NEED OF AN EVOLUTION IN ANALYSIS AND DESIGN METHODOLOGY FOR SMART PHONE APPLICATION DEVELOPMENT

Prof. R. A. Soni*

Abstract

As we had been in a spontaneous technological changes environment since decades, we have to move forward and adopt as well as to get compatible with those technological and non-technological changes made so far. A need arise now to have a crucial need of such kind of refinement in the methodologies adopted for software development which will lead us to adopt those as far as the particular platform and development area concern. This paper presents the idea of having the methodological change and adaptation as well as combination of some of the most adopted methodologies to get into use for the development of the applications which are going to be used by smart phone users.

Keywords— Methodology, Smart Phone, Analysis, Design, Conventional

* Asst. Professor, Master of Computer Application Department, LCIT, Bhandu

I. INTRODUCTION

We have been using so many methodologies and models which will lead us to develop a robust application with the savings of time, cost. The basic idea behind to adopt a development methodology is to analyse the problem and design a physical model which is best suited to the needs of an end – user. But have ever we identified that on which platform the developed application will going to run? Or say which type of application it is? Or say who is an end – user who is not directly related to the application developers? Have we ever identified whether it is a generic product or a product having developed after taken the requirements of an end – user? NO. So this is the paper which will help us to raise as well as to answer those questions which are very much crucial to taken into consideration especially for the application development which are going to be used by smart phone users.

II. HISTORY OF SOFTWARE DEVELOPMENT METHODOLOGY

Today the most adopted development methodologies are System analysis and design techniques, Object oriented system development and Agile methodology. We also going to see other methodologies and the techniques which are to be considered as the part of development of an application or say a software as per the characteristics consigned by those.

A. BISAD (Business Information System Analysis and Design-Honeywell)

B. HIPO (Hierarchy Input Process Output-IBM)

C. Structured systems analysis and design

D. Object oriented system analysis and design

E. System prototype

F. Rapid Application Development

G. Agile modeling

G.A.1 Extreme Programming

G.A.2 Scrum

G.A.3 Crystal

G.A.4 Feature Driven Development (FDD)

G.A.5 Rational Unified Process

G.A.6 Dynamic Systems Development Method (DSDM)

G.A.7 Adaptive Software Development (ASD)

If we derive the much older methodologies which can be separated by whether it's an analysis or design methodology at that time we can have the following list. [1], [3]

The analysis methodologies were

A. DeMarco structured analysis
B. Yourdon modern structured analysis
C. Martin information engineering analysis
D. Bailin object-oriented requirements specification
E. Coad and Yourdon object-oriented analysis
F. Shlaer and Mellor object-oriented analysis

The design methodologies were

A. Yourdon and Constantine structured design
B. Martin information engineering design
C. Wasserman et al. object-oriented structured design
D. Booch object-oriented design
E. Wirfs-Brock et al. responsibility driven design

Structured Systems Analysis and Design Methodology (SSADM) is a systems approach to the analysis and design of information systems. SSADM was produced for the CCTA, a UK government office concerned with the use of technology in government, from 1980 onwards. [2]

Use cases, originally from Jacobson et al. [1992], although not strictly object-oriented, are increasingly being used to define functional requirements even in fairly traditional IS academic programs. These approaches include the spiral model and the concept of risk [Boehm, 1988], rapid development [McConnell, 1996], eXtreme Programming (XP) [Beck, 2000], and agile modeling [Ambler, 2002]. The Unified Process is a comprehensive OO system development methodology originally developed by Jacobson, Booch, and Rumbaugh [1999]. The focus on risk and iteration is grounded in the spiral model developed by Barry Boehm [1988]. [4]

Many practitioners (Booch, 1994; Coad and Yourdon, 1991; Coleman, Arnold, Bodoff, Dollin, Gilchrist, Hayes, and Jeremaes, 1994; Jacobson, Christerson, Jonsson, and Overgaard, 1995; Rumbaugh, Blaha, Premerlani, Eddy, and Lorensen, 1991) believe OOSD to be far superior to

conventional systems development (CSD). OOSD is viewed so highly in some circles that it has been elevated to the "unified software development process"(Jacobson, Booch, and Rumbaugh, 1999). [5]

There are many published object-oriented methodologies such as Object Modelling Technique (OMT) (Rumbaugh et al., 1991), OOSE (Jacobson et al., 1992), OPEN (Henderson-Sellers & Simons, 2000). There are also commercial methodologies available which can be purchased in many configurations such as Rational Rose (Quatrani, 1998) and Rational Unified Process (Jacobson et al., 1999). [6]

