APP DEVELOPMENT PROCESS MODEL

BY

RAZU AHMED

ID: 103-35-148

This Report Submitted for Partial Fulfillment of Degree of Bachelor of Science in

Software Engineering

Supervised By

Dr. Shaikh Muhammad Allayear

Associate Professor
Department of Software Engineering

Software Engineering Department
Daffodil International University
Dhaka, Bangladesh

Date: 07-April-2016

APPROVAL

This Report titled **"App Development Process Model"**, submitted by Razu Ahmed, ID No: 103-35-148 to the Department of Software Engineering, Daffodil International University, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Software Engineering and approved as to its style and contents.

i

DECLARATION

I hereby declare that I have taken this thesis under the supervision of **Dr. Shaikh Muhammad Allayear, Associate Professor, Department of Software Engineering, Daffodil International University.** I also declare that neither this thesis nor any part of this has been submitted elsewhere for award of any degree.

[illegible] 21. 04. 2016
....

Razu Ahmed
ID: 103-35-148

Department of Software Engineering
Daffodil International University

Certified By

[illegible] 23-04-16
....

Dr. Shaikh Muhammad Allayear

Associate Professor
Department of Software Engineering
Daffodil International University

ii

ACKNOWLEDGEMENT

First and foremost, we are thankful to Almighty Allah for his blessings on us to contribute to the knowledge of world. I am particularly grateful to my supervisor, Dr. Shaikh Muhammad Allayear, for his invaluable expertise and advice, inspiring and challenging discussions, and endless patience that have supported us throughout this work.

I would also like to thank our department head, Dr. Touhid Bhuyian sir, for his enthusiastic guidance and excellent comments on my work. I would like to mansion the name of Mijanur Rahman sir, for his initiative inspiration in my study and also like to mention all other faculty members at the Department of Software Engineering and all students of this department, thanks for providing an excellent environment to work in.

The research presented in this thesis was conducted in close cooperation between academia and industry. Therefore, I would like to thank everyone involved at industry. I am grateful to all anonymous participants and their companies who have helped in making the data collection possible for this thesis.

Finally remember in my mind, the name of Prof. Dr. Md. Ismail Jabiullah sir, founder and Ex. Head, department of Software Engineering, Daffodil International University, we always respect his dedication and lovesome time with us.

Last but not the least, I would like to thank our family and friends for constantly reminding me of the most important things in life and for always supporting me in critical situation. All above specially to my parents for their off the record love.

--- Razu Ahmed

ABSTRACT

According to demand of time, from beginning of software Engineering heavy weight software development process as like waterfall model (1971) to present light weight and agile development process has appeared. Spiral Model, RAD, Iterative and Incremental, Scram, Kanban, Lean and other popular development processes, models and technics are appeared time to time based on need. There are some methods as like XP (eXtreme Programming), Test Driven Development (T-DD), Pair Programming, Continuous Integration (CI) and many more invented to make easy the development process, deliver more reliable product, reducing risk, cost, time and finally user/customer satisfaction with their changing requirements demand. Now a days, mobile applications are being more popular and increasing significant number of Apps in the domain area of Education, Health, Office, Finance, Banking, Media, Entertainment, E-Commerce, Ticketing, Social Communication, Marketing, Game and other important area of our daily life. Apps are develop for quick access of information and also developed in short time, quick delivery, small development team and budged, with less functionality, better user friendable interface and ensuring quality and reliability. Smart devices are handheld, small screen size, low memory space and processor where apps are run. Native apps are developed using targeted device API and SDK tools such as android or iOS Apps. Mobile web applications or hybrid applications that can run on any smart devices are developed considering mobile environment. So that, characteristics of mobile Apps and its development process are quite different from desktop or PC software development process. So for it is time to define the Mobile App Development Process Model where have to give emphasize on light weight development, quick delivery, customer satisfaction, changing capability base on requirements and ensuring the security and quality that will fulfill the market demand. Considering current market analysis and above describe issue Agile practice is the most appropriate to define and draw a Mobile Apps Development Process Model.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Introduction:

App is short for application - this can be any type of computer program. Applications have been around for as long as computers, but the term 'app' is associated with the software that runs on a smartphone or tablet device. Apps are usually accessed by clicking or tapping on an icon on the home screen of your smartphone or tablet. This means you don't need to search for a program or key in the address of a website. Many companies make their software available in app form - making it easier for customers to find and use their services. Apps cover a whole range of different type of activities. The most popular are games, news, weather, business, education, health, finance and social networks. Many apps make use of the built-in features of the smartphone such as the camera or GPS - which is used for location-based services and maps. Smartphones and tablets come ready loaded with basic apps, like a browser, calendar, email and chat software, but many more are available for free or at low cost [1].

App is for quick access of information within maximum three touches anywhere and everywhere. PC or computer software systems consist with other sub system with external hardware and software. As example in university ERP system software has many parts with student, teacher, admin, account, employee and other functional panel. Giving attendance of employee has to use external hardware device that is integrated with main ERP system software. Teacher, student manages their courses, credit, semester, CGPA, payments and many more. Accounts make salary following employee data. There is ERP system software combination of other sub-system, hardware and software. But App run on mobile device and access any time any place. So for external hardware as like attendance device is not suitable for App. App realized with some particular part of ERP, such as for student to check result, payment, courses etc. Again in a banking system there are many work have to do but App realize for customer only for checking balance, little amount transaction, account status, transaction report and bill pay service or something little more option will be added.

There is a different philosophe to develop App and PC software. Apps running environment is conscious about mobility, memory, screen size, low power and quick access of information. The mobile application field has been receiving astronomical attention from the past few years due to the growing number of mobile app downloads and withal due to the revenues being engendered. With the surge in the number of apps, the number of lamentable apps/failing apps has withal been growing. Interesting mobile app statistics are included in this paper which might avail the developers understand the concerns and merits of mobile apps. The authors have made an effort to integrate all the crucial factors that cause apps to fail which include negligence by the developers, technical issues, inadequate marketing efforts, and high prospects of the users/clients. As per the various surveys, the number of lamentable/failing apps is growing enormously, primarily because mobile app developers are not adopting a standard App development process model for the development of apps. In this paper, we have developed a mobile application development process model with the aid of traditional software development life cycle phases that are appropriated for mobile App.

1.2 Different Types of App Platforms:

Mobile application development platform (MADP), also known as "mobile enterprise application platform," refers to the all-inclusive group of both services and products that allow mobile applications to be developed. With a wide variety of mobile devices, user groups, and networks, the development of mobile software can be extremely difficult. However, MADPs handle this problem because they are able to manage all of these different devices both when they are deployed and during the entire lifecycle of the mobile solution. MADPs are beneficial because their approach is both inclusive and long-term, which is a major improvement from standalone mobile apps.

Mobile App Development Platform means that a company can develop any given mobile app one time and then deploy the app to many different mobile devices. This included tablets, handheld devices, smartphones, and notebooks. The MADP will ensure that the app is compatible with each device that it is sent to without changing the way that the app functions. It is suggested that a company utilize an MADP if their needs meet the “Rule of Three,” which refers to a need for mobile solutions that either work together with three or more back-end sources of data, are compatible with three or more mobile apps, or work with three or more operating systems.

Typically, MADPs consist of both a mobile client application and a mobile middleware server. The middleware server does not store data, but it manages data through security, system integration, scalability, communications, cross-platform support, and more. The client applications then connect to the server and are the driving force behind both business logic and user interface on any given mobile device.

1.3 Some of Mobile App distribution platforms:

The platform organizations need to develop, deploy and manage mobile apps is made from many components, and tools allow a developer to write, test and deploy applications into the target platform environment. Operating system native platforms and Third-party platforms are software distribution platforms which are used as alternatives for operating system native distribution platforms. Independent operating systems are software collections which use their own software distribution, customized user interface, SDK and API (except billing API which is related only to application store) such as F-Droid, Cydia, Aptoide, GetJar, Opera Mobile Store.

1.3.1 App Store (iOS) :

The App Store is a digital distribution platform for mobile apps on iOS, developed and maintained by Apple Inc. The service allows users to browse and download applications that are developed with Apple's iOS SDK. The apps can be downloaded directly to iOS devices such as the iPhone smartphone, the iPod Touch handheld computer and the iPad tablet computer, or onto a personal computer via iTunes.

Name	App Store
Logo	
OS	iOS
Developer	Apple Inc.
Release	July 10, 2008
Website	www.apple.com/appstore

Table 1.1: App Store information.

The Software Development Kit for iPhone OS was announced at the iPhone Software Roadmap event on March 6, 2008. The SDK allows developers running Mac OS X 10.5.4 or higher on an Intel Mac to create applications using Xcode that will natively run on the iPhone, iPod Touch and iPad. A beta version was released after the event and a final version was released in July 2008 alongside the iPhone 3G. As of January 2, the latest iOS SDK is for iOS 9. This major Roadmap event (coupled with a large distribution program for 3rd-party developers), later became known as the iPhone Developer Program, which currently offers two distribution tracks for 3rd-party developers: Standard, and Enterprise. Applications distributed through the standard program can be sold exclusively through the iTunes Store on Mac and Windows, or on the App Store on the iPhone, iPod Touch, and iPad. Developers who publish their applications on the App Store will receive 70 percent of sales revenue, and will not have to pay any distribution costs for the application. However, an annual fee is required to use the iPhone SDK and upload applications to the store.

1.3.2 Google Play (Android) :

Google Play or Google Play Store, and originally the Android Market, is a digital distribution platform operated by Google. It serves as the official app store for the Android operating system, allowing users to browse and download applications developed with the Android SDK and published through Google. Google Play also serves as a digital media store, offering music, magazines, books, movies, and television programs. It previously offered Google hardware devices for purchase until the introduction of a separate online hardware retailer, Google Store, on March 11, 2015. Google Play was launched on March 6, 2012, with the merger of Android Market, Google Music, and Google eBookstore, marking a shift in Google's digital distribution strategy. The services operating under the Google Play banner are: Google Play Music, Google Play Books, Google Play Newsstand, Google Play Movies & TV, and Google Play Games. The Google Play store has reached over 1.43 million apps published and over 50 billion downloads. Applications are available through Google Play either free of charge or at a cost.

Name	Google Play
Logo	
OS	Android
Developer	Google
Release	March 6, 2012; October 22, 2008 (as Android Market)
Website	www.play.google.com

Table 1.2: Google play information.

The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, and Windows XP or later. As of March 2015, the SDK is not available on Android itself, but the software development is possible by using specialized Android applications. Until around the end of 2014, the officially supported integrated development environment (IDE) was Eclipse using the Android Development Tools (ADT) Plugin, though IntelliJ IDEA IDE (all editions) fully supports Android development out of the box, and NetBeans IDE also supports Android development via a plugin. As of 2015, Android Studio, made by Google and powered by IntelliJ, is the official IDE; however, developers are free to use others. Additionally, developers may use any text editor to edit Java and XML files, then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely).

Enhancements to Android's SDK go hand in hand with the overall Android platform development. The SDK also supports older versions of the Android platform in case developers wish to target their applications at older devices. Development tools are downloadable components, so after one has downloaded the latest version and platform, older platforms and tools can also be downloaded for compatibility testing.

1.3.3 Windows Phone Store (Microsoft) :

Windows Phone Store (previously Windows Phone Marketplace) is a digital distribution platform developed by Microsoft for its Windows Phone platform that allows users to browse and download applications that have been developed by third parties. Like much of the new Windows products, it features "Metro UI"; the UI is presented in a panoramic view where the user can browse categories and titles, see featured items, and get details with ratings, reviews, screen shots, and pricing information.The Windows Phone Store (replacing Windows Marketplace for Mobile) was launched along with Windows Phone 7 in October 2010 in some countries. It was reported on 4 October 2010 that the Windows Phone SDK was downloaded over half a million times. At the end of February 2013, the Marketplace had more than 130,000 apps available. With the rollout of Mango (Windows Phone 7.5) the online web Marketplace was unveiled by Microsoft; it offers various features like silent, over the air installation of apps to the user's device. In August 2012, Microsoft rebranded the "Windows Phone Marketplace" to "Windows Phone Store". The change was seen on Windows Phone devices and on the Windows Phone website; the Marketplace section was changed to "Apps+Games". In 2015, Microsoft announced that Windows Phone Store would be phased out and replaced by Windows Store which would act as a "universal" store for all Windows-powered devices. As part of this process, the Windows Phone Store website was moved to a site that also contained the PC Store creating a single site for Windows Phone 7.x, 8.x, and Windows 8.x applications.

Name	Windows Phone Store
Logo	
OS	Windows Phone
Developer	Microsoft
Release	21 October 2010
Website	www.microsoft.com/en-us/store/apps/windows-phone

Table 1.3: Windows Phone Store information.

Windows Phone 7 application development is based upon Silverlight, XNA, and the .NET Compact Framework. The primary tools used for development are Microsoft's Visual Studio 2010 and Expression Blend. Excluding unlocked developer devices, Windows Phone 7 only runs applications that have first been approved by Microsoft and are only available via the Windows Phone Store. Windows Phone 8 application development allows developers to use Visual C#, Visual Basic and Visual C++ to develop applications for Windows Phone 8. In the release of the Windows Phone 8 SDK, Microsoft actively stopped development of XNA in February 2013 allowing developers to code games in a managed .NET Framework language. The development community has had mixed reaction to this decision; some in quite strong terms. The XNA framework (a wrapper around DirectX) has been discontinued in favor of DirectX for Windows Runtime and Windows Phone apps forcing developers to use native code to build games. There are third party alternatives to XNA such as MonoGame that allow developers to continue using XNA in Windows Runtime, Windows Phone 8 and in other platforms. The emulator has also been changed to use Hyper-V as the hypervisor and now requires Windows 8 Pro or Windows Server 2012 64 bit editions, and a system that supports hardware assisted virtualization. Windows Phone 8 supports a subset of Windows Runtime objects for code reuse between Windows 8 and Windows Phone 8 platforms. Native code support has been added to Windows Phone 8 but it can only be used to build DirectX games or Windows Runtime components. Apps that use XAML must still be built with managed .NET Framework languages. A limited support of a subset of Win32 APIs has been added in the Windows Phone 8 SDK. Microsoft has also added support for third party game frameworks such as Unity to make it easier for developers to port apps to Windows Phone. Developers can still develop Windows Phone 7 apps and it would continue be compatible with Windows Phone 8. Windows Phone App Studio is a one stop source for rapid app building for Windows Phone. With couple of configuration steps (complying with "separation of concerns"), users can generate a production-ready app for Windows Store. Once all the configuration and look and feel is set, this online studio let you publish the app directly to Windows Phone Store.

1.3.4 <u>Ubuntu Software Center (Ubuntu)</u> :

In early 2009 Ubuntu developers noted that package management within Ubuntu could be improved and consolidated. Recent releases of Ubuntu, such as Ubuntu 14.04 (Trusty) included five applications for package management which consume space and other resources as well as provide confusion to users. Applications could be downloaded using the basic Add/Remove Applications or with the Synaptic Package Manager. The Software Updater provided updating for installed packages and Computer Janitor cleaned up packages that were no longer needed. The Software Sources application allowed user selection of the package download location. In November 2015 Canonical announced that development would end and the application would be replaced by GNOME Software.

Name	Ubuntu Software Center
Logo	
OS	Ubuntu
Developer	Canonical Ltd.
Release	October 29, 2009
Website	https://apps.ubuntu.com/cat

Table 1.4: Ubuntu Software Center information.

1.3.5 Marketplace (Firefox OS) :

The Marketplace is a store that features apps designed for any device that runs Firefox OS, Firefox for Android or Firefox desktop. It makes finding your favorite apps and discovering new apps easy. What's great is that the apps you download or purchase can be used on multiple devices and platforms. Firefox OS (project name: Boot to Gecko, also known as B2G) is an open-source operating system made for smartphones, tablet computers and smart TVs designed by Mozilla and external contributors, based on the rendering engine of their Firefox web browser and the Linux kernel. Firefox OS is designed to provide a complete, community-based alternative operating system, for running web applications directly or those installed from an application marketplace. The applications use open standards and approaches such as JavaScript and HTML5, a robust privilege model, open web APIs that can communicate directly with hardware, e.g. cellphone hardware. As such, it competes with commercially developed operating systems such as Apple's iOS, Google's Android, Microsoft's Windows Phone, BlackBerry's BlackBerry 10 and Jolla's Sailfish OS.

Name	Marketplace
Logo	
OS	Firefox OS
Developer	Mozilla
Release	February 21, 2013
Website	https://marketplace.firefox.com

Table 1.5: Marketplace information.

Firefox OS was publicly demonstrated in February 2012, on Android-compatible smartphones. By December 16, 2014, Firefox OS phones were offered from 14 operators in 28 countries throughout the world. On December 8, 2015, Mozilla announced that it will stop sales of Firefox OS smartphones through carriers. Mozilla later announced that Firefox OS smartphones would be discontinued by May 2016 as the development of "Firefox OS for smartphones" would cease after the release of version 2.6. Around the same time, it was reported that Acadine Technologies, a startup founded by Li Gong (former president of Mozilla Corporation) with various other former Mozilla staff among its employees, would take over the mission of developing carrier partnerships, for its own Firefox OS derivative H5OS.

1.3.6 BlackBerry World (BlackBerry) :

BlackBerry World (previously BlackBerry App World) is an application distribution service and application by BlackBerry Ltd for a majority of BlackBerry devices. The service provides BlackBerry users with an environment to browse, download and update third-party applications. The service went live on April 1, 2009. On 21 January 2013, BlackBerry announced that it rebranded the BlackBerry App World to simpler BlackBerry World as part of the release of the BlackBerry 10 operating system. The BlackBerry Priv, the newest BlackBerry phone which runs Android, uses the Google Play Store.

Name	BlackBerry World
Logo	
OS	BlackBerry OS
Developer	BlackBerry Ltd
Release	April 1, 2009
Website	http://appworld.blackberry.com

Table 1.6: BlackBerry World information.

