Suman Kumar Mukherjee

Concept of automated testing of consumer electronic devices just before market entry.
PREFACE

The research started in 2012, to study current impediments causing schedule slippage and increasing 'time to market' for finished products and how the automated tools and methods are helping to address the needs. Manual testing and releasing software was the main process for releasing products from the case company, more or less same process is followed by many units for releasing phone software, so the research began with a study about the usage of manual process followed in different organizations. The main objective of the research was to analyse the current way release testing and identify the ways that are needed to reduce delays in the project through thorough automation.

While the research moved on with the study and discussion with many individuals in the team, it turned up to be a great opportunity to learn scripting tools, automation frameworks readily available to be used in automating release and integrations, all these valuable information shared in the discussions. Also with several years of experience in the industry, in different kind of roles, I gained profound knowledge and best hands on opportunity to express and relate the best ways of using the tools and methods, which would help any individual in software development organizations similar to the one in the case study.

The research should give a fairly good view of best practices in test automation methods and tools for Windows phone environment, and how can automation help in release process of software, correcting missing practices causing wait time and reduce 'time to market'. It brings in a clear picture of applying lean six sigma methodologies in identifying faults inside processes followed inside organization. The solution provided is very much helpful for identifying better, faster feedback to all the stakeholders of software development life cycle with appropriate details to act quickly on any type of failure as and when detected.

This research would have not been possible to complete without many individuals, whom I am sincerely thankful especially my thesis supervisor Thomas R and case company manager Arun J.

Helsinki, November 30th, 2014
Suman Kumar Mukherjee
<table>
<thead>
<tr>
<th><strong>Author</strong></th>
<th>Suman Kumar Mukherjee</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Title</strong></td>
<td>Concept of Automated Testing of Consumer devices before market entry</td>
</tr>
<tr>
<td><strong>Case Company</strong></td>
<td>Leading Manufacturer in Phone Business OY</td>
</tr>
<tr>
<td><strong>Date</strong></td>
<td>30th. November 2014</td>
</tr>
<tr>
<td><strong>Number of Pages</strong></td>
<td>92</td>
</tr>
<tr>
<td><strong>Degree</strong></td>
<td>Master Of Engineering</td>
</tr>
<tr>
<td><strong>Degree Programme</strong></td>
<td>Master’s in Business Informatics</td>
</tr>
<tr>
<td><strong>Specialisation option</strong></td>
<td>NA</td>
</tr>
<tr>
<td><strong>Instructor(s)</strong></td>
<td>Thomas Rohweder, Arun J</td>
</tr>
<tr>
<td><strong>Keywords</strong></td>
<td>Consumer Electronics, Telecommunications, Automation Framework, Continuous Integration Test Automation</td>
</tr>
</tbody>
</table>
## Contents

**PREFACE**

1 Introduction
   1.1 Case Company Context 8
   1.2 Business problem, objective of thesis, expected outcome 11
   1.3 Delimitations 14

2 Methods and Materials
   2.1 Research design & research process 17
   2.2 Data collection & data analysis 21

3 Current state analysis inside case company
   3.1 Analysis of current business problem 24
   3.2 Current release testing for product programs 25
      3.2.1 Continuous Integration overview 25
      3.2.2 Definition of Continuous Integration 25
      3.2.3 Development in Continuous Integration in Case company 26
      3.2.4 Continuous Integration in Case Company 27
      3.2.5 Benefits of Continuous Integration 29
      3.2.6 Disadvantages of Continuous Integration 29
      3.2.7 Continuous Integration Software Tools 29
   3.3 Software Testing Methodologies incase company 33
      3.3.1 Component testing 35
      3.3.2 Integration testing 36
   3.4 Benefits, strengths and weaknesses 37

4 Best Practices of Automated testing in literature 40
   4.1 Black box vs Whitebox testing 41
   4.2 Why testing is important 41
   4.3 Verification 41
   4.4 Validation 41
   4.5 Continuous Integration Software Tools 41
   4.6 Types of testing and opacity 42
      4.6.1 Unit testing 42
      4.6.2 Integration testing 42
      4.6.3 Functional & System testing 42
      4.6.4 Stress testing 42
      4.6.5 Performance testing 43
4.6.6 Usability testing 43
4.6.7 Acceptance testing 43
4.6.8 Regression testing 43
4.6.9 Beta testing 44
4.7 Automation testing framework 45
4.7.1 Test automation framework introduction 45
4.7.2 Advantages of test automation 45
4.7.3 Disadvantages of test automation 46
4.8 Test automation tools 47
4.8.1 Perforce 47
4.8.2 Jenkins 49
4.8.3 Visual Studio Ultimate & MS Build 49
4.8.4 Test build solution 50
4.8.5 Test framework Utility 51
4.9 Summary & Conclusion 52
5 Build proposal for automated testing in case company 54
5.1 Test Automation and Laboratory facilities 55
5.1.1 Importance of CI & Test automation 55
5.1.2 CI & lab infrastructure 56
5.2 Proposal for case company 57
5.2.1 Unit testing 57
5.2.2 Functional testing 59
5.2.3 Certification testing 62
5.2.4 Dashboard and Crash Portal 64
5.2.5 MTBF testing for all application releases 65
6 Testing the prototype proposed in case company 69
6.1 Introduction to testing methodology in case company 70
6.1.2 Validation of proposal by lean six sigma methodology 71
6.2 Feedback & learning 78
6.2.1 Feedback 78
6.2.2 Learning for case company 79
6.3 Summary and final proposal for case company 80
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Conclusions</td>
<td>83</td>
</tr>
<tr>
<td>7.1</td>
<td>Summary</td>
<td>84</td>
</tr>
<tr>
<td>7.2</td>
<td>Practical Implications</td>
<td>84</td>
</tr>
<tr>
<td>7.3</td>
<td>Evaluation</td>
<td>85</td>
</tr>
<tr>
<td>7.3.1</td>
<td>Objective Vs Outcome</td>
<td>85</td>
</tr>
<tr>
<td>7.3.2</td>
<td>Reliability</td>
<td>86</td>
</tr>
<tr>
<td>7.3.3</td>
<td>Validity</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td>References</td>
<td>88</td>
</tr>
<tr>
<td></td>
<td>Appendix</td>
<td>90</td>
</tr>
</tbody>
</table>
1. Introduction
1.1 Case Company Context

My research topic is concept of automatic testing in consumer electronics products before market entry e.g. shipping and selling of products in retail shops. Consumer electronics (abbreviated CE) are electronic equipment intended for daily purpose usage, most often entertaining, communicating and official productivity enhancements. Earliest of the very useful and popular product has been radio receiving systems. Nowadays, products include personal computers, telephones, MP3 players, audio equipment, televisions, calculators, GPS automotive electronics, digital cameras and players and recorders using video media such as DVDs, VCRs or camcorders. Increasingly these products have become based on digital technologies, and have largely merged with the computer industry.

Consumer electronic products tend to have ever-falling prices over a period of time. This is driven by gains in manufacturing efficiency and automation, lower labor costs as manufacturing has moved to lower-wage countries, and improvements in semiconductor design. There is an ever increasing need to keep product information updated and comparable, for the consumer to make an informed choice. Style, price, specification and performance are all relevant.

Out of consumer electronic products I narrow out my scope of work to telecommunication domain. Telecommunication in the modern era is the science and practice of transmitting information by electromagnetic means. A revolution in wireless telecommunications began in the first decade of the 1900s with pioneering developments in radio communications. A revolution in wireless telecommunications began in the first decade of the 1900s with pioneering developments in radio communications by Nikola Tesla and Guglielmo Marconi. Marconi won the Nobel Prize in Physics in 1909 for his efforts. Other highly notable pioneering inventors and developers in the field of electrical and electronic telecommunications include Alexander Graham Bell (telephone). The world's effective capacity to exchange information through two-way telecommunication networks are below in table form:
<table>
<thead>
<tr>
<th>Year</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1986</td>
<td>281 petabytes</td>
</tr>
<tr>
<td>1993</td>
<td>471</td>
</tr>
<tr>
<td>2000</td>
<td>2.2 exabytes</td>
</tr>
<tr>
<td>2007</td>
<td>65 exabytes</td>
</tr>
</tbody>
</table>

**National and International Facts:**

Given this growth, telecommunications play an increasingly important role in the world economy and the global telecommunications industry was about a $4.7 trillion sector in 2012. The service revenue of the global telecommunications industry was estimated to be $1.5 trillion in 2010, corresponding to 2.4% of the world’s gross domestic products.

Approximately three quarters of the world’s population now has access to a mobile phones. The number of mobile phone subscriptions has skyrocketed over the past 12 years. Fewer than 1 billion mobile subscriptions were active in 2000, while there are six billion subscriptions active today. Last year alone, mobile users downloaded more than 30 billion apps. Additionally, the vast majority of today’s mobile subscriptions (5 billion) are in developing countries, a sign of mobile technology’s importance in countries which haven’t gone through the expensive and difficult process of building land-based communications infrastructure. The report highlights mobile-based development programs in several countries, including the use of mobile payments in Kenya and job-search apps in Palestine. Mobile apps are also playing a role in disease response, education and monitoring government abuses throughout the developing world. Smartphones and wireless data packages are still out of the reach of most people. Therefore, many apps designed in developing countries are built as effective low-bandwidth solutions, often employing WAP, SMS and other technologies now considered “low-tech” by developers working with cutting-edge software and hardware.

I am working in a leading mobile manufacturer company in Espoo. This is a Finnish multi-national communications and information technology corporation. Its principal products are mobile telephones and portable IT devices. A significant portion of the revenue also comes from offerings of internet services including applications, games,
music, media and messaging. This company also offers free-of-charge digital map information and navigation services through its sole owned Navteq.

Company has around above 50,000 employees across 120 countries, sales in more than 150 countries and annual revenues of around €30 billion. It is the world’s one of the leading mobile phone maker over more than a decade. This mobile manufacturer company is a public limited-liability company listed on the Helsinki Stock Exchange and New York Stock Exchange. It is the world’s 143rd-largest company measured by 2011 revenues according to the Fortune Global 500.

The case company was the world's leader vendor of mobile phones from 1998 to 2012. However, over the past five years it has suffered declining market share as a result of the growing use of smartphones from other vendors, principally the Apple iPhone and devices running on Google's Android operating system. Case Company was the world's one of the largest vendor of mobile phones from 1998 to 2012. However, over the past five years it has suffered declining market share as a result of the growing use of smartphones from other vendors, principally the Apple iPhone and devices running on Google's Android operating system.

Today's telecom prudent generation is not taking sole interest in buying a cellular phone based on the attractive user interface. The mobile-hungry crowd especially the teenagers are more on the robust eco-system. Here is the catch, the more robust eco-system handset manufacturer is building and more rapidly the eco-system is built, more is the revenue earned. Case Company had put significant efforts to build Linux based mobile handset product from Meego platform has being a hot cake in the market, before the case company decided not to develop any more phones from that platform at all. The planned successors of Meego product did have advanced hardware, camera and more intrusive design changes to make the phone look sleeker and run multi-tasking faster in terms of speed. In mid-year 2011, the case company has cancelled its work on Linux platform totally.

Strategies and other quality statement:

Couple of years earlier, Case Company has gone long term agreement with Windows Operating system licensing. In that agreement Case Company has agreed that
from that time onwards their primary mobile phone devices from Case Company will have Windows operating system. Beginning of the year 2011 Case Company has seen a bit of tough time with sales and marketing of Windows operating system. After two quarters Case Company devices were shipped to the market and supply had risen up, but competitors also announced new products with even lesser price, which made the demand for Case Company products not so called hot cake in market. As Android and iOS products are already at matured phase, their cost of production for individual parts and assembling them together was very easy, whereas for Case Company to make the real Windows phone it had to do so much research, negotiate with other parties and it cost a lot to get to a stable position in initial days. In the last quarter 2012, Case Company has done profit in good margins and there has been significant increase in stock price of Case Company share values. Work is going on at a very fast pace inside Case Company to ship out new devices already which are in planned pipeline sometime this year. In terms of quality Case Company has always been the best, though it was bit sad that many of the earlier products (mobile handsets) came much late in the market than planned. Reading about the surveys from many new sources and telecom analysts it’s a careful observation that one of the major problem inside Case Company is devices coming late than planned. My thesis work primarily addresses on finding a very long term durable solution for this issue. Case Company products have always been of supreme quality whether be it terms of user experience, hardware and also multi-tasking. There has been some cases where battery life of the devices had some issues in US and UK. Case Company has replaced all the batteries to all the customers who faced that problem within a week. There was an announcement from Case Company specifying customers could visit any nearby retail shop and get this change done. The devices which are shipped to retail shops from Case Company undergo a set of rigorous testing over a very long period of time, in technical term it’s called Long Period Testing (LPT).

