AUTOMATION OF A CLOUD HOSTED APPLICATION
Performance, Automated Testing, Cloud Computing

JYOTHI SPANDANA PENMETSA
This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Masters in Electrical Engineering with emphasis on Telecommunication Systems. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author:
Jyothi Spandana Penmetsa
Personal Number: 940625-8443
E-mail: jype15@student.bth.se

External advisor:
Debasish Nayak
Senior Manager, Enterprise Applications
EXFO Electro-Optical Engineering India Pvt. Ltd

University advisor:
Prof. Dr. Kurt Tutschku
Department of Communication Systems
School of Computing
BTH, Karlskrona

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden
Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57
ABSTRACT

Context: Software testing is the process of assessing quality of a software product to determine whether it matches with the existing requirements of the customer or not. Software testing is one of the “Verification and Validation,” or V&V, software practices. The two basic techniques of software testing are Black-box testing and White box testing. Black-box testing focuses solely on the outputs generated in response to the inputs supplied neglecting the internal components of the software. Whereas, White-box testing focuses on the internal mechanism of the software of any application. To explore the feasibility of black-box and white-box testing under a given set of conditions, a proper test automation framework needs to be deployed. Automation is deployed in order to reduce the manual effort and to perform testing continuously, thereby increasing the quality of the product.

Objectives: In this research, cloud hosted application is automated using TestComplete tool. The objective of this thesis is to verify the functionality of cloud application such as test appliance library through automation and to measure the impact of the automation on release cycles of the organisation.

Methods: Here automation is implemented using scrum methodology which is an agile development software process. Using scrum methodology, the product with working software can be delivered to the customers incrementally and empirically with updating functionalities in it. Test appliance library functionality is verified deploying testing device thereby keeping track of automatic software downloads into the testing device and licenses updating in the testing device.

Results: Automation of test appliance functionality of cloud hosted application is made using TestComplete tool and impact of automation on release cycles is found reduced. Through automation of cloud hosted application, nearly 24% of reduction in level of release cycles can be observed thereby reducing the manual effort and increasing the quality of delivery.

Conclusion: Automation of a cloud hosted application provides no manual effort thereby utilisation of time can be made effectively and application can be tested continuously increasing the efficiency and the quality of an application.

Keywords
Software testing, Black-box testing, White-box testing, Automation, Agile development software process
ACKNOWLEDGMENT

I wish to express my deep sense of gratitude to my supervisor Prof. Dr. Kurt Tutschku for his significant suggestions and endless support in every aspect to accomplish this thesis work. That helped enhance the quality of my thesis. His encouragement, patience and keen interests in discussions have benefited me to the extent that cannot be spanned by words. He channeled my thoughts and extended timely suggestions for which I am deeply indebted to him.

I articulate my profound gratitude to my external supervisor Sri. Debasish Nayak, Senior Manager, Enterprise Applications at EXFO, for his indispensable encouragement with salient guidelines and suggestions throughout the work. In fact, he helped me in crucial stages of my work for which I am indebted to him.

I am pleased to thank Mr. Ajay Kumar Jadhav, Mr. Bimal Kumar Patra, and Mr. Prasad Rane at EXFO who have rendered me their constant encouragement and endless help in bringing my project work make successful.

Also, I am pleased to acknowledge my deep in debt to all those who devoted themselves directly or indirectly to the successful completion of my thesis.

To my beloved ones, who shared their support and stood by my side during hard times. Thank you.

Above all, I would like to thank my parents in providing me the ability to complete this program. Their constant love and support helped me every day to come up with a new spirit.
ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon EC2</td>
<td>Amazon Elastic Compute Cloud</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>AWS</td>
<td>Amazon Web Services</td>
</tr>
<tr>
<td>CSV</td>
<td>Comma Separated Values</td>
</tr>
<tr>
<td>DAS</td>
<td>Distributed Antenna Systems</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>IaaS</td>
<td>Infrastructure as a Service</td>
</tr>
<tr>
<td>OR</td>
<td>Object Repository</td>
</tr>
<tr>
<td>OOP</td>
<td>Object Oriented Programming</td>
</tr>
<tr>
<td>PaaS</td>
<td>Platform as a Service</td>
</tr>
<tr>
<td>QA</td>
<td>Quality Analyst</td>
</tr>
<tr>
<td>QOE</td>
<td>Quality of Evaluation</td>
</tr>
<tr>
<td>QOS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RRH</td>
<td>Remote Radio Heads</td>
</tr>
<tr>
<td>SaaS</td>
<td>Software as a Service</td>
</tr>
<tr>
<td>TAL</td>
<td>Test Appliance Library</td>
</tr>
<tr>
<td>TDL</td>
<td>Test Data Library</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Allocation</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>XP</td>
<td>Extreme Programming</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS

Contents

ABSTRACT ............................................................................................................................................. I

ACKNOWLEDGMENT ......................................................................................................................... II

ABBREVIATIONS .................................................................................................................................. III

TABLE OF CONTENTS ......................................................................................................................... IV

LIST OF FIGURES ............................................................................................................................. VI

LIST OF TABLES .............................................................................................................................. VII

1 INTRODUCTION ............................................................................................................................. 1

1.1 PROBLEM STATEMENT .............................................................................................................. 1