In 2003, RIM launched the Mobile Data Service to enable customers to access Java-based third-party enterprise applications using the secure real-time push-based BlackBerry infrastructure. Later on October 21, 2008, RIM announced at the BlackBerry Developer Conference that the company would open an application store for their devices. It was also announced that the store was scheduled to be open in March 2009, and would work in conjunction with PayPal's services. On January 19, 2009, RIM began accepting submissions of applications from developers. On March 4, 2009, RIM officially named the store "BlackBerry App World" (previously called the BlackBerry Application Storefront). It was also confirmed that the service would not initially be available for desktops, and only a web-based catalog would be accessible from non-BlackBerry devices. On April 1, 2009, at CTIA's trade show, RIM announced that App World had gone live. At the BlackBerry sponsored Wireless Symposium, it was announced that an average of one million apps were being downloaded each day. On August 19, 2010, BlackBerry App World 2.0 was released. This new version introduced BlackBerry ID - a single sign, account system that can be used on both the BlackBerry client and the BlackBerry App World desktop storefront. In addition to BlackBerry ID, BlackBerry App World 2.0 also introduced direct credit card billing and carrier billing for AT&T Wireless subscribers. On December 3, 2010, Research in Motion announced that daily downloads were two million apps per day. On February 2, 2011, BlackBerry App World 2.1 was released. This version introduced in-app purchases of digital goods, allowing for add-ons to be purchased within applications. On January 21, 2013, BlackBerry rebranded the BlackBerry App World to simpler the BlackBerry World. On June 18, 2014, BlackBerry announced an official relationship with Amazon, which includes access to Amazon Appstore in BlackBerry 10.3.

1.4 <u>Different Types of App</u>:

Mobile apps are basically little, self-contained programs, used to enhance existing functionality, hopefully in a simple, more user-friendly way. Take one of today's modern smartphones. They all come with powerful web browsers, meaning you can do pretty much anything you can do on a desktop computer in a phone's browser. Normally, when people talk about apps they are almost always referring to programs that run on mobile devices, such as Smartphones or Tablet Computers. There are thousands of apps available falling into many different categories. Nowadays there seems to be an app for everything. Whether it's checking up on breaking news, chatting with friends via social networking or even booking last minute holidays there's an app out there to help us.

There are three types of Apps: Native App, Web App and Hybrid App.

1.4.1 <u>Native App</u>:

Native App has been developed for use on a particular platform or device. A native mobile app is a Smartphone application that is coded in a specific programming language, such as Objective C for iOS and Java for Android operating systems. Native mobile apps provide fast performance and a high degree of reliability. They also have access to a phone's various devices, such as its camera and address book. In addition, users can use some apps without an Internet connection. However, this type of app is expensive to develop because it is tied to one type of operating system, forcing the company that creates the app to make duplicate versions that work on other platforms. Most video games are native mobile apps. Usually, when people hear "Mobile App" they assume you mean Native App. This is a program that runs on a handheld device (iPhone, tablet, etc) which has a "smart" operating system which supports standalone software and can connect the internet via wifi or a wireless carrier network. Usually people download native mobile apps from app stores such as the apple app store or the Android market. A Native app can only be "Native" to one type of mobile operating system: iOS, Android, Blackberry, Symbian, Windows Phone, WebOS etc. If you want to also make your app experience available to Android or Blackberry users, you will need to develop and maintain a separate piece of software. That gets complicated and expensive.

1.1 <u>Web App</u>:

Web App stored on a remote server and delivered over the internet through o browser. Web apps are not real apps; they are really websites that, in many ways, look and feel like native applications. They are run by a browser and typically written in HTML5. Users first access them as they would access any web page: they navigate to a special URL and then have the option of "installing" them on their home screen by creating a bookmark to that page. In contrast, a mobile web app is software that uses technologies such as JavaScript or HTML5 to provide interaction, navigation, or customization capabilities. These programs run within a mobile device’s web browser. This means that they’re delivered wholly on the fly, as needed, via the internet; they are not separate programs that get stored on the user’s mobile device. Web apps became really popular when HTML5 came around and people realized that they can obtain native-like–functionality in the browser. Today, as more and more sites use HTML5, the distinction between web apps and regular web pages has become blurry. Mobile web apps can be designed to run reasonably well via almost any smart mobile web browser — from the full-featured browsers such as the ones available for iPhones and Android phones, to the mid-range browsers such as you see on many BlackBerry phones.

1.4.3 Hybrid App:

Hybrid Apps are like native apps, run on the device, and are written with web technologies (HTML5, CSS and JavaScript). Hybrid apps run inside a native container, and leverage the device’s browser engine (but not the browser) to render the HTML and process the JavaScript locally. A web-to-native abstraction layer enables access to device capabilities that are not accessible in Mobile Web applications, such as the accelerometer, camera and local storage. Hybrid, by definition is anything derived from heterogeneous sources, or composed of elements of different or incongruous kinds. A hybrid app is one that is written with the same technology used for websites and mobile web implementations, and that is hosted or runs inside a native container on a mobile device. It is the marriage of web technology and native execution. Often, companies build hybrid apps as wrappers for an existing web page; in that way, they hope to get a presence in the app store, without spending significant effort for developing a different app. Hybrid apps are also popular because they allow cross-platform development: that is, the same HTML code components can be reused on different mobile operating systems, reducing significantly the development costs. Tools such as PhoneGap and Sencha Touch allow people to design and code across platforms, using the power of HTML.

1.5 App Demand and Market Values:

The last few years have seen an unprecedented number of people rushing to develop mobile apps for iOS and Android. But looking at the installed user base on each platform and information on the payouts made by the different companies, it appears that the vast majority of developers will find themselves with little revenue to show for. As of month of July 2015, Android users were able to choose between 1.6 million apps. Apple's App Store remained the second-largest app store with 1.5 million available apps. The fact that mobile apps are relatively easier to create than computer apps, as well as their considerable lower price has translated into a growing industry which produces every year more and more. In fact, it is impossible to know exactly how many apps are there.

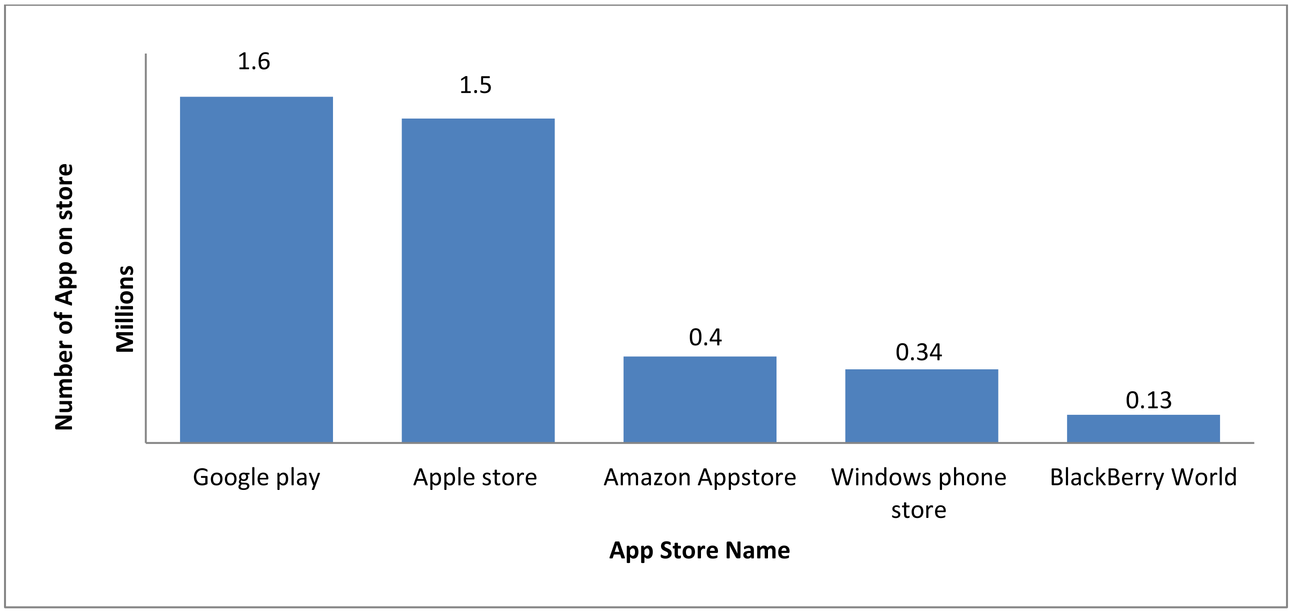


Figure 1.1: Number of App on Various Store As of month of July 2015.

The consensus around the industry is that Google dominates the mobile market with 900 million users, while Apple follows with 600 million iOS devices purchased, and Microsoft comes in third place with an estimated 12 million Windows Phones sold (the vast majority of those, 81%, being sold by Nokia). With different forums aimed at attracting developers, each company handles announcing the size of their markets differently.

Global annual shipments of smartphones using the Android operating system from 2007 to 2016 (in million units), in 2015, the cumulative shipments of smartphones using the Android operating system amounted to 599 million units worldwide.

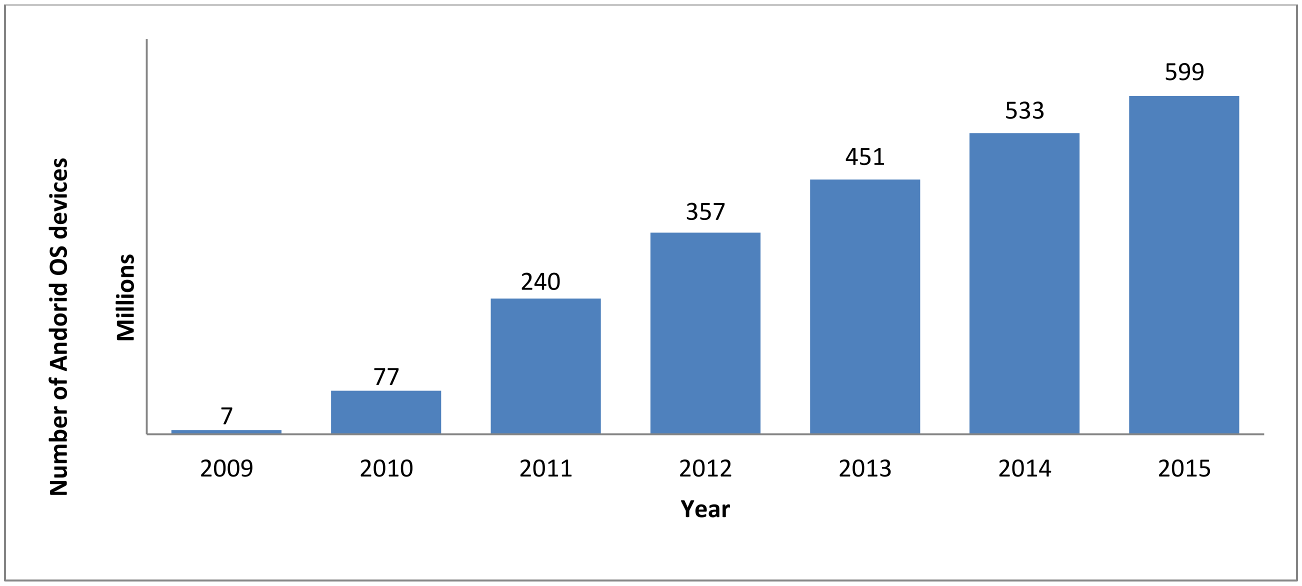


Figure 1.2: Number of Android OS Device Annual Shipment

At the beginning of 2015, 1.4 million mobile apps were available in the Apple App Store. In May 2015, the number of applications submitted for release to the App Store surpassed 40,000 for the first time. The most popular Apple App Store category is gaming with more than 20 percent of available apps belonging to this category. Other leading app categories based on terms of availability are business apps, education apps, lifestyle apps and entertainment apps.

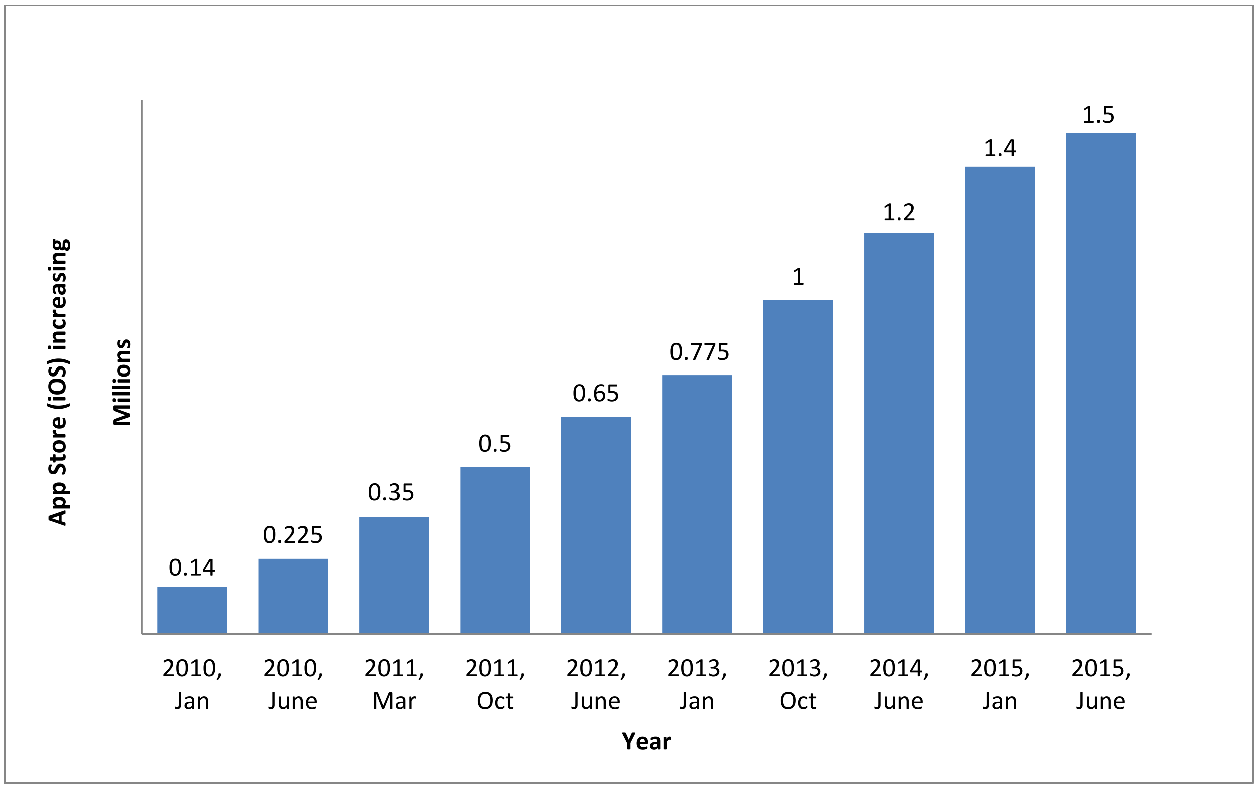


Figure 1.3: App Store Increasing (iOS)

Apple, at its WorldWide Developer Conference, talked about 1.5 million apps in the app store accounting for 100 billion downloads and $40 billion paid off to developers in the last year since 2011. To the company, it is a sign of pride to be able to pay this developer community. Internal data from the app store, gathered from sources close to the company, indicate that the numbers are in line with the actual payments made to developers. The statistic shows the number of cumulative app downloads from Apple's App Store from July 2008 to June 2015. As of the last reported period, Apple announced that 100 billion apps had been downloaded from its App Store.

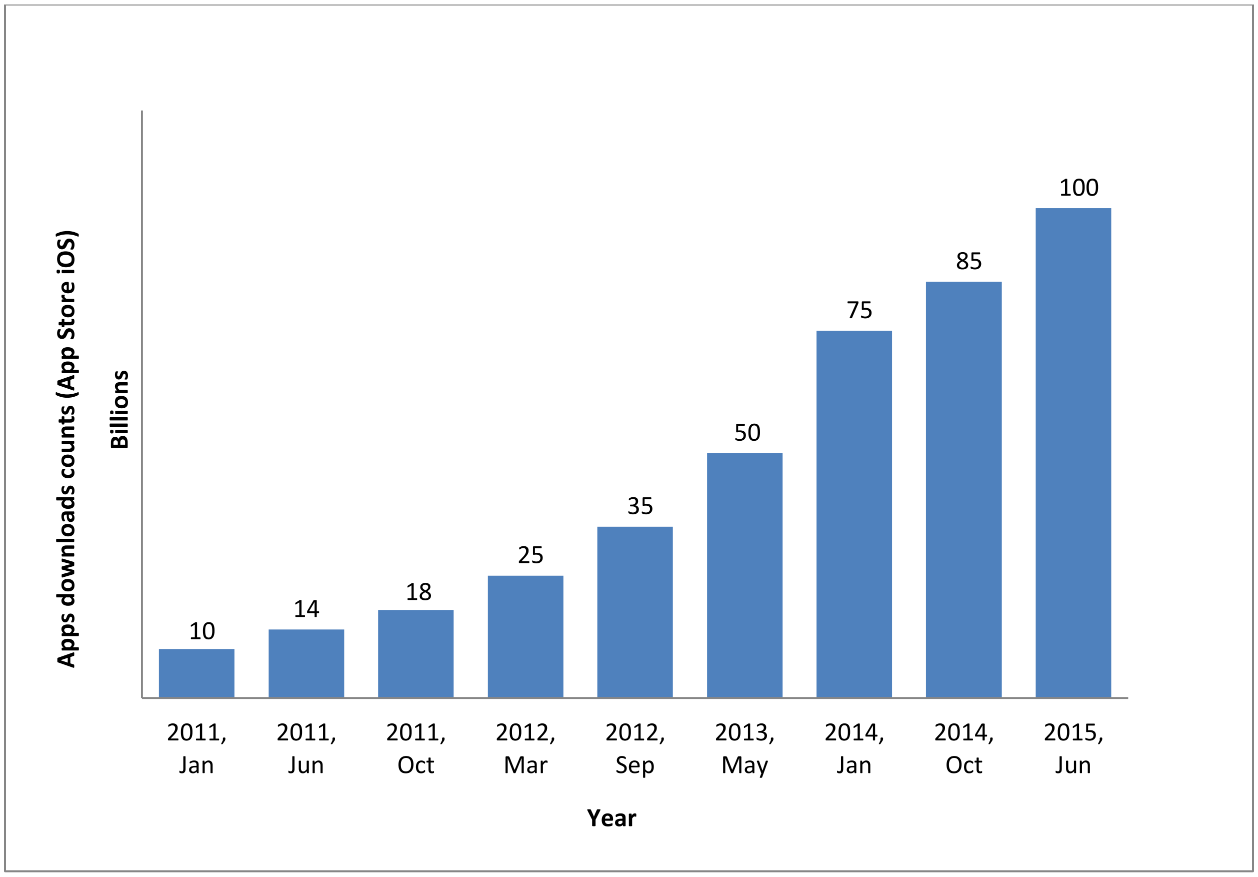


Figure 1.4: Apps downloads counts (App Store iOS)

It is expected that app downloads will grow to 200 billion while mobile app revenues in 2017 will be as huge as$63.5 billion. The transaction value for global mobile payments is projected to grow from $235 billion in 2013 to$721.3 billion in 2017. The main trigger behind rocketing mobile app usage is the growing sales of tablets, smartphones and other mobile devices. At Google I/O, the largest Android developer conference, Google touted 150,000 developers responsible for over 800,000 apps. While the company does not break out revenue numbers on their apps, recent data in their financial filings seemed to indicate somewhere around $900 million in pay-outs to developers “over the last 12 months” and discussions with external research analysts put the number of downloaded apps from the Google Play store at around 48 billion, close to what Apple has claimed. Microsoft, meanwhile, has been claiming 160,000 apps in their store from 45,000 developers. In a recent interview, Microsoft officials claimed that the average user downloaded 54 apps, which would put their download count at 650 million to date. over the last few quarters before and after they introduced their app store shows a variation of up to $100 million since 2011 that could be attributed to the app store. This statistic presents the cumulative Apple App Store earnings of mobile app developers as of January 2016. As of the last reported period, Apple had paid a total of 40 billion U.S. dollars to iOS app developers. Two years ago, this figure amounted to 15 billion U.S. dollars.

