative features enhances its visibility. Modification of the application before release occurs to correct faults, to improve performance and other attributes issues discovered during testing.

4.1.6 Maintenance

Software maintenance is a very broad activity that includes error correction, enhancements of capabilities, deletion of obsolete capabilities, and optimization as further elaborated by question 10, just building your applications, dropping them in the public or private application or system store and running won't ensure rapid and long-term user adoption.

Analyzing and managing your mobile applications to suit the changing demands of your applications users and their technologies will make a dramatic difference in the popularity, usability and lifecycle (and revenue generation) of your applications. Change is inevitable, mechanisms must be developed for evaluation, controlling and making modifications. The applications change due to corrective and non-corrective software actions. The team performs maintenance in order to:

- Correct faults
- Improve the design
- Implement enhancements
- Interface with other systems
- Adapt programs so that different hardware, software, system features, and telecommunications facilities can be used
- Migrate legacy software
- Retire software

4.1.6.1 Question 10 How do we compensate for failures?

Evolution of systems was first addressed by Meir M. Lehman in 1969. Over a period of twenty years, his research led to the formulation of Lehman's Laws, Lehman (1997). Key findings of his research include that maintenance is really evolutionary development and that maintenance decisions are aided by understanding what happens to systems (and software) over time. Lehman demonstrated that systems continue to evolve over time. As they evolve, they grow more

complex unless some action such as code refactoring is taken to reduce the complexity and preventing software performance from degrading to unacceptable levels.

It is impossible to produce applications of any size which do not need to be changed. Once the applications are put into use, new requirements emerge and existing requirements change as the business running those applications change.

Parts of the applications may have to be modified to correct errors that are found in operation, improve its performance or other non-functional characteristics. All of this means that, after delivery, applications always evolve in response to demand for change.

Software maintenance sustains the software product throughout its operational life cycle. Evolution requests are logged and tracked by the team the impact of proposed changes is also determined for action. Usability experts and the daily users provide critically and influential information for this stage.

4.1.7 Nature of the framework

Our framework is a software development framework modeled around a gradual increase in feature additions, a cyclical release and upgrade pattern. The most important advantages of this are:

- Higher product quality and improved implementation of functionalities
- More realistic estimates of time and money,
- Project team is under less pressure,
- Higher quality.

We begin with the planning stage and continue through iterative development cycles involving continuous user feedback and the incremental addition of features to maintenance stage. A cyclic approach involves intensive collaboration between the customer, designers and programmers (multidisciplinary).

If a deliverable does not turn out to be good in practice after a particular stage, it becomes obvious during the loops, thereby allowing adjustment. This way of working also allows customers to request adjustments affirming the flexibility of our framework. There are inner loops as shown in Figure 4:1. Sometimes, it may be necessary to go through these inner loops more than once to affirm feasibility of the product definition, and redesign to fix problems found, but going through them once might be required for all projects.

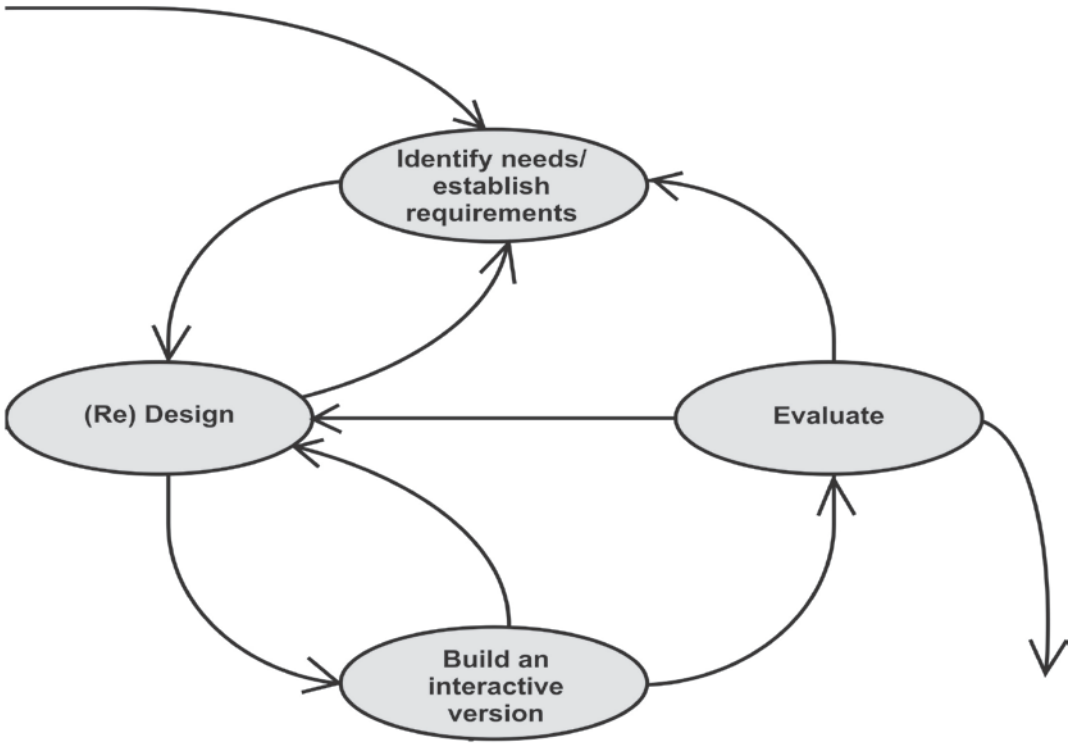

Figure 4:1 Framework's Cyclic Nature

4.2 Integrating Discount Usability into Mobile Agile Process Model

The first question to consider is

- Why one should integrate Usability engineering in to mobile agile process model software engineering and not integrate mobile agile process model software engineering into Usability engineering?
- What are the convergence and divergence points in usability and agile process models?
- Do we really understand them in order to facilitate a smother integration?

4.2.1 Why integrate

In a survey that included people from both professions, Jerome and Kazman report that several Usability engineering practitioners claimed that they collaborated with software engineers frequently, but software engineers believed that they had little or no contact with Usability

engineering practitioners (Jerome & Kazman, 2005). Their interpretation is that software engineers are reluctant to adopt Usability engineering processes, while Usability engineering people try to fit in.

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

One shortcoming of agile development methodologies that became apparent as agile practitioners began developing more interactive and UI-intensive applications is their marginalization of usability issues (Amber, 2008). This is especially true of agile methodologies such as XP as they were originally developed to focus on satisfying development and business needs rather than on end user needs. Agile practitioners and researchers have acknowledged the need to develop systems that meet usability requirements in addition to meeting functional and market requirements and are exploring ways of incorporating usability into agile methods (Sillito & Maurer, 2008).

Another, perhaps more reasonable, interpretation could be that there are lot more software engineers, and very few Usability engineering practitioners. Nielsen surveyed 31 development projects that had usability engineering activities to find how much usability effort was required in projects (Nielsen, 1993). Of the total project effort in person-years, median-sized projects reported using only 6.5% effort for usability. In an ideal situation, Usability engineering practitioners asked for only 10% effort on an average and 21% in the top quartile. The ideal desired usability effort was independent of the project size ($r=0.12$), i.e. smaller projects required relatively more usability efforts, while larger projects did not require lot more efforts. Over the years since Nielsen's survey, project sizes have reduced and importance of usability may have increased especially in mobile devices. On the other hand, there has also been an increase in experience within the Usability engineering community. In an ideal case, perhaps it is

appropriate to expect 10% of the overall effort to be associated with usability and HCI related activities. Usability engineering practitioners are outnumbered by a ratio 1:10 and It seems better if Usability engineering activities are integrated into existing software engineering (SE) processes rather than software engineering processes to be intergrated into Usability engineering. It certainly seems to be a good strategy for organizations, where Usability engineering practices are not yet well established.

4.2.2 Convergence points between agile and usability

Agile methods and usability engineering are built on some of the same principles. One of the key similarities is that both acknowledge that system development is a highly complex and dynamic endeavor that is subject to changing requirements and uncertainties that cannot be known in advance. As a result, both agile methods and usability methods follow cyclical development cycles, focus on early and continuous testing and are inherently human-centered as elaborated below.

4.2.2.1 Human-centered development

Communication is one of the central pillars on which the agile processes are built; both agile methods and usability methods are human-centered in that they both rely extensively on communication and coordination between various project stakeholders. Instead of relying on extensive documentation, teams are expected to communicate and coordinate effort on a daily basis and be able to help each other address problems as they come up.

Processes like XP also have an onsite customer who works regularly with the team to define requirements, answer questions and verify that the system functions as s/he requested (Beck, 2004). Similarly, Usability engineering processes rely on continuous communication and coordination among subject matter experts, usability engineers and end users. Observations of workplaces, usability testing with end users and participatory design techniques ensure that usability engineers understand users, user tasks and the context in which the system will be used.

4.2.2.2 Cyclical development

Agile and Usability methods follow cyclical development processes. This is a way for the system or UI design to be developed iteratively so the developers can verify that the system functions as

specified and can make course corrections as new requirements emerge. In Usability engineering, this relates to the task-artifact cycle where tasks or requirements determine how the artifact is designed. The artifact, in turn, can affect the task that it was originally designed to support. This similarity makes it easier to integrate Usability into mobile agile methods than into more traditional software development methods such as the waterfall development cycle.

4.2.2.3 Continuous testing

Agile methods and Usability engineering methods follow cyclical development processes; both rely on testing to verify that the developed system is meeting the project requirements. Agile methods like XP follow a test driven development cycle where code tests are developed before the functionality itself (Beck, 2004).

In addition, acceptance testing is carried out by the customer representative to verify that the system functions as he or she specified. Similarly, Usability engineering methods rely on a variety of analytic and empirical testing methods both to evaluate and compare different designs and to verify that the implemented system meets end user needs. Usability testing can be viewed as an extension to acceptance testing that contributes to the overall quality of the product.

4.2.3 Divergence points between agile and usability

Given the philosophical similarities between agile methods and Usability engineering methods, many of the difficulties of integrating the two approaches arise due to their different specific practices and the fact that Usability engineering methods focus on end user needs rather than customer needs. Some of the divergence points between the two areas are highlighted below

4.2.3.1 Working software vs design documentation

One of the foundations of the Agile Manifesto is that working software is valued over comprehensive documentation. Past software development projects would often get bogged down by large requirements and design specification documents that were difficult to maintain and would quickly become out of date. In agile methods, high quality working software is valued above all else since that is what is being delivered to customers (Cockburn, 2007). Agile methods like XP strive to minimize documentation to only what is absolutely necessary through practices like onsite customers and close collaboration between team members.

Usability engineering methods like scenario-based design (SBD) would appear to work against this principle as they use a variety of different design artifacts to develop the UI interface before any code is written. In SBD, a variety of different types of scenarios are developed to describe current work practices and the system being developed (Rosson & Carroll, 2002).

In addition, a variety of low and high-fidelity prototyping techniques such as sketches, storyboards and click-through mockups, are used to design and evaluate the interface before it is implemented. These types of prototypes are typically used in formative usability evaluations to get early feedback on designs from users and other stakeholders before implementation begins. Usability engineering methods such as sketches and storyboards are a quick way to prototype designs within an iteration, such practices are increasingly being used in agile usability teams (Nodder & Nielsen, 2008). However more detailed artifacts and practices such as high fidelity prototyping and user modeling are difficult to do within an incremental development cycle.

4.2.3.2 Phased vs incremental approaches

Although both agile methods and Usability engineering methods follow cyclical development processes, they differ in what work goes into each of those cycles and how fast those cycles generally go. Agile methods tend to use incremental development cycles where during each iteration some piece of functionality is designed, implemented and tested. This allows customers to give feedback on the system early and validate that it does what they want it to do.

Usability Engineering methods tend to follow a more ‘layered’ or iterative approach where the requirements are first defined and the system is then completely defined at increasing levels of fidelity. By better understanding the user and the context of use, designers can look at things more broadly and deliver a more cohesive UI design. This divergence point directly relates to the issues of whether the approach should be agile-centric or usability-centric.

Cooper argues that approaches like XP are too development-centric and that developers do not naturally design code to meet end user needs (Cooper, 2004). Cooper and others argues for a ’usability-first’ approach where usability professionals first interact with customers to collect data from end users and develop the UI design before working with developers to implement it using a traditional agile approach (Obendorf & Finck, 2008). Beck counters that such an approach represents ‘Big Up Front Design’ and runs counter to agile practice of continuous development and feedback and would be a waste of resources as developers would have to wait until the UI design was ready.

(Patton, 2002) has advocated a more agile-centric approach where existing agile teams learn about and integrate Usability engineering practices into their day to day tasks. Patton as well as (Meszaros & Aston, 2006) have reported on projects where developers have used some user-centered techniques as they developed systems using an agile approach.

One potential problem with this approach is that effective UI design can be difficult for complex systems with a heavy emphasis on usability. They may require a level of expertise and knowledge that cannot be learned in a short period of time. User interface design and evaluation is not a simple endeavor for systems with a large UI component. Moreover, certain usability tasks are complex and time-consuming and may not be easily handled by developers who also have to implement features.

4.2.3.3 Test driven development vs usability evaluations

Test driven development is one of the most common agile development practices. Agile developers continuously create automated unit tests that define what the software is supposed to do before writing the code itself (Beck, 2004). This practice allows developers to incrementally develop the system while ensuring that the code base remains robust even as it is evolved and refactored. It also allows developers to identify design flaws sooner, discover problems in the requirements and diagnose and fix problems in the code more quickly. In addition, these test suites can be run automatically on a daily basis.

Usability testing does not have the benefit of the level of automation that test driven development supports as they require human intervention both in the sense that they rely on feedback from actual users and they often depend on a usability expert to analyze and interpret that feedback. As a result, usability processes tend to take a less nimble approach than agile methods by focusing more on gathering up-front data beforehand and doing early lightweight prototyping iterations before code development begins. For example, the agile concept of refactoring does not have a clear analog in usability approaches since the impact of making small changes to the user interface can often not be verified until after the next set of usability evaluations are run with users.