1.2 HOW IS THE ISSUE SEEN BY COMPANY? .......................................................................... 1

1.3 WHAT IS CLOUD APPLICATION? .............................................................................................. 2

1.4 WHY AUTOMATION .................................................................................................................... 3

1.5 METHODOLOGIES USED ......................................................................................................... 4

1.6 RESEARCH QUESTIONS ............................................................................................................. 4

1.7 AIM AND OBJECTIVES .............................................................................................................. 4

1.8 SPLIT OF WORK .......................................................................................................................... 5

1.9 THESIS OUTLINE ....................................................................................................................... 6

2 RELATED WORK ............................................................................................................................. 7

3 BACKGROUND ................................................................................................................................... 9

3.1 CLOUD COMPUTING ................................................................................................................... 9

3.1.1 Essential Characteristics of a Cloud ...................................................................................... 9

3.1.2 Cloud Service Models ........................................................................................................... 9

3.1.3 Understanding Public, Private and Hybrid Clouds ............................................................... 10

3.1.4 Advantages and Challenges of Cloud Computing ............................................................... 11

3.1.5 Amazon Web Services and Cloud Computing .................................................................... 12

3.1.6 Amazon Services Cloud Platform ....................................................................................... 12

3.1.7 Amazon Elastic Compute Cloud (Amazon EC2) .................................................................. 12

3.2 SOFTWARE TESTING ................................................................................................................. 13

3.2.1 Basics of software testing ..................................................................................................... 14

3.2.2 Types of Testing .................................................................................................................... 16

3.2.3 Development Testing ........................................................................................................... 16

3.2.4 Release Testing ..................................................................................................................... 17

3.2.5 User Testing .......................................................................................................................... 17

3.3 NEED OF TEST AUTOMATION FRAMEWORK ...................................................................... 18

3.3.1 Components of Automation Framework .............................................................................. 19

3.3.2 Types of Automation Framework ........................................................................................ 19

4 METHODOLOGY ............................................................................................................................ 23

4.1 SOFTWARE DEVELOPMENT METHODOLOGY ................................................................... 23

4.1.1 Software Process .................................................................................................................... 23

4.1.2 Software Process Models ...................................................................................................... 24

4.1.3 Agile Methodology ............................................................................................................... 26

4.1.4 Agile Methods ....................................................................................................................... 26

4.2 SCRUM METHODOLOGY ........................................................................................................ 28

4.2.1 Phases in Scrum .................................................................................................................... 28
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Remote Radio Heads (RRH), Distributed Antenna Systems (DAS), Small Cells and</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Cloud-RAN</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Black-Box Testing</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>White-Box Testing</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>Components of Automation Test Framework</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>Module Based Testing Automation Framework</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>Library Architecture Testing Framework</td>
<td>20</td>
</tr>
<tr>
<td>7</td>
<td>Data Driven Testing Framework</td>
<td>21</td>
</tr>
<tr>
<td>8</td>
<td>Keyword Driven Testing Framework</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>Hybrid Testing Framework</td>
<td>24</td>
</tr>
<tr>
<td>10</td>
<td>Extreme Programming Release Cycle</td>
<td>27</td>
</tr>
<tr>
<td>11</td>
<td>Phases in Scrum Methodology</td>
<td>28</td>
</tr>
<tr>
<td>12</td>
<td>Applications of Cloud Application</td>
<td>30</td>
</tr>
<tr>
<td>13</td>
<td>Applications of Cloud Hosted Application</td>
<td>31</td>
</tr>
<tr>
<td>14</td>
<td>Example for Functionality of Test Appliance Library</td>
<td>32</td>
</tr>
<tr>
<td>15</td>
<td>Testing Device</td>
<td>32</td>
</tr>
<tr>
<td>16</td>
<td>Graph Showing Analysis of Automation and Effort Reduction</td>
<td>35</td>
</tr>
<tr>
<td>17</td>
<td>Graph Showing Analysis of Automation and Defects Reduction</td>
<td>36</td>
</tr>
<tr>
<td>18</td>
<td>Automation Status of Test Data Library</td>
<td>38</td>
</tr>
<tr>
<td>19</td>
<td>Automation Status of Report Analyzer</td>
<td>39</td>
</tr>
<tr>
<td>20</td>
<td>Automation Status of Test Appliance Test Cases for Ongoing Phase</td>
<td>40</td>
</tr>
<tr>
<td>21</td>
<td>Automation Status of File Download Test Cases</td>
<td>41</td>
</tr>
<tr>
<td>22</td>
<td>Distribution of Automated Test Cases over Total Test Cases Across Release</td>
<td>44</td>
</tr>
<tr>
<td>23</td>
<td>Percentage of Automation Achieved across Release Cycles</td>
<td>45</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 1: Table Showing Contributors for Each Section of Thesis Document .......................5
Table 2: Table Showing Automation % & Corresponding Reduction in Manual Effort % ....36
Table 3: Table showing Automation % & Corresponding Reduction in Defects .................37
Table 4: Current Status of Automation on Software Download of Test Appliance Library ..38
Table 5: Current Status of Automation on Operator Test Cases ....................................39
Table 6: Current Status of Automation on Operator Test Cases ....................................40
Table 7: Current Status of Automation of File Download Test Cases ............................41
Table 8: Table Showing Distribution of Automated Test Cases over Total Test Cases across Release Cycles ......................................................................................................45
1 INTRODUCTION

This chapter provides an overview of entire thesis document. The thesis work has been carried at EXFO Electro Optical Engineering Pvt. Ltd, India. This chapter mainly focuses on problem statement and description followed by the brief description of the cloud hosted application which is being automated, followed by research questions, objectives and thesis outline.