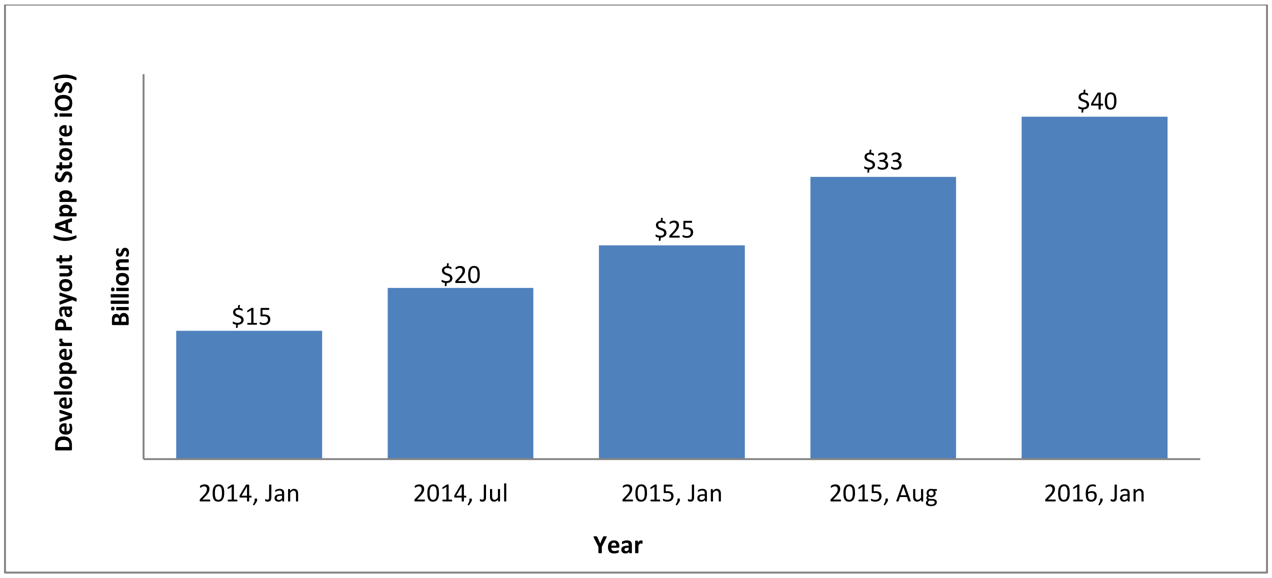


Figure 1.5: Cumulative Developer Payout (App Store iOS)

Applications generate revenue in a number of different ways, such as charging users a small amount of money for the use of an app (an average of 1.02 U.S. dollars per app in the Apple Store), charging for access to premium features of an otherwise free app or simply selling ad space. In 2015, Google Play generated an estimated 6 billion U.S. dollars in. Apple has paid out $40 billion to App Store developers, it said – $8 billion of which was in 2015 alone. For comparison's sake, just over a year ago, Apple said it had paid $20 billion to developers. Taking the data in front of us, we can get a sense of how many apps the average developer creates and what kind of revenue a developer can expect from those apps, on average.

	Google	Apple	Microsoft
Number of App per developer	5	5	3
Number of download per App	60,000	40,000	4,062
Revenue per download	$ 0.01875	$ 0.1	$ 0.1538

Table 1.7: Various App Store revenue and download information.

Multiplying the average revenue per app by its average number of downloads, we can get a sense as to what an average developer can expect to make on an app today. Taking the same number and multiplying it by the number of apps an average developer creates, we get a sense of the revenues one can pull from going that way.

	Google	Apple	Microsoft
Average revenue per App	$ 1125	$ 4000	$ 625
Average revenue per developer	$ 6,000	$ 21,276	$ 2,222

Table 1.8: Revenue per developer and App.

In 2015, the global mobile internet user penetration has exceeded half the world's population, while the average daily time spent accessing online content from a mobile device, such as a smartphone, a tablet computer or wearable, has reached 3.26 hours daily among youths. Accounting for this popularity is also the growing creation and use of mobile applications (shortened to simply apps) – computer programs adapted for mobile use and focused on an extremely wide array of purposes, from gaming to weight loss apps and from mobile messengers to animation software. Currently, the two largest global platforms for app distribution are Apple's App Store, which caters to iOS users, and Google Play, belonging to the eponymous company, which is the official app store for the Android OS.

App goes to call "Bad", the number of lamentable apps/failing apps has withal been growing. "Bad" refers - not conforming to standards. An app can be considered to be deplorable if it has a poor design/UI (native developers), has lot of clutter, has poor navigation, does not meet the user requirement, does not address a specific issue, has security issues, fails at certain essential times, has downloading issues, is not consistent across various platforms, has compatibility issues, consumes lot of battery power, has a very slow replication function, has very high ad frequency, is not appropriately priced, and has no endeavors made to fine-tune issues/concerns raised by the users. Apps must be updated regularly to keep customers focused and engaged. If an app is made and has never been looked at again, then the app could be counted as a bad app. According to the top negative reviews and statistics, 44% verbally express they would expunge a mobile app immediately if app did not perform as expected. The numbers clearly point out that there are good apps, and lamentable apps in the app market. App users not only uninstall the app, but withal provide negative reviews on the app when customers do not relish the app. With social media and word of mouth being so popular negative reviews spread rapidly, which rigorously affects the reputation of developers and poses a threat to their future releases. Hence it is very critical for the developers to understand the criterion for good apps vs. bad apps and develop accordingly.

Chapter 2: Literature Review

2.1 ISO/IEC 12207:1997: (Information Technology—Software Life Cycle Processes):

This international standard contains a set of processes and process contains a set of activities and every activity performs some tasks. Including these sets has tailored a design in respect of software project. This international standard establishes a top-level architecture of the life cycle of software processes rather than provide the step-by-step life cycle model or software development method. This architecture describes interrelationships among the processes and mapping the processes. It also describes continuing responsibilities, effectively establishing the minimum system context, employed to acquire, supply, development, operate and maintaining software life cycle and apply it judiciously. But does not specify the details of how to implement or perform the activities and tasks in the processes [4][5].

For any organization dealings software business or development can be impose this international standard as a condition of trade, agreement concerning for negotiations that must be achieve or performed and a guidance for developing a legally binding contract. This international standard maybe used by a single part as self-imposed tasks and a processes that can be employed for defining, controlling and improving software life cycle model.

This international standard establishes a common framework for software life cycle processes with well-define terminology that can be referenced by the industry. This international standard processes represented with three interrelated life cycle processes of software. Firstly primary life cycle processes with five processes, secondly supporting life cycle processes with eight processes and finally organizational life cycle processes with four processes. Each life cycle process is divided into a set of activities and each activity is further divided into a set of tasks.

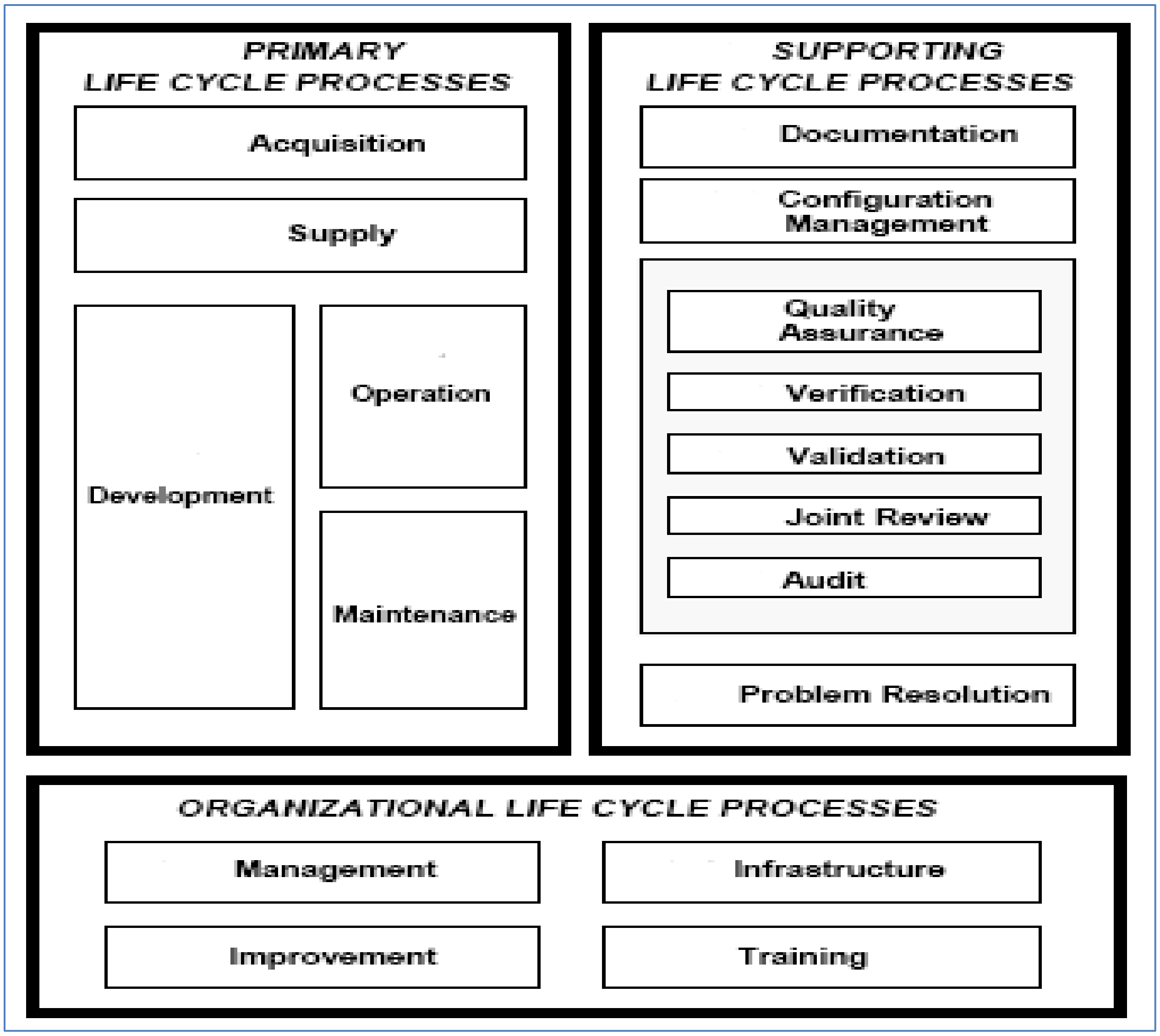


Figure 2.1: Software life cycle processes (ISO/IEC: 12207:1997)

2.1.1 <u>Primary Life Cycle Processes</u>:

Primary life cycle processes perform major roles in software life cycle including five business and technical roles. There are:

I. Acquisition Process;
II. Supply Process;
III. Development Process;
IV. Operation Process;

V. Maintenance Process;

I. <u>Acquisition Process</u> :

The Acquisition Process contains the activities and tasks of the acquirer. The process begins with the definition of the need to acquire a system, software product or software service. The process continues with the preparation and issue of a request for proposal, selection of a supplier, and management of the acquisition process through to the acceptance of the system, software product or software service. The individual organization having the need may be called the owner. The owner may contract any or all of the acquisition activities to an agent who will in turn conduct these activities according to the Acquisition Process. The acquirer in this sub clause may be the owner or the agent. The acquirer manages the Acquisition Process at the project level following the Management Process, which is instantiated in this process; establishes an infrastructure under the process following the Infrastructure Process; tailors the process for the project following the Tailoring Process; and manages the process at the organizational level following the Improvement Process and the Training Process.

List of activities: This process consists of the following activities:

1) Initiation;
2) Request-for-Proposal [tender] preparation;
3) Contract preparation and update;
4) Supplier monitoring;
5) Acceptance and completion.

II. <u>Supply Process</u> :

The Supply Process contains the activities and tasks of the supplier. The process may be initiated either by a decision to prepare a proposal to answer an acquirer's request for proposal or by signing and entering into a contract with the acquirer to provide the system, software product or software service. The process continues with the determination of procedures and resources needed to manage and assure the project, including development of project plans and execution of the plans through delivery of the system, software product or software service to the acquirer.

The supplier manages the Supply Process at the project level following the Management Process, which is instantiated in this process; establishes an infrastructure under the process following the Infrastructure Process; tailors the process for the project following the Tailoring Process; and manages the process at the organizational level following the Improvement Process and the Training Process.

List of activities: This process consists of the following activities:

1) Initiation;

2) Preparation of response;
3) Contract;
4) Planning;
5) Execution and control;
6) Review and evaluation;
7) Delivery and completion.

III. **Development Process** :

The Development Process contains the activities and tasks of the developer. The process contains the activities for requirements analysis, design, coding, integration, testing, and installation and acceptance related to software products. It may contain system related activities if stipulated in the contract. The developer performs or supports the activities in this process in accordance with the contract. The developer manages the Development Process at the project level following the Management Process, which is instantiated in this process; establishes an infrastructure under the process following the Infrastructure Process; tailors the process for the project following the Tailoring Process; and manages the process at the organizational level following the Improvement Process and the Training Process. When the developer is the supplier of the developed software product, the developer performs the Supply Process.

List of activities: This process consists of the following activities:

1) Process implementation;
2) System requirements analysis;
3) System architectural design;
4) Software requirements analysis;
5) Software architectural design;
6) Software detailed design;
7) Software coding and testing;
8) Software integration;
9) Software qualification testing;
10) System integration;
11) System qualification testing;
12) Software installation;
13) Software acceptance support.

IV. **Operation Process** :

The Operation Process contains the activities and tasks of the operator. The process covers the operation of the software product and operational support to users. Because operation of software product is integrated into the operation of the system, the activities and tasks of this process refer to the system. The operator manages the Operation Process at the project level following the Management Process, which is instantiated in this process; establishes an infrastructure under the process following the Infrastructure Process; tailors the process for the project following the Tailoring Process; and manages the process at the organizational level following the Improvement Process and the Training Process. When the operator is the supplier of the operation service, the operator performs the Supply Process.

List of activities: This process consists of the following activities:

1) Process implementation;
2) Operational testing;
3) System operation;
4) User support.

V. <u>Maintenance Process</u> :

The Maintenance Process contains the activities and tasks of the maintainer. This process is activated when the software product undergoes modifications to code and associated documentation due to a problem or the need for improvement or adaptation. The objective is to modify existing software product while preserving its integrity. This process includes the migration and retirement of the software product. The process ends with the retirement of the software product. The activities provided in this are specific to the Maintenance Process; however, the process may utilize other processes in this International Standard. If the Development Process is utilized, the term developer there is interpreted as maintainer. The maintainer manages the Maintenance Process at the project level following the Management Process, which is instantiated in this process; establishes an infrastructure under the process following the Infrastructure Process; tailors the process for the project following the Tailoring Process; and manages the process at the organizational level following the Improvement Process and the Training Process. When the maintainer is the supplier of the maintenance service, the maintainer performs the Supply Process.

List of activities: This process consists of the following activities:

1) Process implementation;
2) Problem and modification analysis;
3) Modification implementation;
4) Maintenance review/acceptance;
5) Migration;
6) Software retirement.

2.1.2 <u>Supporting Life Cycle Processes</u>:

Supporting life cycle processes contributes to the success and quality of the software project. These supporting processes support another processes as an integral part. A supporting process executed as needed by another process. There are eight processes of supporting life cycle processes.

I. Documentation Process;
II. Configuration Process;
III. Quality Assurance Process;
IV. Verification Process;
V. Validation Process;
VI. Joint Review Process;
VII. Audit Process;
VIII. Problem Resolution Process;

2.1.3 <u>Organizational Life Cycle Processes</u>:

Organization life cycle process are employed by an organization to mapping structure associated with software life cycle processes and continuously improve the structure and processes to establish the cycle. Knowledge and experience from project and contracts contribute to improve of the organization. There are four processes of organizational life cycle processes.