Lightweight, or guerilla usability techniques, are commonly used in agile usability teams as a way to get usability feedback within an agile framework (Nodder & Nielsen, 2008).These might include analytic techniques such as cognitive walkthroughs or rapid iterative testing and evaluation-where fixes are made as they are found in a study so that subsequent participants

work on continuously improving the system. These techniques allow usability engineers to get feedback from users and provide guidance to the developers quickly which is essential in agile teams. In fact, one potential benefit of this approach is that the system usability can improve more than using a traditional approach since the team is getting feedback from users earlier and more often. However, it is more difficult to run summative in depth usability evaluations within an agile framework since development moves so quickly and since the UI in development is a piecemeal fashion (Nodder & Nielsen, 2008).

4.2.3.4 Shared understanding vs distinct roles

Agile methods lean towards a generalist approach to software development where all of the developers not only have a shared understanding of the design but are equally qualified to work on any part of the system. Benefits of this approach include improved communication between team members and increased flexibility in terms of what works is done by whom. However, usability engineering is a distinct discipline that software developers are typically not trained to do. Usability engineers are needed for projects where usability is a key quality attribute and the user interface design is nontrivial.

Usability engineers are often not trained as skilled software developers (Sillito & Maurer, 2008). The specialist approach allows each group to leverage their own areas of expertise in developing the system; however it requires careful coordination between the different groups to prevent problems such as drift between the UI design and the implementation. Both approaches highlight a potential conflict of interest that can arise between UI design and software development. Given a limited amount of time to complete a feature, the developer is more likely to sacrifice usability to get the code implemented since functioning code is central to agile methods. In addition, it can be difficult for usability engineers and agile developers to work together given their differing focus areas, backgrounds and concerns.

4.2.3.5 Customer focus vs end user focus

Usability engineering methods and agile methods approaches are ‘human-centered’ in that they value close collaboration with stakeholders. However, they differ in which stakeholders they focus on most. One of the core concepts of agile development is continuous and close collaboration with customers. In XP, there is an on-site customer who joins the team and works

with them throughout the development of the product to define requirements, do acceptance testing and answer questions as they arise (Beck, 2004). In agile teams, the ultimate goal is to efficiently develop a high quality product that meets the user's needs. Usability engineering methods are user-centered, meaning that the ultimate goal is to develop a high quality system that meets the end users' needs.

Usability engineers use a variety of user-focused techniques such as onsite observations, interviews, participatory design and user testing to understand users and ensure that the system meets their needs (Rosson & Carroll, 2002). For many development projects, the customer and Client Company will not be the ultimate end users of the developed system. This can result in conflict between the usability engineer and the agile developers because of their focus on different stakeholders.

The differing focus points between agile and usability can lead to communication and collaboration problems within agile usability teams. Agile developers and usability engineers can come into conflict when their goals do not align. For example, although simplicity in the design is a characteristic that is valued by both agile and usability practitioners, simplicity in the user interface often does not align with simplicity in the implementation. In addition, usability and agile professionals can have problems understanding or accepting each others' practices and worldviews. For example, usability engineers need to understand that business and technical factors can impact the importance of usability as a quality factor in the system. In addition, agile practitioners need to understand that working customers does not guarantee that the resulting system will be usable for end users.

4.2.4 Approach to Integration

Literature on integration of Usability engineering with software engineering (SE) can be classified as

- Process approaches and
- Non-process approaches.

The non-process-based approaches include work in the area of modifying software architecture patterns to make it more responsive to usability concerns, extending SE artefacts to include usability, creating other boundary objects or techniques between the two disciplines, identifying patterns of integrating Usability engineering activities with SE processes, and activity mapping.

The process-based approaches are proposals that aim at integrating Usability engineering and SE processes. These include new process model proposals, and proposals to integrate Usability engineering activities into existing process models such as the waterfall, agile, and RUP. In our case we deal with the process-based approach.

Our approach to integration is characterized by these five important guiding principles:

1. Integration should not disrupt the core values of the SE process model (for example, the waterfall-ness of waterfall model and the agility of agile models.
2. A truly integrated process should integrate and optimise Usability engineering activities and Usability engineering deliverables in the SE process. It should make Usability engineering activities an explicit part of the SE processes, so that Usability engineering design happens purposefully, and not by default.
3. The process should recognize and support the involvement of multi-disciplinary teams. It should recognize that merely representing all professions in the team is not enough. The process should be a script that tells everyone in the team what to do and when.
4. The process should encourage divergence and transformation of the problem space before converging to a solution. The process should support the team to consider many alternatives for the Usability engineering design before making decisions.
5. In addition to managing change, the process should help contain change by proactively anticipating the reasons for change and then accounting for them through creative design solutions.

4.2.5 The Extended Mobile-D Agile Process Model

In order to obtain a set of improvements to a given development methodology, one must first analyze the key method characteristics that have yielded successful results in previous projects. For mobile application development methods, key success characteristics are identified in (Rahimian & Ramsin, 2008). These are agility of the approach, market consciousness, software product line support, architecture-based development, support for reusability, inclusion of review and learning sessions, and early specification of physical architecture. Some of these key features can already be found in the Mobile-D method; however, the method could be improved if more of usability engineering key features could be integrated into it.

Mobile-D comprises five phases:

1. Explore,

2. Initialize,
3. Productionize,
4. Stabilize, and
5. System Test & Fix

Each of these phases has a number of associated stages, tasks and practices as we found out in Chapter Two.

- Explore phase employs different techniques which we can associate with user stories to effectively capture requirements
- In initialize the release planning is notable the most critical event
- Productionize, involve coding and developing the software
- Stabilize insists on continuous integration of the developed objects in productionize
- System Test and Fix confirms if the outcome conforms to the acceptance test criteria proposed
- The added Evolve relies on user feedback to better enhance the system.

A great summary of the above is depicted in Figure 4.2.

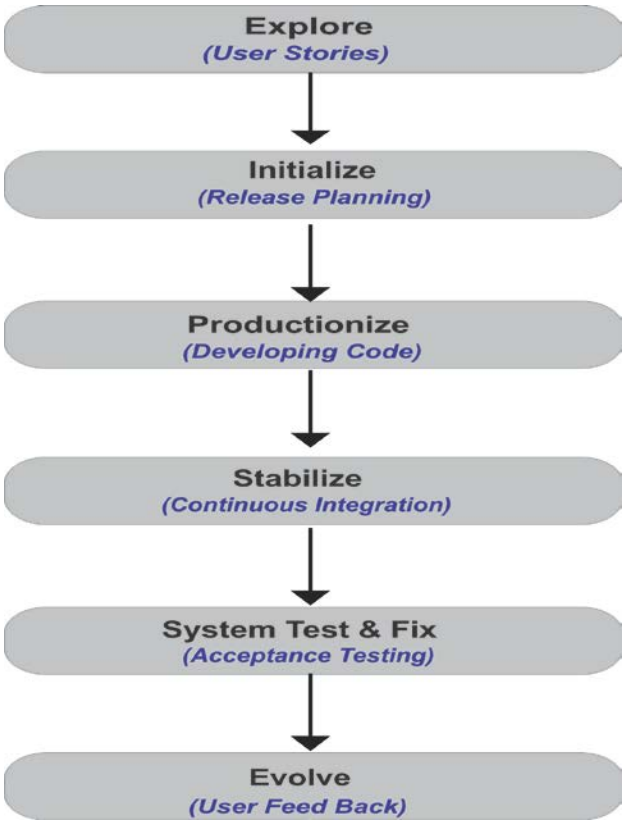

Figure 4:2 Mobile-D associate stages

By using the summarized mobile-D approach in Figure 4:2 and usability activities listed in our framework as the base line. We propose an Extended Mobile-D process model.

We link each mobile-D activity to the 10 phases that we presented in our Framework and by further using some lightweight discount usability practices we identified different possibilities to make mobile devices software development interesting and designer friendly throughout the development process.

The four adaptations we made are:

1. Use of Scenarios along with User stories in Exploration phase
2. Card Sorting as part of Release Planning in initialize phase
3. Usability Heuristic Evaluation during Productionize and Stabilize phase
4. Thinking aloud technique as part System test and fix phase.

4.2.5.1 Explore

Explore means to setup initial characteristics version of the project requirements and establishing the project plan. The main purpose of explore phase is to highlight the scopes and requirements within the project. In agile development user stories are used to capture requirements' and Mobile-D being built on agile processes still uses user stories to fully capture requirements. Mobile-D demonstrates its strengths in iterative software development, where requirements may change as a system is incrementally put into use, the question of how to devise an initial design is largely unanswered. User stories do not fit into fully expressing usability requirements (Jokela & Abrahamsson, 2004). The customer is allowed to refuse a user story, even though there is a successful acceptance test for that story, for instance, if a feature is not usable.

User stories are short narratives which can serve as use cases, describing interaction at a technical level while a scenario is a description of a person's interaction with a system where people who do not have any technical background can understand it. Stories serve to connect even technical tasks to the use context, and worked together with non-programming tasks (e.g. paper prototyping) can shift the general focus of the process away from programming to design.

Many user stories often fell short of improvement.

Scenarios are appropriate whenever you need to describe a system interaction from the user's perspective. Scenarios and user stories are a natural match; the enriched stories serve to connect even technical tasks to the user context. Usability testing with scenarios does not require functioning software. Low fidelity prototypes using paper can yield useful usability data very

inexpensively (Kane, 2003). Together combined, the scenario and story provide a complete picture of the user at the explore stage.

4.2.5.2 Initialize

In Initialize, the development team and all active stakeholders understand the product in development and prepare the key resources necessary for production activities, such as physical, technological, and communications resources.

This phase is divided into three stages: project set-up, initial planning and trial day which are better stated as release planning. Identifying the resources within the project technically and physically is one of the key points of this phase providing the communication channel between the developer and stakeholders is another important point. According to (Bankston, 2003), the planning game in agile methods has two problems, one is that the customer needs help to understand, verbalize, visualize and organize their requirements and second developers have little opportunity to consider how exactly the interface will work because the conversion of requirement to interface is implicitly assumed to take place within the estimation process .

The basic philosophy of release planning is that a project may be quantified by four variables; scope, resources, time, and quality, (Well, 2009). Card sorts are a well established technique for eliciting knowledge from people (Maiden, 2009) by which better external quality can be obtained by involvement of actual end users.

According to (Patton, 2003), Usage centered design works well as a process framework to facilitate requirement gathering, designing and planning sessions. It is valuable to involve users when determining the features to be implemented for a piece of software using card-sorting technique. Card sorting technique with the help of end users as a part of Planning game (Release Planning) in agile process methods can increase the chance for successful usable software.

4.2.5.3 Productionize and Stabilize

The Productionize phase mainly means the implementation of functionalities that are collected within the Explore and Initialize phases of the project.

Stabilize collects and combines iterations together to finalize the product (product finalization). To stabilize the application, one of the vital stages is to integrate all the parts and put them

together as a system, coding and continuous integration of the product actively takes place within this stages.

In Agile methods programming, the customer is to test that the overall system to see if it is functioning as specified by acceptance tests (also known as Functional Tests). When all the acceptance tests pass for a given user story, that story is considered complete. A story can have one or many acceptance tests, whatever it takes to ensure the functionality works, (Well, 2009). However, an acceptance test does not deal with non-functional requirement like usability.

Heuristic evaluation is an approach used by the developers to improve the usability of software by applying a small collection of usability principles to the design and development of the software before testable elements are presented to users (Kane, 2003). Usability Evaluation solves the problem of ad-hoc input (Hussain, et al, 2008).

The simplicity of heuristic evaluation is beneficial to Productionize and Stabilize stages so that improvements can be made as part of the iterative design process Heuristic 2014. It provides some quick and relatively inexpensive feedback to designers. (Sharp, et al, 2008) further suggests heuristic evaluation can be done in each of iterations in agile based programming methodologies.

4.2.5.4 System Test & Fix

System Test & Fix is normally the final phase of Mobile-D agile methodology without the added Evolve phase and it's based on frequently testing the application while completing the documentation of the application. (Kane, 2003) stated that unfortunately, user-interface design and usability are largely overlooked by the agile processes.

Thinking aloud allows you to understand how the user approaches the interface and what considerations the user keeps in mind when using the interface. This testing is essential and preferred in System Test & Fix in Mobile-D agile process where the designer can get quick feedback about their design work. Thinking aloud method can be applied effectively in "Small release" where decision has to be made if some changes occur.

4.2.5.5 Extended Mobile-D with added Evolve phase

The added Evolve phase deals with continuously integrating end-user feedback on the delivered product into future releases.

Feedback can come from multiple sources, such as consumer and peer reviews, or data generated by the application itself (usage statistics and crash reports). The feedback plays an important role in the improvement of future and current developments.

Figure 4:3 best summaries our Extended Mobile-D with evolve phase as shown below.