1.2 Business problem, Objective of Thesis and expected outcome

A product based company e.g. Case Company in telecom industry is remembered only by the amount, quality and quantity of the mobile handsets sold on quarter to quarter basis, then over an era year on year basis profit margins. Time to market and quality to market is vital in consumer electronics. One of the very key stages of the product development process in terms of time and quality is the testing stage of finished prod-
ucts before they enter the market place. Suitable automated testing tools which help manufacturers in this stage exist but unfortunately Case Company is not utilizing them. Without usage of proper automated testing tools, rapid development of code doesn’t add value to the company. Testing is equally important as of development. So in principal for every full-time developer there is a dedicated full-time tester allocated. The scope of my thesis is to establish a concept for automated testing to reduce time and quality slippage in the testing stage of finished consumer electronics before they enter the market using the case of Case Company mobile phones as a source when building this concept.

**Business problem definition:**

Time to market and quality to market is vital in consumer electronics. One of the very key stages of the product development process in terms of time and quality is the testing stage of finished products before they enter the market place. Suitable automated testing tools which help manufacturers in this stage exist but unfortunately all producers are not utilizing them. Supporting statements with figures:

Case Company’s once cluttered mobile trophy cabinet hasn’t had much silverware to boast of in recent years as the company has been moving from its legacy smartphone OS to Windows Phone, shedding massive amounts of market share in the process, and watching fleet-of-foot Android OEMs expand to fill the gap. Despite this smartphone switch, Case Company has continued to sell a lot of basic mobile phones — running its S40 OS (it shipped 76.6 million mobiles in its Q3, for instance) — which has helped it stay on top of the annual global cell-phone rankings. Case Company has been global leader in telecom sector for quite a decade e.g. 1990’s to 2007. Year 2007 first time Case Company noticed a tough competitive market for the first time in history of telecom industry. Before year 2007, Series 60 or Series 40 based SF phones manufactured and sold from Case Company has been almost only choice for all mobile consumers. Apple launched its cool product iPhone in the year 2007 and also Android mobile Linux platform have had been developed for quite a long time in a steady pace. Apple and Samsung are the two major competitors which emerged into the mobile market with their bunch of attractive products, and they have shown popularity in mobile market building their own eco-systems. The main problem with Case Company in past few years has been delayed delivery of finished mobile handsets to market. There have been many reasons for these series of delays where Case Company has prom-
ised for e.g. a promising attractive products coming before Christmas time but actual delivery has been in next summer. There has been cancellation of many of the variants of the mother products due to lack of resources of proper testing which could have added to significant revenue to Case Company. Despite of so much investment in architecture design, hardware manufacture, branched development and manual testing those products had to be cancelled at matured phases due to lack of testing. In parallel if we analyze Samsung’s success has been built on its ‘fast follower’ strategy for design and manufacturing and automated testing implementation -- which sees it fire out myriad devices across the price spectrum to build volume.

Samsung’s share of global smartphone shipments is expected to rise 8 points to 28 percent in the year 2013, up from 20 percent in 2011. In contrast, it said Case Company will suffer the biggest decrease: with its share forecast to plunge by 11 points to 5 percent in 2012, down from 16 percent in 2011.

A quick comparison of world’s leader smart device manufacturers:

<table>
<thead>
<tr>
<th>Rank</th>
<th>Company</th>
<th>2011</th>
<th>2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Samsung</td>
<td>24%</td>
<td>29%</td>
</tr>
<tr>
<td>2</td>
<td>Nokia</td>
<td>30%</td>
<td>24%</td>
</tr>
<tr>
<td>3</td>
<td>Apple</td>
<td>7%</td>
<td>10%</td>
</tr>
<tr>
<td>4</td>
<td>ZTE</td>
<td>6%</td>
<td>6%</td>
</tr>
<tr>
<td>5</td>
<td>LG</td>
<td>6%</td>
<td>4%</td>
</tr>
</tbody>
</table>

Source: HIS iSuppli December 2012

Figure2

In telecom industry modern customers are interested in attractive user interface, functional capabilities, hardware capabilities and available applications. For a telecom manufacturer it’s an absolute necessity that single piece of its manufactured hardware is tested very robustly before shipping to the retailers. Manual testing adds to lot of extra cost to the company and it’s slow and inaccurate. Manual testing is totally dependent on the availability of skilled short term labor. Automated testing has very wide range of advantages. There are many frameworks which support end to end automated testing, but Case Company is not effectively using the full potential of the frameworks to develop practice of functional testing, non-functional testing, reliability testing, compatibility testing and localization testing. Case Company has seen significant problem with programs those operated in manual testing mode. Case Company has seen many of its products getting cancelled due to lack of resource for proper testing. The idea of Case Company would be to reduce the operational cost and make maximum profit utilizing maximum resource available for the competitive advantage. Goal is short term vision of the company. Technically speaking, to develop the phone OS and let’s say we have some about 100 applications coming
as built-in. Each application has about around 25 functional and 15 non-functional test cases for release acceptance testing. Then minimum number of test cases needed to be executed for a basic Acceptance Testing (BAT) = 100*40=4000. There are applications coming from the market place every week e.g. 50. So, we have another 4000 test cases for a Release Acceptance Testing (RAT). So, total number of test cases for a really good release to market is approximately 6000. Current measurement has been always about highlighting code coverage using different tools e.g. Gcov and Lcov. Test automation is there inside Case Company and we also measure how much percentage of the test cases has been automated. Each project coach and architect has review meeting every month about this. All the teams inside should have minimum of reliable 60% test cases automated working in the framework. But there has been no measurement about the false-alarms and there has been no measure to correct the false alarms. So, automation testing results are not considered as final go-live release gate till now. But goal for case company is clear now i.e. to gain back leading position in the United States, Europe and India. To do that Case Company has to do very aggressive marketing and get the products in pipeline in time to the retailers. All development centers are having these two graphs printed in display, so that analyzing company’s significant drop in market share, they act more patriotically.

**Objective of thesis:**
Establish a concept for automated testing process to reduce time and quality slippage in the testing stage of finished consumer electronics before they enter the market using the case of Case Company smart devices as a source when building this concept.

**Expected outcome of the study will be:**
A process or model or guidelines for test automation utilizing the philosophy of enabling full automated testing for applications and services through continuous integration.

1.3 Delimitations:

My research topic doesn’t include other factors which affected time to market and quality to market for Case Company products. Just to highlight other factors are a) lack of proper resource allocation b) time to money c) work culture d) attitude of employees and last but most important e) strategies.

Below are examples of trend charts about Nokia and Apple market shares from year 2007-2012.
Global market share held by Nokia smartphones from 1st quarter 2007 to 4th quarter 2012

Figure 3

Global Apple iPhone sales in the fiscal years 2007 to 2012 (in million units)

Figure 4
2. Methods and Material
2.1 Research Design and Research Process:

Description of my research process

(1) IDENTIFYING BUSINESS PROBLEM & SETTING TARGET

(6) CONSTRUCTING “FINAL” AUTOMATED TESTING GUIDELINES FOR CASE COMPANY

(2) CURRENT STATE ANALYSIS

(5) TESTING / VALIDATING THE PROPOSAL IN PRACTICE

(3) BEST PRACTICES OF AUTOMATED TESTING / CONCEPTUAL FRAMEWORK

(4) BUILDING PROPOSAL FOR CASE COMPANY

Figure 2.1

Chapter II will describe the steps showed in Figure 2.1 in details one after another which is accepted project plan or skeleton. Steps in the flow chart are described in subsequent paragraphs in details sequentially.

(1) Identifying business problem and setting objective:

Procedures followed (AS IS) not enough, from previous lessons learnt higher Case Company management has shown a very positive response on test automation framework development and 100% usage, as over a period of let’s say 10 years there has been significant delays in product deliveries to market and cancellation due to lack of usage of automation testing from beginning days. As mentioned in paragraph 3.5, in a bi-weekly release cycle it’s not affordable for Case Company to finish manual testing for 6,000 test cases with reasonable amount of resources’ time and money allo-
cated for testing. So, automated testing with reliable results is MUST. So, idea is of the thesis is to ensure for each unit of code there is automated unit test case which gets executed in automation framework and gets reliable result. Clearly this needs a change to make faster product release to market. Faster new features are tested, faster we make the phone more look more sophisticated, faster those are shipped to retailer, faster the eco-system is built and faster we reach to our customers.

Unit, functional, non-functional test cases would be written by the developers, along with their code coverage. The major advantage of test automation is developer develops the test cases along with the code development. Then he pushes the test development package along with the development code to the version control. The version control in turn builds the new code changes, generates the new binary. If building of new code is error-free then it pushes the new test asset to test automation framework and gets the results in common understandable format. The beauty of test automation is maximum utilization of time. Supposing developer commits his code by end of the day along with unit test cases. Version control triggers the build with the latest change-list of developer. If build is successful, it triggers execution of functional and non-functional test cases in the framework. Those test cases can be executed over the night, so that we utilize whole of the time. Devices which are supposed to be shipped to market undergo rigorous continuous testing day and night. There is actually very high potential of Long-Period-Testing (LPT) in quality assurance. As, many issues could be found out which apparently stay hidden in normal testing.

(2) Current state analysis:

Case Company management has run a program called lessons learnt, in which each scrum team has given feedback of the hurdles they faced in their work in previous work, difficulties, hindrances, problems and comment on what could have been done better. Senior management of all projects cumulative have had long meetings and figured out that lack of automated testing has been a major drawback for Case Company to ship its finished devices to retailers in time with proper quality. Case Company has bi-weekly all hands line meeting inside the organization, where individual team members sit with line managers to discuss the problems inside the organization which slows the work or hinders somehow the progress of work done. So, from previous programs & platforms almost all employees have agreed that lack of automated testing in place from very beginning of any product life cycle had resulted in schedule slippage and little bit less mature quality products; which had a ripple effect in employees having had to spend extra hours in office or cancelling pre-planned vacations. In
meetings testers, developers and integrators meet regularly to discuss any problems in execution of individual projects. Thesis work takes input from various colleagues on the run time to take into account and address current problems too.

(3) Reading about best practices of automated testing:

Thesis work includes studying the best practices of automated testing. I’d study the rich literature from internet about the various options of finished products or automated tools which are already available in the market suitable for the needs of consumer company e.g. for me Case Company. Smart devices unit of Case Company produces mobile devices based on Windows OS. So, thesis work would only concentrate on Windows world. There are many automated tools which are suitable for testing functional, non-functional test asset which is being developed. Tools provide various plugins which could leverage testing done in very fast and accurate manner. Idea of the thesis is to identify the tools or set of tools which best suites testing of current finished products with best quality (zero defects).

(4) Building proposal for case company:

Projects inside Case Company are managed by project-coaches and architects have had a common goal now that the KPI for measurement of release code. My thesis aims to bring the user-manual for these KPI’s in place inside Case Company.

100% Unit code ready for each unit of code.
>95% Utility for Unit test code through automatic testing
100% reliability from results obtained through automatic testing through usage of best tools available from market which is achieved by reading internet literature.
Automated testing will be the release gate of code from Case Company. Here is sample example of how it goes where my thesis concentrates ONLY on Continuous Integration Test Automation (CITA):
(5) Testing/Validating the proposed framework

The test framework or manual or guidelines produced in previous step to be piloted inside testing organization as a whole in parallel to continuing manual testing. Testers should be trained in all possible ways to understand the automated testing architecture. Testers should understand and be able to interpret the results of automated testing. These results should be compared to the manual testing results for the same cycle of release candidate. Test automation team along with the CI engineers help to identify the deviation in test results. Constructive feedback will be taken from colleagues, managers, co-workers on any needed improvements which may help in
making more appropriate choice of tools needed. All these feedback along with the changes in strategy of company will help to build the final proposal for the company.

(6) Constructing the final proposal for Case Company

The feedback gathered from pervious steps after pilot will be used to build the final manual or guidelines for the case company. Once this manual or feedback is taken into actual practice, the business problem will be eradicated totally from Case Company. Time to market for all finished products will be way beyond simple and easy to achieve as promised by the communication team.

2.2 Data collection and data analysis methods:

Concept of field data:

Case Company releases consumer handsets to customers all across 150 countries. Release of product to different geographic locations is influenced by marketing analyst based reports and operator preference. For product based company time-to-market is THE MOST IMPORTANT plus value added with BEST QUALITY.

My thesis will rely on field data gathered from experience in Series 60 SF and Meego platforms inside Case Company. In earlier phone Operating System based time frame, time to market for Case Company products was significant. There were delays for product shipment and schedule slippage for almost all Case Company mobile handsets. One of the most significant reason for end of Series 60 SF phone era is time-to-market (TOM). According to market research this is what we have for average TOM for finished Case Company product: "It took 22 months to get a Series 60 SF phone out of the door." Meego product had a TOM of 24 months and even more for the latest Case Company product N9. Huge TOM is the biggest threat for any product based company in this super-competitive mobile telephony market.

