International Journal of Computer Applications Technology and Research
Volume 3– Issue 11, 668 - 675, 2014, ISSN: 2319–8656

The Need to Integrate Usability Engineering into Agile Process Models for Mobile Applications and Devices Development

Denish Omondi Otieno
School of Computer Science and Information Technology
Jomo Kenyatta University of Agriculture and Technology (JKUAT)
Nairobi, Kenya

Wilson Cheruiyot
School of Computer Science and Information Technology
Jomo Kenyatta University of Agriculture and Technology (JKUAT)
Nairobi, Kenya

Michael Kimwele
School of Computer Science and Information Technology
Jomo Kenyatta University of Agriculture and Technology (JKUAT)
Nairobi, Kenya

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 application 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 platform-based devices. 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 paper we focus on the need to integrate usability engineering in to agile process models for the enhancement of mobile application and devices development.

Keywords: usability engineering, agile process models, mobile devices

1. INTRODUCTION

The operation of human-computer interface is becoming more complicated due to the fast development in the digital technology. 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 platform based 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 application 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. These models are basically not user-centered and most of them provide limited support for usability activities.

1.1 Human computer interaction

Human-computer interaction (HCI) is a multi-disciplinary field with a focus on the interaction between humans and computers it is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them Keith Andrews (2013). Humans are Individual users, a group of users working together, a sequence of users in an organization. Computers involve, desktop computers, large-scale computer system, Pocket PC, embedded system etc.

1.2 Mobile platform based devices

Mobile application 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; mobile devices have wireless capability to connect to the Internet and home computer systems.

1.3 Usability
Usability is defined in Part 11 of the ISO 9241 standard 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.” Effectiveness is the accuracy and completeness with which specified users can achieve specified goals in particular environments. Efficiency is defined as the resources expended in relation to the accuracy and completeness of the goals achieved. Satisfaction is the comfort and acceptability of the work system to its users and other people affected by its use.

2. WHY USABILITY ENGINEERING
Human-Computer Interaction (HCI) discipline provides the foundations to develop usable 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). Usability engineering deals with issues such as system learnability, efficiency, memorability, 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.

3. UNIQUE DEVELOPMENT CHALLENGES FOR MOBILE PLATFORM BASED DEVICES
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 application development.

3.1 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 apps 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.

3.2 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 app designer has to consider the screen real estate.

3.3 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 application developers must anticipate the targeted device(s).

3.4 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 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 application design and development. They offer new and valuable mechanisms to make mobile apps more powerful and useful than applications with a more limited array of input possibilities.

3.5 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, 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 application 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.

3.6 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 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 team 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.

3.7 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 application development is needed so that the requisite attention is paid by developers when analyzing these requirements resulting in better designed context-aware applications.

3.8 Behavioral consistency versus specific HCI guidelines

Ideally, a given mobile app 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 application 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.

3.9 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 engineering to specify and analyze mobile application requirements.

3.10 Mobile application build and delivery

The strong business motivation to deliver mobile applications into the market quickly has made mobile development projects typically to have extremely aggressive time lines. Inception-to-delivery periods of a few months are common. The pressure to deliver mobile apps 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 application for multiple operating systems.

3.11 Testing of applications

Another area where mobile application 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.

4. HARDWARE CHALLENGES

Due to the limitations of size and weight for portability purpose, the interface design for mobile devices comes with more hardware challenges when compared to other regularized devices such as desktop phones or printers; these challenges include limited input facilities, limited output facilities, and designing for mobility.

4.1 Limited input facilities

According to Muhanna (2007), there are three main input facilities for mobile devices that are on the market:

- The keyboard,
- The stylus with the touch screen, and
- The scroll wheel.

The keyboard allows a user to hit a key to perform a task or navigate through the mobile menu functionalities; the stylus with the touch screen allows a user to hit the screen to do the task; the scroll wheel can be scrolled and pushed by a user to do a task and also navigate through the menus and submenus. The design of keyboards for mobile devices has been a challenge because the space for key installation on a mobile device is limited.