Since the object-oriented paradigm promised to revolutionize software development, in the 1990s, demand for object-oriented software systems increased dramatically; consequently, several methodologies have been proposed to support software development based on that paradigm. [7]

Jim Rumbaugh joined Rational in 1994, and we locked him and Grady in a room together, told them to work together and come up with a methodology, and didn't let them out until October of 1995. That year, at OOPSLA, they introduced the Unified Method, which would become the UML. At that time, the Unified Method was both the language and the process that went along with it. 1 Once Ivar joined Rational, we threw him into the room, gave them another couple of years to collaborate (during which they decided to separate the language from the process), and in 1997 submitted the Unified Modeling Language to the Object Management Group (OMG) for standardization. Now, contrary to popular belief, Rational does not own the UML, although we continue to work on it. The UML belongs to the OMG. If you go to the OMG Web site, you can download the PDF version, which I keep on my laptop because it's a great reference. And speaking of reference, I think it's time we started exploring the UML, don't you? [8]

Since OOA&D became popular in the late 1980s, there have been a variety of techniques or tools for OOA&D (e.g., Booch, 1986; Coad and Yourdon, 1991; Embley et al., 1992; Rumbaugh et al., 1991; Martin, 1993) [9]

III. METHODOLOGIES IN PRACTICE NOT SUITABLE FOR SMART PHONE APPLICATION DEVELOPMENT

Structured System analysis and design methodology doesn't prove itself to be the best suited for smart phone applications as:

- Structured System analysis and design methodology puts special emphasis on the analysis of the system and its documentation which causes the danger of over-analyzing and can be very time and cost consuming.
- Due to various types of description methods, checks of consistence cannot be carried out. Especially with large systems, the outline diagram can become very unclear, because all relevant data flows have to be included.

Object oriented analysis and design methodology can also not to be consider as the alternative for Smart Phone application as:

- It requires significant planning and scheduling effort.
- It requires significant time and effort.
- It requires trained and experienced personnel.
- There is no place in the methodology to build a complete functional model.
- There is no single diagram that shows all of the interfaces between objects.

One closer look to an agile methodology for Smart Phone applications:

- When trying to compare smart phone application characteristics to those of an agile method, difficulty comes partly from the fact that boundaries of agile methodologies are not clearly established.
- It was identified through the analysis of patterns that agile methodologies are not well suited for projects involving databases, embedded development and computationally complex projects.
- Agile provides tremendous value, but not necessary cost savings.

Its lack of documentation, even though in Agile it is usually about things going forward, but it is also necessary to be able to look back at what has been done..

IV. CHARACTERISTICS TAKEN INTO CONSIDERATION WHILE DEVELOPING A SMART PHONE APPLICATION

1) **Interaction with other applications** – the applications of smart phone devices may have possibility of interactions with other applications.
2) **Handling of sensors and like built in features** –smart phones include an accelerometer that responds to the device movement, a touch screen that responds to number of gestures, along with the real or virtual keyboards, a global positioning system, a microphone usable by applications other than voice calls, one or more cameras, and multiple networking protocols.
3) **Existence of Hybrid applications** –smart phone devices often include applications that invoke services over the Internet via a web browser and affect data and displays on to the device.
4) **Complexity of hardware and software integration** –smart phone devices may have to support applications that were written for all of the various devices supporting the operating system, and also for different versions of the operating system.
5) **Security** –most of the available smart phone platforms are open which allows the installation of new applications that can affect the overall operation of the device, including the surreptitious transmission of local data by such an application.
6) **User interfaces** –a smart phone 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.
7) **Complexity of testing** – while native applications can be tested in a traditional manner or via a PC-based emulator, smart phone applications are particularly challenging to test. Not only do they have many of the same issues found in testing web applications, but they have the added issues associated with transmission through gateways and network.
8) **Power consumption** – many aspects of an application affect its use of the device's power and thus the battery life of the device. Dedicated devices can be optimized for maximum battery life, but smart phone applications may inadvertently make extensive use of battery-draining resources.