Data analysis:

Case Company has been global market leader in mobile handset technology for an era. There has been millions of devices from different product families shipped in mobile market. N95 and E71 could be named as most successful handsets ever produced.
Internal study reports show that building OS and very first hardware design for N95 started around May 2005 whereas for E71 it had been 20 months of full-fledged product shipment time. Detailed analysis of significant delay in finished product time to market is identified internally as follows:

- Manual testing resulted in decreased operational efficiency.
- Shortage of resources to build automated framework for testing.
- Improper quality gating.
- Budget constraints.
- Experts of current mobile world say in this fast growing ecosystem and competitive environment time to market for finished devices should be as short as 6-8 months if not an year.
3. Current state analysis inside Case Company
3.1 Analysis of current business environment:

The research plan includes study of different research and development units inside Case Company, to get field data for how much importance is given to automation testing Units studied:

- Smart Devices (Windows phones based on Windows Operating System)
- Mobile Phones (Linux based phones)
- CTO (Case Company Intellectual Properties)

I am working in “Smart Devices” unit, as principal automation engineer. I interviewed and co-worked with different test leads inside Case Company who are responsible for releasing of individual applications to the Release & Integration team. All applications and services are thoroughly tested and then released to marketplace to all consumers. Most of the steps in between development of codebase and releasing to market involves much of manual testing. Manual testing is inaccurate, time consuming, costly and sometimes erroneous. In this tremendous competitive market it’s important for Case Company to keep steady pace in bringing best devices in market with fast pace eco-system, meaning lots of utility applications along with the products. There are potentially well-equipped frameworks for automated testing. Inside Case Company, there is usage of those automated frameworks in bits and pieces. As a consequence defects ranging from minor to show-stopper surface at much matured phase of product development. Bug fixing at matured stage of product has a potential threat of introducing regression. Fixing regression is waste of crucial quality time, effort, resource and possible cost. The importance of test automation in mobile devices market is tremendous, to keep running business in tight schedule and dramatically time to market. The continuous integration test automation system is developed as a framework where Blackbox and Whitebox testing are done which reduced drastically manual work and utilising full automation.

The measurements which are aimed for this year for automated testing framework needs significant awareness and vivid monitoring amongst developers and testers concerned. Case Company is spread across on the globe with persons of different backgrounds, nationalities and culture. Proper and efficient communication is most important driving factor which could bring in quick change inside Case Company. If the organization needs to implement a policy, it should be agreed and followed by all employees who are working in different countries, different time zones and different roles.
This change from manual to automated testing method implementation has been communicated from very high level management. For this reason it’s considered as part of goals and objectives for each developer, integrator and tester. As a test lead my job would be successfully creating automated test asset and integrating to the code and also getting efficient, MOST RELIABLE common format and understandable results from test automation framework to which could be used in Quality Release Gating.

3.2 Current Release testing for Case Company product programs (Continuous Integration and Test Automation background)

3.2.1 Continuous Integration overview:

Software development is an extremely complex process that consists of sub-processes. In many cases the sub-processes are also divided into many sub sub-processes. However, software may consist of smaller components or features, each of which has its own development cycle. The features can be developed by several teams which cannot be collocated geographically. Once the features are developed they are enthusiastically integrated to see fully developed and functional software. It is very normal that the excitement turns into disappointments with discovery of errors. This can cause uncertainty and disappointment among the developers and integrators, and can take several iterations before an acceptable limit is achieved and can be costly. Continuous integration (CI) was first discussed by Martin Fowler and Kent Beck at the end of 1990s, Continuous integration addresses late discovery of errors and prevents delay of software releases by integrating and testing the software components on an average daily basis.

3.2.2 Definition of Continuous Integration:

Continuous integration can be defined as: Application of software engineering tools and processes to speeding up development with decrease in integration time, in more automated way [20]. It is not only about using tools in certain defined processes but it is about a culture in which development, integration, builds and testing is done. The culture is to set best practices for development and to choose set of efficient tools and then defining and applying processes. Therefore, CI best practices is about establishing a culture where the dependencies over tools and processes are reduced.
to a minimum possible extent while they all are integrated to complete the task, that is, release of the software by reducing integration time, in a robotic way.

![Basic Continuous Integration system](image)

**Figure 3.1**

### 3.2.3 Development in the CI Environment

CI is a boost to an iterative and agile software development practice, in which developer commits, compiles and verifies new software change(s) more often and in more automated way. In software development, more frequent and faster feedback have high importance, and these can be attained by improving processes and tools with added automation. The feedback shows the visibility and evolution of the software and also showing early errors and integration problems. By resolving the identified problems at an early stage, better quality can be achieved and the shipment time to the market can be reduced.
Figure 7 shows that the errors rate is higher in the early development phase than later but the cost to solve the errors is much lower in the premature phase than afterwards. However, the defects found in different stages are gradually increasing which is because of feature integration.

There are, however, a few action points for a developer to maintain a better development culture, and they include

- Commit changes at least once per day
- Code little but commit to the repository
- Check feedback reports (build and tests error logs etc.)
- Immediate error fixing
- Maintain team cooperation and communication [20]

### 3.2.4 Continuous Integration inside Case Company

Martin Fowler describes key practices for continuous integration, to work efficiently in daily lives, which are briefly discussed below:

1. **Maintain a single source repository with regular backup history:**
   A revision controls system (or code repository) should be used and all the artifacts needed for the build are placed in the repository. Mainline version (or trunk) should be the place for the working software.

2. **Automate the build + test system:**
   A build should have started by triggering a single command by using one of several available build tools. For example Make, Ant, MSBuild or IBM Rational Build Forge are some of the widely used tools in continuous integration environments.
iii. Make build self-tested:
One of the best practices is Test Driven Development (TDD) in which developer first writes a test of required functionality first and then the actual code is written to make the test pass. All the tests are rerun once the whole code is written.

iv. Commit to mainline or branches (whichever needed) incrementally:
Committing changes to the main development line prevents code conflicts, feature instability with other features, finding and fixing problems quickly. This is a good agile practice too as developers can break down hours into smaller chunks which helps track progress and provides sense of advancement.

v. Commit should build the mainline:
If the build starts on every commit and it fails then it should be fixed immediately, which prone to fewer errors and a stable environment is available for others to check in their changes.

vi. Tend to give the build feedback fast:
The point of CI is to provide rapid feedback, so that the problems can be quickly identified.

vii. Test in sandbox or clone of the production environment:
Sandbox environment setup should be close to the production environment because production environments can be a lot different. Also enhancement features should be tested in sandbox before pushed to production. There can be unidentified errors left in the software if the test environment in not closer to production environment.

viii. Executables are easily accessible:
The tests are performed on several levels other than CI build tests. For those testers and stakeholders the ready image (or executable) should be published to a shared location which will save testers’ time to build and rebuild for any broken feature.

ix. Progress should be visible:
Feedback, reports and logs are the backbone of CI which helps in identifying causes of errors and build breaks. The build process should be visible to every one who is related to the project.

x. Automate deployment:
There can be several test environment setups on either a single or a cluster of live test servers, for a build to execute. In a complete automated build environment, an application should automatically be deployed to the live test server where everyone has access.
3.2.5 Benefits of Continuous Integration

There are many advantages of continuous integration in the software development and production environments, some of them are mentioned below:

- Defects are detected and fixed sooner because CI system runs a build several times or at least once a day. No last minutes fixings before releasing.
- Health of software is measureable such as complexity can be tracked over time.
- Immediate unit testing helps discovering bug(s) or test failures, in such cases the code can be reverted back to a bug free state.
- Early warnings of broken, incompatible code and conflicting changes.
- Constant availability of build for testing, demo or release purposes.
- Instant feedback to developers on the quality, functionality or system-wide impact of code.
- Modular and less complex code is an outcome by developers because of frequent code check-ins.
- Metrics generated from automated testing and CI (such as code coverage, code complexity, features complete) focus developers developing functional, quality code and help develop momentum in a team.

3.2.6 Disadvantages of Continuous Integration

CI, where it has several benefits to enhance the development and production of the software, it has a few disadvantages too, which include

- Increased overhead in maintaining the CI system which involves several automated and manual processes.
- Too much change as there are too many processes that need to change to achieve CI for their legacy.
- Too many failed builds, as the CI cycle keeps on running on every code check-in by the developers, there can be unfixed errors which might cause several builds to fail in a sequence.

3.2.7 Continuous Integration Software tools:

Finding the best automated software tools for an environment and development process requires having clear understanding and defined needs. Of course, the best tool would be the one which will serve longer and reduce pain from the development team.

There is a number of software tools available for CI from repository to build to test to feedback reporters, but the most important criteria in choosing a tool is whether it does
what is required? In the following an overview of a few CI tools is given under the categories:

- CI Build tools
- CI Build schedulers

**CI build tools:**

Build tools are used to compile the code and make a build. Essentially a build tool should provide following functionality:

<table>
<thead>
<tr>
<th>Name of tools</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code compilation</td>
<td>A main ingredient in building software, MSBuild is used inside Case Company for compilation of Windows based OS code written in C#.</td>
</tr>
<tr>
<td>Component packaging</td>
<td>After compilation software typically needs to be bundled into deployable component such as Java JAR files or Windows EXE files, Windows packages</td>
</tr>
<tr>
<td>Program Execution</td>
<td>Good support to invoke a program in a target platform or in command-line interface.</td>
</tr>
<tr>
<td>File manipulation</td>
<td>Creating, copying, and deleting files.</td>
</tr>
<tr>
<td>Development test execution</td>
<td>Execution of automated developer of tests for the software</td>
</tr>
<tr>
<td>Version control tool integration</td>
<td>Automated responses to version control tools (or code repositories)</td>
</tr>
<tr>
<td>Documentation generation</td>
<td>Automated generation of API documents when build is run along with release notes.</td>
</tr>
<tr>
<td>Code quality analysis</td>
<td>Generation of code coverage reports by analyzing the code using static code analysis tool.</td>
</tr>
</tbody>
</table>

Table 3.1 describes the basic functionality of a build tool of choice. It is always difficult
to find the right tool, that is, all the features in one tool but some tools provide good features and almost all the required functionalities, which can, of course, be customized.

In table 3.2 there are a few examples of widely used build tools.

<table>
<thead>
<tr>
<th>Name of tools</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSBuild</td>
<td>The Windows based Build Engine is a platform for building applications. This engine, which is also known as MSBuild, provides an XML schema for a project file that controls how the build platform processes and builds software. Visual Studio uses MSBuild, but it doesn't depend on Visual Studio. By invoking msbuild.exe on your project or solution file, you can orchestrate and build products in environments where Visual Studio isn't installed.</td>
</tr>
<tr>
<td>Make</td>
<td>Make is a widely used utility to automate building executable programs and libraries from source code. Configuration files are called 'makefiles' which specify how to drive the target program. Makefiles, however, are not typically good for cross platform dependencies. The structure of make is not a programming language but it has a fair amount in common with declarative programming languages.</td>
</tr>
<tr>
<td>ANT</td>
<td>Apache ANT is Java based build tool, similar to 'Make' and the execution requires Java platform and is best suited to build java code. But it is also a tool of choice for other development platforms. Ant uses XML files for configuring builds. Use of xml files resolved the issue of creating platform independent configuration files, and on the other hand configuration files can be messy and can easily increase complexity. The XML file has properties, which are set to configure a build, for example, build number, paths, project name et cetera; and targets, which are the tasks or processes, these targets are called to invoke the operations.</td>
</tr>
<tr>
<td>Maven</td>
<td>Apache Maven is another open source tool that is able to build software project, run developer tests, produce a number of useful source quality reports, and generate a Web site to contain the output of all of these steps</td>
</tr>
</tbody>
</table>
Rake

Rake is a build tool like Make, but the configuration files, known as ‘rakefiles’ syntax is defined completely in Ruby and also has number of other differences.