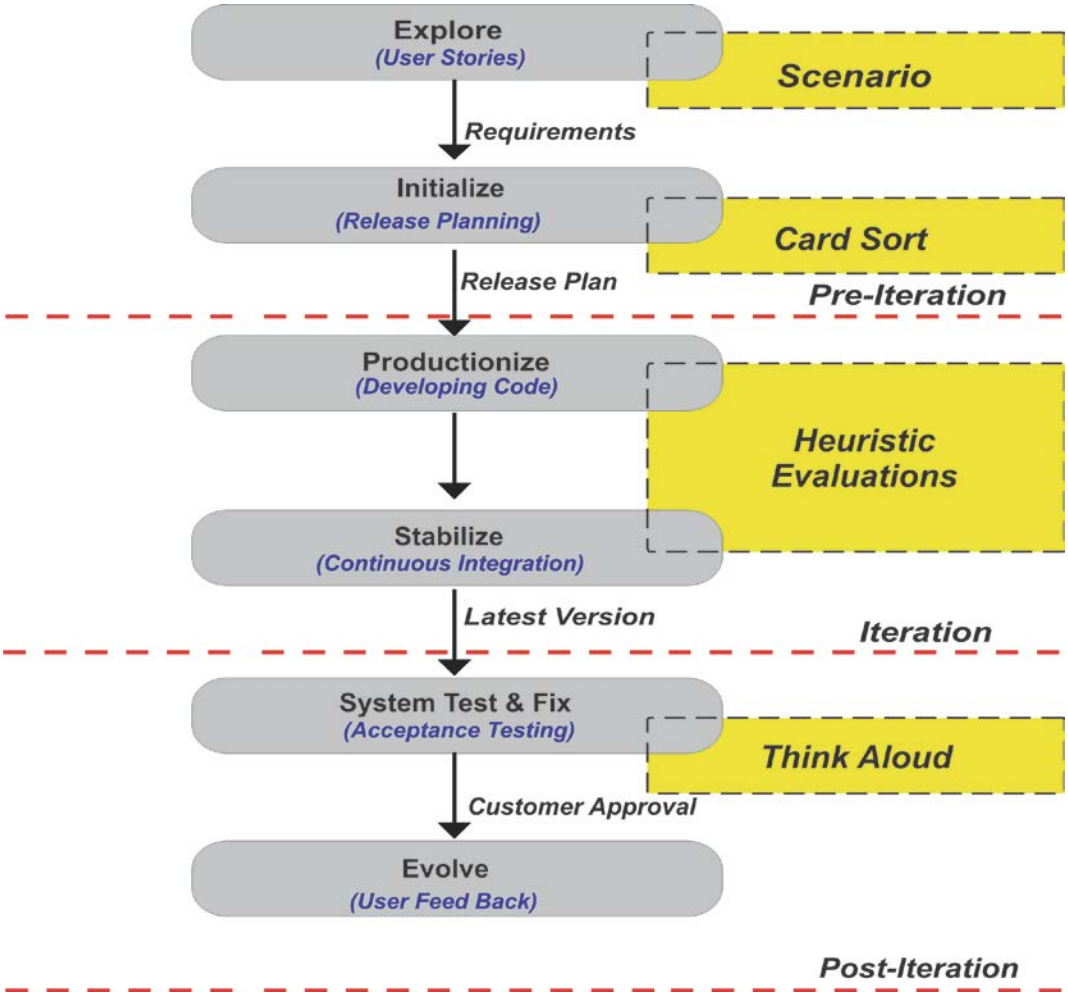

Figure 4:3 Extended Mobile-D with Evolve phase

To fully understand the Extended Mobile-D model within our framework we categorized each activity in the mobile-D agile process in line with our framework and the evolve phase was found to be a perfect match to the maintenance phase.

4.2.5.5.1 Planning

The **Planning phase** of our framework is set to enhance our Extended Mobile-D agile process to be able to fully capture the user requirements.

The planning stage establishes a bird's eye view of the intended software and product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the

project, and describe appropriate management and technical approaches before designing even the first bit of the user interface, the HCI activities of user, market and task analysis, product definition, and usability evaluation must be performed after the user studies are done, usability techniques implementation should be a multi-disciplinary activity where HCI and SE practitioners, business stakeholders, and user representatives are involved. These activities of Question 1 and 2 in our framework are aimed at providing a smoother and a perfect match to the explore phase of the model.

4.2.5.5.2 Analysis

Analysis enhances the intialize stage by further ensuring the development team and all active stakeholders understand the product in development. Question 3 in our framework improves this phase making it include a broader reviewer pool, collaborative inspections, cognitive walkthroughs and usability evaluations ensure product specific requirements are well understood and verified.

4.2.5.5.3 Design

Design takes as its initial input the requirements identified in the explore and initialize phases, for each requirement a set of one or more design elements will be produced as a result of interviews and or workshops. Once a high-level product definition is in place the HCI team should do the detailed design of the interface, create a prototype, and evaluate it with users as in question 4, 5 and 6 of our framework. It is particularly important to do this carefully, as it could set the direction for the rest of the project. Design elements describe the desired software features in detail, and generally can include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudocode, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional inputs.

4.2.5.5.4 Development and Implementation

In development and implementation question 7 of our framework comes in handy to enhance productionoze and stabilize phase as the detailed product is developed to be implemented. Designers explore the finer details of the product such as labels, icons, and behavior of widgets, Design engineers play a major role here.

4.2.5.5.5 Testing

Since agile processes release early and release often, it is also possible for HCI teams to evaluate early and evaluate often, at this point all test cases are run to verify the correctness and completeness of the software. This is normally a summative evaluation as depicted by question 9 of our framework. It may also be possible to influence the development team to fix problems by ongoing interaction. It may not be necessary to do a summative evaluation in every iteration, but it would be very important to do them in the first few iterations as early versions of the products becomes a reality. In longer projects, more evaluations may be needed. It may be a good idea to outsource this evaluation occasionally to a third party. It will act as a checkpoint for the entire team.

4.2.5.5.6 Maintenance

Maintenance is really evolutionary development and maintenance decisions are aided by understanding what happens to systems (and software) over time.

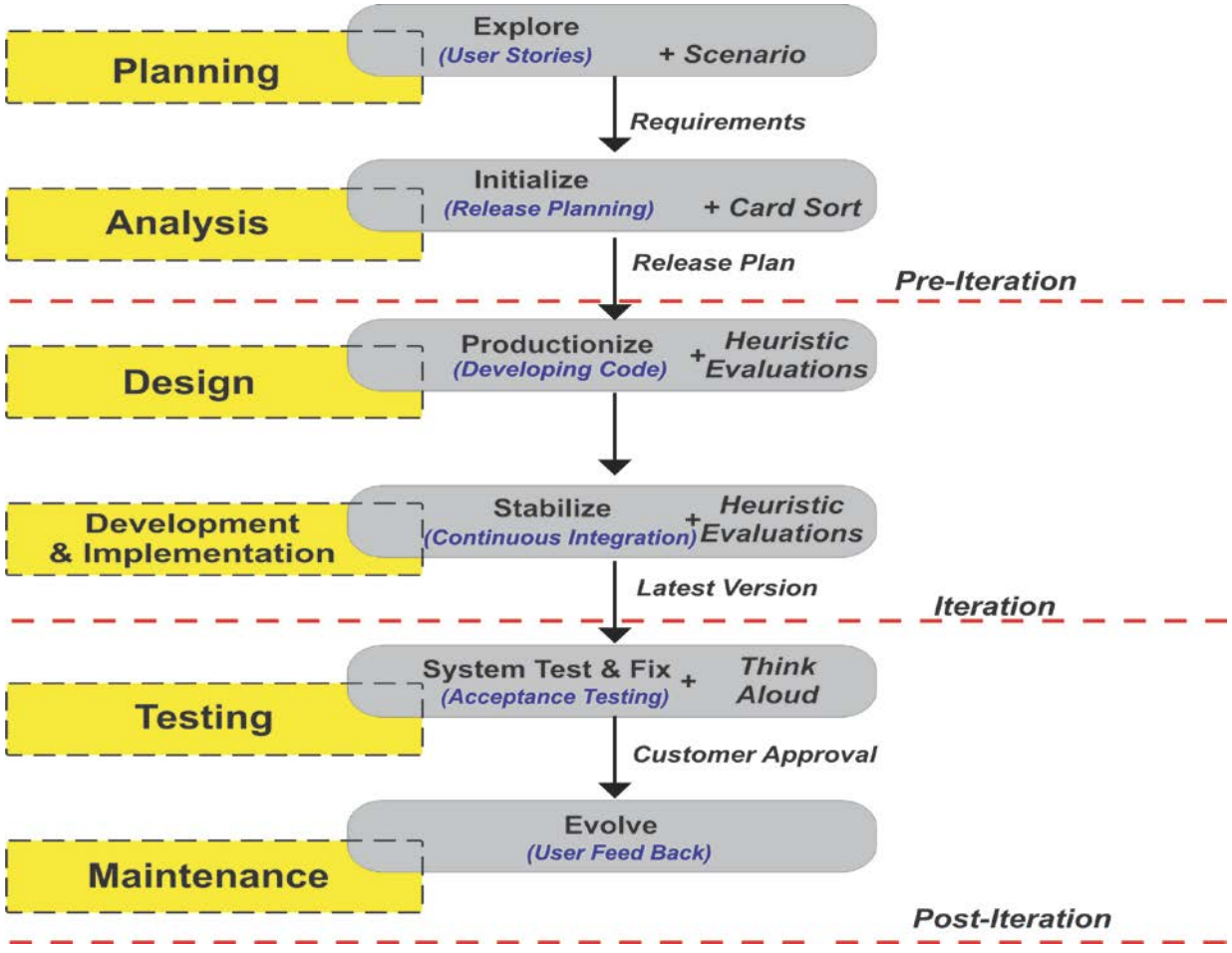

Figure 4:4 Extended Mobile-D model in line with Multidisciplinary Framework

Question 10 tries to make this phase as simple as possible to perform. Figure 4:4 best indicates the integration of the Extended Mobile-D model in line with our multidisciplinary framework.

4.3 Evaluation of Effectiveness of Integration of Discount Usability into Software Engineering

Having proposed a process framework and used it as a baseline for integrating the essential discount usability techniques into Mobile-D model the research question we were dealing with was,

- Is our model efficient? “How can we prove that our process model is working and consistently is leading to quality usable products?”

Usability evaluations of systems and systems design models are important part of the overall development activity. Evaluation is concerned with gathering data about the usability of a design or product by a specified group of users for a particular activity within a specified environment or work context. The main activities involved in an evaluation include:

- Capture: collecting usability data, such as task completion time, errors, guideline violations and subjective ratings;
- Analysis: interpreting usability data.
- Critique: suggest solutions or improvements to mitigate problems.

Usability evaluation is a demanding process

- First because software products vary a lot. They are targeted to different users, to be used in different contexts, with different frequencies, and are deployed on different platforms.
- Secondly, design and development processes vary a lot.

Teams follow different software development processes and follow them to a different extent. Further, skills, creativity, knowledge, and experience of the team members vary. All this affects the outcomes of the project and makes it difficult to measure the quality of the process.

4.3.1 Usability Factors for Software Engineering Methodology

When end users i.e. project managers, software engineers, or organizational executives choose or adopt a software engineering methodology, successful adoption process occurs in following steps Zafar et al, (2014)

1. Understanding

2. Learning
3. Applying
4. Effectiveness/Usefulness for future projects
5. Satisfaction of End Users.

The first step is to understand the methodology to answer the question “what to do?” Understanding involves the description and elaboration of concepts about methodology and its elements i.e. process, product, people, method(s) etc so that the user can understand easily.

Understanding is required for the clarity of concepts and to know about the philosophy and process of the methodology to solve the problems.

After understanding the next step is to learn the methods, techniques, modeling language, implementation language, and tools to answer the question “how to do?”

Learning solves the problem of effort required to acquire technical skills by using the methods, techniques, tools, utilities etc. Learning is required to develop the essential skills required to complete the tasks and activities in order to achieve desired milestones.

After learning the next question is “how to apply?” the methodology on organizational small, medium and large projects.

Applicability of methodology is concerned with acquiring required resources, establishing the software development environment, and organizational culture. How much convenient it is to establish development environment, organizational infrastructure, and to apply the methodology on organizational projects.

After successfully applying the methodology the next question is “how useful?” the methodology is to solve real problems and to develop future projects for the organization.

Effectiveness or Usefulness is the answer and can be evaluated by methodology completeness, coverage of phases, strength capability and expressiveness of modeling language and implementation technology, efficient utilization of resources i.e. time, people, money etc. How much effective or useful the methodology is for the organization to develop current and future projects.

The last thing that can be evaluated for the usability or usefulness of a methodology is End user satisfaction like analysts, managers, software engineers etc. After understanding, learning, and applying the methodology how much of the end user is satisfied during and after applying the methodology is what is key here and is s/he willing to apply the methodology in future projects.

To empirically evaluate the value of a specific technique, it would be necessary to evaluate the same project repeated under conditions employing the technique verses not employing the technique, while controlling for skill, motivation, SE approach, and other possible differences between the two teams. Further, this challenging experiment would have to be repeated with different project teams, different software engineering frameworks, and on different projects in order for the results to achieve statistical validity.

Assuming that n = 15 projects would give us the statistical validity required, and assuming that each project would have 10 control conditions, and further assuming that on an average, it costs 100,000 to do the project once, the budget of such an experiment would be in excess of ` 1,500,000 quite clearly well beyond the scope and budget of our research.

Then how can one tell whether a process model helps a team achieve its goals, and whether it consistently leads to usable products? This brought us into the area of usability measurement tools.

Assume that it is possible to express numerically the extent to which a process model is followed; further, assume that it is possible to express numerically the extent to which a team achieves its product goal by following a prescribed process model to the extent X a project could achieve its goals to the extent Y.

if we can demonstrate that for every X2 that is greater than X1on the X axis, Y2 is greater than Y1 in most cases on the Y axis, we can conclude that the process model in question works.

4.3.2 Findings

Primary data was collected between September and November 2014 using a questionnaire. One hundred and ten (110) questionnaires were issued to randomly selected Mobile applications software engineers in Industry and institutions of higher learning. Eighty nine (89) questionnaires were returned representing an 81% response rate.

The response rate is considered adequate given the recommendations by (Saunders, Lewis & Thornhill, 2007) who suggest a 30-40% response, (Mugenda & Mugenda, 2003) advise on response rates exceeding 50% and (Hager, Wilson, Pollack & Rooney, 2003) recommend 50%. Based on these assertions, this implies that the response rate for this study was adequate.

4.3.2.1 Sample demographics

In analyzing the demographic characteristics of the respondents the following items were considered; respondents' gender, age, number of years they have in the software development

industry, position in the software development team, type of training they have on software development and participation in software development projects.

4.3.2.1.1 General Information

This study was done with participants who had experience and formal background in Information Communication Technology (ICT). Many participants had an aptitude for design and most of them had formal ICT education.

Participants came from mixed educational backgrounds such as Computer Science 26.97% Information Technology 53.93% Electrical and Electronics 12.36% and other related disciplines 6.74%. The software development industry is still a male dominated field. Out of the 89 questionnaires we analyzed 73 of the participants were of the male gender representing a percentage of 82.02% with only 16 candidates being of the female gender standing for 17.98%.

Majority of the respondents were aged below 40 years with most 52.81% of them being in the age group of between 18 to 28 years, 38.20% fell within the age of between 29 and 39 years. 6.74% were between 40 to 50 years and the rest of respondents 2.25% were over 50 years of age. A significant majority of the respondents were in the youth bracket which is mainly between the ages of 18 to 35 years.

This is in line with the common belief that youths are more in technology than any other age group. Industry experience of participants varied between 1-7 years and above. 85.39% (n=76) of the respondents had worked in the software development sector for four years and above, 14.61% had worked for three years and less. This finding suggests that majority 85.39%of the respondents joined the sector after year 2010 which is in line with the growth experienced in the past few years in the sector.