V. PROMISING AREAS FOR METHODOLOGICAL IMPACT OF SMART PHONE APPLICATIONS

5.1 The User Experience

Using a smart phone 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 smart phone applications. The smaller display and different styles of user interaction also have a major impact on interaction design for smart phone applications, which in turn has a strong influence on application development. The smart phone user interface paradigm is based around widgets, touch, physical motion, and keyboards (physical and virtual) rather than the familiar WIMP (Windows, Icons, Menus, Pointer) interface style of Apple's iOS and Microsoft Windows. Other context dependencies may also play a role in the user experience, including such aspects as physical location, proximity to other smart phone devices, and the activation of various device features smart phone platforms include their own UI libraries and guidelines, so native applications for a device will share a common "look and feel." It's in the interest of the application developer to adhere to platform standards, especially on touch-screen devices, where users expect to use the platform's standard set of gestures, which differs for each platform. With the challenge of making the best possible use of limited screen space, user interface design takes on greater importance than ever. Smart phone users are often seeking to quickly complete a simple task, and can't take advantage of the full range of functionality provided by a traditional Web application. The user interfaces for smart phone applications may borrow from traditional web applications, but must often be redesigned to highlight the most commonly used functions and to make most effective use of the screen and the smart phone user interface paradigm, including both the user input and the associated motion and location information.

These observations raise some research issues, including:

1) How does one determine which functions should be present in a smart phone version of a traditional application? Are there techniques that can assure the maximum reuse of code among different versions?

2) What is the comparable effort to build a native smart phone application (or a set of them for different platforms) compared to a web application? Is there a measurable difference in user satisfaction or productivity with either of these?

3) Is there a need for specialized scenario development processes and tools for smart phone applications? Does the smart phone UI require a different contextual design process to support a different set of use cases?

4) How does a software designer integrate the various forms of input and sensor data in application design? The user experience is also strongly affected by other industrial design issues related to the device itself, e.g., weight and size, but these items are largely outside the domain of software development, and not discussed further here.

5.2 Non-functional Requirements

The success of any application, smart phone or otherwise, depends on a lengthy list of non-functional qualities. Among those most relevant to smart phone applications are performance (efficient use of device resources, responsiveness, scalability), reliability (robustness, connectivity, stability),quality (usability, installation ability), and security. Many of these issues have been addressed for web applications, and that knowledge provides an excellent starting point for studying smart phone application requirements. The smart phone environment, with its dependence on different kinds of networks, differs from traditional environments and thus raises some new research questions, such as:

1) Do smart phone web applications behave differently when connected using the telephone network (3G, 4G) than when using an 802.11 (WiFi) or 802.16 (WiMax) connection? Are there differences in security? Is there a significant difference in responsiveness? Are traditional fallback and exception-handling techniques adequate, or does the higher likelihood of a dropped connection (or intermittent connectivity) require additional mechanisms?

2) Are there new techniques needed for assuring data integrity, or will the synchronization techniques from traditional client-server computing suffice? Does potential loss of connectivity or battery power represent a risk to program and/or data integrity if such an event occurs during a transaction or system update?

3) Should applications be designed differently depending on the speed of the network on which they are being used?

4) How does a developer create applications that will maximize battery life and resource usage?

5.3 Processes, Tools, and Architecture

As smart phone applications become more complex and mission critical, development organizations must introduce processes that address more aspects of the development process than are covered in today's agile processes and development environments. As previously noted, the user experience is especially critical, so there is a greater need to create prototypes of the user interface(s), particularly when multiple devices will be supported. Testing is another important area for smart phone methodology adaptation research. One question involves the development of testing methods for product families, such as Android devices. It's insufficient to merely test an Android application on an emulator; it must be tested across many different Android devices running different versions of the operating system on various telecom networks, perhaps with l10n and i18n options. Integrated test suites would simplify this process. Another area for research involves application maintenance in the rapidly changing world of smart phone platforms. While "early adopter" consumers are often willing to update their device and their applications, most enterprise users are less likely to do so. In many cases, their companies will have policies discouraging them from doing so. One particularly interesting question involves the use of virtualization technology on these devices as a way to support various platforms.

5.4 Portability

Application developers quickly developed apps for the iPhone platform following Apple's creation of the AppStore. As noted above, other providers of smart phone platforms and devices have done the same (or are in the process of doing so). An important issue for the application developer is to decide which platform(s) to support in the highly fragmented world of smart phone development. Today, there are at least five important platforms (iPhone, Android, BlackBerry, Windows Phone, Symbian). From the standpoint of the application developer, it's quite expensive to support multiple platforms, especially when there are multiple versions and variants for each of them.

The application developer has several options:

1) Develop for a single platform only and use, to the extent possible, a common subset of the features available across all variants and versions of that platform; thus, for example, the developer would have only a single code base for an application that would run on different versions of the iPhone, the iPad, and possibly the iPod Touch. While that approach would

simplify the developer's work, the resulting application would not be able to take advantage of all of the differentiating features of each device.

2) Develop native applications for each platform and variant, trading off the development and maintenance costs against the ability to optimize the application for each platform.