Groovy

It is an agile and dynamic tool mainly builds upon Java but has additional power features inspired by languages like Python, Ruby and Smalltalk. It makes writing shell and build-scripts easy, simplifies testing by supporting unit testing and mocking out-of-the-box and compiles straight to Java bytecode. Mostly used in all post build step to parse the log results and generate the build state to either “SUCCESS” or “PROJECT FAILURE” or “SYSTEM FAILURE”

CI build schedulers:

Build schedulers are used to schedule a CI build either once, twice or several times a day. In case company context schedulers could be used to trigger builds from developer or tester commits to the version control system. Some most typical functionalities of a widely used build scheduler are summarized in table 3.2

<table>
<thead>
<tr>
<th>Name of tools</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build execution</td>
<td>Sync the latest code from version control system and trigger automated builds, then trigger automated testing according to project configuration. Triggered by commit from developer or tester.</td>
</tr>
<tr>
<td>Version control and build tool integration</td>
<td>Perforce is used as version control tool for Case Company software development and test asset.</td>
</tr>
<tr>
<td>Email notification feedback</td>
<td>Sending email through email notification plugin is only very quick way of providing feedback to developers or continuous integration team, on build/test success, abort, failure.</td>
</tr>
<tr>
<td>Build labeling</td>
<td>Marking of artifacts and deliverables from particular builds to testers as inputs in terms of labels.</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Interproject dependencies</td>
<td>Execution of any dependent build to make a main build successful. This is how a build scheduler is configured to know if there are dependent builds.</td>
</tr>
<tr>
<td>User interface</td>
<td>For modifying and enhancing configuration, checking, copying or downloading artifacts and watching build status Case Company uses Jenkins.</td>
</tr>
</tbody>
</table>

### 3.3 Software Testing methodologies in Case Company

Notion of software testing has a very huge and wider scope, it is about testing of functions, modules, components consisting of several modules, sub-system which consists of components and then fully functional system which can be comprised of several sub-systems. These tests can be performed either manually or in an automated environment. Setting up an automated environment is an extremely difficult process, but once it is ready, the chaos of executing tests manually, creating and delivering tests results and reports to the specific development teams is ended. It reduces a lot of development and testing costs. Software testing is a set of activities conducted with a goal of discovering software errors. Also, it is a software development process intended to assess a software item (such as system, sub-system, unit), features (functionality, performance) against the given set of system requirements. Testing is never an easy job, neither are all the defects identified within software. Testing is a process to produce optimized, bug-free code to a level of required satisfaction, meaning that the functional software should perform what it is expected to do with integrity with compliance mutually with other software modules, components and applications. However, in practice, some bugs or defects remain stick in test cases. Hence, the testing can be broken down into component testing and integration testing. Finally, acceptance testing is performed once the system is fully developed and ready.
Finally, acceptance testing is performed once the system is fully developed and ready.

*Component testing* is the functionality of identifiable components like functions, or groups of methods collected together into a module of object are tested. *Integration testing*, the second stage, the components are incorporated to form a sub-system or a complete functional system and tests are performed to check integrity and interactions between the components and performance of the system as a whole.

There is one more test phase and that is *acceptance testing phase*. Generally, *acceptance testing* is known for testing customers or end user requirements specifications if they match with the system, subsystems, components or units.
Figure 3.4 shows that acceptance tests are part of component testing, integration testing stages as well as, acceptance itself is another stage, for instance, interface testing. Therefore, acceptance testing is an integral part of both component and integration testing.

Engineering department takes care of developing software and unit test asset for the development code. Execution of the component test happens under quality organization. Quality organization also takes care of the release and integration of last known good software to certification team. Continuous Integration team under Quality organization takes care of preparing an environment for building software code, maintaining and enhancing the test laboratory to facilitate automated functional and non-functional testing. Quality organization also develops and owns automated test asset inside Case Company.

3.3.1 Component testing:

Testing is a very important phase in product development life cycle. World’s leading product based companies believe that there should be one tester to test the code written by one developer, but that’s the ideal scenario. Most of the companies can’t afford to allocate so much budget for testing in today’s market scenario. Developers perform coding in Windows based developing technology e.g. C#. They develop unit test cases to cover testing of the functions written. We have code coverage tools which measure the percentage of code which is covered in testing and traversed during testing. Developers commit the changes to the code to the perforce stream they work in. Each commit triggers a build in the continuous integration system. Once the build is complete which gives assurance that the latest code is good, unit test
job is triggered. Unit tests are executed automatically in the laboratory. Unit tests suite is used currently to execute unit test cases. Our organization works mostly on testing the top-most layer of the OS stack.

![Diagram of layers in Windows OS](image)

Figure 3.5

### 3.3.2 Integration testing:

Release process of application software from Case Company takes time to understand, little complex and timely procedure. Complexity lies in the bureaucracy and more time is spent in cross-culture communication between teams spread across different parts of the globe and participating in building the software. The image is assembled of layers from Windows OS for phone then middle layer adaptation and on top lies application software applications delivered. Application software deliveries come on the top most layer of phone.

Release image build is made by the integration team taking all the accepted deliverables from all teams.

Acceptable deliverables contents are decided in regular meetings where the test leads and project managers meet to decide the content to be released.

Variant phones are different target hardware phones from the same base OS and engine.

Devices under test are flashed with the latest image and testers start testing the devices manually.

Quality Center is the testing tool used inside Case Company which contains the whole functional and non-functional test cases to be executed per round of release.

Testing is done manually for all applications under our organization.
Bugs are filed for the failed test components in Nzilla. Error management team tries to understand all the bugs and proposes the most important to be solved in order or priority.

3.4 Summary & conclusion, strengths and weaknesses

Case Company as an organization has been successful and most prominent mobile manufacturer for more than a decade. Case Company has always manufactured excellent products winning hearts of significant percentage of mobile customers for years. The devices manufactured brought across rich user interface experience and extremely good hardware which lasted for more than a decade. Customers have been interviewed who reported that “We have been using Case Company products for years and don’t wish to switch to new hardware as we are very much satisfied”. Over the years this competitiveness in mobile market has been increasing all the time. The business issue with Case Company has very few times been with the quality of products shipped or delivered to customers, but delays and schedule slippage. This thesis work follows six-sigma model.

Smart devices unit has Devices Engineering and Quality organization doing phone business research work.

Engineering productivity mainly focuses on the development of the applications coming to marketplace. Architects along with product program management and design team decides the user interface for applications. They work very close to marketing division for their inputs. The strength lies in developing application features according to needs of consumers and simplifies the release cycle of applications. The faster applications released to market square, more robust the windows phone ecosystem it results to. The engineering productivity team is not totally aligned to believe in developing unit test cases for the developed code of applications. This is one of the major weak points in current working model of engineering development. This is fatal because as codebase increases in lines and modules the complexity increases and there is no way of testing initially how the integration code behaves unless there is proper unit testing written from the very first of codebase development. This leads to many of the design and inter-process errors detection to be delayed till the last phase of delivery cycle. The more error detection is delayed, more is the cost involved for case company side to correct the error. Moreover error correction at matured phase of
product development delays the time to market for finished mobile products to consumers. This is because software delivery is delayed to flash the phones and variants across markets of different regions. The thesis process highlights and addresses of possible remedies in first draft proposal of current company.

Current Quality team how it operates is explained here along with merits and demerits of current process map. Delivering the products to consumers with the highest quality is the main goal. There are various checks which happen in build, API scan results, functional, certifications and non-functional testing. After all these testing cycle is over on a release candidate, then it’s sent for certification. A certified application is then placed to marketplace wherefrom consumers get updates. These updates include applications as well as OS items. The whole process of testing functionality of services and applications rely mostly on manual testing. The merits of manual testing are many. For instance manual testing gives very reliable or in other words too much accurate results than any other testing in certain dimensions (e.g. when highly skilled testers are available). Reliable results are key to success and biggest strength for time to market for products. It is important to understand that testing and getting results from phone software is more difficult than testing big Windows. Manual testing has many disadvantages if continued for very long time in product development release life cycle. Phone testing is more time consuming than desktop testing. So, at initial stage of product development life cycle with parallel OS development manual testing is only way to get very good results to determine release candidate. Cost of automated testing setup along with proper test laboratory infrastructure is huge initially. Manual testing is very costly for companies which produce phone hardware as well as develop phone software. On the other hand, sometime due to poor tester skills results are erroneous which leads to faulty releases. In case company most of the times this has happened in localization testing teams. As external vendor testing is time based and booked in advance, due to faults of internal releases case company lost lots of money and faced delayed schedules. Moreover in this mobile world, manual testing doesn’t improve the skillsets of the testers for long time which leads to boredom in mundane work involved. Mundane or routine work sometime leads to low energy and loss of creativity and in the broader sense leads to stagnant career curve for a bigger set of organizational fellows. Also in manual testing cycle it’s not possible to execute all the test cases before releasing the candidate, reason being the time constraints. Thesis explains how when automated testing is taken is taken into use full bandwidth of the device pool can be exploited to get the results on time, e.g. testing overnight the commits made by develop-
ers and automation engineers, so that results are ready by morning times each day which can heavily speed up the release cycles.

At par with engineering and quality organization who are directly involved in the R&D works, there is Sales, Marketing, Product Program Management, Supply Chain and IT organizations who are supporting the services with respect to products. The brighter side of design organization inside Case Company is that always ergonomics of human beings are taken good care right from beginning till end of design phases. Samples are taken from places where Case Company brand is very strong. Weak points are ergonomics are missing where Case Company market is less prominent. This gets the sampling process bit biased on geographical locations and market entry in those locations get further delayed. The prime important market for Case Company phones are Europe and Asian countries. Market for Case Company is slowly increasing in US side, along with UK. Phone accessories demand more aggressive sampling from different human beings which affect the demography of sales according to consumer preferences. All the short comings and weaknesses are covered in thesis process with best possible remedies studied from internet literature and taking inputs from colleagues, managers and lean six sigma experts.
4. Best practices of automated testing in literature
4.1 **Blackbox versus Whitebox testing:**

Software testing is the process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of all the software items including dependencies.

4.2 **Why testing is important:**

Listed are the categories where more and more testing yields better results:

- **Mistake** – a human action producing an incorrect result.
- **Fault [or Defect]** – an incorrect step, process, or data definition in any program.
- **Failure** – the inability of a system or component to perform its required function within the specified performance requirement.
- **Error** – the difference between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition.
- **Specification** – a document that specifies in a complete, precise, verifiable manner, the requirements, design, behaviour, or other characteristic of a system or component, and often the procedures for determining whether these provisions have been satisfied. A mistake committed by a person becomes a fault (or defect) in a software artifact, such as the specification, design, or code. This fault, unless caught, propagates as a defect in the executable code.

4.3 **Verification:**

Verification means the process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. *Are we building the product right?*

4.4 **Validation:**

Validation means the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements. *Are we building the right product?*

4.5 **Blackbox & Whitebox testing:**

**Black box** testing (also called functional testing) is testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions.

**White box** testing (also called structural testing and glass box testing) is testing...
taking into account the internal mechanism of a system or component.

4.6 Types of testing and opacity:

There are six types of testing which can be classified amongst blackbox and whitebox as mentioned below:

4.6.1 Unit Testing:

**Opacity:** Whitebox testing

**Specifications:** Low and high level design and/or code structure

Unit testing is the testing of individual hardware or software units or groups of related units. Using white box testing techniques, testers (usually the developers creating the code implementation) verify that the code does what it is intended to do at a very low structural level. For example, the tester will write some test code that will call a method with certain parameters and will ensure that the return value of this method is as expected.

4.6.2 Integration Testing:

**Opacity:** Blackbox and Whitebox testing

**Specifications:** Low and high-level design.

Integration test is testing in which software components, hardware components, or both are combined and tested to evaluate the interaction between them. Using both black and white box testing techniques, the tester (still usually the software developer) verifies that units work together when they are integrated into a larger code base.

4.6.3 Functional and system Testing

**Opacity:** Blackbox and Whitebox testing

**Specifications:** Low and high-level design, requirement specifications.

Functional testing involves ensuring that the functionality specified in requirement specification works. System testing involves putting the new program in many different environments to ensure the program works in typical customer environments with various versions and types of operating systems and/or applications. System testing is testing conducted on a complete, integrated system to evaluate the system compliance with its specified requirements.

4.6.4 Stress testing

Testing conducted to evaluate a system or component at or beyond the limits of its specification or requirement.
4.6.5 Performance testing
Testing conducted to evaluate the compliance of a system or component with specified performance requirements.

4.6.6 Usability testing
Testing conducted to evaluate the extent to which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component.

4.6.7 Acceptance Testing

Opacity: Blackbox testing
Specifications: Requirement specifications
Acceptance testing is formal testing conducted to determine whether or not a system satisfies its acceptance criteria (the criteria the system must satisfy to be accepted by a customer) and to enable the customer to determine whether or not to accept the system.

4.6.8 Regression Testing

Opacity: Black- and white-box testing
Specifications: Any changed documentation, high-level design
Throughout all testing cycles, regression test cases are run. Regression testing is selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements. Regression tests are a subset of the original set of test cases. These test cases are re-run often, after any significant changes (bug fixes or enhancements) are made to the code.

The following guidelines should be used when choosing a set of regression tests (also referred to as the regression test suite):
• Choose a representative sample of tests that exercise all the existing software functions;
• Choose tests that focus on the software components/functions that have been changed; and
• Choose additional test cases that focus on the software functions that are most likely to be affected by the change.
A subset of the regression test cases can be set aside as smoke tests. **A smoke test** is a group of test cases that establish that the system is stable and all major functionality is present and works under “normal” conditions.

**4.6.9 Beta Testing**

**Opacity**: Black-box testing  
**Specification**: None.

When an advanced partial or full version of a software package is available, development organization can offer it free to one or more (and sometimes thousands) potential users or **beta testers**. These users install the software and use it as they wish, with the understanding that they will report any errors revealed during usage back to the development organization. These users are usually chosen because they are experienced users of prior versions or competitive products.