1.1 Problem Statement

In a hierarchical telecommunications network, the backhaul portion of the network comprises the intermediate links between the core network, or backbone network and the small subnetworks at the "edge" of the entire hierarchical network.

Mobile broadband services are undergoing a period of dramatic growth causing a tremendous increase in data traffic. This rising tide of traffic is being driven by the growing number of mobile subscribers, particularly smartphone users, who are connecting to faster networks and consuming bandwidth-hungry video content. Such is the rate at which mobile subscribers are consuming more data that new technologies alone cannot keep up.

Mobile operators are continuously making their networks more efficient by investing in new generations of mobile technology (e.g. 5G) and rolling out ever increasing numbers of cellular base stations as well as public Wi-Fi. However, these network upgrades will not be sufficient to meet growing demand - mobile services unless until the network deployment and test equipment is properly tested and analyzed.

As a result of ever-increasing competition in the telecom industry, service providers need to turn up next-generation Ethernet services more quickly to get revenue faster. What’s more, they need to manage a multitude of ever-evolving technologies and deploy mobile backhaul, data-centre interconnect and Ethernet business services more efficiently, which requires proper testing of the network equipment and analyzing the network results before deployment. Also continuous testing of the network equipment without manual intervention is essential for minimizing human effort and to increase the productivity of an organisation. So automation is considered essential for testing the network equipment and to analyze the network results before deployment.

Conversely, field technicians need to rapidly learn and gain proficiency in all these new technologies to keep pace with a host of new testing requirements. This poses a challenge to many service providers, who have limited budgets for test equipment and training, and are struggling to reduce repeat calls and network issues.

1.2 How is the issue seen by company?

To support mobile network operators (MNO) in rolling out additional capacity and new cells in shorter time spans, company provides a complete solution covering turn-up, activation and troubleshooting of remote radio heads (RRH), distributed antenna systems (DAS), small cells and cloud-RAN. The increase in the network performance is significant when the potential throughput increases from 100Mbps to 1Gbps and above.
This requires fiber-optic media that has no defects and transmits data to the users with proper estimated quality. In other words, customer satisfaction is related to the Quality of Service (QoS) provided by the Network service Providers which is in turn dependent on the Quality of evaluation (QoE) of the networks.

The company has found a solution, launched an application called “Cloud Application” for verifying and validating the network performance. This is done by validating the test reports generated by the service provider by setting target thresholds and filtering the valid criteria. Key features of Cloud Application are:

- Provides suitable environment for uploading, downloading and testing the validity and verification of network reports by allowing the users to set their own compliance criteria for testing.
- Acts as a cost-effective huge data storage base for the customers.

1.3 What is Cloud Application?

Cloud Application is a comprehensive, cloud-based, data-management service that is specially designed to optimize telecom field-tests workflow operations. Company’s Cloud Application is a unified web based solution that provides inventory management, automatic software downloads to test platforms, and also acts as a repository for historical test data.

It is based on client-server architecture; with server hosted on cloud while client running on test platforms.
Cloud Application helps to:

- Get the global view of all test platforms.
- Get Snapshot of platform applications installed on the platforms
- Find what modules were installed on the platforms
- Find license status and download a new one, if available.
- Download latest products packs and service packs to the platforms.
- Upload and view test reports on a centralized data server.
- Exchange any type of file(s) within platforms.

The software services delivered by Cloud Application include:

- Near real-time test result handling with a closed feedback loop to correct things quickly.
- Powerful analytics to track the quality issues against budgeted thresholds and record for historical tracking.
- APIs for north bound data and process integration.
- A powerful reporting engine
- 24/7 global user access through a Web-based interface,
- Secure communication links for both users and test equipment,
- Application programming interfaces (APIs),
- A fully managed infrastructure for security and backups.

So the main aim of this thesis is to automate the cloud hosted application known as in order to keep track of the inventory management, automatic software downloads into the test platforms. The following section gives brief description on the importance of automation and why it is needed.

1.4 Why Automation

Software testing is the process of assessing quality of a software product to determine whether it matches with the existing requirements of the customer or not. It is done by eliminating bugs in the software and hence evaluating the features of the software product. Software testing is one of the “Verification and Validation,” or V&V, software practices. The two basic techniques of software testing are Black-box testing and White box testing. Black-box testing focuses solely on the outputs generated in response to the inputs supplied neglecting the internal components of the software. Whereas, White-box testing focuses on the internal mechanism of the software of any application. To explore the feasibility of black-box and white-box testing under a given set of conditions, a proper test automation framework needs to be deployed. A “Test Automation Framework” is scaffolding used to provide an execution environment for the automated test scripts generated. Testing framework focuses on developing, processing and executing the automation test scripts for each part of the application efficiently.

Cloud is an internet-based computing where resources are shared and huge database information is exchanged. Large number of clients can access the cloud through the telecom application hosted in it. To withstand the evolving requirements, there is a need for updating applications and its features regularly, as well as continually increase the scale of operations and capacity handling. Also there is a need to simulate an application in real-time environment using several automation frameworks before deploying an application in a cloud. An application needs to be validated using large number of test cases to meet its designed expectations. These test cases will both be regression test cases (test cases used to ensure whether previously implemented features are functioning correctly or not) and new test cases.
cases that are used to validate newly implemented features in the application.