I. Management Process;
II. Infrastructure Process;
III. Improvement Process;
IV. Training Process;

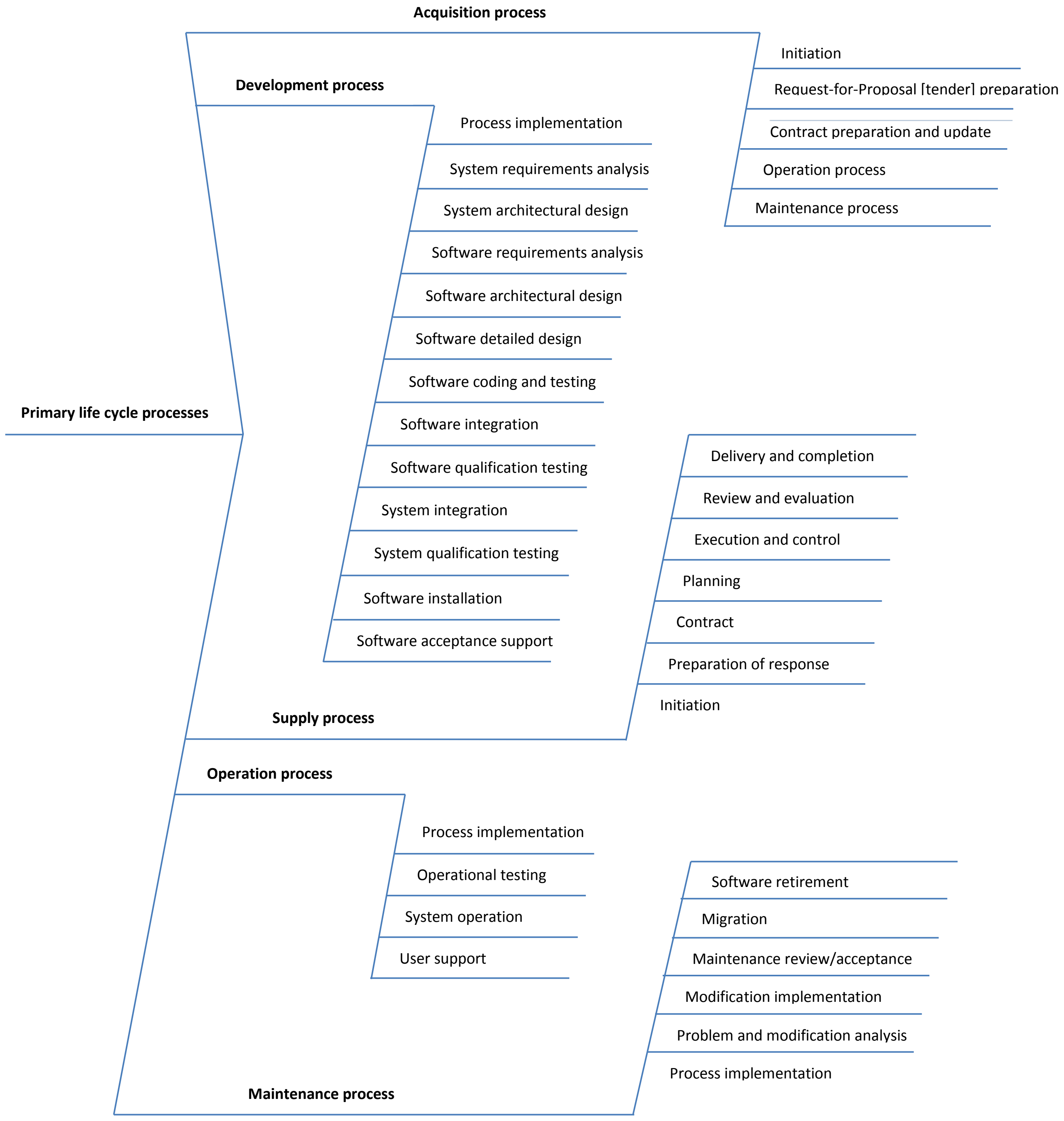


Figure 2.2: Primary life cycle processes with activities.

2.2 <u>IEEE Std. 12207:2008</u>: <u>(Systems and software engineering-Software life cycle processes)</u>:

This International Standard establishes a common framework for software life cycle processes, with well defined terminology, that can be referenced by the software industry. It contains processes, activities, and tasks that are to be applied during the acquisition of a software product or service and during the supply, development, operation, maintenance and disposal of software products. Software includes the software portion of firmware. This International Standard applies to the acquisition of systems and software products and services, to the supply, development, operation, maintenance, and disposal of software products and the software portion of a system, whether performed internally or externally to an organization. Those aspects of system definition needed to provide the context for software products and services are included. This International Standard also provides a process that can be employed for defining, controlling, and improving software life cycle processes.

The purpose of this International Standard is to provide a defined set of processes to facilitate communication among acquirers, suppliers and other stakeholders in the life cycle of a software product. This International Standard is written for acquirers of systems and software products and services and for suppliers, developers, operators, maintainers, managers, quality assurance managers, and users of software products [8].

selecting a life cycle model for the software project and mapping the processes, activities, and tasks in this International Standard onto that model. The parties are also responsible for selecting and applying the software development methods and for performing the activities and tasks suitable for the software project. This International Standard is not intended to be in conflict with any organization’s policies, procedures, and standards or with any national laws and regulations.

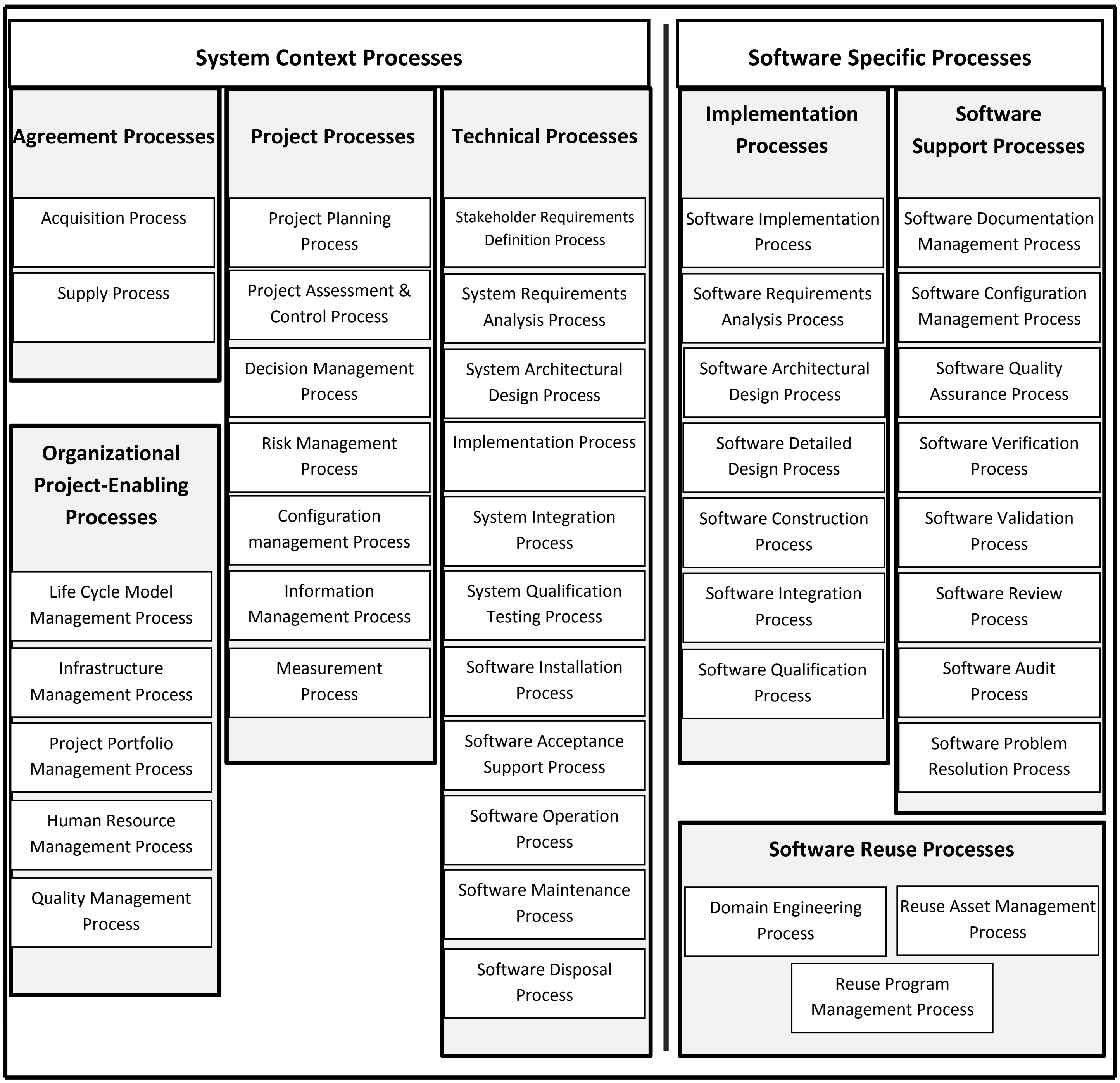

System Context Processes
Agreement Processes
Acquisition Process
Supply Process
Organizational Project-Enabling Processes
Life Cycle Model Management Process
Infrastructure Management Process
Project Portfolio Management Process
Human Resource Management Process
Quality Management Process
Project Processes
Project Planning Process
Project Assessment & Control Process
Decision Management Process
Risk Management Process
Configuration management Process
Information Management Process
Measurement Process
Technical Processes
Stakeholder Requirements Definition Process
System Requirements Analysis Process
System Architectural Design Process
Implementation Process
System Integration Process
System Qualification Testing Process
Software Installation Process
Software Acceptance Support Process
Software Operation Process
Software Maintenance Process
Software Disposal Process
Software Specific Processes
Implementation Processes
Software Implementation Process
Software Requirements Analysis Process
Software Architectural Design Process
Software Detailed Design Process
Software Construction Process
Software Integration Process
Software Qualification Process
Software Support Processes
Software Documentation Management Process
Software Configuration Management Process
Software Quality Assurance Process
Software Verification Process
Software Validation Process
Software Review Process
Software Audit Process
Software Problem Resolution Process
Software Reuse Processes
Domain Engineering Process
Reuse Asset Management Process
Reuse Program Management Process

Figure 2.3: Software life cycle processes (IEEE std: 12207-2008).

2.3 Three Dimensions of Software Engineering & the Basic Software Development Process:

On David Vernon software engineering notes [7], make software engineering useful to think of it in three dimensions and each dimension being concerned with one particular aspect.

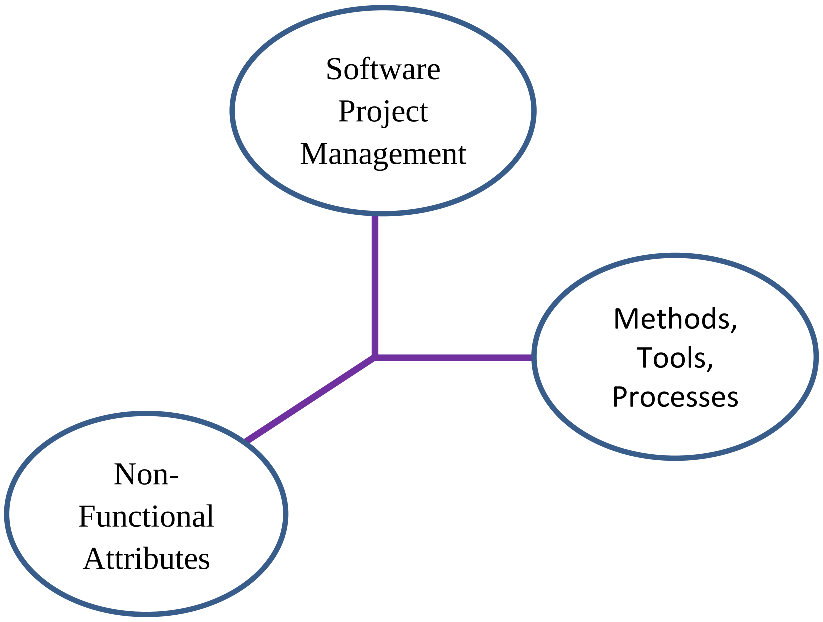

Software Project Management
Methods, Tools, Processes
Non-Functional Attributes

Figure 2.4: Three dimension of software engineering.

One side of dimension contains all of the methods, tools, techniques and processes to development software.

Second side of dimension contains the management techniques that attentive to organize software projects successfully. Effectively monitor the development and improve the development process.

The third dimension addresses the non-functional attributes of software development. Dependency, security, composability, portability and interoperability those attributes refer not to what the software dose but instead to the manner in which it does, that called non-functional attributes.

Here software development process describe with a number of activities and a number of outcomes. In the following diagram activities are the arrows and the outcomes are the bubbles.

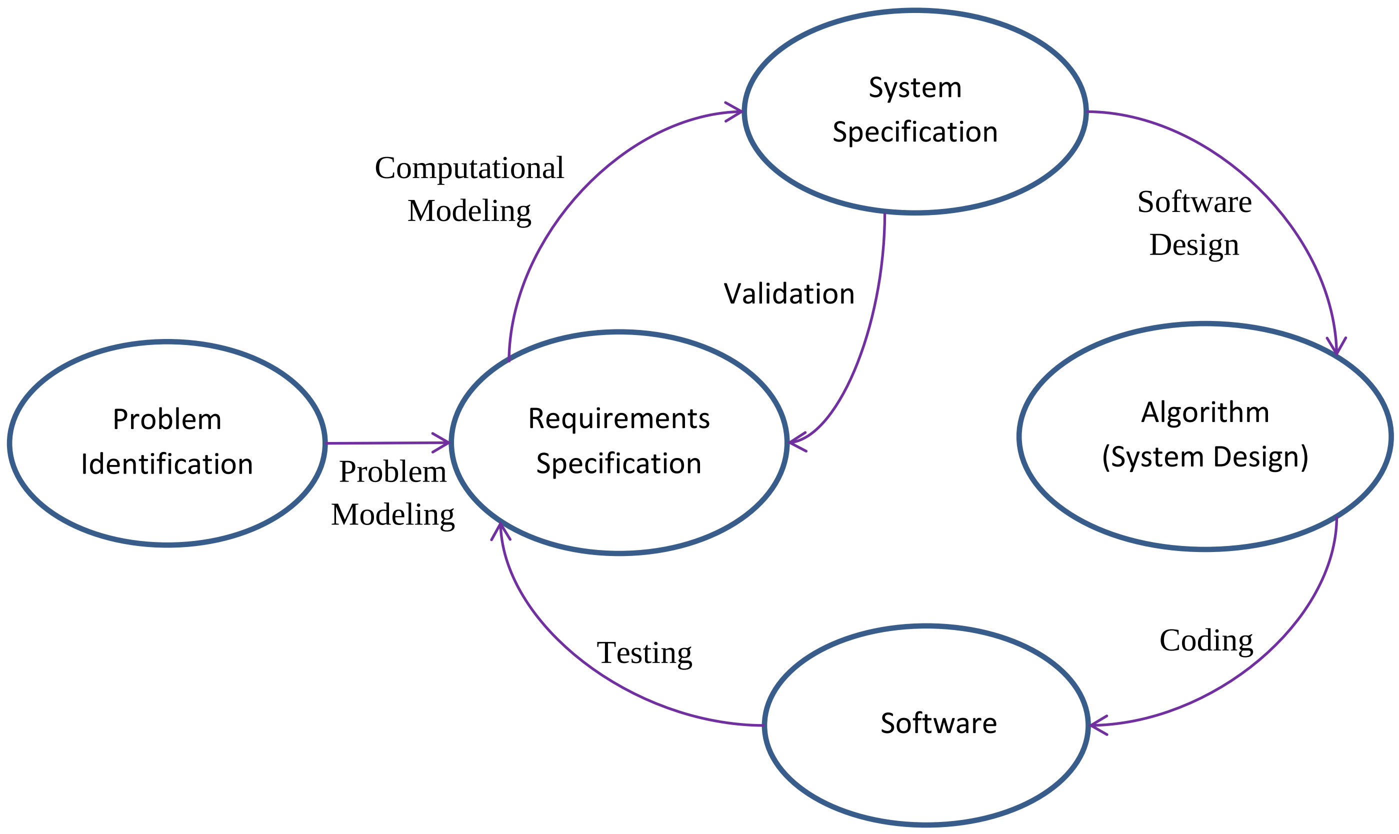


Figure 2.5: The Basic Software Development Process.

2.4 Agile Software Development Method:

The software industry, software technology, and customers expectations were moving very quickly and the customers were becoming increasingly less able to fully state their needs up front [9]. As a result, agile methodologies and practices emerged as an explicit attempt to more formally embrace higher rates of requirements change.

In February 2001, several software engineering consultants joined forces and began to classify a number of similar change-sensitive methodologies as *agile* (a term with a decade of use in flexible manufacturing practices which began to be used for software development in the late 1990's). The term promoted the professed ability for rapid and flexible response to change of the methodologies. The consultants formed the Agile Alliance and wrote The Manifesto for Agile Software Development and the Principles behind the Agile Manifesto. The methodologies originally embraced by the Agile Alliance were Adaptive Software Development (ASD),

Crystal, Dynamic Systems Development Method (DSDM), Extreme Programming (XP), Feature Driven Development (FDD) and Scrum.
Agile methods are a subset of iterative and evolutionary methods and are based on iterative enhancement and opportunistic development processes. In all iterative products, each iteration is a self-contained, mini-project with activities that span requirements analysis, design, implementation, and test. Each iteration leads to an iteration release (which may be only an internal release) that integrates all software across the team and is a growing and evolving subset of the final system. The purpose of having short iterations is so that feedback from iterations N and earlier, and any other new information, can lead to refinement and requirements adaptation for iteration N + 1. The customer adaptively specifies his or her requirements for the next release based on observation of the evolving product, rather than speculation at the start of the project.
With agile methods, iteration lengths vary between one to four weeks, and intentionally do not exceed 30 days. Research has shown that shorter iterations have lower complexity and risk, better feedback, and higher productivity and success rates.

The Agile Alliance documented its value statement as follows:
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.

2.4.1 <u>Agile The Principles</u>:

The Agile Alliance also documented the principles they follow that underlie their manifesto. As such the agile methods are principle-based, rather than rule-based. Rather than have predefined rules regarding the roles, relationships, and activities, the team and manager are guided by these principles [19]:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter time scale.