The advantages of running beta tests are as follows:

- Identification of unexpected errors because the beta testers use the software in unexpected ways.
- A wider population search for errors in a variety of environments (different operating systems with a variety of service releases and with a multitude of other applications running).
- Low costs because the beta testers generally get free software but are not compensated.

**The disadvantages of beta testing are as follows:**

- Lack of systematic testing because each user uses the product in any manner they choose.
- Low quality error reports because the users may not actually report errors or may report errors without enough detail.
- Much effort is necessary to examine error reports particularly when there are many beta testers.

Various types of testing in a snapshot:
4.7 Automation testing introduction:

4.7.1 Test Automation Framework

Test Automation Framework (TAF) is a set of theories, ideas and tools that offers robotic software testing provided that the maintenance cost of the framework is low. Implementing the correct logic in methodical manner pays back in terms of cost and efficiency. TAF development is part of software development life cycle inclusive business cases, requirement management, planning. Implementation includes right choice of tools, scripts, technologies and methodologies. Proper planning for development of TAF is mandatory to understand test scenario requirements, specifications and goals of test automation outcomes inside the organization.

4.7.2 Advantages of test automation

Automated software testing answers to many questions and provides solution to several problems which software developers are facing today. If test automation is implemented correctly, it has many advantages; some of them are below.

- **Execute more tests more frequently.** A clear benefit of automation is ability

![Table: Testing Types and Specifications](image)
to execute more tests in less time and hence to make it possible to execute more frequently. This will lead to greater confidence in the system.

- **Earlier time to market** Automated testing allows to quickly find and fix Bugs and the time elapsed, as compare to manual testing, is much shortened. This helps in releasing qualitative software timely to the market.

- **Perform tests which would be difficult or impossible to do manually.** For instance, a test requires input from 200 users which seems difficult and costly, which can, otherwise, be simulated using automated tests. Another example could be a graphical user interface (GUI) object which may trigger some event that does not produce any immediate output. A test execution tool may be able to check that the event has been triggered, which would not be possible to check without using a tool.

- **Reducing the time and cost of software testing.** It demands considerable amount of time, money and skills to produce a worth having, reliable and efficient TAF. In return the system becomes more cost effective than the cost was spent to build the system. Efforts and time decreases in terms of test data generation, test execution, test results analysis, monitoring error status and its correction, report creation and other migrating factors.

- **Better use of resources.** Automating tedious and boring tasks, such as repeatedly inputting same test data, gives greater accuracy as well as improved staff morale, and frees skilled testers to put more effort into designing better test cases. Machines which usually are idle during the nights can be used to execute automated tests.

- **Reliability and repeatability of tests.** Tests that are repeated automatically will be repeated exactly every time (at least the inputs will be; the outputs may differ due to timing, for example). This gives a level of consistency to the tests which is very difficult to achieve manually.

**Reusability.** The effort put into deciding what to test, designing the tests, and building the tests can be distributed over many executions of those tests. Tests which will be reused are worth spending time on to make sure they are reliable because an automated test would be reused several more times than manual testing.

**Increased confidence.** Knowing that extensive sets of automated tests have Run successfully, there can be greater assurance that there will not be any horrid shocks when the system is released, providing that the tests being run are good tests. Useless and unused functions and code which can either be removed or used to being efficiency to the software.

### 4.7.3 Disadvantages of test automation

In test automation life cycle we face some problems sometimes which comes to us as real surprises on run time. Proper knowledge in fixing the tools, interfaces and using proper parameters are essential.

- **Poor testing practices.** Automated testing is not recommended if the test asset
is not matured enough. This happens when test cases are written with improper documentation, may be they are little inconsistent and they are insufficient to find bugs in development codebase.

- **Technical problems.** Sometimes it happens so that infrastructure is improper due to budget constraints. E.g. sometimes automation laboratory has poor facilities or equipment’s or test devices used for testing.

- **False alarms.** Poor framework development sometimes lead to false positive test results or false alarms when test cases actually passed. It’s not good to take in to use immature test asset or framework for production purpose.

### 4.8 Test automation Tools

**CI & Test Automation inside Case Company background**

Case Company has been world leader in mobile market for more than a decade, both hardware and software being manufactured and developed entirely from inside Case Company. A part of software development has always been outsourced to subcontractor companies. In February 2011, Case Company took a strategic decision to develop smart devices based on Windows OS. The tools needed to support development of software codebase as well as test automation are totally based on Windows OS. To complete search literature on best practices for automation testing with respect to Windows based testing tools and related in depth technologies. Developer submits code to Perforce (version control tool), a build is triggered in Jenkins (build machinery) per commit. After build succeeds, automated testing is triggered via Jenkins to the test devices. Results for tests are gathered in “xml” & “trx” format and collected back to Perforce. Automation test developers check the results and correct failing test cases and add new test cases.

**Test Automation Tools relevant to Case Company**

#### 4.8.1 Perforce

**Perforce** is a commercial, proprietary revision control system developed by Perforce Software, Inc. The Perforce server manages a central database and a master repository of file versions. Perforce supports both Git clients and clients that use
Perforce’s own protocol. A Git client can communicate with the Perforce server over SSH, and other Perforce clients communicate with the server via TCP/IP using a proprietary RPC and streaming protocol. Developers, QA leads, and CI automation leads submit changed files together in changelists, which are applied as Atomic commits.

**Basic Concepts**

P4V connects computer to the Perforce versioning service and helps you move files between Perforce *depots* and your *workspace*, as shown below.

The definitions for these Perforce terms are as follows:

- **Workspace**: folders or directories on our workstation where you work on revisions of files that are managed by Perforce.
- **Perforce application**: P4V (or another Perforce application), running on the workstation, which makes requests from the Perforce versioning service and delivers the results of those requests (files, status information, and so on).

![Diagram of Perforce components](image)

**Figure 4.2**

**Advantages of Perforce tool:**

- **Perforce** – Very robust version control system, capable to handle huge data traffic.
- **Roles and Workflow** – In Perforce functionality and use cases vary from user perspective and the tasks associated with the workflow for each. System administrators can modify create streams and delete the same. Users can only submit tasks to the streams. Local admins can edit stream information in addition to submitting tasks.
- **Codebase** – Practices, terminology and procedures relating to codebase management. Full history of files are maintained.
4.8.2 Jenkins

In the year 2005 - Hudson was first release by Kohsuke Kawaguchi of Sun Microsystems. In year 2010 Oracle bought Sun Microsystems. Due to a naming dispute, Hudson was renamed to Jenkins. Oracle continued development of Hudson (as a branch of the original). Studying other tools from internet best option is Jenkins tool. When setting up a project in Jenkins, out of the box we have the following general options:

- Associating with a version control server.
- Triggering builds.
- Polling, Periodic, Building based on other projects.
- Execution of shell scripts, bash scripts, Ant targets, and Maven targets.
- Artifact archival.
- Publish JUnit test results and Java docs.
- Email notifications.

Advantages:

- Jenkins is a highly configurable system by itself.
- The additional community developed plugins provide even more flexibility.
- By combining Jenkins with Ant, Gradle, or other Build Automation tools, the possibilities are limitless.
- Jenkins is released under MIT license, there is large support community and thorough documentation.
- Easy to write plugins and can be easily customized.
- Jenkins generate test reports.
- Integrated very well with version control system e.g. perforce.

4.8.3 Visual Studio 2012 & MSBuild

The Windows Build Engine (MSBuild) is a platform for building applications. MSBuild provides an XML schema for a project file that controls how the build platform processes and builds software. Although MSBuild is integrated into Visual Studio, it is not dependent on it. Developers can orchestrate and build products in environments where Visual Studio is not installed. To start developing your apps, you'll need to install the right tools to get the job done. The Visual Studio 2012 family of de-
Development tools offers a powerful development environment to build great apps quickly. Two free versions of Visual Studio 2012 are available to build your apps for Windows Phone 8 and Windows 8.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual Studio Express 2012 for Windows 8</td>
<td>Tool to build Windows Store apps. It is free, and includes the Windows 8 SDK, Blend for Visual Studio, and project templates.</td>
</tr>
<tr>
<td>Visual Studio Express 2012 for Windows Phone</td>
<td>Tool to build Windows Phone 8 apps. It's free and includes the Windows Phone SDK 8.0, Blend for Visual Studio, and project templates.</td>
</tr>
</tbody>
</table>

With new version of Visual Studio, we are successfully able to use plugins and create the test packages for automated testing.

4.8.4 Test build solution

Test build solution is a utility which is developed for proprietary purpose Case Company neededs. This utility helps in building zip files of packages used in test automation. Test build solution uses 7zip in the backend which creates the zip packages. This is a wrapper project to take other input parameters needed to ensure all the binaries are present in the zip package to be tested.

**Advantage:** This tool acts as a wrapper over 7zip which is customized to get input parameters to prepare the zip file for automation testing. Tool is very robust and is flexible enough to be used during build or after build just before executing the test cases. This tool is invoked from the Continuous Integration automated scripts if the build for a particular application or service passes. Tool has many advantages on top of 7zip tool. There are lots of customizations which are added and many features are getting developed by tool owners. Development team is working on requirements which engineering and quality divisions inside Case Company provide them with.
4.8.5 Test Framework Utility

This is a highly specialized tool built for managerial reporting and engineering extensibility. It is tool built and designed by a project manager to help other project managers stay abreast as to the progress their QA team is making.

**Manage test case efficiently**

Test Framework Utility makes it easy to create new tests by eliminating a lot of the complexity that has slowly been added to so many other tools. This tool differentiates itself by keeping its test cases simply structured and by focusing test engineers on the most important parts of the test case: the instructions on how properly execute the test.

**Create test plans quickly**

When it comes time to build a test plan, this tool allows us to quickly create a plan comprising all the test cases with a given keyword or tag. Just enter in the tags and keywords relevant to your testing goals and this tool generates a test plan composed of the test cases with the tags you entered.

**Quickly view passed, failed and blocked test cases**

Once testing begins perhaps one of the most important tasks performed in monitoring the progress of the testing process is the inspection of test cases and their status. Blocked and failed test cases may help identify areas of functionality with high degrees of risk associated with them, while test cases with other statuses can help you to hone your process, more evenly distribute the testing load across your team and help your product converge on a high degree of quality more quickly.

**Integrated with bug tracking system**

This utility is very much well integrated with Nzilla, Nzilla is used as bug triaging system inside Case Company. When developers commit a fix to Perforce, they can mention the bug fix in the commit, then Nzilla bug is automatically updated with the status To Resolved/fixed.
4.9 Summary & conclusion:

Case Company has always given importance in adapting to new technologies over the years. In recent years there has been a tough competition for Case Company in the mobile market to be in leadership position both from software and hardware point of standards. Competitors have been super-fast in getting attractive products to market. Time to market is now a prime important use-case which my company is trying to improve radically. As mentioned earlier in my thesis, testing is very important in product development and maintenance life cycle. Taking a prudent and pragmatic view, it’s not possible to test all the applications and in-market use cases in all devices manually. The process is erroneous, time consuming and very costly. Quality gating should be more robust and fast process reducing delays in “time to market” radically. Faster we test the products we builds, faster we ship the products to commercial market with assured quality.

I have identified that the existing continuous integration tools are best suited for supporting and enhancing importance of automation testing environment. Jenkins is the build tool which we use to build the development codebase. I have explained in details in this chapter earlier. Jenkins is very well integrated to the automated testing web reporting. We use Perforce as version control tool for codebase, Jenkins is very well integrated to Perforce. Unit test, functional test, certification test, compatibility test, localization test suites are getting developed by respective test automation teams across sites. Test code is piloted to be executed in CI laboratory and is corrected so that all test cases give 100% accurate results for a significant period of time. This gives me to build the initial proposal for case company, which is Case Company in coming chapter V. I have studied the time taken to perform one round of manual test preloaded applications in the in-market devices. In coming chapter I present and describe the draft proposal to the case company. Thesis helps in identifying how test automation helps in speeding up the production, supply chain and time to market for finished products from case company. Engineering management, architects and quality organizations had representatives are brought under common platform to understand the difficulties and impediments in project work. All impediments are identified and prioritized in the thesis process which helps in evaluating the most prominent factor which leads to delay in time to market for finished products from Case Company delivery.
4.9 Conceptual framework as per literature on automated testing process:
5. Build the proposal for automated testing in case company
5.1 Test Automation and Laboratory facilities

5.1.1 Importance of CI & Test Automation:

Testing is very important in all product development life cycle. Robust and proper quality gating yields good commercial products to consumers luring them sticking to the same brand for years and even generations. Current mobile market is kind of saturated and slow. Competition is huge in this market, target for all product based companies is to get the finished products to the market as soon as ready with good quality. Manual testing of products developed is time-consuming, boring, erroneous and very costly for the company. To get the finished products out to market the whole supply-chain and testing procedure needs a big change and relying on automation testing. Automated testing tools are sumptuously available and being developed and customized according to the needs of product companies everyday as plugins.