The results also indicate a stable and a sticky Information Communication and Technology job environment which shows that more youths have employments in this field and that ICT has created jobs in the country. Data show that majority of the respondents 57.30% worked as programmers 20.23% were mainly software testers 13.48% were project managers and only 8.99% were within other related ICT disciplines.

This kind of distribution could have been influenced by the fact that programming or developing code and testing it, are normally viewed as the key areas that champion innovations in the ICT set up. Table 4:2 details a summary of the general information.

Table 4: 2 Summary of general Information

1)Total Respondents According To Gender

Gender	Frequency	Percent
Male	73	82.02
Female	16	17.98
TOTAL	**89**	**100.00**

2) Total of Respondents according to Age

Age	Frequency	Percent
Below 18 years	0	0.00
18-28 years	47	52.81
29-39 years	34	38.20
40-50 years	6	6.74
51 years and above	2	2.25
TOTAL	**89**	**100.00**

3) Total of Respondents according to number of years they have worked in the software development industry

Levels	Frequency	Percent
Below 1 year	0	0.00
1-3 years	13	14.61
4-6 years	42	47.19
7 years and above	34	38.20
TOTAL	**89**	**100.00**

4) Total of Respondents according to position in the software development team

Position	Frequency	Percent
Programmer	51	57.30
Project manager	12	13.48
Software tester	18	20.23
Others	8	8.99
TOTAL	**89**	**100.00**

5) Total of Respondents according to formal training received in software development or related disciple

Training type	Frequency	Percent
Formal	89	100.00
Non-Formal	0	00.00
TOTAL	**89**	**100.00**

6) Total of Respondents according to scientific field of formal training

Field	Frequency	Percent
Computer Science	24	26.97
Information Technology	48	53.93
Electrical and Electronics	11	12.36
Others	6	6.74
TOTAL	**89**	**100.00**

4.3.2.2 Summary of products development information

4.3.2.2.1 Products Development Information

Investments in software development have increased recently since software is capable of having a positive impact on the economic development front. This in turn has lead to more demand for more software hence the need for more projects to be development

The results in Table 4:3 shows the product development experience of (n=89). 50.56% had succefully developed and delivered to the industry 5 projects and more out of this 15.73% had more than 11 projects under their belt while 49.44% reported to have succefully built less than 5 projects. These figures are a reflection that Software offered an opportunity for industries to improve their incomes and hence better return on assets. To date many organizations have embraced the use of software.

Table 4: 3 Totals of respondents according to software development

1) Total of Respondents according to Software systems they have participated in developing

Software systems	Frequency	Percent
None	0	0.00
1-5	44	49.44
5-10	31	34.83
11 and above	14	15.73
TOTAL	**89**	**100.00**

Our respondents recorded to have participated in the development of different products ranging from mobile based applications, management information systems, security mitigation software and others, on asking about the version on the best product they have ever developed (n=46) 51.69% said the first version of their best product was the best ever rated, 28.09% had to refine their product into version two, 13.48% ticked version three and 6.74% talked about version four.

The developed software's had different work environments and on querying about what is the work place of the best product they had developed 73.03% said their best ever product was working in a business critical environment, 15.73% had their best product in the life critical environments, 7.87 % had theirs in the learning environments and only 3.37% reported their best ever product to be in the gaming industries.

Only 8.99% involved HCI practitioners during their software development process, 91.01% developed their software without any involvement of any HCI personnel this can be attributed to the fact that some process models pay little attention to users, usability, and HCI design. In fact, in agile process models agile teams primarily consist of software engineers, and working code is considered the primary deliverable. Anyone who does not deliver code (e.g. a designer) does not easily fit in culturally and hence there is no need to have them onboard.

4.3.2.3 Summary of model survey

4.3.2.3.1 Model Survey

A. Understandability

Software developers need to read and understand development models, platforms and other software artifacts. The increase in size and complexity of software drastically affects several quality attributes, especially understandability. False interpretation of development guide lines often leads to ambiguities, misunderstanding and hence to faulty development results. Despite the fact that development models understandability is vital and one of the most significant components of the software development process, development models understandability is often assumed to be clear to the developers in most cases.

IT industry schedules are often tightly restricted because of the consumer pressure and misinterpretation of a process increases the potential for defects, leading to problems with the software that include incomplete design, poor quality, high maintenance cost and also the risk of loosing customer satisfaction.

Software needs to be modified necessarily, (Sommerville, 2011) this process of modification or maintenance is usually carried out by programmers, whom may not have developed that software and they need to read and understand the development process, source programs and other relevant documents. Even for the developers of the system, after a gap of few years, it may not

be an easy task for them as they themselves might have forgotten the intricacies of the software. False interpretations can lead to misunderstandings and to faulty development results, without an understanding and the ability to articulate the processes in use it is likely that they will not be effective. Therefore, understandability of the development process has a lot of influence on the factors that directly or indirectly affect software quality.

75.28% agreed that our design objectives and steps are clearly stated in each section, out of this 22.47% strongly agreed this implies that most respondents found the objectives to be clear and simple for the current software development industry Table 4:4.

Table 4: 4 Summary of Understandability

(Area rated (Understandability)	% SD	% D	% N	% A	% SA	mean
a. The design objective and steps are clearly stated in each section	0.00	2.25	22.47	52.81	22.47	3.96
b. The structure keeps me focused on what is to be designed	0.00	4.49	20.22	53.93	21.35	3.92
c. The ordering of steps and sequences is logical	0.00	0.00	13.48	59.55	26.97	4.13
d. I found the various methods of the model well intergraded	0.00	3.37	19.10	55.06	22.47	3.97
e. My interaction with it is clear and understandable	2.25	10.11	15.73	49.44	23.60	3.85
Grand Mean						**3.92**

53.93% agreed that our structure keeps them focused on what is to be designed and when a team is focused on a goal chances of deviating from the original plan are slim plus our model comes in handy to ensure the focus is on the end product. 86.52% said the ordering of the steps and sequences were logical and the high score is attributed to the fact we did not change the software process we were enhancing but only integrated additional usability features into it, thus designers found the steps and sequences familiar.

On integration a mean 3:97 was recorded implying most of our respondents found the usability features integrated into software engineering a perfect match. The significance of understandability is very obvious that can be perceived as ‘If we can't learn something, we won't understand it. If we can't understand something, we can't use it - at least not well enough to avoid creating a money pit. 73.04% reported to have found their interaction with our model to be clear and understandable, thus understandability of the processes was not a major challenge. Understandability of software documents is very important; we can not develop and make changes to a system if we don not understand it development cycle.

Despite the fact that understandability is vital and highly significant to the software development process, it is poorly managed, (Aggarwal, et al, 2003). Researchers and Practitioners advocated that understandability aspect of software is highly desirable and significant for developing quality software. Literature survey reveals that there are various aspects of software, including understandability factor that either directly or indirectly influence testability of software, (Jimenez, et al, 2005). Aforementioned facts reveal that understandability is a key factor to testability.

Software systems tend to depart more and more from the principle of simplicity and become increasingly complex. The increase in size and complexity of software drastically affects several quality attributes, especially understandability and maintainability. Software developers and maintainers need to read and understand several documents of software and keeping them as simple as possible in this industry is the key to quality.

B. Learnability

The learnability of a process is based on comprehensibility if you can not understand it you can not learn it and vice versa. When we talk about the learnability of a process, we are generally discussing how hard it is for a user to learn how to use it. 21.35% strongly agreed that they can learn to use our model quickly and this might be attributed to the simplicity of our model plus the fact that most of the respondents had some background in computing if not fully engaged in the software development industry Table 4:5.

Skills are desired in the computing industry and the good the skills the better the product, 52.81% agreed they would quickly become skillful with our model while 25.84% strongly agreed that they could easily remember to use the model. Learn ability isn’t about teaching users

how to use your product. It's about making them not have to think or at least doing your best to prevent the amount of mental mind-work that they're required to do. People don't like to think after all when using something that's new. Thinking too much is what creates rejected systems and if the above percentage says they can easily remember to use our model then the amount of mental mind-work required to use the model is desirable.

Table 4: 5 Summary of Learnability

(Area rated (Learn ability)	% SD	% D	% N	% A	% SA	mean
a. I can learn to use it quickly	4.49	6.74	19.10	48.31	21.35	3.75
b. I can quickly become skillful with it	2.25	3.37	17.98	52.81	23.60	3.92
c. I can easily remember to use it	6.74	10.11	22.47	34.83	25.84	3.63
d. The data grouping is reasonable for easy learning	0.00	2.25	11.24	55.06	31.46	4.16
e. Learning to use it is easy	0.00	5.62	15.73	52.81	25.84	3.99
f. I think that I would need the support of a technical person to be able to use this model	21.35	32.58	23.60	13.48	8.99	2.56
Grand Mean						**3.67**

86.52% concluded that the data grouping was reasonable and none complex, out of this 31.46% strongly agreed this made the process of learning to use our model an easy as depicted by the results of question e, which also recorded a mean of 3.99 in a scale of 0 - 5. 53.93% indicated they would not need the services of a technical person in order to be using this model and this is because of the multidisciplinary nature of the model where several experts work together to achieve a common goal.

The learnability of modern devices is still very immature for this reason; it is easy to understand that modern technology is not taken in use. Moreover, new technological devices very often cause human beings to feel frustration, anger, panic, chaos and fatigue and consequently, their resistance to using technological devices is understandable. Harmfully, people often experience unpleasant emotions when they interact with a technological device for the first time. As a result, they may never purchase the same product again or they may never return to using anything from the same product family by gaining a deeper understanding of the phenomenon and process of learnability, we can design products and services that are much better, (Mika, 2007).

C. Applicability

Applicability of a methodology is concerned with acquiring required resources, establishing the software development environment, and to apply the methodology on organizational projects. Lately the number of organizations adopting agile practices and concepts is increasing. This increase consists not only of more small teams developing simple applications, but also of large teams successfully developing complex systems, (Highsmith, 2002). This is a surprise, because initially agile development was considered suitable only for small organizations producing simple applications (Boehm, 2003). Empirical evidence has shown that embracing agile practices yields many benefits, (Barnett, 2006), (Law & Charron, 2005), (Schatz & Abdelshafi, 2005), (Barnett & Schwaber, 2004), (Kuppuswami, 2003), (Williams, 2000), including:

- Early return on investment, Short time to market, Improved quality, Enhanced client relationships, Better team morale. Table 4:6 gives our findings on applicability of our model

Our model was simple to use, designer friendly and flexible with all of the three key questions recording a mean of above 3.50 from the respondents. 66.29% affirmed that the model provides the clearest steps possible to accomplish what they would wish do with it confirming that the steps were unambiguous.

The Extended mobile-D model can easily be used without written instructions as 61.8% agreed with 19.10% out of the 61.8% strongly agreeing that they could easily use it without written instructions and this is because it is not a completely new development but just a slight improvement kept more simple and clear for enhancement of software products development.

Table 4: 6 Summary of Applicability

(Area rated (Applicability)	% SD	% D	% N	% A	% SA	mean
a. It is simple to use	1.12	2.25	21.35	55.06	20.22	3.91
b. It is designer friendly	0.00	1.12	22.47	51.69	17.98	3.66
c. It is flexible	2.25	5.62	19.10	50.56	22.47	3.85
d. It provides the clearest steps possible to accomplish what I would wish do with it	1.12	4.49	28.09	49.44	16.85	3.76
e. I can use it without written instructions	8.99	12.36	16.85	42.70	19.10	3.51
f. I don't notice any inconsistencies	0.00	0.00	30.34	43.82	25.84	3.96
g. Using it I can recover from mistakes quickly and easily	4.49	7.87	26.97	46.07	14.61	3.58
h. It is easy to apply it in designing of a software system	0.00	0.00	10.11	60.67	29.21	4.19
Grand Mean						**3.80**

The respondents stated that they did not notice inconsistencies with the model with 69.66% concluding that they don't notice such this can be attributed to the fact that we did alter during our improvement process the consistence of the parent model we were enhancing.

Mistakes are bound to happen in any development situation and how to handle them is normally the critical issue, a mean of 3:58 computed implies that our model can assist the developers recover easily from mistakes they have done in the design process and this was facilitated by the

fact that our steps are brief and clear. 60.67% agreed that our model is easily applicable to the software development industry with 29.21% strongly agreeing to this.

D. Usefulness

Is it easy to use? As important as that question is, there's one that's more important: Is it useful?

Table 4: 7 Summary of Usefulness

(Area rated (Usefulness)	% SD	% D	% N	% A	% SA	Mean
a. It can help me be more effective	0.00	4.49	21.35	51.69	22.47	3.92
b. It can help me be more productive	0.00	6.74	24.72	47.19	21.35	3.82
c. It can makes the things I want to accomplish easier to get done	0.00	5.62	30.33	48.31	15.73	3.74
d. Using it would save me time	0.00	3.37	23.60	43.82	29.21	3.99
e. It meets a designers needs	0.00	4.49	32.58	42.70	20.22	3.79
f. I find it useful in my job	1.12	3.37	31.46	41.57	22.47	3.81
g. It gives me a clear insight of specific activities per stage	0.00	0.00	19.10	55.06	25.84	4.07
h. Using it would lead to better products	0.00	0.00	24.72	53.93	21.35	3.97
Grand Mean						**3.89**

First and foremost, a product, website, application or development model should solve a problem, fill a need or offer something people find useful. In fact, people are willing to put up with poor usability if a product delivers something of great perceived value. Consider how much

time you would spend learning to use software if you knew you'd have a guaranteed way to double investments in the stock market? Conversely, it doesn't matter how easy to use a product is if people don't find it useful. Usefulness is the holy-grail of product design, it's often even more important than revenue. Table 4:7 summaries our finding on usefulness of our model.

When you are effective most likely you will be productive, 74.16% said our model would help them be more effective and similarly 68.54% expected the model to enhance their productivity. 64.04% confirmed that the model makes the things they wish to accomplish easier to get done and this would definitely reduces the development time of a product with other development factors kept at normal levels and thus 73.03% reported that using the model would save them time.