3) Develop smart phone web applications, thus minimizing the amount of native code for each platform; it remains uncertain whether this approach will meet the needs of the market, or;

4) Use one or more layer(s) of abstraction that can map a "write once" application into native executable programs that will run on multiple platforms. Each of these approaches presents a set of research questions, and suggests the need for customized tools to support cross-platform development and testing. [10]

VI. SUGGESTED FOCUS, KNOWLEDGE AND SKILLS FOR MOBILE SOFTWARE ENGINEERING RESEARCH AND EDUCATION

6.1 Questions regarding the complexity of development

Mobile software engineering research and education should focus on developing and teaching methods considering the specific challenges of mobile application development. Such challenges are, for instance, short time to market and high quality, especially user experience, security, and flexibility. According to our philosophy mentioned above, requirements engineering, UI and interaction design, or architectural design approaches for mobile business apps should be user centric, lightweight, iterative, and integrative. Mobile software engineering methods need to provide answers to practitioner, for instance, with respect to the following questions:

- How can we achieve a great user experience?
- How can we design for multiple platforms?
- What is the right scope of an app?
- What do we need to consider when integrating an app in an existing IT infrastructure?

6.2 Required skills for mobile software engineering

In order to provide adaptive development frameworks capable of abstracting its behaviour in favour of expressiveness without losing the possibility of adjustment to the execution platform, a

multi-disciplinary approach is required. It should combine traditional computer science knowledge (algorithms, data structures, network, compiling, operating systems, and programming languages) with additional knowledge that is normally not in the mainstream of a computer science course.

This additional knowledge would be based on the following ideas:

- Model-based design must be the focus of education: i.e., the software engineer must be capable of defining and using different models as well as relating them through transformations;
- New abstraction models must be explored to capture the semantics of the end-user application. To this end, natural language processing, visual languages, and agent systems theory, for instance, can be considered;
- Verification, Analysis, and Testing concepts are required to support the software synthesis process and evolution;
- Optimization theory can be applied in the synthesis process to deal with the execution constraints and to the run-time adaptation;
- Fault tolerance is also a required knowledge for the platform provider as a means to cope with the escaped faults both in the hardware and in the software platforms;
- Metrics and assessment techniques, which are quite different in the mobile domain w.r.t. the traditional software development;
- Parallel programming for MPSoC platforms, since this looks like the future of mobile platform [11]

6.3 The Smartphone as the Primary Computing Device

Mobile devices are changing from being an adjunct for occasional remote usage to being the primary way that broad swaths of workers handle information. This creates new challenges in creating usable interfaces for those workers. A new breed of mobile applications now needs to be created that assumes the app is the primary way of creating and consuming information. How does this affect mobile app creation? It means that companies need to create apps that can handle all of the information needed for an application or function, and not just writing apps that are oriented primarily to reading information on the device or exposing a subset of the information.

It is actually a good practice to not try to stuff all objects into a single app. Rather it is important to make sure that there are apps that cover all business objects in a backend application.

- **Data Synchronization** - One important capability is data synchronization. If you are going to use your smart phone as your primary device you need to be able to create and edit data on it. This does not happen without data synchronization. If the only connection to the backend is via live synchronous connection (as with a web browser), users do not have enough confidence to use their devices for work. Just as users adopted email on devices once synchronized email became commonplace, enterprise app usage will not become bidirectional until a solution for offline data and synchronization is in place.
- **Cross Platform Portability** - Another challenge to building highly functional mobile enterprise apps that are an employee's primary computing tool is handling various form factors. Single apps with one codebase need to work across smart phones and tablets of various sizes. Current approaches to mobile interface development required extensive redevelopment for minor changes in screen resolution, which says nothing of the effort required to target a new device.
- **Faster and Easier Development** - Finally, smart phone app development needs to be easier if all enterprise applications are going to be moved to the smart phone in a timely way. Those apps then immediately work across all popular Smart phones. It also simplifies the development process if a new source adapter needs to be written.[12]

VII. DIFFERENCE BETWEEN CONVENTIONAL APPLICATION DEVELOPMENT AND SMART PHONE APPLICATION DEVELOPMENT

Conventional SW Development	Smart Phone application development
- Applications - Direct manipulation - Directly drawn, dynamic graphics - Conventional deployment - Binary representations favored - Development based on established engineering principles - More formal development - Target environment specifically intended for applications - A wide variety of development approaches available	- Native/other Application - Indirect Manipulation' - Needed Layout design plugins - Worlwide Deployment - Representation suggested by SDK provider - Development based on the type and usage of application - More complex in terms of UI design - Target environment may not intended be intended in case of upgradable hardwares - Development approach if to be suggested by the SDK provider