4. Business people and developers must work together daily through the project.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.

8. Agile processes promote sustainable development.

9. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

10. Continuous attention to technical excellence and good design enhances agility.

11. Simplicity – the art of maximizing the amount of work not done – is essential.

12. The best architectures, requirements, and designs emerge from self-organizing teams.

13. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

2.5 Extreme Programming (XP):

Extreme Programming (XP) originators aimed at developing a methodology suitable for "object oriented projects using teams of a dozen or fewer programmers in one location [21]."
The methodology is based upon five underlying values:
Communication, Simplicity, Feedback, Courage and Respect.

- **Communication:**
  XP has a culture of oral communication and its practices are designed to encourage interaction. The communication value is based on the observation that most project difficulties occur because someone *should have* spoken with someone else to clarify a question, collaborate, or obtain help. "Problems with projects can invariably be traced back to somebody not talking to somebody else about something important."

- **Simplicity:**
  Design the simplest product that meets the customer's needs. An important aspect of the value is to only design and code what is in the current requirements rather than to anticipate and plan for unstated requirements.

- **Feedback:**
  The development team obtains feedback from the customers at the end of each iteration and external release. This feedback drives the next iteration. Additionally, there are very short design and implementation feedback loops built into the methodology via pair programming and test-driven development.

- **Courage:**
  The other three values allow the team to have courage in its actions and decision making. For example, the development team might have the courage to resist pressure to make unrealistic commitments.

- **Respect:**
  Team members need to care about each other and about the project.

In general, XP relies on “documentation” via oral communication, the code itself, and tacit knowledge transfer rather than written documents and artifacts. However, while oral communication may work for small groups, it is not a recommended procedure for large systems, high-risk systems, or systems that require audit-ability for legal or software reliability engineering reasons. In these cases, the following “tools” may need to be more formally managed, recorded/preserved and regularly re-visited as part of a more “formal” and traceable XP process.

- **User story cards:**
  Paper index cards which contain brief requirement (features, fixes, non-functional) descriptions. The user story cards are intentionally not a full requirement statement but are, instead, a commitment for further conversation between the developer and the customer. During this conversation, the two parties will come to an oral understanding of what is needed for the requirement to be fulfilled. Customer priority and developer resource estimate are added to the card. The resource estimate for a user story must not exceed the iteration duration.

- **Task list:**
  A listing of the tasks (one-half to three days in duration) for the user stories that are to be completed for an iteration. Tasks represent concrete aspects of a user story. Programmers volunteer for tasks rather than are assigned to tasks.

- **Customer acceptance tests:**
  Textual descriptions and automated test cases which are developed by the customer. The development team demonstrates the completion of a user story and the validation of customer requirements by passing these test cases.

- **Visible Wall Graphs:**
  To foster communication and accountability, progress graphs are usually posted in team work area. These progress graphs often involve how many stories are completed and/or how many acceptance test cases are passing.

2.5.1 <u>Below the 13 primary technical practices of XP are briefly described</u>:

- **Sit together,** the whole team develops in one open space.

- **Whole team,** utilize a cross-functional team of all those necessary for the product to succeed.

- **Informative workspace,** place visible wall graphs around the workspace so that team members (or other interested observers) can get a general idea of how the project is going.

- **Energized work**, XP teams do not work excessive overtime for long periods of time. The motivation behind this practice is to keep the code of high quality (tired programmers inject more defects) and the programmers happy (to reduce employee turnover). Tom DeMarco contends that, “Extended overtime is a productivity reducing technique.”

- **Pair programming**, refers to the practice whereby two programmers work together at one computer, collaborating on the same design, algorithm, code, or test.

- **Stories**, the team write short statements of customer-visible functionality desired in the product. The developers estimate the story; the customer prioritizes the story.

- **Weekly cycle**, at the beginning of each week a meeting is held to review progress to date, have the customer pick a week’s worth of stories to implement that week (based upon developer estimates and their own priority), and to break the stories into tasks to be completed that week. By the end of the week, acceptance test cases for the chosen stories should be running for demonstration to the customer to drive the next weekly cycle.

- **Quarterly cycle**, the whole team should pick a theme or themes of stories for a quarter’s worth of stories. Themes help the team reflect on the bigger picture. At the end of the quarter, deliver this business value.

- **Slack,** in every iteration, plan some lower-priority tasks that can be dropped if the team gets behind such that the customer will still be delivered their most important functionality.

- **Ten-minute build,** structure the project and its associated tests such that the whole system can be built and all the tests can be run in ten minutes so that the system will be built and the tests will be run often.

- **Test-first programming,** all stories have at least one acceptance test, preferably automated. When the acceptance test for a user story all pass, the story is considered to be fulfilled. Additionally, automated unit tests are incrementally written using the test-driven development (TDD) practice in which code and automated unit tests are alternately and incrementally written on a minute-by-minute basis.

- **Incremental design**, rather than develop an anticipatory detailed design prior to implementation, invest in the design of the system every day in light of the experience of the past. The viability and prudence of anticipatory design has changed dramatically in our volatile business environment. Refactoring to improve the design of previously-written code is essential. Teams with robust unit tests can safely experiment with refactoring because a safety net is in place.

2.5.2 Below the 11 corollary technical practices of XP are briefly described:

- **Real customer involvement**, the customer is available to clarify requirements, is a subject matter expert, and is empowered to make decisions about the requirements and their priority. Additionally, the customer writes the acceptance tests.
- **Incremental deployment,** gradually deploy functionality in a live environment to reduce the risk of a big deployment.
- **Team continuity,** keep effective teams together.
- **Shrinking team,** as a team grows in capacity (due to experience), keep their workload constant but gradually reduce the size of the team.
- **Root cause analysis,** examine the cause of a discovered defect by writing acceptance test and unit test to reveal the defect. Subsequently, examine why the defects was created but not caught in the development process.
- **Shared code**, once code and its associated tests are checked into the code base, the code can be altered by any team member. This collective code ownership provides each team member with the feeling of owning the whole code base and prevents bottlenecks that might have been caused if the "owner" of a component was not available to make a necessary change.
- **Code and tests,** maintain only the code and tests as permanent artifacts. Rely on social mechanisms to keep alive the important history of the project.
- **Negotiated scope contract,** fix the time, cost, and required quality of a project but call for ongoing negotiation of the scope of the project.
- **Pay-per-use,** charge the user every time the system is used to obtain their feedback by their usage patterns. Though not one of the "official" XP practices, essentially all XP teams also have short
- **Stand-Up Meetings,** daily. In these meetings, the team stands in a circle (standing is intentional to motivate the team to keep the meeting short). In turn, each member of the team tells the group:
    - What he or she accomplished the prior day.

- What he or she plans to do today.
- Any obstacles or difficulties he or she is experiencing.

Often the pair-programming pairs are dynamically formed during the daily meeting as the tasks for the day are discussed and the two programmers that are best equipped to handle the task join together.

2.6 Scrum Development Model:

In the Scrum process puts a project management “wrapper” around a software development methodology. The methodology is flexible on how much/how little ceremony but the Scrum philosophy would guide a team towards as little ceremony as possible. Usually a Scrum teams works co-located. However, there have been Scrum teams that work geographically distributed whereby team members participate in daily meeting via speakerphone. Scrum teams are self, directed and self-organizing teams. The team commits to a defined goal for an iteration and is given the authority, autonomy, and responsibility to decide how best to meet it.

There are three main artifacts produced by Scrum teams, the Product Backlog, the Sprint Backlog, and the Sprint Burn down chart [24]. All of these are openly accessible and intentionally visible to the Scrum Team.

- **Product Backlog:** An evolving, prioritized queue of business and technical functionality that needs to be developed into a system and defects that need to be fixed during the release. For each requirement, the Product Backlog contains a unique identifier for the requirement, the category (feature, enhancement, defect), the status, the priority, and the estimate for the feature. It is kept in a spread sheet like format.

- **Sprint Backlog:** A list of all business and technology features, enhancements, and defects that have been scheduled for the current iteration (called a Sprint). The Sprint Backlog is also maintained in a spreadsheet-like format. The requirements are broken down into tasks. For each task in the backlog, the spreadsheet contains a short task description, who originated the task, who owns the task, the status and the number of hours remaining to complete the task. The Sprint Backlog is updated each day by a daily tracker who visits the team members to obtain the latest estimates of the work remaining to complete the task. Estimates can increase when the team member realizes that the work was underestimated.

- **Sprint Burndown chart:** The hours remaining to complete Sprint tasks are graphed and predominantly displayed for the team.

2.6.1 Roles on Scrum:

- **Product Owner:** The person who is responsible for creating and prioritizing the Product Backlog, choosing what will be included in the next iteration/Sprint, and reviewing the system (with other stakeholders) at the end of the Sprint.

- **Scrum Master:** Knows and reinforces the product iteration and goals and the Scrum values and practices, conducts the daily meeting (the Scrum Meeting) and the iteration demonstration (the Sprint Review), listens to progress, removes impediments (blocks), and provides resources. The Scrum Master is also a Developer (see below) and participates in product development (is not just management).

- **Developer:** Member of the Scrum team. The Scrum Team is committed to achieving a Sprint Goal and has full authority to do whatever it takes to achieve the goal. The size of a Scrum team is seven, plus or minus two.

2.6.2 Scrum Process:

The Scrum process is composed of the following:

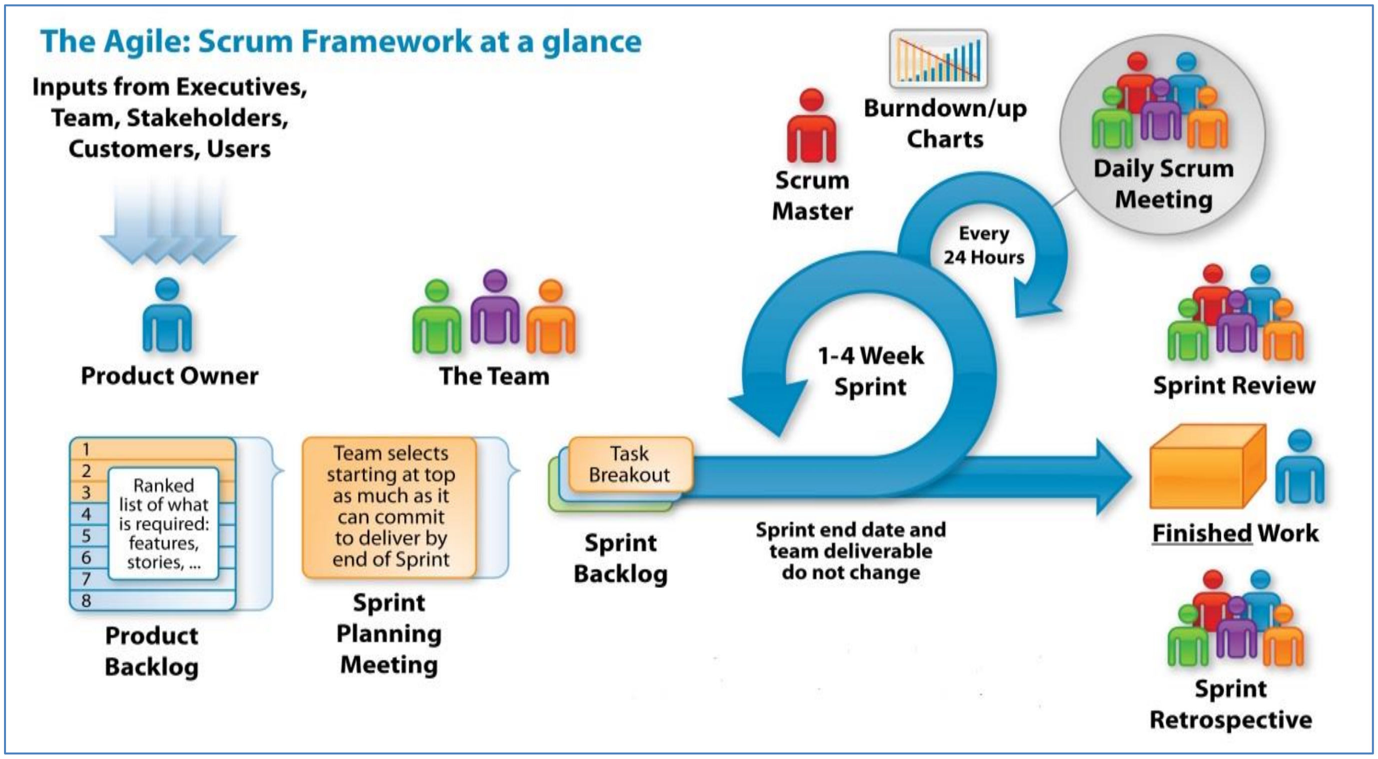

The Agile: Scrum Framework at a glance
Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Burndown/up Charts
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
Sprint Review
1
2
3
4
5
6
7
8
Ranked list of what is required: features, stories, ...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and team deliverable do not change
Finished Work
Sprint Retrospective

Figure 2.6: Scrum Software Development Process.

- **Sprint Planning**, meeting is held with the development team, management, and the Product Owner. The Product Owner is a representative of the customer or a contingent of customers. The Product Owner creates and prioritizes the Product Backlog. In the planning meeting, the Product Owner chooses which features are included in the next 30-day increment (called a Sprint) usually driven by highest business value and risk.

Additionally a Sprint Goal is established which serves as a minimum, high-level success criteria for the Sprint and keep the Scrum Team focused on the big picture, rather than just on the chosen features. The development team figures out the tasks and resources required to deliver those features. Jointly, they determine a reasonable number of features to be included in the next Sprint. Once this set of features has been identified, no re prioritization takes place during the ensuing 30-day Sprint in which features are designed, implemented and tested.

- **During a Sprint**, code is integrated and regression tested daily.

- **Short 15-minute Scrum Meetings**, are held daily. These meetings are similar to XP Stand Up Meetings described above because XP's meetings are based on the success Scrum had with its Scrum Meetings. The meeting is held in a room with a whiteboard so that tasks and blocks can be written down. While others (such as managers) may attend the Sprint Meeting, only the team members and the Scrum Master can speak.
  Each team member answers the following questions:

  - What have you done since the last Scrum?
  - What will you do between now and the next Scrum?
  - What got in your way of doing work?

The Scrum Meeting is an essential component of the methodology. Social promises are made in the meeting which seems to increase responsibility and follow-through and to keep the project on course. However, these meetings can become unmanageable if they are run with too many people; it is recommended that each team has a maximum of seven members. For use with larger teams, the team subdivides into smaller groups, each having its own Scrum meeting. One representative from each of the smaller groups attends a "Scrum of Scrums" meeting. This representative answers the Scrum questions, highlighting the activities of his or her own sub-team. In this way, essential information is passed between sub-teams.

- **At the end of a Sprint, a Sprint Review**, takes place to review progress, demonstrate features to the customer, management, users and the Product Owner and review the project from a technical perspective. The meeting is conducted by the Scrum Master. The Product Owner and other interested stakeholders attend the meeting. The latest version of the product is demonstrated in which the functions, design, strength, weaknesses, and trouble spots are shared with the Product Owner. The focus is on showing the product itself; formal presentations (such as with PowerPoint slides) are forbidden.

- **The cycle continues with a Sprint Planning,** meeting taking place to choose the features for the next Sprint.

2.7 **<u>Continuous Test-Driven Development</u>:**

Test Driven Development is the craft of producing automated tests for production code, and using that process to drive design and programming for every bit of functionality, you first develop a test that specifies and validates what the code will do[19]. You then produce exactly as much code as necessary to pass the test. Then you refactor (simplify and clarify) both production code and test code.

Continuous testing is a technique in modern software development in which the source code is constantly unit tested in the background and there is no need for the developer to perform the tests manually. We propose an extension to this technique that combines it with well-established software engineering practice called Test- Driven Development (TDD). In our practice, that we called Continuous Test-Driven Development (CTDD), software developer writes the tests first and is not forced to perform them manually. We hope to reduce the time waste resulting from manual test execution in highly test driven development scenario.

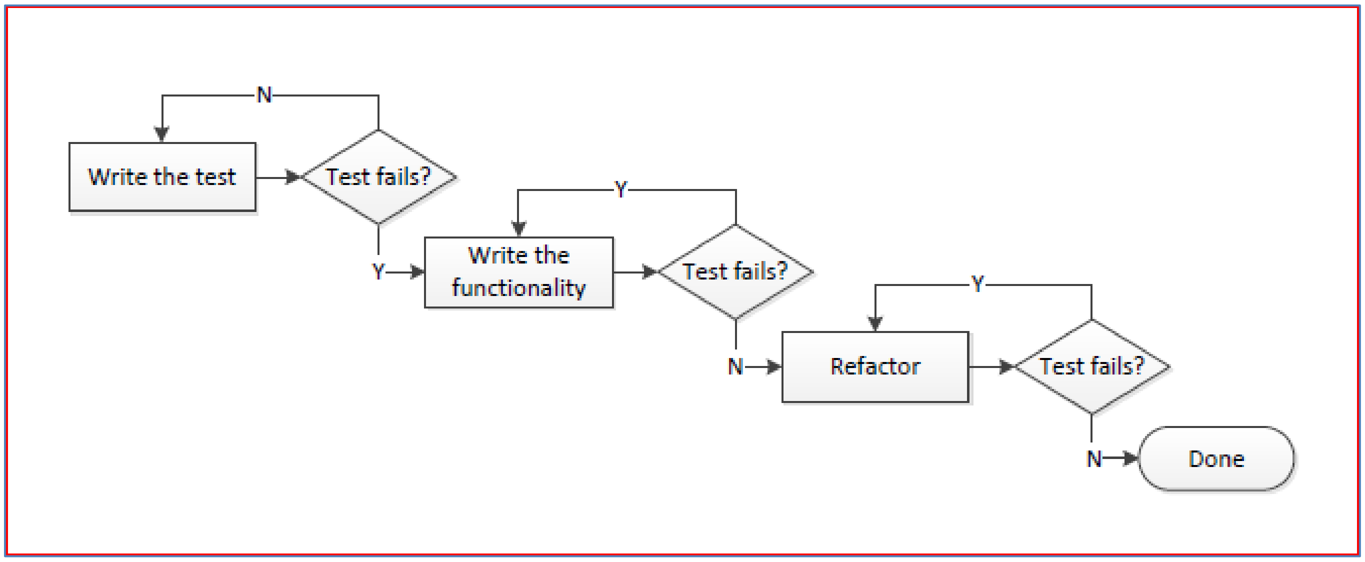


Figure 2.7: Continuous Test-Driven Development activities.