The cyclic nature of our model contributed to the high score (62.92%) on meetings designers needs as the model provides continuous internal loops that allow for reinterpretations and redesign of a product. This is a very useful component in a multidisciplinary development context hence more designers found our model to useful in their jobs recording a mean 3.81 when asked if they found our model to be useful. 25.84% strongly agreed that the model gave them a clear insight in to specific activities per stage with 55.06% agreeing to this. The respondents summarized this section with 75.28% alluding to the fact that using this model would lead to better products.

E. Satisfaction

When we have a great food experience at a new restaurant, we usually want to go back. Positive evaluations result in greater customer satisfaction, which leads to customer loyalty and product repurchase. Customer satisfaction is influenced by perceived quality of product and service attribute. Question A to C focused on the simplicity of development brought by our model and the respondents' attitude towards the model was impressive as shown in table 4:8 below.

In questions D to J we focused on overall satisfaction measure (Emotional) this questions reflected on the overall opinion of a consumer's satisfaction experience with the model and 60.67% said using our model was quite fun, 91.01% felt they would wish to have the model, 86.52% would recommend it to a friend and 85.39% felt confident in using the model. Satisfaction can influence other post-experience actions like communicating to others through word of mouth and social networks.

Table 4: 8 Summary of Satisfaction

(Area rated (Satisfaction)	% SD	% D	% N	% A	% SA	Mean
a. Using it would improve my job performance	0.00	2.25	19.10	53.93	24.72	4.01
b. Using it in my job would enable me to accomplish tasks more quickly	0.00	4.49	20.22	49.44	25.84	3.97
c. Using it would make it easier to do my job	0.00	3.37	23.60	50.56	22.47	3.92
d. Using it is fun	2.25	7.87	29.21	41.57	19.10	3.67
e. I feel I need to have it	0.00	0.00	8.99	60.67	30.34	4.21
f. I would recommend it to a friend	0.00	0.00	13.48	62.92	23.60	4.10
g. I feel very confident in using this model	0.00	0.00	14.61	56.18	29.21	4.15
h. I would have no difficulty using the model	4.49	7.87	29.21	41.57	16.85	3.58
i. I think I would like to use this model frequently	0.00	0.00	15.73	52.81	31.46	4.16
j. I am satisfied with it	0.00	0.00	26.97	51.69	21.35	3.94
Grand Mean						**3.97**

Additional post-experience actions might reflect heightened levels of product involvement that in turn result in increased search for the product or information 58.42% would have no difficulty

using the model, 84.27% would like to use it frequently and 73.04% were generally satisfied with it.

4.3.2.3.2 Extended Mobile-D and Task Performance

Assume that it is possible to express numerically the extent to which a process model is followed; further, assume that it is possible to express numerically the extent to which a team achieves its product goal by following a prescribed process model and with that assumption we record that the confidence levels in using our model increased as we moved down the steps, 41.57% strongly disagreed that if they follow our model to step 1 only they would record better results, 30.34% was recorded for step 2 and 23.60% was the data in step 3, Table 4:9 and we did link this to the fact that steps 1 to 3 involved mostly the establishments and refinement of requirements and since no product was visible at this point so the results.

There was a slight change from step 4 onwards, 37.08% would record better results by following our model only up to this stage, 66.29% up to step 5, 67.42% up to step 6, 74.27% up to step 7, 78.65 up to step 8 and this is attributed to the fact that this steps involve the design, redesign and expert evaluations, however there was a decrease from step 9 with a record of 61.79% and 61.8% and we attributed this to poor follow up on maintenance and evolution of products after development.

If the steps are on the X-axis and scores (sd, d, n, a, sa) represent the product on the Y-axis then we comfortable conclude that by following our prescribed process model to the extent X your project would achieve its goals to the extent Y, because we have seen that as you move down the steps the confidence level in having a better product increases, that is people feel more better in step 2 than 1, 3 than 2, 4 than 3,5 than 4,6 than 5, 7 than 6 and 8 than 7 and thus we concluded that for X2 that is greater than X1, Y2 is greater than Y1 in most cases.

A diagrammatic illustration of each of the scores against the steps is as shown from Figure 4:5 to 4:9 the dotted red line depicts the normal.

Table 4: 9 Summary of steps against the scores

Steps	%SD	%D	%N	%A	%SA
1	41.57	44.94	12.36	1.12	0.00
2	30.34	40.45	21.35	7.87	0.00
3	23.60	37.08	26.97	10.11	2.25
4	14.61	16.85	31.46	21.35	15.73
5	6.74	8.99	17.98	28.09	38.20
6	2.25	14.61	15.73	31.46	35.96
7	1.12	7.87	16.85	33.71	40.45
8	0.00	6.74	14.61	37.08	41.57
9	0.00	7.87	30.34	29.21	32.58
10	0.00	5.62	32.58	35.96	25.84