From the findings and the above mentioned study we can have a clear difference between the development of the conventional software and smart phone application

- Here we can see that at the conventional level we can have either a web application or a desktop application while in case of the smart phone application development we can have both native and hybrid application so that an application which uses the web services.
- We can manipulate or say change the data in either manner in case of the conventional development while smart phone application's data cannot be manipulated unless to upload a new version of the same application.
- The graphics or say the layout design in case of the conventional development can be dynamic as per the user interaction or we can have the readily available templates for the same. While in case of the smart phone application development we need to design such a layout which must have a layout design plugins.
- The deployment of the conventional softwares were made either to the customer who is in need for that or who has placed an order to the development for that. While in case of the smart phone application, it will be get available to all those users whose devices get compatible to use those either in a paid manner or at free.
- Basically the conventional softwares has been represented in a machine language code so that it get supported to all the platforms, while in case of the smart phone application it has been decided by the SDK provider who is going to publish the application.
- The development of a conventional applications has been based on the established engineering principles so that the methodology which has been chosen for the development has been applied to each and every stage as well as to the model which has been adopted for the development, while in case of the smart phone application development no established principles has been adopted but mere requirement as well as the type and usage of an application has been taken into consideration.
- The design of the conventional software is very much formal as the problem domain can be easily identified by identifying the basic client requirements while in case of the smart phone application the problem domain as well as the UI design of the application becomes complex to adopt and design.

- The targeted environment in case of the conventional development has been intended in advance while in case of the smart phone application development it depends on the hardware as well as the version of the operating system and its future updating.
- The variety of the approaches are available for the conventional development while in case of smart phone applications, its fully depends on the SDK provider.

VIII. CONCLUSION

From the findings and the differentiation of the design and analysis methodology as well as the adaptation and requirement needed for smart phone application development and conventional development we can derive a conclusion that there is a significance difference between the procedure of not only analysis and design but also the way they get developed are also different, so it's an area where we can have a tremendous focus so that we can develop and derive such a methodology for analysis and design of the smart phone application development such that by using or say adopting that methodology we can have the best way to design and develop a robust smart phone application.

REFERENCES

[1] Robert G. Fichman and Chris F. Kemerer: Massachusetts Institute of Technology, "Object oriented and conventional Analysis and design methodologies- Comparison and critique", 0018~9162/9?/1000-0022$01.00 0 1992 IEEE COMPUTER

[2] Structured Systems Analysis and Design Methodology by ITC Infotech India Ltd.

[3] Donald Millington: Structured Systems Analysis and Design using standard flowcharting Symbols, THE COMPUTER JOURNAL, VOL. 24, NO. 4, 1981

[4] J.W. Satzinger and R.B. Jackson: Making the Transition from OO Analysis to OO Design With the Unified Process, Communications of the Association for Information Systems (Volume12, 2003)659-683

[5] Richard A. Johnson: Object-Oriented Systems Development: A Review of Empirical Research, Communications of the Association for Information Systems (Volume 8, 2002) 65-81

[6] Linda Dawson and Peta Darke: THE ADOPTION AND ADAPTATION OF OBJECT-ORIENTED METHODOLOGIES IN REQUIREMENTS ENGINEERING PRACTICE, ECIS 2002, June 6–8, Gdańsk, Poland

[7] Luiz Fernando Capretz: A Brief History of the Object-Oriented Approach, ACM SIGSOFT Software Engineering Notes vol 28 no 2 March 2003

[8] Terry Quatrani: Introduction to the Unified Modeling Language, UML Evangelist, Rational Developer Network RUC 2001.

[9] Shouhong Wang: TEACHING THE OBJECT-ORIENTED APPROACH FOR BUSINESS INFORMATION SYSTEMS ANALYSIS AND DESIGN, Journal of Informatics Education and Research, Volume 4, p.p. 17-26

[10] Anthony I. Wasserman: Software Engineering Issues for Mobile Application Development, FoSER 2010, November 7-8, 2010, Santa Fe, New Mexico, USA., 2010, ACM, 978-1-4503-0427-6/10/11.

[11] Érika Cota, Luigi Carro, Lucio Duarte, Leila Ribeiro, Flávio Wagner, XModel: an Unified Effort Towards the Development of High- Quality Mobile Applications, PPGC - Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS) Po Box 15064 – Porto Alegre, RS, Brazil {erika,carro,lmduarte,leila,flavio}inf.ufrgs.br

[12] Top Trends in Smart phones, rhomobile.com/resources, WHITE PAPER MAY 2011