Mobile interfaces can be quite tricky and cumbersome to use when compared to the fully-blown GUI, especially for those with poor manual dexterity or fat fingers and those who have difficulty in selecting tiny buttons on mobile devices, Siek et al (2005). Research directions on this limitation have come up with different alternatives and solutions. Green et al (2004) described a specialized keyboard 'Stick' that maps row to decrease the physical space. However, a drastic key reduction in order to achieve sufficient portability decreases text entry performance, and requires additional effort to learn a new typing method. The stylus and touch screen which are widely used in personal digital assistants and smart phones can be a good alternative for the keyboard in some cases. However, touch input would be problematic if the screen of a mobile device is small and that would lead a user's fingers to occlude the graphical elements he wishes to work with.

4.2 Limited output facilities

There are various output facilities that are used on mobile devices. The small-sized screen is one of the mainly and most commonly used output facilities for mobile devices. Designing the screen for outputting is a trade-off challenge that needs to be experimentally studied to find out which is the efficient and most effective size of the screen that can be used for the different types of mobile devices Muhanna (2007). For example, having a larger screen can solve a limited output facilities challenge; however, it will bring up another challenge of designing for mobility.

The audio output is another output facility that is commonly used on mobile devices. It can be a good output facility for feedback messages to the user, and can be used in conjunction with the graphics and text messages to have an effective interaction between the human and the device Muhanna (2007).

4.3 Designing for mobility

A mobile device should be portable and easy to be held by the user, and this brings up the big challenge of designing for mobility, Myers (2004).The power facility in a mobile device is the main challenge of designing for mobility that is characterized by limited and dynamically varying available resources and stringent application requirements. Ashwini et al (2006) indicated that the power consumed by an application depends on the performance level requested by the user or application, and that the mobile device can be viewed as the collection of devices. Therefore, it is very crucial to design a power management unit which collects information in hardware so that the performance of the system is not degraded Hwang (2008).

5. THE GAPS IN INDUSTRY PRACTICES

Jerome and Kazman analyze the gaps between SE and usability in HCI in practice Jerome, et al (2005) from a survey of 63 HCI practitioners and 33 software engineers; they found that the state of practice is not very encouraging. They report that there is substantial lack of mutual understanding among software engineers and HCI practitioners and the two disciplines hardly follow each other. They also do not collaborate much in projects. 68% software engineers report that they made key software design decisions that affect the user interface without consulting HCI practitioners. Even greater proportion of HCI practitioners (91%) believe that software engineers were making key design decisions without consulting any HCI practitioners. When collaboration does occur, it usually happens too late. Only 1 out of 21 software engineers and 2 out of 60 HCI practitioners reported that they collaborated during the specifications phase below we explore the challenges.

5.1 Usability engineering inputs are not taken during requirements specifications

Usability engineering inputs are needed early in the process before requirements are finalized. Use cases in requirements documents routinely over-specified the usability design, including details such as the sequence, the contents of dialog boxes in the application, 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.

5.2 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 usability design and the corresponding requirements.

5.3 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.

5.4 Usability engineering skills do not have process support

Software Engineering projects have some involvement of Usability engineering practitioners, though they still ended with unresolved usability issues that they knew could be solved Jerome, et al (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.

5.5 Too little and too late is not good enough

In projects, Usability engineering practitioners are pulled in towards the end when too many obvious usability problems surfaced Jerome, et al (2005). In these situations, Usability engineering 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 Usability engineering 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.

6. AGILE PROCESS MODELS

Agile process models have come to represent the iterative nature of software development as shown in figure 1 below. Several process models have emerged. 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 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.