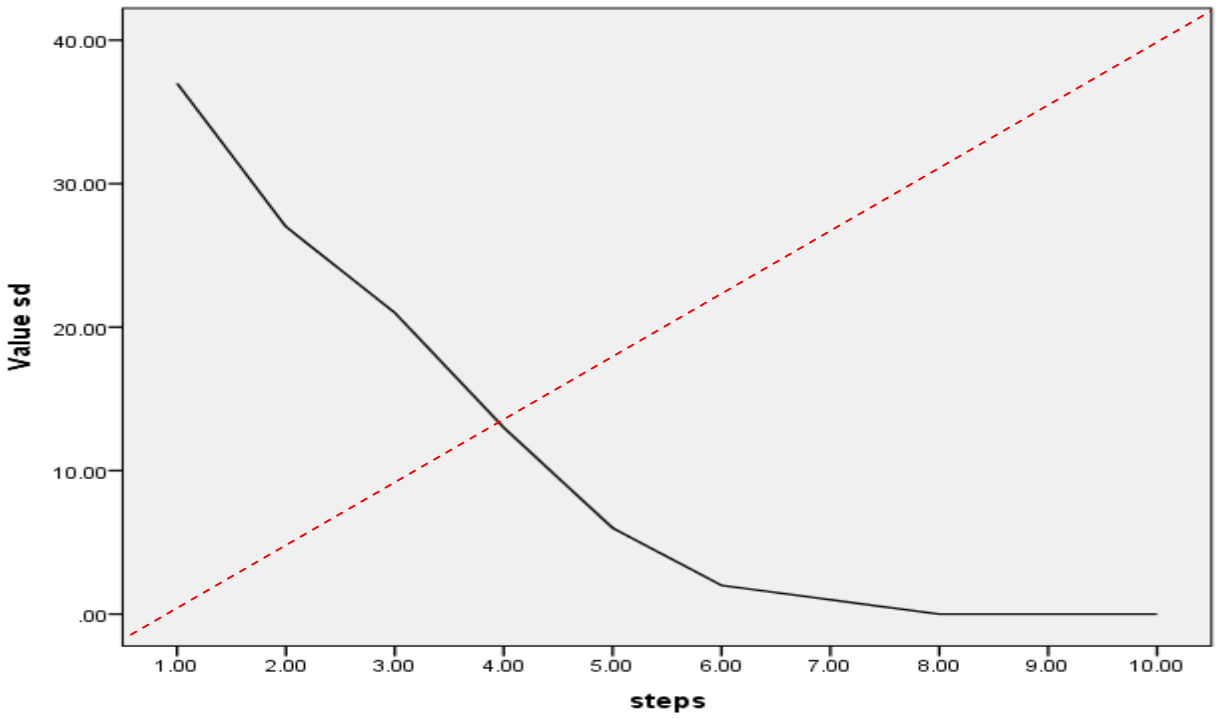

Figure 4:5 X against Y in Sd

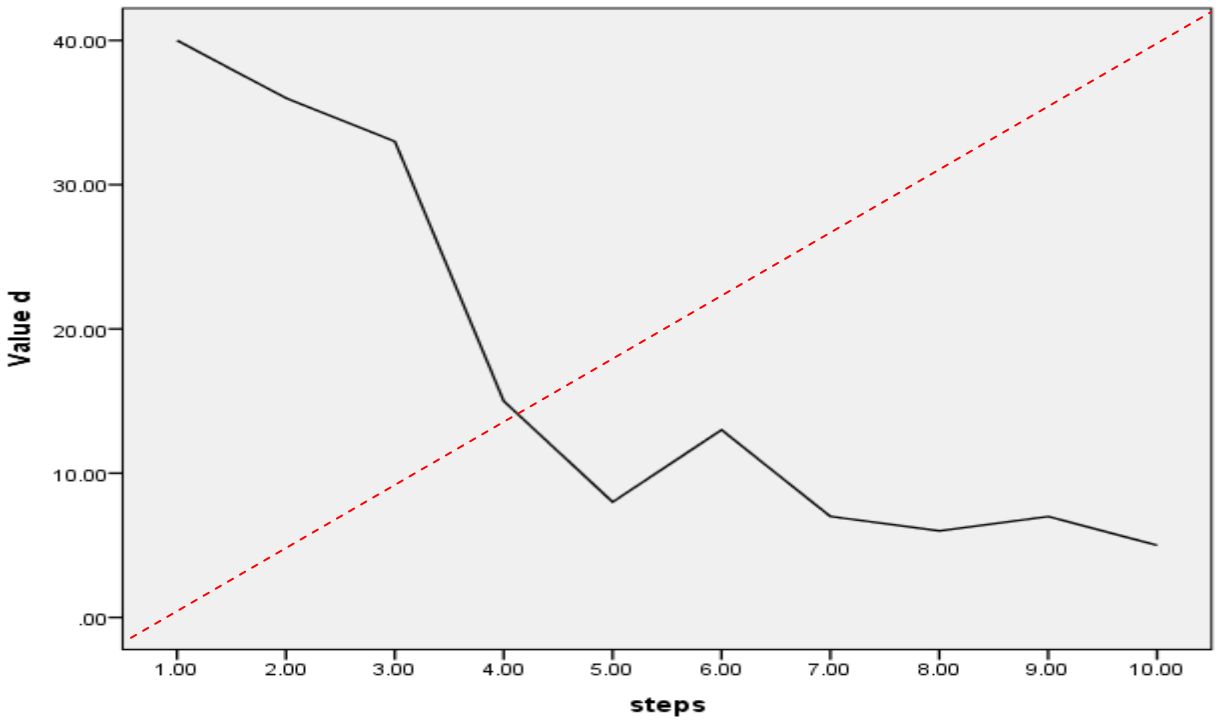

Figure 4:6 X against Y in d

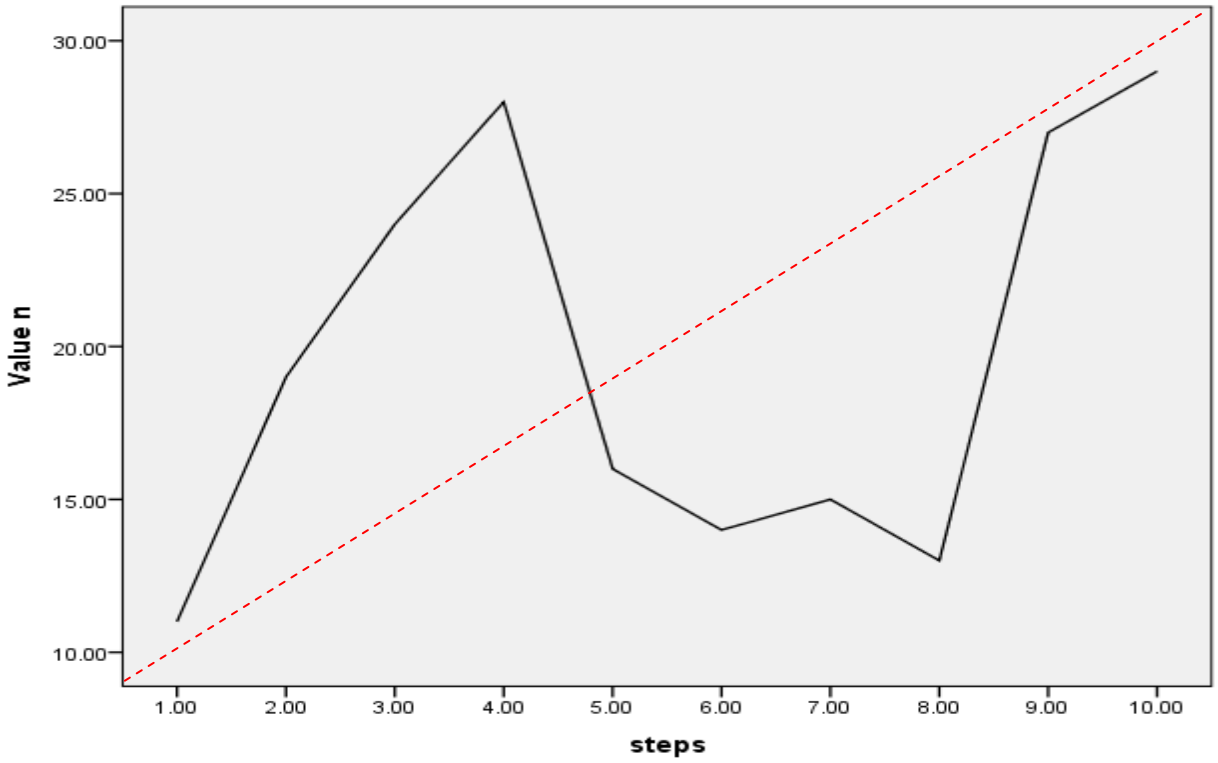

Figure 4:7 X against Y in n

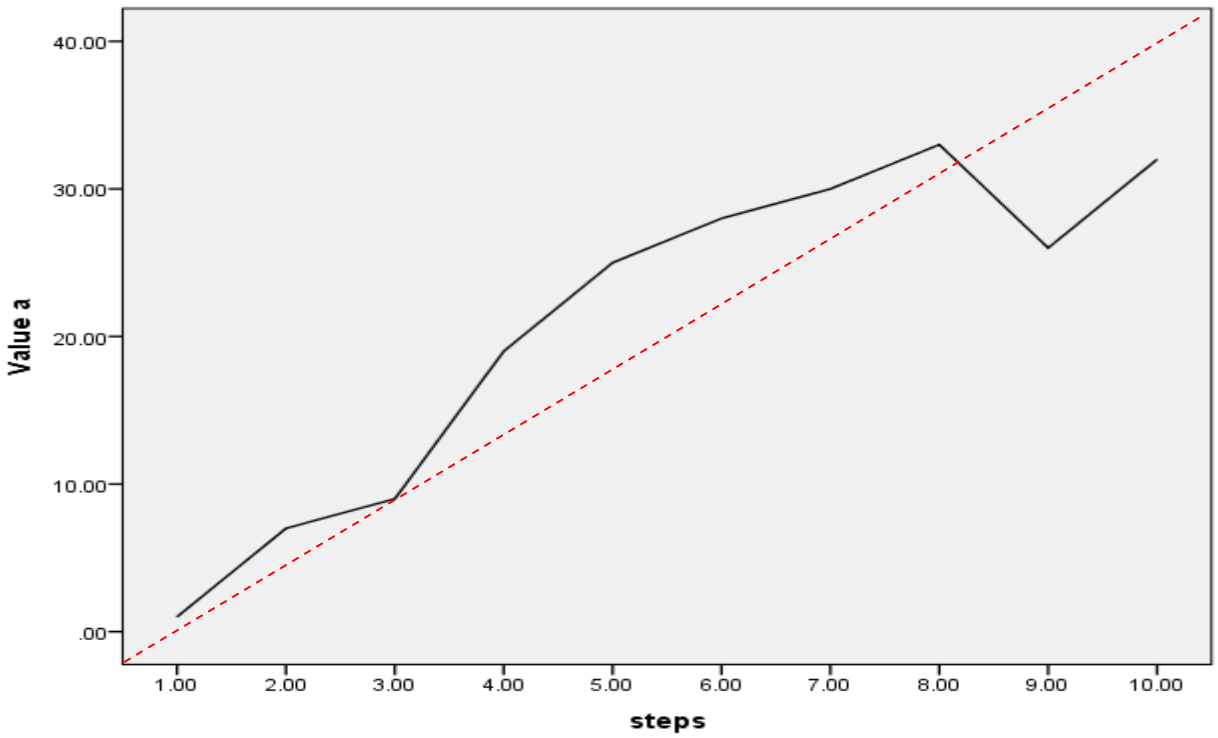

Figure 4:8 X against Y in a

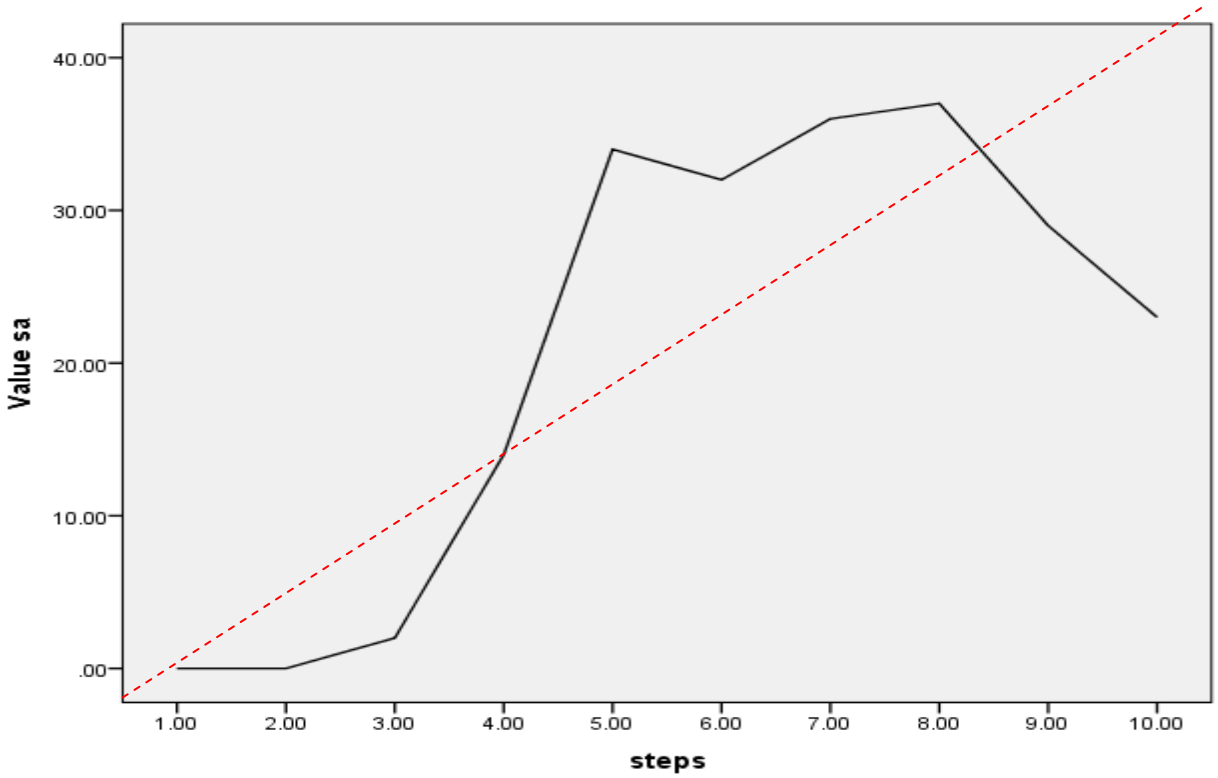

Figure 4:9 X against Y in Sa

4.3.3 Principle-based analytic evaluation of Extended Mobile-D

It would be great to ask people a few questions to see if they would both use and then purchase a product. Asking people about a new product is notoriously unreliable. We can and should be critical of data based on users' predictions of their future behavior from focus groups, surveys or even the most complex statistical analysis.

In this section we include an analytic analysis of the Extended Mobile-D model with respect to key agile and usability values/principles. We will show how the Extended mobile-D model which takes the general approach of integrating usability practices into an agile processes is an agile process by showing how it adheres to the core agile values detailed in the Agile Manifesto.

4.3.3.1 Evaluating whether Extended mobile-D model is an agile process

Agility is not defined by any one or set of specific practices. Certainly, practices such as incremental development, on-site customer collaboration and test driven development are common among practicing agile teams and specific methodologies. However, common ground among agile methods was established through the Agile Manifesto which captures the core values that underlie all agile methods. Thus, in determining whether Extended mobile-D model can be said to be an agile process, one should evaluate whether it adheres to the core values as stated in the manifesto.

The Agile Manifesto is reproduced below Table 4:11

Table 4: 11 Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

These values highlight the central importance of adaptiveness and responsiveness to change that the agile practitioners felt were the most important features of agility, (Cockburn, 2007). A common misconception among developers unfamiliar with agile is that in agile methods practitioners do no planning and generate no documentation, (Ambler, 2007). However, agile teams do have to do those things to work effectively with varying levels of intensity. It is also important for agile methods to focus on the items on the left of each of the four values in order to ensure that a product is efficiently developed that meets customer needs. The four subsections 6.3.1.1 to 6.3.1.4 will describe in detail how Extended mobile-D model is in alignment with each of the core values.

4.3.3.1.1 Individuals and interactions over processes and tools

Agile methods stress the central importance of people in development process and how well they communicate with each other. The abilities and dedication of the agile team members will always trump any particular process or tool that is used. This is in contrast to many past software development efforts where people are treated as resources that can be compared interchangeably with time, (Brooks, 1995). For example, XP, one of more popular agile methods, has a number of practices that depend on this principle. The customer is working with the developers through the entire project and should be available at all times to answer questions, (Beck, 2004). In addition, pair-programming is used as a way for developers to collectively become familiar with the whole system and enhance team-wide communication.

The following practices from Extended mobile-D model illustrate how it adheres to the agile principle of individuals and interactions over processes and tools.

- In Extended mobile-D model, there is a requirement not only for continuous contact with the customer, but also continuous contact with users, (representative) and collaboration among team members (see framework section disciplines involved Chapter Four section 4:1). The end user representative should be someone knowledgeable about the domain in which the system being designed will be used and be familiar with the tasks it will support. This person should be in continuous contact with the team especially usability experts and designers to answer questions and to help coordinate site visits and evaluations with end users.
- One focus of Extended mobile-D model is to increase collaborations and communications with end users (see framework section usability techniques Chapter Four section 4:1). Practices such as observations, interviews and usability evaluations, will help usability

experts and designers to better understand the users and the context in which the system is to be used which in turn will help developers develop a more usable end product.

4.3.3.1.2 Working software over comprehensive documentation

Agile methods value working software over comprehensive documentation because the software as the primary measure of process because the delivered system is the ultimate goal of the development process, (Cockburn, 2007). Agile practitioners and other software developers have noted that documentation can easily become difficult to understand, out of date and can take a significant amount of time to maintain.

Although Extended mobile-D model requires documentation related to the design and evaluation of the system, the focus is always on delivering a functioning, usable end product (see framework section impact Chapter Four section 4:1)

• The primary goal of using Extended mobile-D model is to efficiently develop a usable software system. Thus, for Extended mobile-D model team, the primary measure of progress is not just working software, but working software that meets high-level project goals. The design documentations are necessary too and are needed as they help the team to meet that primary goal.

4.3.3.1.3 Customer collaboration over contract negotiation

Agile methods value customer collaboration over contract negotiation to stress the central importance of customer involvement in the development project. Also implicit in this statement is that simple negotiating a contract at the beginning of the project is insufficient. Within the incremental agile development process, continuous collaboration is needed. Customers are needed to define and refine requirements as they review developed functionality and answer questions and concerns of developers as they come up.

The Extended mobile-D model adheres to and extends this value to include not only customer collaboration but also end user involvement through things such as site visits, interviews and usability evaluations throughout the process. Optimally, the customer representative will be from the client company and will work onsite with the team throughout the development project (see framework, section disciplines involved Chapter Four section 4:1). However, other people can take this role if necessary.

4.3.3.1.4 Responding to change over following a plan

Agile methods accept the fact that requirements and project circumstances will continuously change during the project. Rather than rely on precisely constructed plans and try to adhere to them, agile methods focus on adaptability and the ability to respond to changing requirements and circumstances, (Koch, 2004). This means using flexible, short plans and continuously prioritizing and reviewing requirements and the system is developed. Much of the Extended mobile-D model was developed with this value in mind and the Extended mobile-D model, like all agile methods, is a cyclical process.

The system is incrementally designed and is validated by regularly running usability evaluations to verify that the design is sound and is meeting the high-level design goals (See section 4.7)

4.4 Chapter Summary

In this Chapter sub section 4.1, we defined a multidisciplinary framework in which different Software development disciplines, techniques and outcomes fit together. We further identified Usability Engineering activities that could be considered essential to the framework and for integration with SE process model.

In 4.2, we use the framework as the basis to integrate the Discount Usability Engineering Techniques in to Mobile Agile process model. Look at why should we integrate, the approach to integration and what are the convergence and divergence points between Agile based process models and Usability engineering methods. Based on the details, we proposed extensions to the current Mobile-D based Agile SE process model and presented an Extended Mobile-D model.

In sub section 4.3 we found out the effectiveness of integration of discount usability into software engineering having established a process framework and used it as a baseline for integrating all the essential discount usability techniques into the Extended Mobile-D model. We further described the results of an analytic evaluation of the Extended Mobile-D model whose purpose was to show how it adhered to key agile values.

CHAPTER FIVE

5.0 CONCLUSIONS AND FURTHER WORK

In this Chapter, we synthesize the results of this thesis, and summarize the important findings. We then review and critically analyze the thesis to determine how successful it is at answering the research questions posed in Chapter One, and whether the contributions arising from this work answered the research questions. Finally, we briefly discuss future directions for extending the research carried out in this thesis.

5.1 Findings and Contributions

Agile organizations have begun to develop more user-facing, UI-intensive systems; they have identified the need to find ways to develop more usable systems without sacrificing key benefits of agile methods.

The single greatest predictors of customer satisfaction are the customer experiences that result in attributions of quality. Perceived quality is often measured in one of three contexts:

- Overall quality
- Perceived reliability and
- Extent of customer's needs fulfilled

Our development of the Extende Mobile-D was motivated by the need to find ways to integrate Usability engineering into SE to satisfy quality mobile platform based product development.

We focused on integration of discount usability techniques specific to mobile devices into the core values of SE process model without disrupting the same values. In Chapter Two we discussed agile development for mobile applications a relatively new approach to mobile platform-based devices applications development, presented a lengthy review of the current state-of-the-art in the design of usable mobile platform based devices. Saw Usability engineering Issues with Agile Processes, the unique development challenges for mobile platform based devices and the gaps in industry practice leading us to consider the best way possible to address the challenges for a better mobile platform based devices applications development environment.

To address the challenges we first proposed a process framework in Chapter Four section 4:1 in which different Software development disciplines, techniques and outcomes fit together. In this framework we identified the essential discount usability techniques, methods, deliverables, and

skills relevant to mobile devices software engineering. We divide the framework into phases, each phase consisting of one or more activities and each activity being associated with one or more techniques.

Each method in the framework requires specific skills and could be associated with a particular discipline to address a specific concern in the software development life cycle and each activity undertaken results in specific deliverables. We further identified Usability engineering activities that are essential for integration with the six traditional software engineering process steps and organized these activities in ten phases, which we described in terms of ten questions.

Our framework proved to be a flexible way of understanding and communicating the work of Usability engineers in different contexts, being modeled around a gradual increase in feature additions, a cyclical release and upgrade pattern the framework presents the following important advantages if implemented correctly:

- Higher product quality and improved implementation of functionalities,
- More realistic estimates of time and money,
- Project team works under less pressure,
- Higher quality of work being done.

The cyclic approach involves intensive collaboration between the customer, designers and programmers (multidisciplinary).
In Chapter Four section 4: 2 we used this framework as a baseline for integrating the essential discount usability techniques in to mobile agile process model and presented the Extended Mobile-D process model.

The integration was made simple by the fact that both agile methods and Usability engineering are built on some of the same principles. One of the key similarities is that both acknowledge that system development is a highly complex and dynamic endeavor that is subject to changing requirements and uncertainties that cannot be known in advance. As a result, both agile methods and usability methods follow cyclical development cycles, focus on early and continuous testing and are inherently human-centered.

Literature on integration of Usability engineering with software engineering (SE) is classified as

- Process approaches and

- Non-process approaches.

The non-process-based approaches include work in the area of modifying software architecture patterns to make it more responsive to usability concerns, extending SE artefacts to include usability, creating other boundary objects or techniques between the two disciplines, identifying patterns of integrating Usability engineering activities with SE processes, and activity mapping.