Main focus of an organization is to automate all regression test cases. At the same time the new test case that arises for the current feature will be regression test case for the next feature. When multiple features of an application are developed simultaneously by different team in an organization, it is ensured that functionality of the code developed by them is not interrupted. This requires continuous testing of the application and huge effort if done manually. For successful operation of the cloud by different organizations after hosting it in a remote environment, specific challenges are to be met and need to be addressed. To confront these challenges, Automation is required and is expected to bring necessary efficiencies so that as much regression testing is done without manual intervention.

1.5 Methodologies Used

In order to automate the cloud hosted application, methodology followed is scrum methodology which is agile software development process. Agile process is an alternative to the traditional waterfall model, which helps teams respond to unpredictability in the product development, through incremental, iterative work cadences known as sprints. Scrum is a popular and simplified framework implemented in agile methodology. It reduces the complexity involved in building the software product. In scrum methodology, a product can be implemented in series of fixed length iterations called sprint lengths where the software of a product can be made shippable to the customers in the respective sprint lengths. Also, the product with working software can be delivered to the customers incrementally and empirically with updating functionalities in it. In order to implement scrum methodology, the system should be well automated enough to have product deliverables within sprint length.

1.6 Research Questions

The aim of this thesis is to answer the research questions RQ1, RQ2, RQ3 listed below and the research questions are answered by performing automation of a cloud hosted application by conducting an experiment.

RQ1) How to identify and analyze a right automation test data driven framework for automating cloud hosted application?

RQ2) What level of release cycles can be reduced to maximize the efficiency of automated testing of cloud hosted application?

RQ3) How to evaluate the tool for the automation testing of cloud hosted application?

1.7 Aim and Objectives

The aim of this thesis is to automate the software testing process of company’s cloud hosted application which is a web interface or customer interface application. Here, the main aim is to verify the Test Compliance Library, a functionality of cloud hosted application using the right automation framework, employing the software testing tool called the TestComplete, where the application is tested continuously in order to determine whether the software upload and test file upload into server is made incrementally or not. Using this functionality, the customer can become aware of the software uploads and test file uploads into the server periodically and empirically.
The following are the objectives to be studied while automating cloud hosted application:

- Study of the software testing tool called the TestComplete.
- Detailed study on the design of proper automation framework for software testing of cloud hosted application.
- Evaluation of the testing tool based on management goals, testing requirements and the technology.
- Detailed study of Test Compliance Library which is a functionality of the cloud application.

1.8 Split of Work

This section gives details about the work contribution of this thesis document.

<table>
<thead>
<tr>
<th>SECTION</th>
<th>TOPIC</th>
<th>CONTRIBUTOR</th>
</tr>
</thead>
</table>
| 1.1 to 1.5 | 1.1 Problem Statement  
1.2 How is the issue seen by company  
1.3 What is Cloud Application  
1.4 Why Automation  
1.5 Methodologies used | Jyothi Spandana Penmetsa, SriKavya Chavali |
| 1.6 | Research Questions | Jyothi Spandana Penmetsa |
| 1.7 | Aim and Objectives | Jyothi Spandana Penmetsa |
| 1.9 | Thesis Outline | Jyothi Spandana Penmetsa |
| 2 | Related Work | Jyothi Spandana Penmetsa, SriKavya Chavali |
| 3 | Background | Jyothi Spandana Penmetsa, SriKavya Chavali |
| 4 | Methodology | Jyothi Spandana Penmetsa, SriKavya Chavali |
| 5 | Experiment | Jyothi Spandana Penmetsa |
| 6.1 | Common Results Between Both Thesis | Jyothi Spandana Penmetsa, SriKavya Chavali |
| 6.2 | Individual Results: Results for Automation of Test Appliance Library | Jyothi Spandana Penmetsa |
| 7.1 | Conclusion | Jyothi Spandana Penmetsa, SriKavya Chavali |
| 7.2 | Future Work | Jyothi Spandana Penmetsa, SriKavya Chavali |
| 8 | Answering Research Questions | Jyothi Spandana Penmetsa |

Table 1: Table Showing Contributors for Each Section of Thesis Document
1.9 Thesis Outline

Rest of the thesis is organized as mentioned below:

Chapter 2: Chapter 2 is about related work. This chapter portrays the research work referred by us in order to strengthen our background knowledge.

Chapter 3: It is about background knowledge, it explains the concepts related to cloud computing, software testing and automation testing. This chapter gives clear view on software testing and types of software testing. Also concepts of automation is explained in this chapter.

Chapter 4: It is about methodology used in our research. The methodology used for the company's product delivery is scrum methodology. Various software development models are explained in this chapter. Also, scrum methodology which is the most popular approach in agile methodology is described in this chapter.

Chapter 5: It is about experimentation. In this chapter, functionality of cloud hosted application called test appliance library is described and the testing device used for testing the test appliance library functionality is explained.

Chapter 6: This chapter is about numerical results. Common results across the both thesis are explained with graphical representation and values are tabulated in this chapter. Also, individual results are explained with graphical representation and tabular forms.