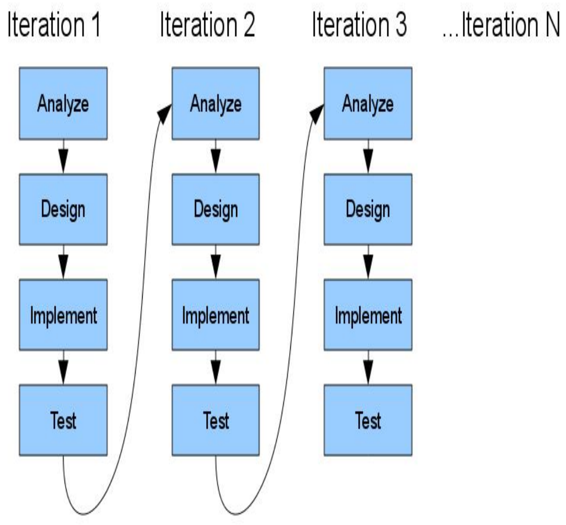

Figure. 1 Agile process

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

6.1 Software engineers are asked to design

The most important issue with agile process models is that they pay little attention to users and Usability Engineering. Agile methods do not acknowledge that Usability Engineering 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 Usability Engineering design. While Nielsen is not against Usability Engineering 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).

6.2 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 Usability Engineering 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.

6.3 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 Usability Engineering 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. Usability Engineering activities can help in anticipating many of the intra-lifecycle changes that arise out of human needs and business processes.

6.4 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 Usability Engineering 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 pp. 77-82). It is difficult to imagine how a scenario can be chopped or merged just so that it can be developed in three weeks.

6.5 Short Iterations

An important Usability Engineering issue is that breaking down product development into small parts and constant change can potentially undermine the totality of the user experience. While some Usability Engineering 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.

7. DISCUSSION

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, Usability Engineering 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 Usability Engineering 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 usability Engineering. 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 and Reiterer, (2002). We need strategies and tools that support engineering organizations. Little research has been done on integrating methods and tools of usability in to software engineering development process for the enhancement of interactive mobile platform based devices and on gathering knowledge about Usability Engineering activities in a form that can capture relationships between mobile platform development contexts, applicable methods, tools and their impact on the engineering process.

8. CONCLUSION

Early computer systems were expensive and were developed mainly for particular tasks, like advanced number-crunching; as such, these systems were employed only by specialist computer users. Often the systems had command-line interfaces, with obscure commands known only by these specialist users. Thus, the user had to adapt to the system, and learning how to use the system required much effort. Computing systems, however, are no longer the province of the specialist user. As the price of PCs and computer-based technologies has fallen, the ownership of these types of goods by non-specialists has widened. The need for the design and development of user interfaces that support the tasks people want to do and that can be used easily by a variety of people with varying abilities has become an important issue. Users are more comfortable with mobile platform based devices that are easy to use, easy to understand, and enable them to attain their goals with minimum frustration. Poor or bad user interfaces design leads to user frustration and dissatisfaction and that's why we highlight different issue to be addressed in regards to achieving better mobile applications and devices.

9. REFERENCES

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

[2] A. Muhanna, "Exploration of human-computer interaction challenges in designing software for mobile devices," master's thesis, University of Nevada, Reno, USA, 2007.

[3] A. Oulasvirta, M. Wahlström, and K. Anders Ericsson, "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, Mar. 2011.

[4] Agile Manifesto 2001, Accessed June 1, 2009, http://agilemanifesto.org/.

[5] B. A. Myers, J. Nichols, J. O. Wobbrock, and R. C. Miller, "Taking handheld devices to the next level," *IEEE Computer Journal*, vol. 37,no. 12, 36−43, 2004.

[6] Bevan N Classifying and Selecting UX and Usability Measures, International Workshop on Meaningful Measures: Valid Useful User Experience Measurement, 2008.

[7] Beyer H, Holtzblatt K, and Baker L, An Agile Customer-Centered Method: Rapid Contextual Design, XP / Agile Universe, 2004.

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

[9] C. Lee and D, S, McCrickard, "Towards extreme(ly) usable software: exploring tensions between usability and agile software development," Proc, AGILE 2007 conference, (Agile '07), IEEE Press,2 007, pp, 59-71.