It can be succinctly described by the following set of rules:

- Write a "single" unit test describing an aspect of the program.
- Run the test, which should fail because the program lacks that feature.
- Write "just enough" code, the simplest possible, to make the test pass.
- "Refactor" the code until it conforms to the.
- Repeat, "accumulating" unit tests over time.

2.8 Kanban Software Development Method :

Kanban is a lean approach to agile software development. Actually, Kanban means many things. Literally, Kanban is a Japanese word that means "visual card". At Toyota, Kanban is the term used for the visual & physical signaling system that ties together the whole Lean Production system. Most agile methods such as Scrum and XP are already well aligned with lean principles. In 2004, however, David Anderson pioneered a more direct implementation of Lean Thinking and Theory of Constraints to software development. Under the guidance of experts such as Don Reinertsen, this evolved into what David called a "Kanban system for software development", and which most people now simply refer to as "Kanban [22]".

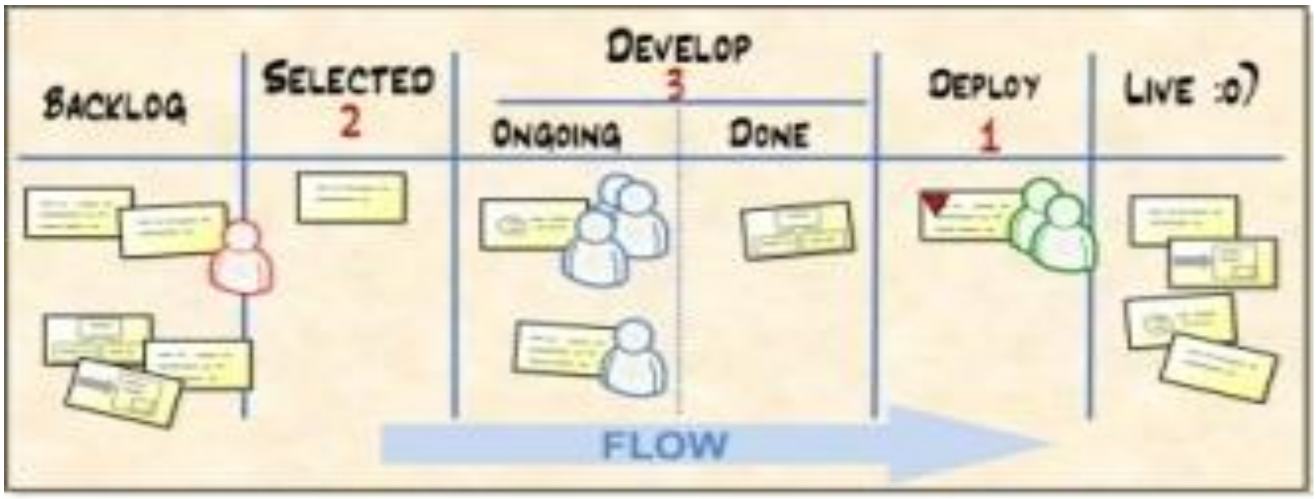


Figure 2.8: Kanban board, with WIP limits.

2.8.1 How does Kanban work?:

There are many flavors, but the core of Kanban means:

- **Visualize the workflow**:
    - Split the work into pieces, write each item on a card and put on the wall.
    - Use named columns to illustrate where each item is in the workflow.

- **Limit WIP:** (work in progress) – assign explicit limits to how many items may be in progress at each workflow state.

- **Measure the lead time:** (average time to complete one item, sometimes called "cycle time"), optimize the process to make lead time as small and predictable as possible.

Here's Figure 2.8 an example of a simple Kanban board, with WIP limits in red.

2.9 Mobile-D:

Mobile-D was the first attempt to incorporate Agile practices for the development of mobile applications. Mobile-D was introduced in 2004 by Abrahamsson, as a development methodology inspired on Extreme Programming, Crystal Methodologies and Rational Unified Process (RUP). It is recommended to be used by a small, co-located team, working in a short development cycle.

Mobile-D is structured in five phases:

i. Explore.
ii. Initialize.
iii. Productionize.
iv. Stabilize.
v. System Test Fix.

Each of these phases has a number of associated stages, tasks and practices. The complete specifications of the method are available in (VTT Electronics, 2006).

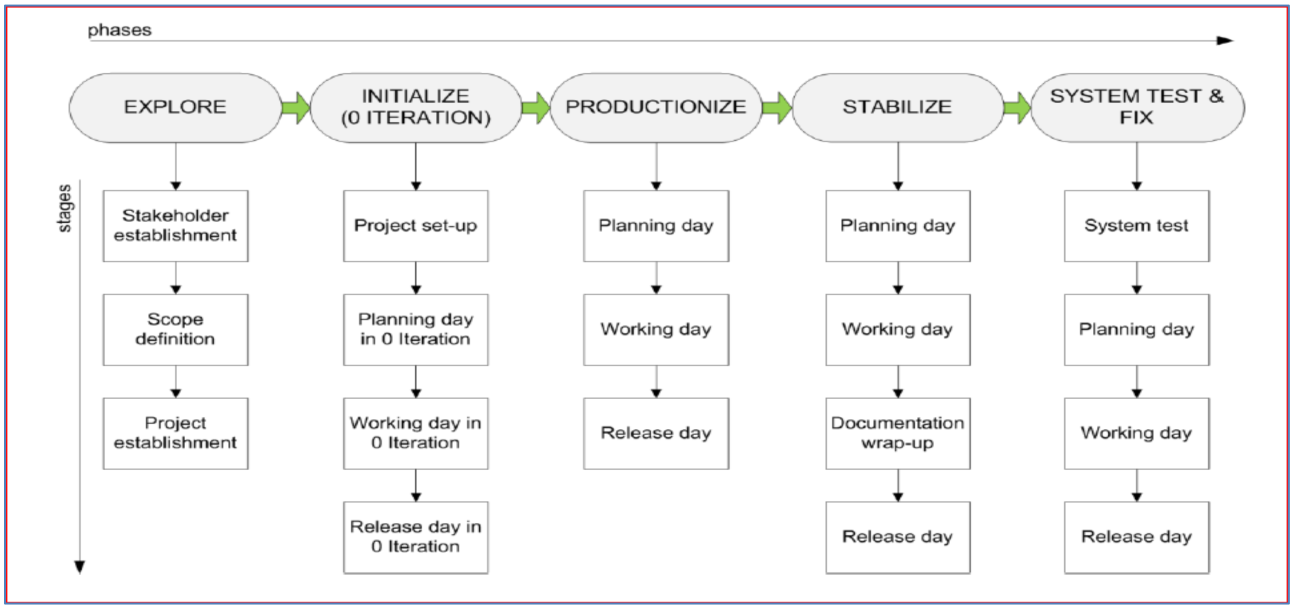


Figure 2.9: Mobile-D with added Evolve Phase. (Adapted from VTT Electronics, 2006).

2.9.1 <u>Description About Mobile-D</u>:

i. **<u>Explore</u>:**

The purpose of Explore phase is the planning and establishment of the incipient project. “A well planned is half done” is a saying to be remembered also in the software development context. Explore phase can be timely unattached to the latter phases of Mobile-D and also overlap with 0 Iteration phase. Explore phase is an important phase to set the ground for controlled implementation of the software development product regarding, for example, issues related to product architecture, software development process and environment selection. Different stakeholder groups are needed to provide their expertise in the Explore phase.

- **<u>Goal</u>:**
  - Establish the stakeholder groups
  - Define and agree the goals and scope.
  - Plan the project regarding environment, personnel and process issues.
- **<u>Process</u>:**
  - Stakeholder Establishment.
  - Scope Definition.
  - Project Establishment.
- **<u>Roles</u>:**
  - Project team.
  - Support group.
  - Customer group/Customer.
  - Steering group.
  - Exploration team.

ii. **<u>Initialize</u>:**

The purpose of the Initialize phase pattern is to enable the success of forthcoming project phases by preparing and verifying all critical development issues so that they all are in full readiness in the end of the phase for implementing requirements selected by the customer.

- **<u>Goal</u>:**
  - Gain a good overall understanding of the product.
  - Prepare physical, technical and human resources as well as customer communication, project plans and all critical development issues.
  - Initial requirements document.

        - Project plan.
        - Base process description.
        - Measurement plan.
        - Training plan.
        - Architecture line description.
        - Product backlog.
    - **Process:**
      The Initialize phase pattern can be conducted via the following stages:
        - Project Set-up.
        - Initial Planning.
        - Trials Day.
    - **Roles:**
        - Project team
        - Project team/project manager
        - Project team/architect
        - Support group
        - Customer group
    - **Risks:**
        - Plans are not flexible.
        - Critical development issues are not ready in the end of the phase.

iii. **Productionize**:
The purpose in the Productionize phase is to implement the required functionality into the product by applying iterative and incremental development cycle.

- **Goal:**
    - Implement the customer prioritized functionality to the product.
    - Focus on the crucial core functionality
- **Process:**
    - Planning day.
    - Working day.
    - Release day.
- **Roles:**
    - Project team.
    - Support group.
    - Customer group.
    - Steering group.

iv. **Stabilize**:
The purpose of the Stabilize phase pattern is to ensure the quality of the implementation of the project.

- **<u>Goal</u>:**
    - Finalize the implementation of the product.
    - Enhance and ensure the quality of the product.
    - Finalize the documentation of the product.
- **<u>Process</u>:**
    - Planning day.
    - Working day.
    - Documentation Wrap-Up.
    - Release day.
- **<u>Roles</u>:**
    - Project team.
    - Project team/the architect.
    - Support group.
    - Customer.
    - Steering group.

v. **<u>System Test Fix</u>**:

The purpose of System Test & Fix is to see if the produced system implements the customer defined functionality correctly, provide the project team feedback on the systems functionality and fix the found defects.

- **<u>Goal</u>:**
    - Test the system.
    - Provide information.
    - Allow the project team to plan fix.
    - Fix the defects.
    - Produce as error free system as possible.
- **<u>Process</u>:**
    - System test.
    - Fix.
    - Planning day.
    - Working day.
    - Documentation Wrap-Up.
    - Release day.
- **<u>Roles</u>:**
    - Project team.
    - Support group.
    - Customer.
    - Steering group.
    - System test group.

2.10 <u>Mobile Development Method Proposed by Rahimian & Ramsin</u>:

The mobile development methodology pro-posed by Rahimian & Ramsin in 2008 is created in four iterations and the beginning point is the generic software development lifecycle, as Analysis, Design, Implementa-tion, Testing. The first iteration details the methodology by adding those practices that are found within the agile methodologies. The second iteration details the process con-cerned with introducing new product/service on the market. In the third iteration, concepts of Adaptive Software Development are inte-grated into the methodology and in the final iteration the prototyping was added to miti-gate likely technology-related risks.

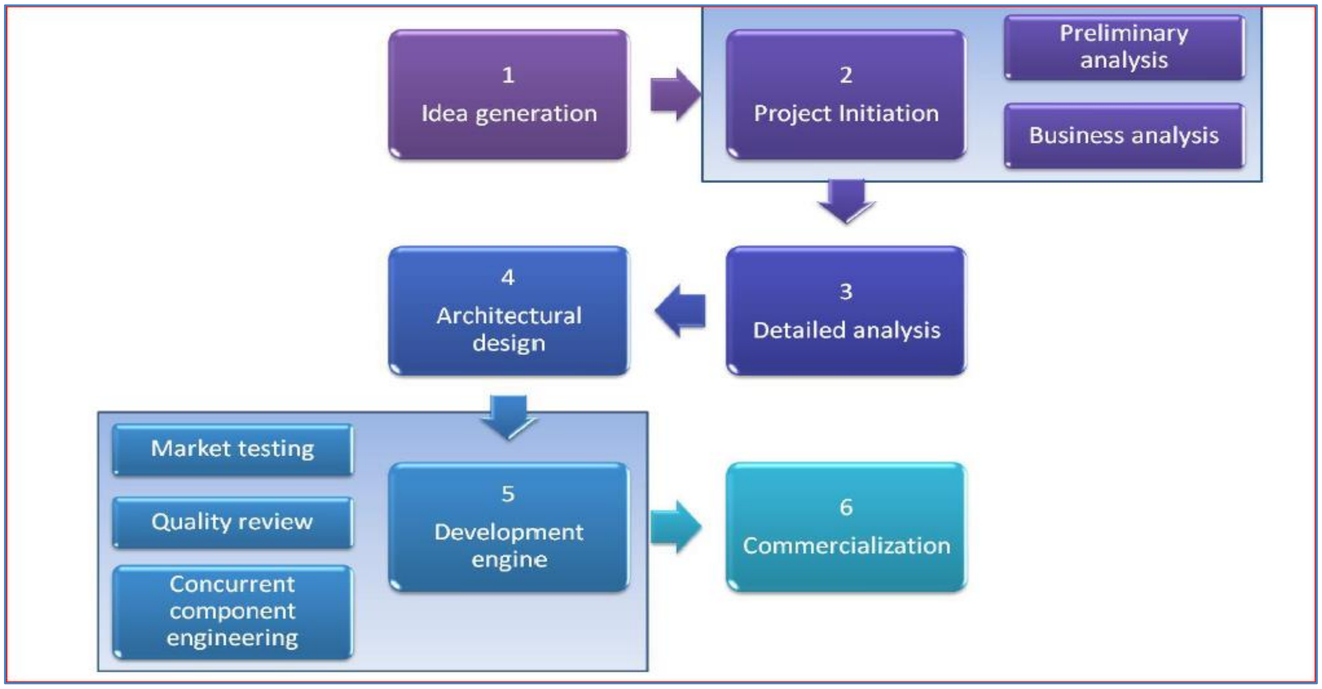


Figure 2.10: Adaption of the Mobile Development Method Proposed by Rahimian & Ramsin.

Chapter 3: Research Methodology

3.1 Research Type:

This paper was a research-based descriptive study. The key outcome of the investigation was the identification of App Development Process and discussion of their appropriateness to various types of App and platforms.

3.2 Research Methods Employed:

The primary research method employed throughout the course of writing this paper was browser-based Internet searches. The literature reviewed included textbooks, journal articles, and magazine articles referenced by a select set of online resources. The full text articles from journals and magazines were located and subsequently downloaded.

App development process important, demand, usefulness have identified to draw the model. Actor, action, activities have fined out and try to identify relation, communication, dependency and sequence work follow. Existing model that was more closer or could be applicable on App development have analyzed. Agile practice was the most perfect for making App development process model. Data has collected from various sources as like thesis, research, recent statistics, IT scholar opinions throw publications, survey and online. After data have been collected and doing data analysis we got findings. Then findings from various data have matched with each other; find out the similarities, deference, demand and gap. After all decision has taken.

3.3 Research Tools, Sources and Elements :

Research data has collected from various resources, analyzed using various tools and technique. Thesis paper, research paper, recent statistics collected from publications, download from online, survey over IT scholar who has particular working experiences and particular working area.

As example,

-Project manager	-Team leader	- Programmer	-Designer
-System Analyst	-Quality Tester	-Student	-Teacher

Working experience in on average five (5) years, where particularly has one (1) year to 10 years.

-Thesis papers	-Publications	-Recent Statistics	-Books
-Magazines	-Online Articles		

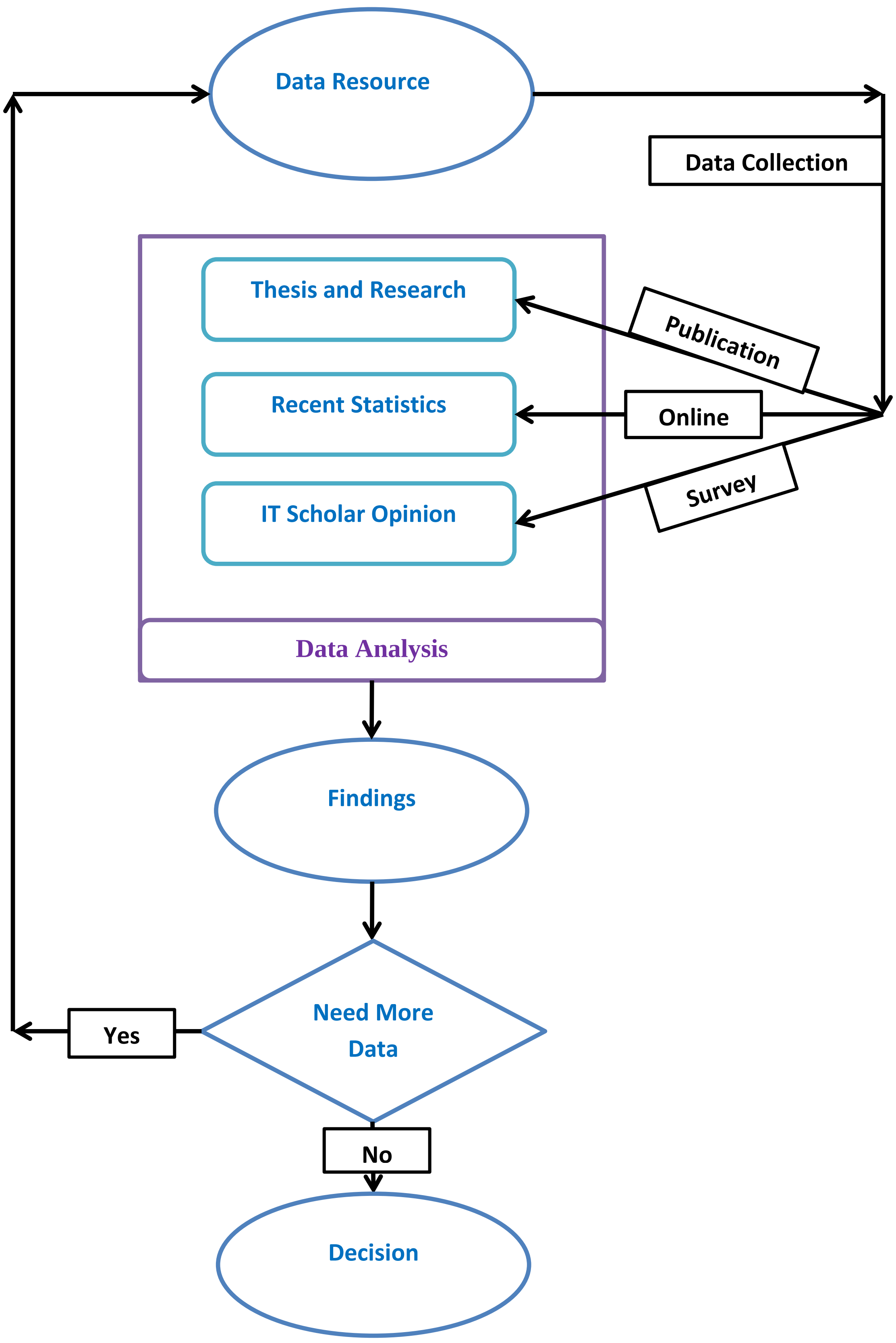


Figure 3.1: Data collection, Analysis & Finding Work Flow.