The process-based approaches are proposals that aim at integrating Usability engineering and SE processes. These include new process model proposals, and proposals to integrate Usability engineering activities into existing process models such as the waterfall, agile, and RUP.

In our work we preferred the process-based approach and by using the summarized mobile-D in Chapter Four section 4:2 plus our framework in Chapter Four section 4:1. We presented the Extended Mobile-D process model in which we did link each mobile-D activity to the 10 Usability engineering activities that we identified in our Framework and further using some lightweight discount usability practices we identified different possibilities to make mobile devices software development interesting and designer friendly throughout the development process, the four adaptations we made are:

1. Use of Scenarios along with User stories in Exploration phase
2. Card Sorting as part of Release Planning in initialize phase
3. Usability Heuristic Evaluation during Productionize and Stabilize phase
4. Thinking aloud technique as part System test and fix phase.

Our Extended mobile –D is characterized by these three important principles:

1. It is a model that has integrated Usability engineering into SE without disrupting the core values of the SE process model. (The agility of agile model).
2. The process supports and recognizes the involvement of multi-disciplinary teams in the development process.
3. It encourages divergence and transformation of the problem space before converging to a solution allowing the team to consider many alternatives before making decisions.

Having proposed a process framework and used it as a baseline for integrating all the essential discount usability techniques into Mobile-D model the final research question we were dealing with was,

- Is our model efficacious? “How can we prove that our process model is working and consistently is leading to quality usable products?”

To empirically evaluate the value of a specific technique, it would be necessary to evaluate the same project repeated under conditions employing the technique verses not employing the technique, while controlling for skill, motivation, SE approach, and other possible differences between the two teams. Further, this challenging experiment would have to be repeated with different project teams, different software engineering frameworks, and on different projects in order for the results to achieve statistical validity. Assuming that n = 15 projects would give us the statistical validity required, and assuming that each project would have 10 control conditions, and further assuming that on an average, it costs 100,000 Kenyan shillings to do the project once, the budget of such an experiment would be in excess of ` 1,500,000 Kenyan shillings quite clearly well beyond the scope and budget of our research. Then how we tell whether our process model helps a team achieve its goals, and whether it consistently leads to usable products? This brought us into the area of usability measurement tools.

We did assume that it is possible to express numerically the extent to which a process model is followed and furthered assumed that it is possible to express numerically the extent to which a team achieves its product goal by following a prescribed process model to the extent X a project could achieve its goals to the extent Y. Then if we could demonstrate that for every X2 that is greater than X1 on the X axis, Y2 is greater than Y1 in most cases on the Y axis then we could conclude that the process model in question works.

One hundred and ten (110) questionnaires were issued to randomly selected Mobile applications software engineers in Industry and institutions of higher learning. Eighty nine (89) questionnaires were returned representing an 81% response rate. The response rate was considered adequate given the recommendations by (Saunders, Lewis & Thornhill, 2007) who suggest a 30-40% response, (Mugenda & Mugenda, 2003) advise on response rates exceeding 50% and (Hager, Wilson, Pollack & Rooney, 2003) recommend 50%.

The participants had experience and formal background in Information Communication Technology (ICT). Many having an aptitude for design and most of them had formal ICT education. They came from mixed educational backgrounds such as Computer Science 26.97% Information Technology 53.93% Electrical and Electronics 12.36% and other related disciplines 6.74%. Industry experience of participants varied between 1-7 years and above. 85.39% (n=76) of the respondents had worked in the software development sector for four years and above, 14.61% had worked for three years and less. Majority of the respondents 57.30% worked as programmers 20.23% were mainly software testers 13.48% were project managers and only

8.99% were within other related ICT disciplines. This kind of distribution could have been influenced by the fact that programming or developing code and testing it, are normally viewed as the key areas that champion innovations in the ICT set up, from further statistical analysis in Chapter Four section 4:3 we found that by following our prescribed process model to the extent X your project would possibly achieve its goals to the extent Y, and for every X2 that was greater than X1, Y2 was greater than Y1 in most cases as detailed by data in figure 4:8 (X against Y in a) and figure 4:9 (X against Y in Sa).

The research contributions arising from this thesis have worked towards achieving the specific objectives of this work in line answering the sub-questions associated with them.
Research question 1 was largely answered by presenting the concepts in Chapter Two digging deep into mobile platform based devices applications development Usability engineering Issues with Software Engineering Processes, the unique development challenges and the gaps in industry practice.

Research question 2 was addressed by stating the tool characteristics needed to support the design of secure and usable systems in Chapter Two (agile manifesto 2001); these characteristics were illustrated by developing a multidisplinary process framework in Chapter Four section 4:1.

Research question 3 was answered right from Chapter Four and by using the framework in Chapter Four section 4:1 as a base line for integration we integrated the essential discount usability techniques in to Mobile-D and presented an Extended Mobile-D process model in Chapter Four section 4:2.

Research question 4 is our model efficacious? “How can we prove that our process model is working and consistently is leading to quality usable products?”Was tackled by questionnaire feedback and statistical analysis as delivered in Chapter Four section 4:3

5.2 Future work

In this section, we propose the possible areas for future research. The development and use of the Extended Mobile-D represents an initial contribution to how usability can be integrated into an agile organization which in itself is a complex and multifaceted problem. Future work could include developing tools to support the use of Extended Mobile-D for example, a tool could be developed to integrate it with existing project management tools and additional studies of the Model are encouraged to further evaluate it.

REFERENCES

Abrahamsson, P. (2007). Agile Software Development of Mobile Information Systems. In *Advanced Information Systems* (pp. 1-4). Berlin: Springer.

Abrahamsson, P. (2005). Mobile software development - the business opportunity of today. *Proceedings of the International Conference on Software Development*, (pp. 20-23). Reykjavik, Iceland.

Abrahamsson, P., Hanhineva, A., Hulkko, H., Ihme, T., Jäälinoja, J. and Korkala, M. (2004). Mobile-D: an agile approach for mobile application development. *Conference on Object Oriented Programming Systems Languages and Application; Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications* (pp. 174-175). Vancouver: ACM.

Abrahamsson, P. Warsta, J. (2003). New directions on Agile methods: A comparative analysis, International Conference on Software Engineering, (pp. 244 – 254).

Aggarwal, K. K. Singh, Y and Chhabra, J. K. (2003). A Fuzzy Model for Measurement of Software Understandability, International Symposium on Performance Evaluation of Computer & Telecommunication Systems, Montreal, Canada.

Ambler, S. W. (2008). Agile practices and Principles Survey Results: July 2008. Ambyoft, Inc. Retrived from: http://www.ambysoft.com/surveys/practicesPrinciples2008.html.

Ambler, S. W. (2007). Introduction to Agile Usability: User Experience Activities on Agile Development projects.

Balagtas-Fernandez . F, Forrai . J, and Hussmann . H. (2009). Evaluation of user interface design and input methods for applications on mobile touch screen devices, *Human-Computer Interaction*, (pp. 243–246).

Bankston. (2003). Usability and User Interface Design, In XP White Paper, [Online]. Retrived from: http://www.ccpace.com/Resources/documents/UsabilityinXP.pdf.

Barnett, L. (2006). *Agile Survey Results: Solid Experience And Real Results Agile Journal.*.

Barnett, L. and Schwaber, C. (2004). *Adopting Agile Development Processes; Improve Time-To-Benefits For Software Projects* Forrester Research.

Baskerville Richard L. (1999). Investigating information systems with action research. Communications of the Association for Information Systems, 2(article 19):1–32.

Beck, K. (2004). Extreme Programming Explained: embrace change (2nd Edition). Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

Beck, K. (1999). Extreme Programming Explained: Embrace Change, Boston, MA: Addison-Wesley. ISBN 0-321- 27865-8.

Beyer H, Holtzblatt K, and Baker L (2004). An Agile Customer-Centered Method: Rapid Contextual Design, XP / Agile Universe, Springer.

Blomkvist S. (2005). Towards a Model for Bridging Agile Development and User-Centred Design, in Human Centred Software Engineering, Seffah A, Gulliksen J, and Desmarais M (eds.), Springer.

Boehm B. W. and Turner R. (2003). *Balancing Agility and Discipline*, Addison-Wesley Professional, Boston.

Boehm, B. W. (1988). *A Spiral Model of Software Development and Enhancement.* Vol. 21. TRW Defense Syst. Group, Redondo Beach, CA, USA.

Brooks Jr., F. P. (1995). The Mythical Man-Month: Essays on Software Engineering (20th Anniversary Edition), Addison-Wesley Professional, Boston, MA.

Buxton B. (2007). Sketching User Experiences, Morgan Kaufmann publishers.

Carroll J. (2009). Human Computer Interaction, Interaction-Design.org.

Christelle Scharff, Ravi Verma (2010). Scrum to support mobile application development projects in a just-intime learning context, Proceedings of the 2010 Workshop on Cooperative and Human Aspects of Software Engineering - CHASE 2010, pp. 25-31.

Constantine L (2002). Process Agility and Software Usability: Toward Lightweight Usage Centered Design, Information Age.

Cockburn, A. (2007). Agile Software Development: The Cooperative Game (2nd Edition). Pearson Education.

Cooper A Allen's (2008). Keynote at Agile Conference, Toronto Canada.

Cooper, A. (2004). The Inmates are Running the Asylum (2nd Edition). Pearson Higher Education.

Cooper A and Reimann R (2003). About Face 2.0. The Essential of Interaction design, 2nd Edition Wiley.

Da Cunha, T.F.V. Valeria L. L. Dantas, Rossana M. C. Andrade (2011) SLeSS: A Scrum and Lean Six Sigma Integration Approach for the Development of Software Customization for Mobile Phones, 25th Brazilian Symposium on Software Engineering, (pp. 283-292).

Dawson, C. (2009). *Introduction to Research Methods: A practical guide for anyone undertaking a research project*. How To Books Ltd, 3 Newtec Place, United Kingdom.

Dawson, C. (2002). *Practical Research Methods: A user friendly guide to research.* How To Books Ltd, 3 Newtec Place, United Kingdom.

Dey, J. Anind K. and Hakkila, (2008). Context-Awareness and Mobile Devices.

Dillon, Andrew. (2001) *Beyond Usability: Process, Outcome and Affect in human computer interaction.* Toronto : s.n.

Dyba, T., and Dingsoyr, T. (2009). What Do We Know about Agile Software Development? *IEEE Software , 26*, 6-9.

Eduard Metzker and Harald Reiterer. (2004). Integrating Usability Engineering Methods into Existing Software Development Processes via Evidence-Based Usability Engineering revue d'Interaction Homme-Machine, pp. (61-64)

Fowler, M. (2005). The New Methodology. Retrived from http://www.martinfowler.com/articles/newMethodology.html

Fowler, M and Highsmith, J. (2001). The Agile Manifesto, Retrived from http://agilemanifesto.org/.

Fox, D., Sillito, J., and Maurer, F. (2008). Agile Methods and User-Centered Design: How These Two Methodologies Are Being Successfully Integrated In Industry. In Proc. Agile '08, 63-72.

Fritz, Bauer, (1968). *Software Engineering: A Report on a Conference Sponsored by NATO Science Committee*, NATO.

Goldsbury R. Christopher (2012). The Agile Management Fad, [Online] Retrived from http://anagilestory.com/2012/08/21/theagile- management-fad/

Heuristics evaluations. (2014) [Online]. Retrived from: http://usability.gov/methods/test_refine/heuristic.html

Highsmith, J. (2002). *Agile Software Development Ecosystems*, Pearson Education, Indianapolis, USA.

Hofer, T. Schwinger, W. Pichler, M. Leonhartsberger, G. Altmann, J. and Retschitzegger, J. (2003) Context-awareness on mobile devices - the hydrogen approach, in *36th Annual Hawaii International Conference on System Sciences.*

Holler, R. (2011) Mobile Application Development: A Natural Fit with Agile Methodologies Version One, LLC. White paper. [Online]. Retrived from www.versionone.com/pdf/mobiledevelopment.pdf.

Hulkko, H., and Abrahamsson, P. (2005). A Multiple Case Study on the Impact of Pair Programming on Product Quality. *Proceedings of the 27th international conference on Software engineering*, (pp. 495-504). St Louis.

Hussain, Z. Wolkerstorfer, P. Tscheligi, M. Lechner, M. Shahzad, S. Sefelin, R and Milchrahm, H. (2008) Probing an Agile Usability Process, Proceeding of CHI 2008, (pp. 2151-2157).

Hyndman, R. (2008). *Quantitative Business Research Methods*. Department of Econometrics and Business Statistics, Monash University (Clayton campus).

IFIP WG 2.7/13.4 (2012). on user interface engineering Copenhagen, Denmark

ISO 9241-11:1998 Ergonomic requirements for office work with visual display terminals (VDTs) -- Part 11: Guidance on usability.

IXDA (2009). About Interaction Design, Interaction Design Association.Retrived from http://www.ixda.org/

Jerome, B. and Kazman, R. (2005). Surveying the Solitudes: An Invetigation into the Relationships between HCI and SE in Practice, in Human Centred Software Engineering, Springer.

Jimenez, G. Taj, S. and Weaver, J. (2005). Design for Testability, in the Proceedings of the 9th Annual NCIIA Conference.

Jordan B. Barlow, Justin Scott Giboney, Mark Jeffery Keith, David W. Wilson, Ryan M. Schuetzler, Paul Benjamin Lowry and Anthony Vance. (2011). Overview and Guidance on Agile Development in Large Organizations, Communications of the Association for Information Systems Vol. 29, Article 2, (pp.25–44)

Jokela, T. and Abrahamsson, P. (2004) Usability assessment of an extreme programming project: Close co-operation with the customer does not equal to good usability, In 5th International Conference, PROFES 2004, pp.393-407 Kansai Science City, Japan.

Kane D. (2003). Finding a Place for Discount Usability Engineering in Agile Development: Throwing Down the Gauntlet, Proceedings of Agile Development Conference (ADC'03), (pp. 1-7)

Kannan N. (2011). Mobile development: Why using an agile methodology makes sense. [Online]. Retrived from http://searchsoftwarequality.techtarget.com/tip/Mobiledevelopment-Why-using-an-Agile-methodologymakes-sense.