My case company has not yet explored full band-width of the test automation plugins available. This thesis gives a step by step guideline or process to leverage, utilize and expedite development and deploy of test automation framework to reduce the “time to market” for all products under development and on the future roadmap. Mobile world market leaders believe that testing is equally important as development for software codebase. Testing and more testing highlights defects and more testing we do at the early stages of software development, more we save time as defects and bugs are caught at very early stage. In earlier days Case Company has faced severe errors at very late stage of software development, where entire architecture needed to be re-invented to get the products in market, which caused large scale delays in shipping finished products to consumer market. Manual testing has been the main reason for these delays. Too much code complexity (e.g. cyclometric complexity), complexity in writing unit test cases for code, un-availability for automated testing results led to poor quality of products till matured phase. Bugs found in later stages of product development life cycle led to frequent architecture changes, regressions and delays in “time to market” for products from Case Company many times. The reason behind management to decide delaying products was Case Company always wanted to have the best consumer experience image in customers’ eyes. Defective products hurt consumer sentiments and lowers confidence. This thesis deeply studied internal gaps between all stake holders producing finished products inside the company in earlier chapters. Thesis work addresses the problems & hindrances (practical and theoretical) faced in development and testing life cycle of earlier products from all angles. Thesis works on
improvements in way of working. Thesis introduces test automation and importance of test automation in testing products developed from early stages of development life cycle. Test automation is solely dependent on the CI & infrastructure laboratory which is a direct outcome of the thesis process as described below in details:

5.1.2 CI laboratory & infrastructure

This laboratory is developed as part of thesis proposal to utilize full potential of test automation and fasten release process to market. The continuous integration test automation laboratory is centralized testing centre for automated tests getting executed. Laboratory is equipped with latest device hardware’s for testing. Testing is done across sites over a centralized pool of devices, which is maintained and updated as and when there is any latest image available. Testing of devices include all kinds of prototypes in combination.

Prototypes are classified as follow:

- Before public launch; **SECRET/CONFIDENTIAL**
  - Visual design
  - Technical design
  - Features
  - Software design
  - Functionality of product

- After public launch, but not yet commercially available; **CONFIDENTIAL**
  - Technical design
  - Software design
  - Functionality of product

- Product commercially available; **CONFIDENTIAL**
  - Functionality of product

In principle, secret prototypes cannot be taken outside Case Company premises at all. Confidential prototypes (after public launch), can be taken outside office with special NDA permission for field testing. Public devices mean they are already available in commercial market stores.

The infrastructure of the test laboratory can be divided in following categories:
1. Test devices mainly proto-types.
2. Test PCs’.
3. Updated Windows test environment from PC side (Windows PATH for tools, etc.).
4. Updated images flashed in test devices.
5. Test network through wireless LAN as well as SIM internet in phones.
6. Accessories needed for testing purpose.
7. All the build and test PC’s get automated common base script from CI team on per commit basis. Whenever there is a commit in CI script base from centralized CI team an automated sync job is triggered in Jenkins tool where all the existing build, test slaves are configured for automated update. In this particular way CI team ensures all the slaves receive latest features developed by CI team as well as bug fixes.

5.2 Proposal for Case Company (Case Company) in test automation

5.2.1 Unit Testing

Unit testing is very much important for product development life cycle. In test driven development (TDD) unit test cases should be developed even before code is written. When test cases pass in a controlled environment, then code is considered to be complete in theory. Unit test cases reduces uncertainty in units. By testing parts of the program and then testing sum of its parts makes integration testing much easier. The procedure is to write test cases for all functions and methods so that whenever a change causes a fault, it can be quickly identified. Unit test cases are developed in Visual Studio 2012. There are mainly three different frameworks to develop test cases enabled via Visual Studio.

In earlier days unit test cases didn’t find much importance in development. Code & architecture complexity was never managed or controlled. According to interviews with developers this is a major problem in way of working and general awareness to build unit test cases from beginning of any application till it reaches market square. Developers across Case Company have started developing unit test cases for each application along with codebase development right from the beginning days when requirements are gathered in Accept360 tool till end. All test cases are written in C# language but in different frameworks, according to comfort level and expertise of developer. Continu-
ous Integration and test automation team helps these test cases to be integrated and executed in laboratory. Support is extended in all available Windows based test frameworks so that developers can write test cases in any of the frameworks. Unit test cases are executed per commit in CI laboratory and results are uploaded in Perforce for developers.

Developers get feedback on failing test cases and they correct those in next commit which triggers a build in Jenkins again so on and so forth.

Developers perform following steps to write unit test cases along with code development:

- Open Visual Studio and click File and choose Add.
- In New Project dialog box, expand Visual C# and choose Test.
- From template projects choose Unit test project
- In name box enter "ProjectName.UnitTest" project and choose OK.
- The UnitTest project gets added to the main ProjectName solution.
- In UnitTest project a reference to ProjectName is added.
- In configuration xml file the test solution is added to build on per commit basis.

**Action taken from CI team:**

CI automation team developed python scripts which are part of the thesis development process to allow support development in all Windows OS based testing frameworks. Scripts are modified, all xml based project configuration files for all applications are modified and developed for this features.

**Advantages:**

1. Developers across Case Company have unified understanding about the importance of unit test cases.
2. Higher management is closely monitoring amount of test cases developed (whether or not) for all applications.
3. CITA team has enabled full support to execute test cases written in all frameworks.
4. CITA team has generated label information (discussed later) to show to higher management which development teams are missing unit test case development from early stages of produce life cycle.
5. Product development life cycle has been easy and less complex from the beginning of application development.
6. Quick feedback from CI team about the test cases improves quality of test asset complying with code development.
7. Developers get very quick feedback on code written and submitted to Perforce stream, the faster feedback they get, developers write more refactored code. Quality of code developed gets matured and less error-prone (from middle layer level to OS side). More matured code written from early phase of product development leads to shorter application life cycle and time to market reduces a lot.

5.2.2 Functional Testing

Functional testing is one of the most prime important factors for development of codebase. Functional test cases have the ability to evaluate individual features of an application. They are familiar with typical application behaviour and have the skills needed to look objectively at a feature and see what’s wrong. It’s perhaps worth to mention that more automated test cases an organization aims to develop and test more the product is built robust and most importantly at great pace. As mentioned earlier in thesis background that testing is very important to reduce time to market for
finished products to market. Functional testing will only be successful if an organization’s underlying quality fundamentals are solid and everyone clearly understands how testing helps achieve the goals of the business. Functional testing is only one of many activities that collectively comprise a comprehensive testing strategy. Depending on the needs and expectations of your company, different testing activities such as performance, load, and security testing should be considered. Functional testing differs from other types of testing in that it most closely reflects the experience of the users. While performance effects the experience and security issues add risk to the experience – how the application functions is the experience. Inside Case Company automation testers write code for functional testing. Test cases are developed in C# language. Test cases imitate all use-cases which a common user does while using different applications in a Case Company phone.

**Action taken from CI team:**

As part of the thesis process the centralized automation team gathered data from marketing team. Data consists of a year of consolidated statistics for applications which have been the most commonly downloaded and used applications in Case Company phones. Case Company Research and Development division analyses the market consumer demand for applications as well as downloads from Market Place. These data are used to prioritize development of applications and related services quickly and adding features and releasing newer versions of applications. Scrumworks is followed with product application backlog items as well as consumer requirements. Automation team checks requirements for prioritized (decision from marketing and higher management approval data analysis) applications and develops applications. Functional test cases are developed in Code-base developed UI test framework based on priority. Test case solution files are included in the build solution files to get built. Test cases to be automated are chosen from Quality Centre. The automation team lead decides which test cases to be automated on priority. Thesis work includes data analysis from marketing, management awareness how to radically reduce the cycle time for application release.

**Thesis work** stresses how manual testing can be erroneous, time-consuming and of resulting poor quality of products in market. Six sigma project is used to prove with 95% confidence that “cycle time” for products with pre-loaded applications is reduced a lot with introduction of automation to test the applications. Also its proved that quality of the products coming to market have improved a lot via tools introduced in six sigma. Thesis work highlights time taken to execute functional test cases before
making internal release to get bunch of automated testing getting executed. Also thesis works shows time taken to execute functional test cases in Application Software Development CI laboratory mentioned earlier Functional test cases are executed per commit in CI.

**Advantages:**

1. Testers get feedback about the application behavior (newly implemented features as well as old functionality) immediately after build and testing is over per commit.

2. Testing is performed whole night, using full bandwidth of the test automation devices connected to the test network pool. With this setup, the developers and automation engineers can push the latest fixes in office hours before leaving and the results are ready by an over-night test execution next morning before arriving office. This saves lot of business man hours of developer, tester and automation engineer as a whole speeding up the application development life cycle. Moreover, this saves company money, as manual testing is boring and erroneous. As applications grow functionalities needed to be tested also increases. It’s expensive and risky business for Case Company to invest so much money and book resources for so much of testing always. So, automated functional testing gives reliable test results with assured quality. In continuous integration system, faster feedback provided faster is action is taken, delayed feedback loses its importance and is of no worth. Following is picture which diagrammatically displays how the process of manual functional testing in release integration is slowly eradicated and automated functional test reporting is used in a wider audience.
5.2.3 Certification Testing

Certification testing or non-functional testing is also another important aspect for software development life cycle. Many organizations use "xresponse" tool to calculate the response timings for starting up the applications once tapped application icon from menu or closing any application once pressed application close button. Requirement comes from architects as well as UI design engineering organization. Moreover requirements also come from marketing organizations, common understanding about the non-functional requirements come from meetings and conclusive decision of standards are proposed. Decisions are made according to the best possible values supported from current OS as well as racing towards perfection in this competitive mobile market. Response timings vary from application to application, depending on complexity, the backend services loaded on tapping front end application.

Actions taken from CI team:
As part of thesis process, requirements from marketing department are finalized and refined. Previously non-functional bench-marking values had been unrealistic in earlier phone business models. In meetings, testers, architects and developers
came to common conclusion about the time values for various important consumer applications. These are approved by global quality triads before handed over to tables of testers and automation testing team who writes the automation test cases. Automation team developed certification test cases for all applications of Case Company phones. Whenever developers commit any code changes test cases get executed in CI laboratory devices and values of test results are pushed back to Perforce for analysis. Results are generated in ".trx" and ".xml" formats to ease developers’ interpretations. Integration of Case Company Crash Portal, details are coming in next section.

**Advantages:**

1. Most important activity which thesis proposed and proceeded work on is regarding certification testing is integration for Crash Portal to CI builds and dashboard implementations.

2. Certification testing automation is huge time saver, as results which are used in review meetings have values filled from automated test results, so manual efforts are reduced to a minimal. Time is saved and also speed is increased for test results.

3. CI team made a comparison for the manual results from historical data and the new values for automated data to testify the results from automation are not erroneous. This analysis continued for about a month. In the first phase there was 20% difference in values, but the automation tools is correctly configured and properly calibrated to minimize the error. In second phase of certification testing after a month automation team noticed only 5% difference between manual and automated test results which is very much acceptable. Control on this environment will be reflected in appendix phases of the thesis environment.

4. The whole application release life cycle has reduced a lot due to overnight automated testing process implementation and reduction of manual erroneous work. Following is a picture which demonstrates how the manual process or release cycle testing is replaced with automated testing for any application/service in Case Company Windows Phone Solutions finished or unfinished products.

5. There are not many advanced tool chain for testing application launch timings as well as response times and as explained previously manual testing results are inaccurate. So, testing all applications and services automatically gets rid of manual testing dependencies totally. In addition the tool gives detailed reports on memory usage for processes, applications on run time. This tool for certification testing,
5.2.4 Dashboard & Crash Portal

One of the pioneer way of improvement which this thesis process concentrates is visibility to higher management about the scope of betterment of processes followed inside Case Company to develop and build mobile products. This is accomplished by developing scripts to integrate in crash portal as well as developing user interface to display the trends and charts for all applications’ test results. Scripts are written in python and dashboard UI is developed in C# language. Script is integrated to the CI system, whenever there is single run of functional or non-functional test cases, any crash from a device is automatically uploaded to a shared network drive. A chrome job is created which identifies if there is any "cab" file present in the network drive, if there is a cab file the file is extracted and processed automatically. The data from crash is uploaded to the crash portal database and displayed to the dashboard.

Actions taken from CI team:
As part of thesis process, CI team led by me studied the importance of displaying test results from all sorts of testing for all applications to the dashboard. Managers inside the open development center can anytime check which applications failed in execution of functional, non-functional test cases and for how long. Failures are marked as red and passed test cases are marked as green. There are various dashboards which display different results from test assets.