[10] Carroll J Human Computer Interaction, Interaction-Design.org, 2009

[11] Christer Nordberg (2010), Exploring the text free interface for illiterate users Designing an icon-based prototype for mobile phones

[12] Constantine L Process Agility and Software Usability: Toward Lightweight Usage Centered Design, Information Age, 2002

[13] Cooper A and Reimann R About Face 2.0, Wiley, 2003.

[14] Cooper A Allen's Keynote at Agile 2008.

[15] Da silva, t. S. *Et al.* "User-Centered Design and Agile Methods: A Systematic Review". In: Agile COnference. 2011, pp. 77-86.

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

[17] F. Bomarius et al. (Eds.): PROFES 2009, LNBIP 32, pp. 386–400, 2009.The Waterfall Model in Large-Scale Development, Springer-Verlag Berlin Heidelberg 2009.

[18] Fowler M The New Methodology, December 13, 2005.

[19] Ferreira, j.; sharp, h.; robinson, h. "User experience design and agile development: managing cooperation

through articulation work". *Softw. Pract. Exper.*,New York, NY, USA, vol. 41, August 2011, pp. 963-974.

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

[21] Gulliksen J, Cajander A, and Eriksson E Only Figures Matter? – If Measuring Usability and User Experience in Practice is Insanity or a Necessity, International Workshop on Meaningful Measures: Valid Useful User Experience Measurement, 2008.

[22] H. S. Ashwini, A. Thawani, and Y. N. Srikant, "Middleware for efficient power management in mobile devices," in *Proceedings of the 3rd International Conference on Mobile Technology, Applications and Systems*, 2006.

[23] IXDA About Interaction Design, Interaction Design Association, 2009.

[24] J. Dey, Anind K., Hakkila, "Context-Awareness and Mobile Devices," 2008.

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

[26] Kai Petersen, Claes Wohlin, and Dejan Baca1 *The Waterfall Model in Large-Scale Development,* LNBIP 32, pp. 386–400, 2009.

[27] Kay H. Connelly, Katie A. Siek, Valerie Lafond Favieres, and Gisele Bennett. Planes, pains, and phosphorane: Usability studies in non-traditional environments. In Adjunct Proc. From Interact 2005, 2005.

[28] Katie A. Siek, Yvonne Rogers, and Kay H. Connelly. Fat finger worries: How older and younger users physically interact with PDAs. In Proc. of Interact 2005, pages 267–280. LNCS 3585, 2005.

[29] Keith Andrews, *Human-Computer Interaction Course Notes* Version of 28 May 2013

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

[31] Nielsen J Agile Development Projects and Usability, November 17, 2008

[32] N. Green, J. Kruger, C. Faldu, and R. Amant, "A reduced QWERTY keyboard for mobile text entry," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2004, pp.1429−1432.

[33] p, McBreen, "Quality assurance and testing in agile projects," McBreen Consulting, [online] Available: http://www.mcbreen.ab.caltalksiCAMUG.pdf [Accessed: December 2009]

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

[35] Software Engineering Institute CMMI for Development Version 1.2, August 2006.

[36] Sohaib, o.; khan, k. "integrating Usability Engineering and Agile Software Development: A Literature Review". Computer Design and Applications ICCDA 2010 International Conference on, vol. 2, 2010,

[37] Sharples, M., Taylor, J., & Vavoula, G. (2007) A Theory of Learning for the Mobile Age. In R. Andrews and C. Haythornthwaite (eds.) The Sage Handbook of Elearning Research. London.

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

[39] Usability Professionals Association, Usability Body of Knowledge, 2004, 2010.

[40] Wells D User Stories, Extreme Programming, 2009. http://www.extremeprogramming.org/rules/userstories.html

[41] Welle-Strand, A., & Thune, T. (2009, April 24). *Store Norske Leksikon*. Retrieved May 30, 2010, from Analfabetisme: http://www.snl.no/analfabetisme

[42] Y. S. Hwang, S. K. Ku, C. M. Jung, and K. S. Chung, "Predictive power aware management for embedded mobile devices," in *Proceedings of the 2008 Conference on Asia and South Pacific Design Automation*,2008, pp. 36−41.