Chapter 7: This chapter is about conclusion and future work. Conclusion for this research work is portrayed in this chapter. Also, future work is described here.

Chapter 8: This chapter is about answering research questions of this thesis. The research questions mentioned in chapter 1 are answered in this chapter.
2 RELATED WORK

In reference [31], the author's explained in detail about the basics of software testing, a verification and validation practice throughout the entire software development lifecycle, the two basic types of software testing, black-box and white-box testing, types of testing that involves both black-box and white-box testing, strategies for writing fewer test cases and also template usage for writing repeatable and defined test cases. This reference helped us in gathering background knowledge of software testing and various type of testing techniques. Also, this reference helped us in defining and writing test cases for automation of a cloud hosted application.

In reference [2], author's described in detail about agile methodology and most common agile methodologies like Extreme Programming (XP), agile modelling (AM) and scrum methodology are discussed in this research paper. Here author's discussed the working principles of agile development and proposed a new practical approach for agile software methodology. This reference paper helped us in gathering background knowledge regarding agile methodology which is a proposed methodology for delivering the final product to the end users.

In reference [3], author's discussed in detail about scrum process which is an agile methodology. In this research paper, the agile requirements refinery is presented which is an extension to scrum process that enables product managers and developers in a team to cope with larger requirements in an agile software development. Here author's described about scrum development method, agile software product management and requirements refinery. The main aim of agile methodology is to develop software quickly, focusing on people and frequent delivery of software. This paper helped us in gathering background knowledge of scrum methodology which is mostly widely applied process in agile methodology. It is widely applied due to its ability to complement other methods and processes.

In reference [4], author's introduced an automated test case generation approach for industrial automated applications which are specified by UML state chart diagrams. In this research paper author's discussed manual approach, semi automated approach and automated approach for generating test cases. The author's also discussed and gave a clear view on test case planning, test case generation, test case implementation and test reporting on various levels. This research helped us in planning the test cases, generation of test cases, implementation of test cases and reporting them across various levels for automating the cloud hosted application which is a web based application.

In reference [5], author's described in brief about data-driven automation framework and keyword-driven automation framework. Here, author's described in detail about automation framework and this research paper shows how to design and implement an automated testing framework with strong reusability, expandability and high robustness. Here, author's described in detail about the model of automation testing framework. This research paper helped us in designing test data driven framework for automating the test cases of test appliance library functionality of a cloud hosted application.

In reference [6], author's described various automation tools like Selenium, TestComplete, Ranorex and Quick Test Professional (QTP) for testing a web application. Also, author's described various test automation frameworks like keyword driven automation framework, data driven automation framework, library driven automation framework for testing a web based application. This research helped us in gathering background knowledge about automation frameworks and various automation tools in automating the cloud hosted application which is a web based application.
In reference [7], author's described automation tools for testing windows based applications. Here various automation scripts are developed using Graphical User Interface (GUI) tools like UFT (QFT), TestComplete, Ranorex etc. to check functionality of GUI in order to avoid manual effort in testing web based applications. Also some basic factors of these automated tools in order to perform GUI testing are explained in this paper. This research paper helped us in automating cloud hosted application which is a web based application using TestComplete tool and various features like object spying, insight object and error handling are performed for automating the cloud hosted application.

In reference [8], author's discussed various advantages for GUI testing automation. A data driven automation framework for GUI functional test is explained in this paper which helped us in developing automation framework for testing a cloud hosted application. Also test driven concept is introduced in this paper which is designed to develop communication between test cases and test execution engine. The concepts explained in this research paper helped us in developing a test data driven automation framework for automating the cloud hosted application which is a web based application.
3 BACKGROUND

This chapter gives readers, the background knowledge about cloud computing and Amazon Cloud Front which is a web service provider for Cloud hosted Application. Also gives users, the background knowledge about software testing and automation.

3.1 CLOUD COMPUTING

What is a cloud?
A cloud refers to a network or internet that can be accessed from remote location. It provides services over public or private network and can be accessible from anywhere.

What is Cloud Computing?
Cloud Computing is a networking paradigm which involves the concept of sharing computing resources, by providing dynamically scalable infrastructure capable of hosting end user applications, data and file storage rather than a local server or a personal computer. It helps in significant cost reduction, increased storage, and flexibility. It acts as an interface between service suppliers and group of multiple service customers.
The following definition of cloud computing has been developed by the U.S. National Institute of Standards and Technology (NIST):
Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models.

3.1.1 Essential Characteristics of a Cloud

- Self-Service of Resources on Demand: Customers are allowed to request and access their own computing resources on demand.
- Resource Sharing: Resource sharing which simplifies infrastructure planning is the promise of cloud computing.
- Broad Access to the networks: This feature offers the customers to use shared network resources over the internet or other private networks.
- Pooling of Resources: This allows the customers to access different pooled computing resources in remote data centres from various remote locations.
- Cloud scaling/elasticity: Resources and other cloud services can be scaled to larger or smaller according to the requirement of the customers.
- Measured Service: The cloud services provided to the customer are measured and billed properly in accordance to their usage.
- Reusability of IT Capabilities: The main idea behind cloud computing is reusability of IT capabilities.
- Scalable Infrastructure: Cloud computing offers highly scalable infrastructure capable of hosting end-customer applications and billed basing on their utilisation [9].