Koch, A. S. (2004). Agile Software Development: Evaluating The Methods For Your Organization. Artech House Publishers, Boston, MA.

Ko Dooms and Roope Kylmäkoski, (2005). Comprehensive documentation made agile – experiments with RaPiD7 in Philips, In Proceedings of the 6th International Conference on Product Focused Software Process Improvement - PROFES, (pp 224-233).

Kombo, D.K., and Tromp, D.L.A. (2009). *Proposal and Thesis Writing: An Introduction*. Paulines Publications Africa, Don Bosco Printing Press, Nairobi Kenya.

Kothari, C. (2004). *Research Methodology: Methods & Techniques.* 2nd edition. New age International Publishers, New Delhi, India.

Kunz, T., and Black, J. (1999). An Architecture For Adaptive Mobile Applications. *Proceedings of Wireless 99, the 11th International Conference on Wireless Communications*, (pp. 27-38).

Kuppuswami, S. Vivekanandan, K. Ramaswamy, P. and Rodrigues, P. (2003). *The effects of individual XP practices on software development effort,* SIGSOFT Softw. Eng. Notes, 28, (pp. 6-6).

Kurt Lewin. (1946). Action research and minority problems. Journal of Social Issues, 2(4):34–46.

Lavrakas, P. (2008). *Encyclopedia of Survey Research Methods Vol. 1 & 2*. Sage Publications, Los Angeles, United States of America.

Law, A. and Charron, R. (2005). *Effects of agile practices on social factors, Proceedings of the 2005 workshop on Human and social factors of software engineering*, St. Louis, Missouri: ACM Press.

Lee C. and D, S, McCrickard (2007). Towards extreme(ly) usable software: exploring tensions between usability and agile software development, Proc, AGILE 2007 conference, (Agile '07), IEEE Press, (pp, 59-71).

Lehman M.M. (1969). The Programming Process, IBM Research Report RC 2722, IBM Research Centre, Yorktown Heights, NY.

Lehman M.M. (1997). Laws of Software Evolution Revisited, In Proceedings of EWSPT'96, Nancy, LNCS 1149, SpringerVerlag, pp. 108-124.

Luis Corral, Alberto Sillitti and Giancarlo Succi. (2013). Software Development Processes for Mobile Systems: Is Agile Really Taking Over the Business?, 1st International Workshop on Mobile-Enabled Systems (MOBS 2013) in connection with ICSE 2013, (pp. 19-24). IEEE.

Mark A. Hager, Sarah Wilson, Thomas H. Pollak and Patrick Michael Rooney. (2003). Response Rates for Mail Surveys of Nonprofit Organizations: A Review and Empirical Test. *Nonprofit and Voluntary Sector Quarterly 32(2): 252-267.*

Mika Laakkonen, (2007). Learnability Makes Things Click A grounded theory approach to the software product evaluation, University of Lapland, Lapin Yliopisto.

Meszaros, G., and Aston, J. (2006). Adding Usability Testing to an Agile Project. In Proc. Agile '06, 289-294.

Metzker, E. and Reiterer, H. (2002). *Evidence-based Usability Engineering*. in Computer-aided Design of User Interfaces (CADUI2002). 2002. Valenciennes, France.

Miller, G. J., and Yang, K. (2008). *Handbook of Research Methods in Public Administration*. Newyork: Auerbach Publications, Taylor & Francis GroupModell.

Mugenda, O.M. and Mugenda, A.G. (2003). *Research Methods: Quantitative and Qualitative Approaches*. Nairobi: Acts Press.

Maiden. N, (2009). Card sort to acquire requirements, IEEE Software, (pp. 85 -86).

Nielsen, J. (2008). Agile Development Projects and Usability.

Nielsen, J. (1993) Usability Engineering, Morgan Kaufmann.

Nielson, (1992) Usability engineering Iifecycle, IEEE computer vol. 25, issue 3, (pp 12-22).

Nodder, C. and Nielsen, J. (2008). Agile usability: best practices for user experience on agile development projects. Nielsen Norman Group, Fremont, CA.

Obendorf, H. and Finck, M. (2008). Scenario-based usability engineering techniques in agile development, Acm.

Oinas-Kukkonen, H., and Kurkela, V. (2003). Developing Successful Mobile Applications. *Proceedings of the International Conference on Computer Science and Technology*, (pp. 50-54). Cancun, Mexico.

Oulasvirta A, Wahlström M, and Anders Ericsson K. (2011). What does it mean to be good at using a mobile device? An investigation of three levels of experience and skill, *International Journal of Human-Computer Studies*, vol. 69, no. 3, (pp. 155-169).

Patton, J. (2003) Improving on Agility: Adding Usage-Centered Design to a Typical Agile Software Development Environment, In: ForUse2003: Proceedings of the Second International Conference on Usage-Centered Design.

Patton, J. (2002). Hitting the target: adding interaction design to agile software development. In Proc. OOPSLA '02, 1-ff.

Pikkarainen, M., Salo, O., and Still, J. (2005). *Deploying Agile Practices in Organizations: A Case Study*. Springer Berlin / Heidelberg.

Polit, D., and Beck, C. (2003). *Nursing Research: Principles & Methods*, 7th Edition, USA Lippincott, Williams and Wilkins.

Pressman, R. (2005) Software Engineering – a Practitioner's Approach (6th Edition), McGraw Hill.

Rahimian, V. and Ramsin, R. (2008) Designing an Agile Methodology for Mobile Software Development: A Hybrid Method Engineering Approach, Second International Conference on Research Challenges in Information Science, RCIS, (pp. 337-342).

Roman, G. C., Picco, G. P. and A. L. Murphy, A. L. (2000). Software engineering for mobility: a roadmap, in *Proc. of the Conf. on the Future of Software Engineering*, pp. 241–258.

Rosson, M .B. and Carroll, J. M. (2002). Usability Engineering: Scenario-Based Development of Human- Computer Interaction, New York, Morgan Kaufman.

Salo, O. (2006). *Enabling Software Process Improvement in Agile Software Development Teams and Organisations*. Helsinki: VTT.

Saunders, Lewis and Thornhill (2007) Research Methods for Business Students Fourth edition, Pearson Education Limited.

Schach, Stephen (1990) *Software Engineering*, Vanderbilt University, Aksen Association.

Schatz. B and Abdelshafi .I. (2005). *Primavera gets agile: a successful transition to agile development*, Software, IEEE, 22 , (pp. 36-42).

Seffah A, Desmarais M, and Metzker E. (2005) HCI, Usability and SE Integration: Present and Future, in Human Centred Software Engineering, Springer.

Sharp, H., Robinson, H. and Segal, J. (2008). Integrating user centered design and software engineering: a role for extreme programming, [Online]. Retrived from: http://www.ics.heacademy.ac.uk/events/presentations/376_hcie-arp2.pdf.

Sharp, H., Rogers, Y. and Preece, J. (2007) *Interaction Design: Beyond Human-Computer Interaction*, 2nd Edition, Wiley.

Sommerville, I. (2011). *Software Engineering*, 9th Edition, Addison Wesley.

Spataru A. C. (2010). Agile Development Methods for Mobile Applications, Unpublished Master of Science Thesis, United Kingdom: University of Edinburgh.

Unhelkar, B., and Murugesan, S. (2010). The Enterprise Mobile Applications Development Framework. *IT Professional , 12* (3), 33-39.

Varshney, U., and Vetter, R. (2001). A Framework for the Emerging Mobile Commerce Applications. *Proceedings of the 34th Annual Hawaii International Conference in System Sciences, 9*, (pp. 9014-9023).

VTT Electronics. (2006). *Portal of Agile Software Development Methodologies.* Retrieved from Mobile-D Method: http://virtual.vtt.fi/virtual/agile/mobiled.html

Wasserman, A. I. (2010). Software engineering issues for mobile application development, in *Proceedings of the FSE/SDP workshop on Future of software engineering research - FoSER '10*, 397-400.

Wells, D. (2009) User Stories, Extreme Programming. Retrived from http://www.extremeprogramming.org/rules/userstories.html

Wells D. (2009). Extreme programming: a gentle introduction, [Online]. Retived from: http://www.extremeprogramming.org.

Williams, L. Kessler, R. R. Cunningham, W. and Jeffries, R. (2000). *Strengthening the case for pair programming*, Software, IEEE, 17 (pp. 19-25).

Yang-Jae Jeong, Ji-Hyeon Lee, Gyu-Sang Shin (2008) Development Process of Mobile Application SW Based on Agile Methodology, 10th International Conference on Advanced Communication Technology, ICACT 2008, vol.1, (pp. 362-366).

Zikmund, G.W., Babin, B.J., Carr,C.J. and Griffin, M.(2010). *Business Research Methods* 8th ed. South-Western, Cengage Learning.

APPENDICES

Appendix I: Letter of Authorization

Date……

To.....

.....

.....

Dear Sir/Madam,

RE: RESEARCH DATA ON "INTEGRATION OF DISCOUNT USABILITY INTO SOFTWARE ENGINEERING TO ENHANCE DEVELOPMENT OF INTERACTIVE MOBILE PLATFORM BASED DEVICES".

I am a student pursuing a Masters Degree in Software Engineering at Jomo Kenyatta University of Agriculture and Technology. I' am required to undertake a research thesis as partial fulfillment for the award of this degree. My research topic is stated above and kindly request for your assistance in making my research a success.

This purpose of this letter is therefore to request you to grant permission to collect relevant data from your organization from selected respondents among your staff. The information collected will be treated with utmost confidentiality and will be used for the purposes on this research only.

Yours Sincerely

Denish Omondi Otieno

Appendix II: QUESTIONNAIRE

Introduction

Hello,

I am a researcher from the School of Computer Science and Information Technology at Jomo Kenyatta University of Agriculture and Technology (JKUAT) Nairobi, Kenya. Currently am running a survey on the usability of the Extended mobile-D a software development process model we am proposing.

I aim to collect as many different responses from as many different people as possible to validate my results. All am asking for is about a few minute of your time to fill in the survey, I really would appreciate it.

Your participation in this study is completely voluntary. There are no foreseeable risks associated with this research. However, if you feel uncomfortable answering any question you can withdraw from the survey at any point.

Your survey responses will be strictly confidential and data from this research will be reported only in the aggregate.

Thank you very much for your time and support.

Please start the survey now

Directions: Tick the box that best corresponds to your answer

Section A; General Information about you

1. **What's your gender**

Male ☐ Female ☐

2. **What's your age**

☐ Below 18 ☐ 18-28 ☐ 29-39 ☐ 40-50 ☐ 51 + years

3. **For how long have you been in the software development industry**

☐ Below 1 year ☐ 1-3 years ☐ 4-6 years ☐ 7 and above

4. **What's your position in the software team**

☐ Programmer ☐ Project manager ☐ Software tester ☐ Others (specify)

5. **Do you have any formal training in software development or any field related to software development**

☐ Yes ☐ No

6. **If yes specify under which specification**

☐ Computer science ☐ IT ☐ Electricals and Electronics

Section B; General Information on Products development

7. **How many systems have you developed or participated in developing**

☐ None ☐ 1-5 ☐ 5-10 ☐ 11 and above

8. **A brief description of the best product you have ever developed?**

_____

_____

_____

_____

_____

9. **What is the current version of the product?**

1st ☐ 2nd ☐ 3rd ☐ 4th ☐ 5th ☐ 6thand above ☐

10. **What is the work place of the product?**

☐ Life critical

☐ Business critical

☐ Learning environment

☐ Gaming

11. **Do you carry out a feasibility study before starting a new software development project?**

☐ Yes ☐ No

12. **Do you engage the counsel of a Human Computer Interaction Practitioners in the development process?**

☐ Yes ☐ No

13. **If yes why do you normally incorporate them in the development process**

_____

_____

_____

_____

Section C; Model survey

14. How elaborate and clear are the concepts of the process to solve development problems?

Understandability

Statement	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree
a. The design objective and steps are clearly stated in each section					
b. The structure keeps me focused on what is to be designed					
c. The ordering of steps and sequences is logical					
d. I found the various methods of the model well intergraded					
e. My interaction with it is clear and understandable					

15. How easy is it to learn to use the process?

Learn ability (Ease of Learning)

Statement	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree
a. I can learn to use it quickly					
b. I can quickly become skillful with it					
c. I can easily remember to use it					
d. The data grouping is reasonable for easy learning					
e. Learning to use it is easy?					
f. I think that I would need the support of a technical person to be able to use this model					

16. How much convenient is to apply the methodology on organizational projects?

Applicability (Ease of Use)

Statement	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree
a. It is simple to use					
b. It is designer friendly					
c. It is flexible					
d. It provides the clearest steps possible to accomplish what I would wish do with it					
e. I can use it without written instructions					
f. I don't notice any inconsistencies					
g. Using it I can recover from mistakes quickly and easily					
h. It is easy to apply it in designing of a software system					

17. How much effective or useful is the methodology for current and future projects development?

Usefulness

Statement	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree
a. It can help me be more effective					
b. It can help me be more productive					
c. It can makes the things I want to accomplish easier to get done					
d. Using it would save me time					
e. It meets a designers needs					
f. I find it useful in my job					
g. It gives me a clear insight of specific activities per stage					
h. Using it would lead to better products					

18. How much end user satisfaction does the model promise?

Satisfaction

Statement	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree
a. Using it would improve my job performance					
b. Using it in my job would enable me to accomplish tasks more quickly					
c. Using it would make it easier to do my job					
d. Using it is fun					
e. I feel I need to have it					
f. I would recommend it to a friend					
g. I feel very confident in using this model					

h. I would have no difficulty using the model					
i. I think I would like to use this model frequently					
j. I am satisfied with it					

Section D product development enhancement survey

19. In rating our model from section one to section ten at what point do you feel product development is best be enhanced?

Statement	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree
a. Section 1					
b. Section 2					
c. Section 3					
d. Section 4					
e. Section 5					
f. Section 6					

g. Section 7					
h. Section 8					
i. Section 9					
j. Section 10					

20. In your own opinion will using the model lead to better product development and be of benefit to the software development community?

☐ Yes ☐ No

21. If yes kindly explain why you think so.

_____

_____

_____

_____

_____

Conclusion

We greatly appreciate your time and assistance with this questionnaire thank you.