3.4 <u>Research Question</u> :

Question 1:

App Development Process Model, how much effective to manage project, develop within budget & time, delivering successfully and making pleaser user/client?

Question 2:

What are the elements of App Development Process Model? How should elements perform activities, communicate with these and execute them?

Question 3:

What are the expected outputs from App Development Process Model?

3.5 <u>Evaluating App Development Process Model</u> :

Every development processes are complete with some actors and actions. Process model describe these actions how should perform by the respective actors. Which method should be obeyed by the actors to execute the actions. How should communicate with actors to understand inter dependability of actions. How should manage actions from start to end and maintenance from before and after deploy.

Actor of App Development Process Model:

-Team leader	-Requirements engineer	-System analyst
-Designer	-Programmer	-Quality tester
-User/Clients		

Action of App Development Process Model:

-Project manage	-Requirements specification	-System analysis
-Design	-Codding	-Testing
-Feedback	-Communication	-Maintenance
-Documentation		

How should manage action and actor?

How should communicate with action and actor?

What the role over the action by actor?

How depend or independent from actions and actors?

Consideration of App Development Process Model:

-Mobile environment	-Screen size	-Memory size
-Processor Speed	-Power supply	-UI
-Hardware capability	-Mobility	-API
-SDK	-running Platform	-Type of App
-small Budget	-short Time of development	-small Team
-User experience	-Developer experience	-Expert guideline
-Scholar advise	-Technical Suggestion	-Agile practice
-Development Method	-Limitation/problem	-Efficiency
-Reliability	-Security	-Risk management

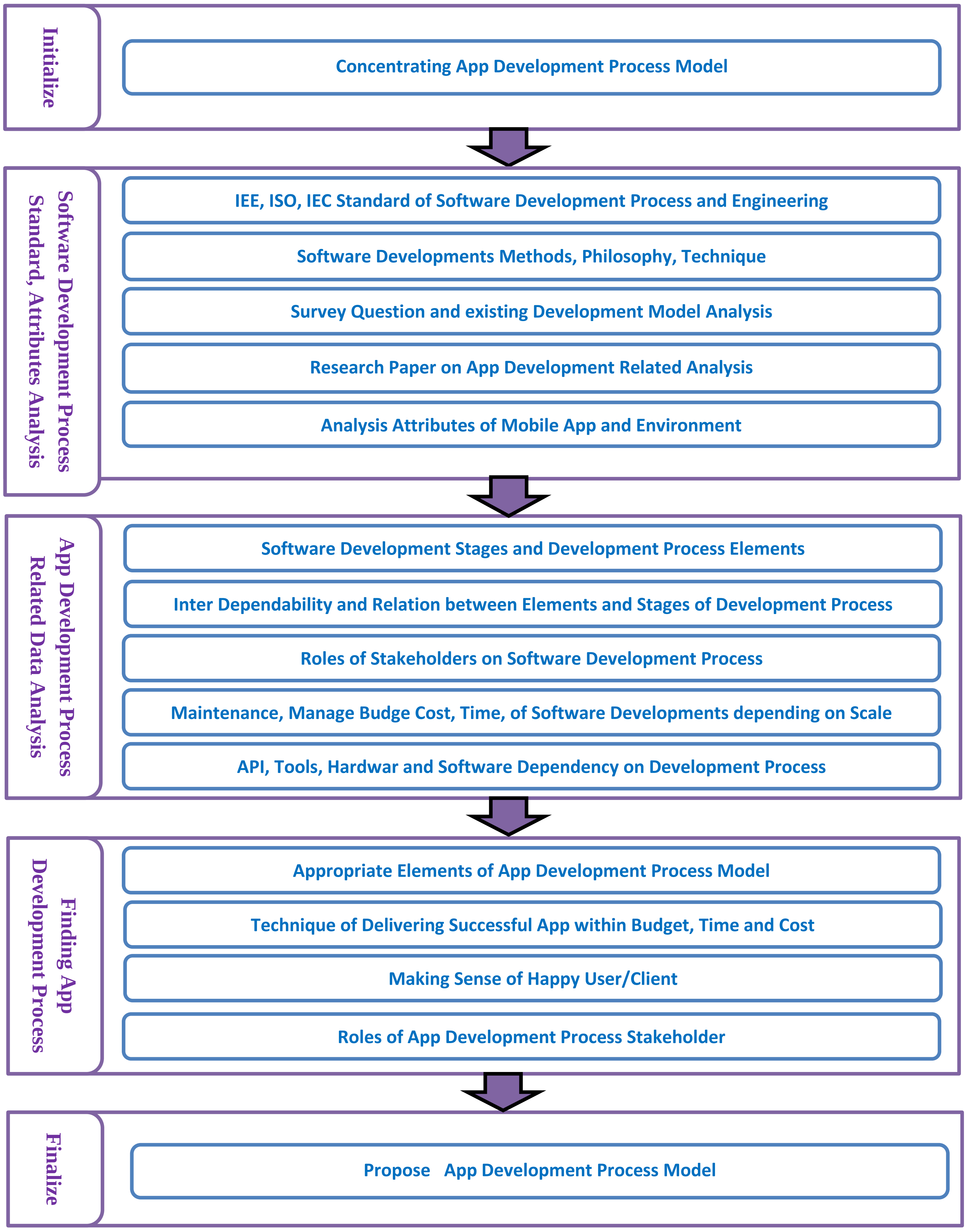

Initialize
Concentrating App Development Process Model
Software Development Process Standard, Attributes Analysis
IEE, ISO, IEC Standard of Software Development Process and Engineering
Software Developments Methods, Philosophy, Technique
Survey Question and existing Development Model Analysis
Research Paper on App Development Related Analysis
Analysis Attributes of Mobile App and Environment
App Development Process Related Data Analysis
Software Development Stages and Development Process Elements
Inter Dependability and Relation between Elements and Stages of Development Process
Roles of Stakeholders on Software Development Process
Maintenance, Manage Budge Cost, Time, of Software Developments depending on Scale
API, Tools, Hardwar and Software Dependency on Development Process
Finding App Development Process
Appropriate Elements of App Development Process Model
Technique of Delivering Successful App within Budget, Time and Cost
Making Sense of Happy User/Client
Roles of App Development Process Stakeholder
Finalize
Propose App Development Process Model

Figure 3.2: Evaluating App Development Process Model.

3.6 Expectation from App Development Process Model :

Expected output results are considerable for validating the App Development Process Model. There are some parameters that could be consider for get output of expectation, here some questions are mention for getting expected output.

Expected Parameters are:

1) -save development time?
2) -Minimize Cost?
3) -Minimize Risk?
4) -Better Management Performance?
5) -Increase team Confidence for Successfully Development & Delivery?
6) -Quick Problem Identify and Quick Solve?
7) -Clear Requirements Conception?
8) -Minimize Bug?
9) -More Quality Products can Delivery?
10) -Easy to Add-Drop Team member incase?
11) -Easy to Re-Engineering?
12) –User/Client Happiness over the Development & delivery?
13) –Pleaser all Member over the Development?

Chapter 4: App Development Process Model

4.1 App Development Process Model:

Apps Development Process Model is including four steps or phases of development process and two supporting stages or activities. On this model development process is a team work with small number of team members. Where team members are including one team leader, one or more Requirements engineer, System analyst, Designer, Programmer, Tester and User/Client. One person may be able to play more than one role and more than one person may be able to play one role exception team leader role. Team member may be co-located or stay unique location. Team leader do interaction with all the team members with all together meeting or one by one depending on respective job. User/client gives feedback about product. The team work together for delivering reliable and quality App on possible shortest time. Development steps or phases will be executed paralleling one step following next step and can move one phase to another phase one by one. Every elements of this model are synchronous inter dependable.

4.1.1 Two Supporting Stages or Activities are:

i. **Manage and Maintenance.** (by team leader)
   -Time.
   -Cost.
   -Risk.
   -Method.

ii. **Inspection.**
   -Continuous Testing.
   -User/Client Feedback.
   - Documentation.

4.1.2 Four Steps or phases are:

i. **Plan.**
   -Requirements Specification.
   -Platform Consideration.
   -System Analysis.

ii. **Design.**
-UI (User Interface).
-Logical.
-Functional.

iii. **Build.**
-Coding.
-Prototyping.
-Integration.

iv. **Approve.**
-Testing.
-Security.
-Quality Assurance.

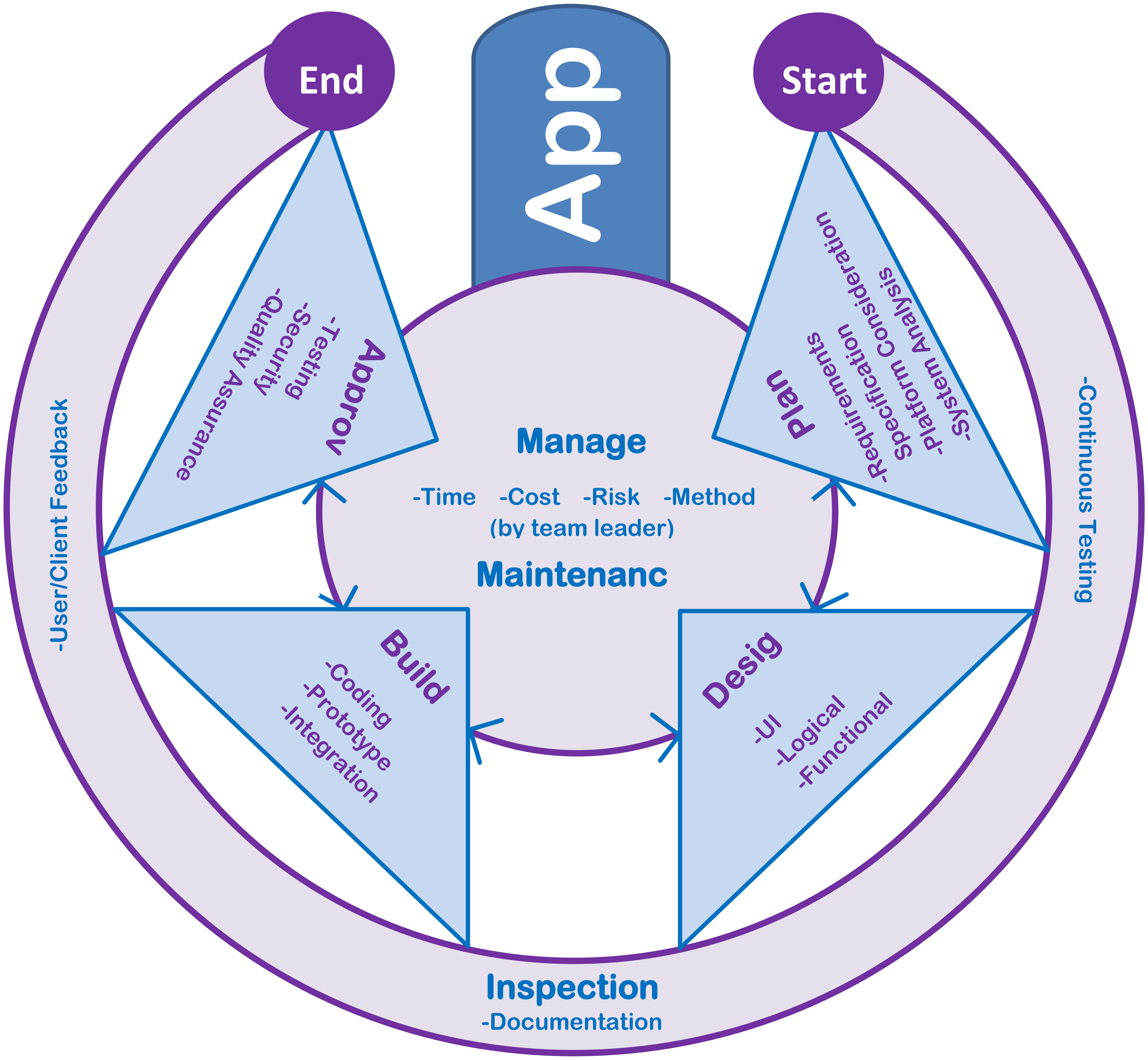


Figure 4.1: App Development Process Model (Proposed).

Steps or phases and supporting stages or activities of App Development Process model are described below.

4.2 <u>Supporting Stages or Activities Description</u> :

There are two stages or activities of this App development process model. Those are given below.

4.2.1 <u>Manage and Maintenance</u>:

-Time -Cost -Risk -Method (by team leader).

There is a team leader of App development project team members. Team leader be always dedicated with other members and user/client of App project and be responsible over the project. Team leader start the project after business agreement with user/client or discovering demand from market analysis. Collecting requirements and system analysis team leader plan to development and select other members. Team leader portray about the App to other members. Team members play their role according to their job and team leader get update from them. Requirements engineer, system analyst, designer, programmer, tester perform their job unerringly and precisely following requirements. One person may be able to doing more than one job as like one person can be requirement engineer, system analyst and designer or something else. Other hand more than one person can be able to doing one job as like three persons can be programmer or something else.

More than one steps or phase can go paralleling under construction following relation among the step or phase. As like design can be initializing following requirements specification after few parts has completed and at a time coding can start when design has completed some parts, during this time requirements analysis may be completed. Ongoing under construction every steps or phases observed the process by user/client and should give feedback to team leader. If come any change from user/client feedback then team leader describe about feedback to responsible member. Following this if any change come on coding step there have to require change on requirements. Under this circumstance backward throw previous phase design to requirements and vice versa requirements to coding throw design phase.

If any technical limitations come out from system analysis or any other stage then team leader discus about it with user/client and try to show alternative solution. Team leader be interactive with other team members over the development process to manage every things and changing feedback. Team leader also be responsible for service and maintenance before and after App deploy.

Managing development process team leader can use Kanban board for orchestrate time, cost and risk. Scrum product back lock very helpful for managing risks and scrum sprint time can help to manage time. Team leader can observe working stage following this sprint. Identifying problem early stage saves cost and time. Meeting with team members for discuss about important issue of development process. That can identify problem early stage. Team members remain on co-located or individual located; they can attend meeting physically or virtually.

4.2.2 **Inspection:**

-Continuous Testing -User/Client Feedback -Documentation.

Documentation, user/client feedback and continuous-testing are process from start to end of App development. Testing is not only after coding or integration. Continuous-testing starts from requirements analysis as well its feasibility study. Is it feasible? Can we draw relation between functionality? Those are one type of testing and have large impact on designing App. To delivering unerring App on budgeted cost and time proper design is very important that comes from testing functional relation properly. If any functional test conflict with any requirements then team leader can discus with user/client then they can take proper decision early stage of the development. It increases reliability, confidence, and possibility. Minimizes cost and save time and assume initial threat. User/client can observe the working process and can give changing feedback and team leader can negotiation with user/clients.

Every final decision, running work and changing feedback should be documented. Documentation starts from agreement. In every step, what has decided what is going on and what change has come? Every record should be documented. Design of apps with low level and high level, coding method that has followed and quality assurance result also should be documented. App user manual is also a part of documentation. Documentation can go sign-off from both side team leader and user/client. In any odd situation both side authority (user/client or project development) can take action according to law following documentation.

4.3 **Developing Steps or phases Description :**

There are four Developing Steps or Phases of this App development process model. Those are given below.

4.3.1 **Plan :**

-Requirements Specification –Platform Consideration –System Analysis.

App requirements deals with establishing the need of user/client or market that are to be performed by the App. IEEE standard define requirements as “A condition or capability need by a user to solve a problem or achieve an objective”. Requirements specification is a description of an App to be developed, define functional and non-functional requirements. Requirements specification establishes interaction and agreement between user/client (in market analysis-driven development project, this role played by market analyst or RnD team) on what the App is to do as well as what is not expected to do. It also provide realistic basis for designing and estimating product cost, risk and schedules. To refine and achieve the success user/client and developer team should continuous communication over the development the App project with requirements.

User/client gives requirements about App that they demand. Sometime tell story about their demand or their working process that we called user history. User history is description of requirement. Sometime concept of App comes from market demand analysis. Sometime organization, person or company wishes to release their own App.

System analyst has to prudent about App type, platform, OS version, Data authentication and authorization, hardware and network resources and also regional government code. Native App develops for its native environment with well-known native programming language, API, IDE and SDK tools. As like Android is native for java programming, Android Studio IDE, API level according to Android version and SDK testing environment according handheld smart device like mobile phone, tablet, smart TV and even more smart watch with memory size, screen size, resolution and processor speed. Highbred or cross platform App is issue of more system analysis. Mobile web application develops considering mobile device environment. Now a days responsible web applications are being more popular that response following screen size and change menu following PC and mobile device.

System analysis also goes under passing feasibility testing of data and its relation. What have to develop and what functionality has to perform? Identifying actor and their action and making UI planning to make with each other. Every things should be documented and get sign-off from client.

4.3.2 Design :

-UI (User Interface) –Logical –Functional.