3.1.2 Cloud Service Models

Cloud providers offer services which are grouped into three categories: Software as a service (SaaS), Infrastructure as a service (IaaS), and Platform as a service (PaaS).
Software As A Service
SaaS is a one-to-many-computing model where Software is delivered to the customer either as a service on demand or it can also be a pay-as-you-go model hosted by the vendor or the internet service provider over a network, typically internet. The customers are given network-based access to single copy of an application through this software on demand model.

Various Benefits with SaaS
- Easy to maintain and provides great accessibility to wide range of services.
- Automatic updating of software is done and eliminates different patch management.
- Integration between various software is done using API (Application Programming Interface).

Infrastructure As A Service
Infrastructure as a service provides basic storage and computing capabilities as standardized services over network. Servers, storage systems (databases) and networking equipment are pooled and made available to users to manage heavy workloads. The customer would typically utilise his or her own software on infrastructure. In infrastructure as a service (IaaS) model storage, database management and computing capabilities are offered on demand to the customer.

Platform as a Service
Platform as a Service benefits the software development world rather than the customer. It provides a computing platform for quick and easy development of software web applications, development and deployment of tools without the necessity of purchasing and maintaining the requirements underneath it. Here, service is offered by encapsulating a layer of software or development environment, upon which various other higher layers of service can be built. Also, the customer is free to develop or build his own application, which runs on the service provider’s infrastructure. In order to meet the scalability requirements of applications, Platform as a Service (PaaS) providers offer a predefined combination of operating systems and application servers like Lamp platform etc.

PaaS can be characterised as follows:
- Provides software development services by integrating various web services.
- Services to develop, test, host, deploy and maintain applications is done in the same integrated development environment [10].

3.1.3 Understanding Public, Private and Hybrid Clouds
There is a necessity to develop and deploy applications on public, private and hybrid cloud. Each organisation needs to deploy a specific depending on their application functionality. Cloud integrators play a vital role in determining the right cloud path for each organisation.

Public Cloud
Public Cloud is owned and it is operated by third parties where delivering of superior economies of scale is made to the customers, as the infrastructure maintenance and costs are spread among the various users. So savings in cost is made possible due to the deployment of public cloud in an organization. Due to spread of infrastructure among mix of users, an attractive low cost is assigned to each individual user, giving pay-as-you-go model.
Advantages of Using Public Cloud

- Public cloud is larger than enterprises cloud.
- All customers share common infrastructure services with limited configuration, security protection and availability variances which will be managed and supported by the cloud provider.

Private Cloud

Private cloud is pertained for a single enterprise or organisation. Private cloud vendor mainly focuses on data security, and provides great control which mostly lacks in the public cloud. Private cloud is categorized as follows:

(I) On-Premise Private Cloud

On-premise private clouds are also called internal clouds. Internal clouds are hosted within own data centers existing in an organization. Internal clouds provides more security as it is hosted within one’s own data center in an organization but limited in the aspect of its size and scalability so that additional capital and operational costs needs to be incurred for the deployment of physical and storage resources. On premise private cloud is suitable for applications which require high security and better configuration of infrastructure.

(II) Externally Hosted Private Cloud

As a name itself indicates, it is hosted externally within a cloud provider and external cloud provides high security and scalability as compared to internal cloud. It is required by the organizations who don’t require deployment of the public cloud. In order to deploy externally hosted private cloud, no additional capital and operational costs are required for the deployment of physical and storage resources.

Hybrid Cloud

Hybrid cloud is a combination of public and private cloud models. Hybrid cloud model increases flexibility of computing. Using hybrid cloud model, service providers can utilize third party cloud providers in a partial or full manner thereby increasing its scalability. It has capability to provide on demand, externally provisioned scale. The capacity to enlarge private cloud by utilizing public cloud services can be used to manage or control any unpredictable surges in the work load of an organization [11].

3.1.4 Advantages and Challenges of Cloud Computing

The following are the benefits of cloud computing:

- **Reduced Cost:** Cloud computing is an approach to experience direct cost benefits and to transform a data centre to variable price environment. So overall cost can be reduced with cloud computing technology.
- **Increased Storage:** As cloud computing offers highly scalable infrastructure and high resource sharing, overall storage of physical and data storage resources can be increased with cloud computing.
- **Flexibility:** Flexibility to use physical and storage resources in an organisation can be increased due to cloud computing.

Challenges of Cloud Computing

- Data protection
- Data recovery and availability
Management capabilities
Regulatory and compliance restrictions

3.1.5 Amazon Web Services and Cloud Computing
Amazon has a quite long history of utilizing decentralized architecture (IT). Due to this arrangement made by the Amazon, development teams in an organization are able to access the physical resources like storage resources and computing resources on demand. Also due to decentralized architecture of Amazon, overall productivity in an organization and agility of the organization can be improved in a huge manner. Also Amazon has spent over millions of dollars in developing, deploying and managing high scale, reliable and more efficient IT infrastructure.

In order to provide benefits to other organizations as well, Amazon has launched Amazon Web Services (AWS) so that other organizations could get benefits from Amazon’s developed large scale, reliable and decentralized IT infrastructure. Today, amazon.com is serving millions of customers by deploying and running a global web platform.