1. Certification, Functional and Unit test asset trend charts
2. Crash data from all applications resulted from testing in tabular formats are displayed in UI for dashboards.
3. Trainings and awareness across sites on how to setup similar dashboards. Back up of the data for one month is kept for reference in another sever by automatic replication.
4. Extending & supporting functionalities according to custom requirements.
5. Unified strategy across sites in crash data display in tabular format and analysis.

5.2.5 MTBF testing for all application releases:

What does MTBF mean to mobile testing?

Mean time between failure (MTBF) refers to the average amount of time that a device or product functions before failing. This unit of measurement includes only operational time between failures and does not include repair times, assuming the item is repaired and begins functioning again. MTBF figures are often used to project how likely a single unit is to fail within a certain period of time.

Target of this project is to utilize the maximum bandwidth of available automation frameworks to the full from initial days of product development. Framework used is MTBF or “Mean Time between Failures” in identifying bugs while developing features from initial product development phase.

MTBF impacts both reliability and availability. Before MTBF methods can be explained, it is important to have a solid foundation of these concepts. The difference between
reliability and availability is often unknown or misunderstood. High availability and high reliability often go hand in hand, but they are not interchangeable terms. 

*Reliability is the ability of a system or component to perform its required functions under stated conditions for a specified period of time.*

In other words, it is the likelihood that the system or component will succeed within its identified mission time, with no failures. 

*Availability, on the other hand, is the degree to which a system or component is operational and accessible when required for use.*

MTBF, or mean time between failures, is a basic measure of a system's reliability. It is typically represented in units of hours. The higher the MTBF number is, the higher the reliability of the product. Equation below illustrates this relationship:

\[
\text{Reliability} = e^{\frac{-\text{Time}}{\text{MTBF}}}
\]

\[
\text{Availability} = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}}, \text{ where MTTR = Mean Time to Repair (or recover).}
\]

Case company has an existing framework of MTBF developed as part of the application tool kit and supplied to all the OEM’s who are interested in testing. As the case company mobile operating system evolves new application kit is releases via Hydra server to us. The MTBF mobile test suite comprises of a rich set of test cases including making and receiving phone calls, send and receive sms, send and receive mms, send receive mails, send receive files through Bluetooth, playing music files, playing videos etc.

Before thesis work started case company didn’t utilize the maximum possible testing from this MTBF suite. MTBF test suite consists of PowerShell scripts and an xml configuration file which dictates how many times the test case should be executed per loop, how many loops there will be and also how much time for which the testing should continue. In addition there are many settings which can be configured in the xml file to set the sim number to the phone wherefrom sms/call should come and be received. Thesis highlights how this MTBF suite can be extended to an extent where each and every application delivery can be made more and more robust through a thorough cycle of MTBF testing.

Previously MTBF testing only was performed on the traditional test cases as executed previously. Test automation and CI team has identified and tested that MTBF test suite can be modified and reused in long period testing where all the appli-
cations and services part of the Applications and Software organization is tested daily. The crashes found from the testing is analysed via xperfview and perl scripts. The script output pinpoints to the .dll file which crashed during execution of certain test cases.

MTBF test suite properly configured for long period testing for all the applications make a very early detection of crashes from all applications developed inside Application Software Development. In this test scenario the applications are invoked several times inside a loop of testing so that automation team leverages the maximum possibility of crashes. Early detection of clashes leads to early analysis of causes of crashes, which leads to early correction of the crashes. This leads to more robust application software development. Statistically it was seen that 80% of the crashes are corrected at very early stage of software development which huge improved the supply chain delivery of application software, OS; in turn the consumer product.

Usage of MTBF in testing all the applications part of Application Software Development, hugely reduced the time to market for delivery of tested applications. MTBF testing used for application software behaves like system testing, which tests interoperability between applications, e.g. camera application invoking gallery to show the pictures is executed 100 times (as an example) per loop. In the first run itself we found there was 23% of failures for some crash in the application tool kit. There are many instances of these kind of crashes detected from very early stage of software application as well as kit development.
Figure 5.4 Bathtub curve to illustrate constant rate of failures

Picture below shows MTBF implementation for application software.

Figure 5.5
MTBF execution as LPT in Application Software Development CI team, cures 75% of crashes at early stage
6. Testing the prototype proposal for case company
6.1 Introduction to testing methodology in case company

Testing is very important for understanding the fruitfulness of the proposed prototype. This gives a comprehensive understanding of business savings we get implementing this solution. The savings can be both hard savings and soft savings. Hard savings mean getting the maximum work done from minimum resource allocated which could be directly monetary savings. This directly reduces operational cost of the company, which directly adds to the revenue of the company yielding to more profit. The soft savings meaning not directly saving in terms of money, but saving time or eliminating bugs inside processes followed to make organization more lean and operate with maximum efficiency. Saving time means faster products to market, which means time to market for consumer products are reduced. This could be applied to all the processes e.g. bug triaging, continuous integration methods followed, faster robust development cycle with least number of trailing bugs and then very efficient testing cycle which gives quick, informative description of errors. The prototype proposed for case company is development of automated test cases and relying on automated test results for applications part of Windows OS and reducing manual work. This reduces the errors caused in manual testing and provides faster feedback to developers who can act quickly to correct errors and whole release cycle for applications to marketplace is reduced significantly. For case company thesis process not only just compare the results of manual and automated testing but also has taken help of lean six sigma technologies in depth to prove with more than 95% confidence that the proposal is excellent. Thesis process uses a tool named “minitab” on a small case project e.g. localization testing to prove how would the new improved process be so much superior than the current existing process of testing and releasing for consumers. Thesis compares the capabilities of existing and proposed processes and at the end highlights the business impact in terms of money as shown in coming paragraphs. The pilot phase of the solution is applied on our language variant testing project team on production where there were pretty many observations of delayed cycles and waste of resource and company money involved. In parallel the solution is also piloted to global release and integration teams for possible usage and feedback to propose the final solution.
6.1.2 Validation of proposal by Lean Six Sigma methodology

This paragraph describes in details how I did the testing of proposal to the case company. The whole idea behind Six Sigma is for organizations to become better and better continuously; better in the sense serving customers better or reducing scrap and rework or consistently providing better products and services with a minimum non-value added activities. Lean six sigma is an approach to sustainable continuous improvement that moves business towards the goal of being a world-class company. It fosters common language and cross-cooperation, helping business to develop expert personnel and leaders. Besides it increases capacity with minimal capital, meets productivity goals and improves customer service.

It is a combination of two powerful strategies, those complement each other to achieve performance organizations.

**Lean:** Lean originates from Toyota, it's both philosophy and methodology. Following this methodology ushers speed.

**Six Sigma:** Six sigma is data-driven disciplined improvement methodology. Focus is to reduce variation in all process steps. From here come stability and accuracy.

In Lean six sigma we follow DMAIC which is an acronym for Define, Measure Analyze, Improve and Control. Inside Application Software Development organization we follow DMAIC methodology in all day to day testing and automation work. Incorporating this helps in statistically proving that the work done is best possible with 95% confidence. Minitab is a very robust and powerful tool widely used inside both Case Company to reflect statistical data analysis.
Figure 6.1 DMAIC is the way in six sigma

Continuous improvement is very closely monitored across all projects in quality organization. The main idea is to eliminate waste from all processes and reduce variance in process (better stability). Applying Lean Six Sigma automation core team did a project to statistically prove with 95% confidence that removing manual steps delivered and increases efficiency of processes followed which in turn helps in lot of improvement in time to market.

Below are shown project graphs from the language variant testing project where performance is hugely improved after introduction of automation. We started with base-lining all projects inside Application Software Development Quality and developed tools for automated testing. This helped not only in faster reporting to developers and less error-prone i.e. increased quality.

Firstly inside language variant testing core team identified the problematic areas where improvement is needed (can be automated) and is really critical to business. This is done through Cause & Effect diagram. This is used as C&E diagram shows us the root causes wherefrom the problems of delayed testing cycle comes up. Next step after identifying the root causes is prioritizing the most important cause(s) amongst many, fixing which would fix 80% of the problems.
After identifying the root causes of problem, we used Pareto chart to identify the most significant issue amongst all, fixing which would fix 80% of the cumulative issues. This data is historical, i.e. data collected from day to day work or process for last six months. This identifies potential leaks of the management process and shows us by quantitative numbers wherefrom the maximum failures are.

Figure below explains in details, we use Pareto chart to distinguish and prioritize different problems identified in C&E matrix.
In quality organization replacing erroneous manual work with automated tool output has been bench marking in continuous improvement methodology. We did a current capability analysis of the project once we got (internal) customer specifications. The graph highlighted all the values are outside customer specification limits. In our example sample project, the customer wanted whole localization sanity testing to be finished in maximum 4.5 hours' time, but due to errors, communication gap, poor tester capability and other budget constraint, the project used to take much longer time to complete. Core CI and automated team developed scripts and automated the whole sanity testing process. This reduced the mundane manual work for testers and also improved quality. Previously in manual testing the device was tested in only 3 different languages, now the script tests the device in all the 55 different languages which our company offers to customers before shipping them to store. Process capability for current manual activity is shown below:

Figure 6.3 Pareto chart highlighting most defective category causing delay
In improved phase we proposed the prototype solution to the internal customer, we piloted the benefits and setup two production machines in Application Software Development CI laboratory where testers could have access and take this solution into use for feedback. In improved process we drew IMR chart comparing the capabilities of previous and improved processes. Here is below the comparison graph.

We recorded statistical data from the project regarding time taken to complete the project with customer expected quality. This following graph proves that after we took this new process into use, the data is well below the customer specifications and number of defects is zero. This was a “shift” project where the end result was an improved mean time for the process followed in the proposed solution. Quality is most important for our customers and we have “war on warranty” project which takes into account all possible reasons of defects found in the shipped devices. Reasons identified are either hardware or software of the products in market. Software issues are faced frequently whereas hardware issues occur seldom.
The level of sigma is defined by current or improved process capability and is highlighted in below diagram.

The major achievement of the project is listed below:

- Observed significant reduction in the mean of BAT validation time with the new process.
- Significantly improved the Sigma value (2.66).
- Significant reduction in the DPMO = improved quality of process and software.
- Majority of the Language testing cycle time values have come within the USL.
- The process mean changed significantly.
- The process standard deviation didn’t vary too much, this is insignificant in this particular sample example project as the majority of values of time taken was within internal project target limit.
- Internal target of project is set to be < customer specifications limit.
Figure 6.6 Sigma value at the end of pilot phase 2.66

**Business improvement from the small pilot project:**

Localization BAT Validation time is reduced from 6 hours to 4.5 hours per application. One and half hour of time costs around > 100 € / per week approximately for expert linguist tester. Manual testing covered 4-5 languages, but automation framework test all the languages per round. Taking in to account last 10 months statistics, 5 applications requests come for Localization testing per week. We save ~ 50K € per annum roughly.
6.2 Feedback & Learning:

6.2.1 Feedback

The Localization testing team is more than happy to use the tools developed by CI and automation engineers to test their release candidate. Localization testing team is centralized team which is involved in localization for all applications delivered from the case company. The deliveries from this Localization testing project itself is very much business crucial as this affects all the variant products releases in the market. This team delivers all the translation strings for all applications in all supported languages.

Feedback from different roles are collected as below:

1. **Developers:** Any project which submits code to the repository, triggers a build which creates the translations for all supported languages. Once the application becomes a release candidate, Localization testers test few of the languages and pass for full functional testing to external vendors. Few languages are tested because of lack of time before the final release could be made, which led to errors and more often developers needed to correct or re-factor the code and submit fixes. Automated tool for Localization testing verification tests each of supported languages and overnight testing gives faster result to them. Developers are very happy with the automated tool output which helps them to develop faster with less errors.

2. **Testers:** Introduction of automated tool give the testers freedom from boring manual work and they are able to do potential productive work in that time. Only few e.g. three of the major languages were tested in each cycle of release testing, but the automated testing tool checks localization in all supported languages. This yields best possible quality and eases release cycle for all applications.

3. **Program managers:** Program managers compared the results of manual versus automated test results. Initial pilot version yielded only 15% deviation of results along with manual testing meaning there were few false alarms and false negatives in the results. This feedback was very important for CI & test automation team (us) and helped a lot in building the final proposal for the case company.

4. **Line managers:** Line management is very happy with automated tool output, cost to return ratio is very high. Senior management has taken initiative on
more and more trainings to be arranged for lean six sigma which would give maximum benefit to the organization in the long run.

6.2.2 Learning for case company

This whole automation project is of great learning for the team as well as the company. In today’s fast moving world, it is a challenge for any company to continuously maintain and improve the quality and efficiency of software systems development. In many software projects, testing is neglected because of time or cost constraints. This leads to a lack of product quality, followed by customer dissatisfaction and ultimately to increased overall quality costs. The main reasons for these added costs are primarily poor test strategy, underestimated effort of test case generation, delay in testing, and subsequent test maintenance.