Designing is a process to transfer user requirements into artifact and visible structure, which help the programmer in coding, team leader in implementation and re-engineering easily. There is need of more specific and detailed requirements in App designing. The output of this process can directly be used into implementation in programming language. So user/client has to give more attention on this architecture and data relation design to be insure his demand.

The designer design the App observant about Environment, API platform, device power, size, resolution, memory capacity where the App will execute. Designer has to identify actor and action by drawing Use Case Diagram (UCD). Design is visible architecture of what to achieve and how to achieve by the App. High level design break the system single-multiple component concept of architectural design into less-abstracted view of sub-system, method and depicts their interaction with each other. High level design focus on how the system along will all of its components implemented in form of modules. State Transaction Diagram (STD), Context Diagram represents the high level design. Low level design is known as detailed design. Details design deals with implementation part of what is seen as a system and its sub-system in. It is more detailed towards modules, functions and their implementations. It defines logical structure to communicate with other modules. Data Flow Diagram (DFD), Entity Relation Diagram (ERD) represents the low level design. A sub-system has different functional parts. During designing functional parts have to be specific for incremental development of App part by part. Then a functional sub-system will be a deliverable product for review of user/client and giving feedback.

User interface (UI) design is one of the biggest challenges of App development. To interact with user, giving physical data entry and mind manipulation done by UI of App. User Experience (UX) has a great impact on UI designing. App UI design should become acquainted according to type of device such as smart TV, Phone, Tab and even more smart watch. Icon view, theme animation and its drawable customization, status and navigation bars and notification style are also part of UI design.

UML (Unified Modeling Language) tools should use for graphical representation of Use Case Diagram (UCD), State Transaction Diagram (STD), Context Diagram, Data Follow Diagram (DFD), Entity Relation Diagram (ERD), User Interface (UI) and other graphical view of APP design. There for user/client can assume the action of App. They can give feedback depending on that and also helpful for programmer and team leader for re-engineering.

Object-Oriented design, Top-Down, Model-Driven design, Structured Method and other designing concept can be followed by designer. Break the system into possible smaller sub-system and show interaction between inter the classes and external entities with classes. It increases efficiency of the App and re-engineering of design. Re-engineering is the vital issue of Agile philosophy and help to make happy user/client. During designing if any requirements conflict logically designer will contract with team leader and share own experience to solve the problem. Every process of design should be documented with detail description.

4.3.3 **Build:**

-Coding -Prototype -Integration.

Coding is physical design of App. According to logical design programmer write code in device executable language. Usually native App has native language like Java for Android, Objective C for iOS and C# for windows phone. Programmers usually use IDE for coding and integration. Android Studio, Aptana Studio, Visual Studio, Eclipse, Net Beans are popular to programmers. Programmers write code in different packages for a App. Though they are co-located or situated on unique-location. Programmers should follow coding principal like Unit Test Development, Test-Driven Development or Continuous Integration Technic or others to writing code and integrating the packages. Unit test is for test of every unit of method, class and every functional unit according to design and requirements. Every functional part has to integrate on following package. During continuous integration of functional part of incremental development on package programmer have to test every integrated functional part. Then a functional part comes as deliverable product. User/client can give their feedback and feel happy. It is very good practice to give prototype for better understand for non-technical user/client. Programmer should write code by giving code reusability, object-oriented and KISS (Keep It Simple Stupid) to increase the efficiency and Agility of code. Changing feedback is also welcome on coding phase.

Programmers are free to choose their programming method like XP (extreme programming), Pair Programming, Cowboy Coding or other they like most. Doing physical design or writing code if occur any conflict with logical design programmer will discuss with team leader and should give better suggestion.

4.3.4 **Approve:**

-Testing –Security –Quality Assurance.

App quality assurance is check the development standard and is it meets the requirements that guided its design. Testing functionality, usability, performance, consistency, efficiency and performance assure the quality of App. Quality assurance increase customer confidence, reliability and company’s creditability. To insuring the quality of App have to test is it computable with targeted device screen size, resolution, memory capacity, sensor and dependent hardware and resources. Changing orientation of display as landscape or portrait view screen should re-draw correctly and UI (User Interface) should look like same. Maintaining its state, activity should not loss data or anything that user has already entered into the UI. The App should not ‘forget’ its place in the current transaction. Mobile device primarily run on battery power. Without charge mobile has gone useless. Develop App to minimize battery use and should test battery performance follow the method that manages battery use during design. Device configuration change in availability of key board or changing in system language, App should update itself to response correctly to the new changing configuration. If App depends on external resources like network access, GPS, Bluetooth and others then should test what happens when the resource is not available. Native App or highbred or cross platform App should also pass some parameter such as response correctly to all kinds of inputs. Perform its function within an acceptable time. Insuring the security issue confirm that the App is not containing any spam or malware coding. App security threat, vulnerability, bug should be should be insure testing properly. To insuring above testing tool and emulator have to use.

To achieve the general result its user/client desire has to document every testing with result. Also have to documented which method, model, framework has used and which ISO standard has followed. Documentation should gone sign-off. According to this documentation user/client or company can take lawful action any need occur. So the documentation make confident and smile both company and user/client.

4.4 <u>Case Study</u> :

In this case an android App has taken for case study. “Easy Wallet” (Income and Expense Manager) android App has developed following proposed App Development Process Model. With a small Team was responsible of this App Development. Agile Scram method was adopted here as method of development.

Easy Wallet (Income and Expense Manager) App:

This app is an android native App, Which works on offline environments. Potential users need to input the required data such as the expense amount, merchant, category, and date when the expense was made. Optional data such as sub-category and extra notes about the expense can be entered as well. The application allows users to track their expenses daily, weekly, monthly, and yearly in terms of summary, bar graphs, and pie-charts. This mobile application is a full detailed expense tracker tool that will not only help users keep a check on their expenses, but also cut down the unrequired expenses, and thus will help provide a responsible lifestyle. An analysis comparing existing expense tracking software with the one being introduced is provided.

Requirement Analysis:

- The system will be a offline mobile application.
- User will entry their income and regular expense.
- There will be different types of category to entry the income and expenses.
- User can add, delete category as per their requirements.
- User can see their income & expense details.
- There will be graph report of income & expense details.

Use Case Diagram:

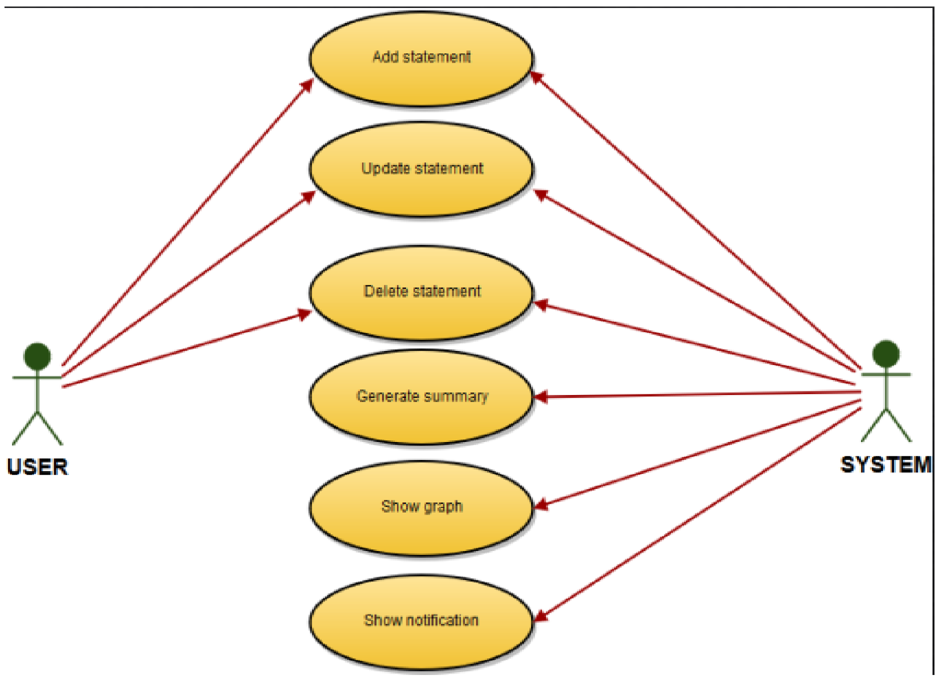


Figure 4.2: Easy Wallet use case diagram.

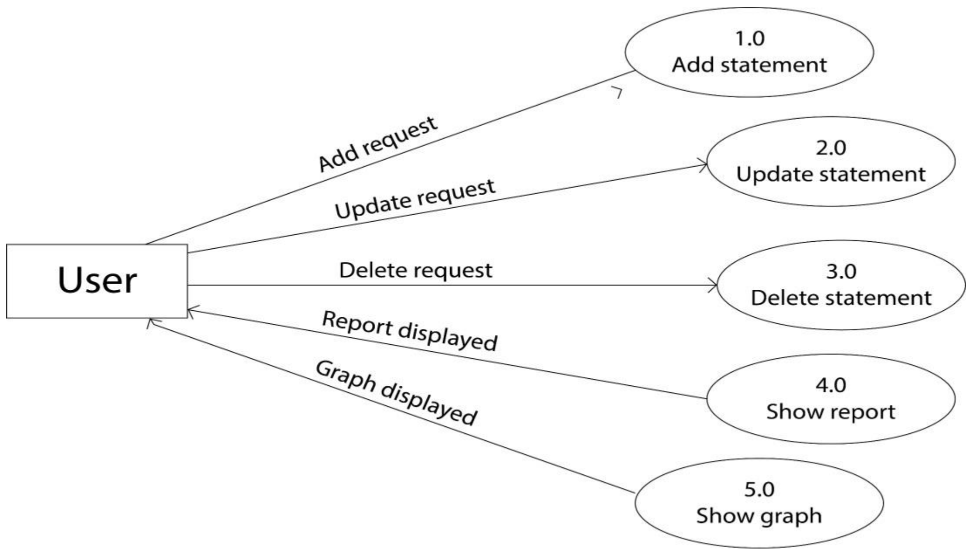
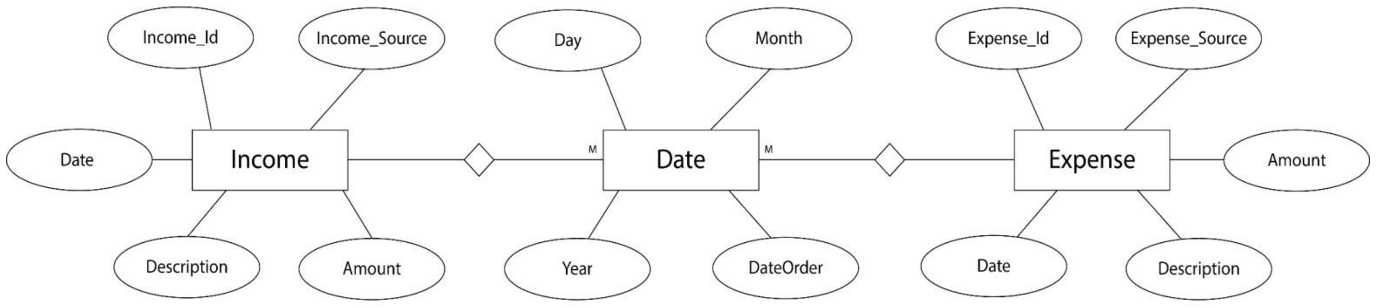


Figure 4.3: Easy Wallet data flow diagram.

ER Diagram:



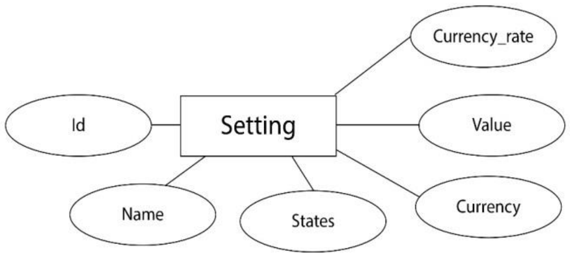


Figure 4.4: Easy Wallet ER diagram.

4.5 Case Study Analysis :

Output of this case study has analyzed by some parameters of expectation. There are four answers for giving result of efficiency of this App Development Process Model. The multiple choice system questions for answer not good, average, better and excellent from various parameters.

Question Parameters are:

1) -save development time?

- Not-Good
- Average
- Better
- Excellent

2) -Minimize Cost?

- Not-Good
- Average
- Better
- Excellent

3) -Minimize Risk?

- Not-Good
- Average
- Better
- Excellent

4) -Better Management Performance?

- Not-Good
- Average
- Better
- Excellent

5) -Increase team Confidence for Successfully Development & Delivery?

- Not-Good
- Average
- Better
- Excellent

6) -Quick Problem Identify and Quick Solve?

- Not-Good
- Average
- Better
- Excellent

7) -Clear Requirements Conception?

- Not-Good
- Average
- Better
- Excellent

8) -Minimize Bug?

- Not-Good
- Average
- Better
- Excellent

9) -More Quality Products can Delivery?

- Not-Good
- Average
- Better
- Excellent

10) -Easy to Add-Drop Team member incase?

- Not-Good
- Average
- Better
- Excellent

11) -Easy to Re-Engineering?

- Not-Good
- Average
- Better
- Excellent

12) –User/Client Happiness over the Development & delivery?

- Not-Good
- Average
- Better
- Excellent

13) –Pleaser all Member over the Development?

- Not-Good
- Average
- Better
- Excellent

Chapter 5: Conclusion and Recommendation

5.1 Conclusion:

Every development processes are complete with some actors and actions. Process model describe these actions how should perform by the respective actors. Which method should be obeyed by the actors to execute the actions. How should communicate with actors to understand inter dependability of actions. How should manage actions from start to end and maintenance from before and after deploy.

A Mobile App Development Process Model has been proposed to bring out a formal lifecycle for mobile application development. App Development Process Model will aid the mobile application developers in developing high-end apps. This App Development Process Model includes the following phases: Plan, Design, Build, and Approve. There are two supporting activities are: Manage & Maintenance and Inspection.

This App Development Process Model also addresses some of the distinguishing characteristics of mobile applications like life span, complex functionalities, fewer physical interfaces, more number of screens for interaction, battery and memory usage, cross platform development and maintenance. Preliminary testing of the App Development Process Model indicates that this App Development Process Model will help developers and project managers in efficiently execute projects and deliver solutions on time.

The thesis has attempted to improve mobile development best practice by considering a popular and mature development methodology and improving it using ideas based on relevant literature. This was done after analyzing the soundness of the fundamental principles of the methodology, and the way these principles fit with mobile development projects. The final major improvement approach was the development of a support tool for mobile developers, which tries to fill a gap in current tool support, as well as sustain the proposed improvements to the methodology.

An important goal of the thesis was to obtain a methodology that is as close as possible to the ideal and end user feedback support, application categories and software product lines principles, App Development Process Model now presents more of the ideal characteristics. Furthermore, the methodology was improved in other aspects not mentioned in the list, such as the inclusion of performance testing in the lifecycle. Another important contribution was the support tool, that not only enables corresponding features of App Development Process Model, but decreases overall development effort. Some characteristics of the tool may even be used outside mobile development; features such as usage logging, log analysis and test case generation can be used in any development scenario, with only minor adjustments to the support tool.

5.2 **Future Works:**

1. App UI (User Interface) Design According to UX (User Experience).
2. App Designing for Re-Engineering.

References:

1. Tejas Vithani and Anand Kumar: “Modeling the Mobile Application Development Lifecycle”.
   -Proceedings of the International MultiConference of Engineers and Computer Scientists 2014 Vol I, IMECS 2014, March 12 - 14, 2014, Hong Kong

2. Barry Boehm: “ A Survey of Agile Development Methodologies.”

3. Andrei Cristian Spataru: “Agile Development Methods for Mobile Applications”.
   -2010

4. AS/NZS ISO/IEC 12207:1997: “Information technology—Software life cycle processes”.

5. ISO/IEC 12207:2008(E), IEEE Std 12207-2008: “Systems and software engineering Software life cycle processes”.

6. Raghu Singh: “International Standard ISO/IEC 12207 Software Life Cycle Processes”.

7. *SWEBO: “* Guide to the Software Engineering Body of Knowledge (Version 3.0)”.

8. James W. Moore: “IEEE/EIA 12207 as the foundation for enterprise software processes.”

9. Robert C. Martin: “Professionalism and Test-Driven Development.”

10. Brian Nielsen and Arne Skou,: “Test Driven Development”.

11. [Online]: http://guide.agilealliance.org/guide/tdd.html

12. Lech Madeyski1 and Marcin Kawalerowicz: “Continuous Test-Driven Development—A Novel Agile Software Development Practice and Supporting Tool”.

13. [Online]: https://www.crisp.se/gratis-material-och-guider/kanban

14. Denis Duka, Lovre Hribar: “Test Driven Development Method in Software Development Process”.

15. Luis Corral, Alberto Sillitti, Giancarlo Succi: “Software Development Processes for Mobile Systems”.

16. Vasile-Daniel: “Methodology Approaches Regarding Classic versus Mobile Enterprise Application Development”.

17. Andrei Cristian Spataru: “Agile Development Methods for Mobile Applications”.

18. Melissa L. Russ and John D. McGregor,: “A Software Development Process for Small Projects”.

19. Ridi Ferdiana,: “Agile Software Engineering Framework for Evaluating Mobile Application Development”.

20. Nabil Mohammed Ali Munassa1 and A. Govardhan: “A Comparison Between Five Models Of Software Engineering”.

21. Tejas Vithani: “Modeling the Mobile Application Development Lifecycle”.

22. Intel IT: “Building a Mobile Application Development Framework”.

23. Harlin K. Flora, Swati V. Chande, Xiaofeng Wang: “Adopting Agile Approach for the Development of Mobile Application”.

24. Harlin K. Flora, Dr. Swati V. Chande: “A Review and Analysis on Mobille Application Development Processes Using Agile Methodologies”.

25. Venkata N Inukollu, Divya D Keshamoni , Taeghyun Kang and Manikanta Inukollu: “FACTORS INFLUENCING QUALITY OF MOBILE APPS: ROLE OF MOBILE APP DEVELOPMENT LIFE CYCLE”
- International Journal of Software Engineering & Applications (IJSEA), Vol.5, No.5, September 2014

End