Benefits of Using Amazon Web Services (AWS)
- Using Amazon Web Services (AWS), an organisation can requisite compute power, compute storage and other required services within minutes.
- Required development platforms and programming models can be requested by organisations whenever they required.
- It is cost-effective solution, because customers can pay for what they use with no advance expenses or without long-term commitments. So Amazon Web Services is a cost-effective solution to deliver applications in an organisation.

Differences That Distinguishes Amazon Web Services (AWS)
- Flexibility, because it enables organisations to use operating systems, development platform whenever they required.
- Cost-effective, because customers can pay for what they use without no longer commitments and without advance expenses.

3.1.6 Amazon Services Cloud Platform
Amazon Web Services is an effective cloud services platform, which offers compute power, compute storage, delivery of required content for a platform and all other required functionalities to an organization. So organizations can deploy applications and services in a cost-effective and way with more flexibility and reliability.

Amazon self-service means that an organization can address their internal plans in advance and they can chose services whenever they required externally.

3.1.7 Amazon Elastic Compute Cloud (Amazon EC2)
Amazon EC2 is a web service which provides cloud, a resizable computing capability. Here cloud can be supplied with resizable computing capacity. Amazon EC2 is developed and deployed to provide web scale computing adoption, flexible for system developers and system administrators in an organization. [12]

Amazon EC2 web service interface allows an organization to acquire and configure capability of cloud with minimal or no friction.

The features offered by Amazon EC2 are:

(I) Auto Scaling: Auto Scaling helps to maintain application availability and allows to scale Amazon EC2 capacity up and down according to the specified conditions by the customer. It is well
suitable for applications that have stable demand patterns or applications those have experience of hourly, daily or weekly variations in usage.

**Features of Auto Scaling**

- Maintains Amazon EC2 instances availability.
- Automatically scale Amazon EC2 fleet [13].

**(II) Elastic Load Balancing**

Elastic load balancing automatically distributes incoming traffic across various EC2 instances whenever a load balance is created and instances with load balancer in one or more available zones are registered.

**Features of Elastic Load Balancing**

- Elastic load balancing increases availability of an application.
- It provides usage of operating systems and instance types supported by Amazon EC2 [14].

**(III) Amazon Work Spaces**

Amazon work spaces is fully managed desktop computing service in the cloud. Usage of Amazon work spaces allows customer to easily provision any cloud based desktop and user access can be provided easily to documents, applications and resources they need from any supported devices like windows, Mac computer etc.

**(IV) Amazon Virtual Private Cloud**

Amazon virtual private cloud allows a customer to provision a logically isolated section of AWS cloud where you can launch AWS resources in a virtual network that user defines, so that complete control over virtual networking environment including selection of Internal Protocol (IP) address range, creation of subnets, super nets, routing table configuration and networking configurations can be made [15].

**3.2 Software Testing**

Software testing is the process of assessing quality of a software product to determine whether it matches with the existing requirements of the customer or not. It is done by eliminating bugs in the software and hence evaluating the features of the software product.

“Testing can only show the presence of errors not their absence”- Edsger W. Dijkstra (Dutch computer scientist).

Software testing is an activity that should be done throughout the whole development process of a product. Software testing is one of the “Verification and Validation”, or V&V, software practices practiced throughout the software development lifecycle.

**Verification:** Are we building the product right?

Verification is the process of evaluating a system or component of the system, to determine whether the products of a given development phase satisfies the conditions which are imposed at the start of that phase. Activities included in verification are testing and reviews.

**Validation:** Are we building the right product?

Validation is the process of evaluating a system or component of the system during or at the end of the development process to determine whether it satisfies specified requirements.

The purpose of software testing is to assess the complete quality level of the code. Its main aim is to make faults visible, so that the faults can be detected and they can be corrected in order to deliver product to the customers without errors in the code.
The goals of program testing are:

- To demonstrate to the customer and developer that the software meets its specified requirements and expectations
- To find out the situation in which the behaviour of the system is incorrect, not desirable or does not confirm to its specifications

During software development cycle, software testing of any product requires proper equipment set up and validation of the test cases.

**Test case**
A test case often referred to as test script is nothing but set of conditions which are determined by a software tester, in order to detect the functionality of an application or software system. A test case is a document containing test data, pre conditions, expected results and post conditions developed for a particular test scenario. A test case determines whether functionality of the software is working as expected or not. Using test case validation, a software tester can determine whether software program or system has passed or failed. Sometimes, many test cases are required to finalize the release of the software. A typical test case consists of following:

- **Test case id**: Id of test case
- **Test Scenario**: Detailed explanation of the scenario which is required to be tested
- **Test Case Description**: Description of the test case
- **Test Steps**: Steps to be executed in order
- **Prerequisite**: Holds pre required conditions to be followed before executing the test case
- **Test Data**: Input data provided by a tester at the time of executing the test cases in order to verify the functionality of an application. Test data is required for a test case to execute any kind of test. Preparing appropriate test data is necessary to execute any test case and preparing valid test data is part of the test setup. In general, testers call it as testbed preparation. Testbed contains all software and hardware requirements are set using predefined data values.
- **Expected Result**: Result expected in prior to the execution of test case
- **Test Parameters**: Input parameters required for the execution of the test case
- **Actual Result**: Actual result to be obtained after the execution of the test case
- **Environment Information**: Information regarding the type of environment required to execute the test case
- **Comments**: Any further information by the tester, if necessary [16]

**3.2.1 Basics of software testing**
Two basic classes of software testing needs to be discussed are black-box testing and white-box testing.