Test automation improved development process of a software product in many cases. The automation of tests is initially associated with increased effort, but related benefits quickly paid off.

The main goal in software development processes is a timely release. Automated tests run fast and frequently, due to reused modules within different tests. Automated regression tests which ensure the continuous system stability and functionality after changes to the software were made lead to shorter development cycles combined with better quality software and thus the benefits of automated testing quickly outgain the initial costs.

Employees in both engineering and quality have understood underlying potential of lean six sigma technology after demonstration of the statistical comparison from current and improved process capabilities. A significant group of testers and developers also shown interest to get training in six sigma methodologies. The trainings make them aware of how to eliminate processes which cause delays or schedule slippage inside organization. Key learning from the exercise is more the organization operates in lean way eliminating old time consuming processes, more efficient the deliveries are. Deliveries from all teams are on time and the product life cycle is shortened, leading significant reduction in time to market for consumer products.
6.3 Summary and final proposal for case company

In this chapter thesis documents the first prototype version of the test automation to the case company. Several trainings are arranged inside the company to make the manual testers aware of the advantages of automation. Also general information sessions are held every week where core CI & test automation team explains in more details

1. What has changed in daily work model than past?
2. What all the release integration teams should do?
3. What all the release integration teams shouldn't do?
4. What all the release integration teams might expect in future?
5. Introduction to coming features
6. Awareness to lean six sigma methodologies

Info sessions inside company has also made the communication system better than past. As the team is spread across different time zone and culture, reaching out to everyone takes lot of time and effort on top of the development work for CI. Manual testers spent a day time in a week understanding how to use automation tools, interpret results from test automation, debug initial failures and report to the developers about unit test failures. Testers all of application software organization participated in the workshop *in phases* so that the business continuity is not at all hampered.

The first prototype solution has many non-standard implementations, the reason behind that mainly lack of time, resource and budget constraint. The non-standard implementations don’t affect the functionality of the prototype at all.

Standardization is a process which includes understanding needs of internal customers, doing a small project in developing the scripts needed, providing the prototype for testing to customers then correcting those scripts according to needs before putting to production usage.
The diagram below shows a comprehensive proposal for case company where manual testing is not relied upon anymore for releasing the applications to OS or marketplace, but automated testing results are taken into considerations. The release candidate is decided in Application Release Decision Review meetings. In these meetings, when manual testing gave the final verdict, that time there was possibility that testing team could, sometimes overlook the results of minor functionality errors in applications and still release it. When automated testing is taken into use, the final label description has all the results from ft (functional tests), ct (certification tests), unit tests, and localization (language testing) results appended. In Application Release Decision Review meeting, even if one of the tests have failed, the quality assurance engineer is questioned about it in details.
Figure 6.8 Final proposal for case company, manual testing replaced with automated testing.

The feedback from Localization and release integration teams are taken into account. All the non-standard implementations e.g. .bat scripts, perl code are re-written into python. All the blocks in above diagram are implemented in python code from beginning till end. Also the Crash Portal mentioned in last chapter script which collects the .cab files produced during test execution.
7. Conclusions
7.1 Summary

This chapter gives the general overview of the thesis, it is a look through of all the steps done in the research process. Validate the research by comparing the objective with the results, how applicable and generalized are the results in another context. And also check how reliable the research process is, if the methods and materials used in data collection and analysis are appropriate and whether this research could be carried on further by another person, in the same way it is done currently.

7.2 Practical Implications

The thesis was aimed to study all possible causes for any possible delays in time to market for finished products from case company. This include that exists in current practices in development methods and tools, in the organisation in case company, in comparison with the best practices in test automation, available from different sources and summarised in the literature part of the thesis, and to propose the solutions would fill the existing gaps and would provide the ways to improve in all way the release cycle of phone software as a whole with better quality for all new products as well as in-market available devices. The current state analysis was done by discussions with many individuals, stakeholders and teams in different kind of roles, in different teams across the release organization. These discussion turned out to be a great source of knowledge and insight into the organisational process, mainly it gave an insight and good understanding of project planning and execution methods. Everyone involved were in highly responsible roles, hence the discussions were more clearly depicting needs for leveraging full utilization of test automation for releasing phone software, which helped to frame the proposals that would address concentrating on automated test suite development in an improved way and help all the key stakeholders of the delivery organization to deliver better quality products with less schedule slippage.

The research was mainly based on the current ways of working, current issues and needs that were desired by the team members. The proposals are improvements that are needed in three different area of using test automation to the full, building very robust test automation reports from test executions, replying on the test results to decide
on releases from applications and services. This is framed based on the research articles in the references. The feedback about the proposal was very positive, senior management showed very much interest on investment in building a very good test laboratory where robust testing can be done utilizing 24 hours a day time as mentioned in this research.

The current research process addresses in developing full automated test cases e.g. functional, non-functional, unit test suites to have better release process for all applications and services part of OS as pre-load or available from Marketplace both inclusive.

7.3 Evaluation

7.3.1 Objective vs Outcome

The main objective of the thesis was to find the ways that would improve quality of all released products’ software for all phones (inclusive new products & in-market devices) along with reducing schedule slippage for all new products as well as reducing delay in OTA updates for in-market products. In the study three main proposals were created to improve the visibility of the situations in the release and testing processes which cause potential delays and totally eradicating last moment fixes which many times might introduce regression in the software. First proposal has been to improve the wide coverage of unit, functional and mtbf test suites for all applications and services delivered in order of importance with very fast pace. Second was to improve the reliability of the test results from automated execution as well as coverage of test cases for all new features. Thirdly the most important part of thesis proposal was to create awareness amongst developers and testers across the whole organization about this project very widely in info sessions and proper trainings. Manual testers are given thorough trainings about automation project status and latest updates so that they are able to interpret the results from automation and give judicial feedback to improve the continuous integration system. The final proposal to the case company comprises the results from these three pieces of proposed stages in thesis clubbed together. The main customer of this thesis is release testing teams across Windows phone world, who is very much benefited by on time release of properly tested applications and services using the philosophy of software integration. All new applications and services released are tested repeatedly over the night to give quick feedback to the developers. Outcome of the thesis is very close to the objective of the thesis, rather I would say in the process of
7.3.2 Reliability

Reliability checking for the thesis is done mainly based on the validating how well the methods and the materials used and well the outcome addresses the objective of the thesis. The methods and materials used in this thesis, is mainly from the referenced articles and books, and the teams and people involved are from the organization in which the case study is done. The objective and the outcome of the study are closely related to the real situation and more relevant to the study topic. In the literature part contents are designed in a way that test automation development tools used to write unit, functional, certification and mtbf test suites are highlighted in details; this report would serve the purpose of referencing as a guideline to design and improve them. Also current state analysis were conducted in a more generalised way, with more generic question about problems in test release organization so that the inputs from those interviews would give a generic overview of the process and the needs for improvements in the project. In this way this study has initiated a research which would further be continued in an elaborate mode, for example improvising the tools needed to develop and release software with very robust quality over and over by time. The newer versions of the tools which give detailed or more customized reports from testing and dashboards help the developers, testers with improved visibility and the continuous integration team with better hands to help them. More features in the tools help in more visibility in reports across teams, piloting the proposals, are some of the key research areas for future studies.

7.3.3 Validity

From the practical implications explained in the beginning of this section, the answer to the research question “How the time to market for all products released can be reduced significantly?” “How can products’ quality be of premium quality with maximum user experience? “How can we minimize the schedule slippages?” are an-
answered in the proposal section and is addressed completely. Hence this states that the research has a consistent and logical approach in providing what is promised in as objective. Also the study could be continued further by another person. The study uses proper techniques and materials to base the outcome.

Lean Six Sigma Black belt course learning methods are used to control various inputs which might affect the output. The output could be either continuous or discrete. Continuous output could be measuring lead ‘time to market’. Discrete could be ‘how many defective products per million’. Similarly there can be case where products could be affected by many external factors (again factors could be continuous or discrete or a combination of both). Six Sigma Black belt training course properly guides with a wide range of toolset which suffice to statistically calculate, predict the quality of good products which could be manufactured in given time. The tools used to statistically prove with 90% confidence that there are no mistakes from the beginning to the end of any product life cycle coming from case company. The tools help in identifying all the bugs inside all the processes followed inside the company. All the processes get improved day by day and the mindset of the employees got aligned to identify and eliminate waste in all possible in day to day work. Better work from all employees means improved productivity, better productivity leads to innovative products to market with minimum ‘time to market’.
References:

https://meego.com/
https://www.testrunonline.com/
http://www.runtestrun.com/features.php
http://www.perforce.com/support-services/elearning

All Windows testing based development tools and ready made packages:
http://msdn.microsoft.com

Denning S (2014), the future workplace is now: How Etsy makes 30 innovations per day

Matthew Goche and Trevor Christiansen (2014), Five steps to make better Apps: A cookbook for mobile application security
http://www.forbes.com/sites/sungardas/2014/04/22/5-steps-to-better-apps-a-cookbook-for-mobile-application-security/

Stuart Parkerson (2014), Perfecto mobile offers support for Windows Phone 8.1 App Testing

Patrick Day (2014), n-Tiered Test Automation Architecture for Agile Software Systems
Procedia Computer Science Vol. 28, pp 332 – 339

Wendy Torell and Victor Avelar (2012), Mean Time Between Failure: Explanation and Standards, Schneider Electric – Data Center Science Center White Paper 78 Revision 1

Scott Speaks “Reliability and MTBF Overview”, Vicor Reliability Engineering

Reliability Analysis Center, Reliability Toolkit: Commercial Practices Editions, Rome Laboratory.

Dorothy Graham Mark Fewster, Software Test Automation, Effective Use of Test Execution Tools Addison-Wesley 2010


John Ferguson Smart ‘Jenkins - The definitive Guide’ www.it-ebooks.info
Jez Humble and David Farley ‘Continuous Delivery’

Cohn, Mike (2009) ‘Succeeding with Agile: Software Development using Scrum, Addison-Wesley


Laurie Williams (2006) Testing overview and Black Box Testing Techniques

Per-Gustaf Stenberg and Joakim Hembrink (2103) ‘The Ultimate Jenkins Setup’ version 0.1
Appendix 1

Questions about the current state analysis before thesis for case company

The data collection plan for case company had a numerous amount of questions regarding the current way of working inside release and test teams.

Questions to developers:

- Are you happy with the code development and refactoring tools?
- What are your comments on current continuous integration system
- How can you rate about fastness of the current feedback system
- In which way the CI system should be improved?
- What features are missing in the system right now?
- Are you over-loaded in work?
- Is this over-load somehow related to the slower feedback system?
- If it’s slower, is it the continuous integration system slower or the release testing cycle slower?
- If both of them are slower, which one amongst the two is severe problem in work life?

See the attached excel for the questions and actions from developers:

Developers.xlsx

Questions to testers:

- Are you happy with manual testing and release cycles?
- Are you overloaded?
- How often there are mistakes due to work overload?
- How often there are schedule slippages?
- Can automated execution of test cases help quality?

Testers.xlsx

Questions to management team:

- In which ways the product development life cycle can be improved for future products?
- Were there any schedule slippages recently?
- How many programs had regression bugs which became blocker?
- Do you need improved visibility to the applications download/crash counts all the time basis?
- How badly product program is affected due to schedule slippage?
- How bad is impact on operators for schedule slippage?
- How can the continuous integration or release testing teams work better?
- What is the key goal for us to make all the future products succeed?
Appendix 2

Comments from Questionnaire

Program Managers:
- Goal is to have best software quality
- All products released on time
- Minimum customer (Operator) complaints
- Minimize the number of over the air updates for any products due to regression

Testers:
- Remove bureaucracy inside organization
- Automation would be of great help in every stage
- Over the night automated testing could be one key help
- Test results should be readable
- More trainings and more feedback needed
- Improved communication across teams help a lot

Developers:
- Unit test case execution should be supported
- Laboratory should have improved hardware
- All latest test devices should be in the Continuous Integration test pool
- Functional/certification test cases should be executed per commit from developer
- Generating binaries and test execution with latest binaries should be atomic operation
- Faster feedback about new code in emails.
- More robust defect detection mechanism through MTBF from the VERY beginning of any product development life cycle.
- More exposure through dashboards about the crash information’s.
- More and more long period testing from all applications and services.
- Standard test result e.g. xml or .trx for all test suite execution, easy to interpret.
• Common naming conventions for application/service codebase guided by CI.

Questions to commercial operators:
• On-time delivery
• All feature requests supported
• Minimal customer complaints

Appendix 3

Questionnaire after finishing the real time project
There is always scope of improvement in real world working environment. The more automation, the more is manual work reduction. After a certain phase comes the era where there is fine lining of all the individual small projects inside the umbrella project. The feedback from developers, testers and management is very much positive and the overall outcome of the thesis is a laboratory facility with increasing budget and attention which facilitates these automated test execution.

Feedback.xlsx