**(I) Black-Box Testing:**

![Diagram of Black-Box Testing](Figure 2: Black-Box Testing)
• Black-Box testing also called closed box or functional testing is a type of testing that ignores the internal functioning or mechanism of the software.
• It can be performed without having much knowledge on the internal working of the software.
• Client requirement specifications can be analysed using black-box testing.
• Black-Box testing can be done without having actual knowledge on the code, so programming knowledge is not required for performing black-box testing. So it can be easily performed.
• Black-Box testing is used to examine the functionality of a software or a web-based application.
• It can be performed by end users and developers.
• Black-Box testing also offers low granularity.
• Black-Box testing is widely applicable to business domains.

(II) White-Box Testing

Figure 3: White-Box Testing

• White-Box testing also known as Glass box/ transparent box/ non-functional/ structural box testing is a testing which is performed by taking into account the internal mechanism of a system or component.
• It is a method of testing in which internal structure is checked and code is verified keeping in mind the design specifications and main aim of white-box testing is to find common defects in the code.
• White-Box testing is applicable at the source code level.
• White-Box testing can be used for testing loops, if-else statements etc. in the code.
• High programming knowledge is required for a tester in order to perform black-box testing.
• Black-Box testing can be performed by testers or developers.
• White-Box testing offers high granularity.
• White-Box testing is widely applicable to all domains.

With black-box testing, the tester does not have access to the source code. The source code itself is treated as big black box to the tester. The tester knows only the information that can be put into the black box and black box sends the output back, according to the input given by the tester. Based on the requirements knowledge, the tester expects what black box sends out. According to that, tester defines test cases and makes sure what black box sends out and what is supposed to send out. Alternatively, white box testing focuses on the detailed internal structure of the code. So software tester or developer designs test cases with certain parameters keeping in mind the internal functioning.
of the software. In the language of V&V, white box testing is used for validation and black box testing is used for verification.

### 3.2.2 Types of Testing

There are several types of software testing that can be implemented on a large software system. Each test has a specification that explains tester the correct behavior, the test is examining and the incorrect behavior or faults can be identified by testing.

### 3.2.3 Development Testing

Development testing includes all testing activities that can be done by the testing team throughout the development of a system.

Phases in development testing are:

(I) **Unit Testing**

Unit testing, a defect testing process is a type of testing where individual components (program units or object classes) are tested at low level. Unit testing should focus on testing the functionality of objects or methods. Units can be individual functions or methods inside objects.

(II) **Automated Testing**

Unit testing can be automated whenever possible, so that testing can be done and software can be checked continuously without manual intervention. In automated unit testing, tester can make use of automation framework to write and run program tests. Generic test classes can be provided using unit testing frameworks, which allows a tester to create specific test cases.

The components of automated test cases are:

- A setup part, where initialisation of the system is done using test cases, namely the expected input values and output values.
- A call part, where the object or method to be tested is called.
- An assertion part, where the result of the call part is compared with the expected result. If assertion is true the test has been successful, if it is false then the test has been considered as failed.

(III) **Component Testing**

Component testing is a testing in which several individual hardware and software units are integrated and tested to create a composite system. Components developed by different team members or sub teams are integrated in component testing. Software components are composite components made up of several interacting objects. Functionality of these objects are accessed through the defined component interface. Therefore, Component testing should focus on testing the component interfaces.

(IV) **System Testing**

System testing also known as integration testing, is a testing in which some or various hardware and software components in a system are integrated and the system can be tested as a whole. It is a collective rather than individual process as component testing. System testing should focus on the component interactions. The main focus of the system testing is to determine the interaction between the software and hardware components by testing the system as a whole. System testing checks whether the components are compatible, interact correctly and transferring the right data or not.

(V) **Use-Case Testing**

Use cases are developed to identify interactions of a system and those use cases are basis of the system testing. Developed use cases contains various system components, so testing those use cases forces components to interact.
(VI) Regression Testing
Regression testing is selective retesting of a system or component, to verify that modifications have not caused any effects or have not broken previously working code. Regression testing works fine with automated testing compared to manual testing. In manual testing process, regression testing is complex and expensive. In automation testing process, regression testing is easy and straightforward. Rerunning of tests can be done with less complexity, whenever a change is made to the program.

3.2.4 Release Testing
Release testing is the process of testing, in which testing of entire software system is done by the separate software tester team before the final product is released into the market. Release testing is a black-box testing process, in which tests are derived only from the software specifications. The primary goal of release testing is to show that the system works as expected according to the specified requirements, so that it does not fail during its normal usage.
Release testing is the form of system testing. The main difference between release testing and system testing is:

- Release testing is carried out by the separate software testing team that has not been involved in the system development. The main aim of release testing is to verify that the system meets its requirements and also to validate it is good enough to fit for external usage.
- System testing is carried out by the development team in order to detect any bugs existing in the system.

(I) Performance Testing
Performance testing is a part of release testing, which involves testing the properties of system like performance and reliability. The main concept behind performance testing is, testing is done by steadily increasing load of the system continuously until the system performance becomes unacceptable.

(II) Stress Testing
Stress testing is the form of performance testing in which the system is intentionally overloaded in order to test the system's failure behavior.

3.2.5 User Testing
User testing also called customer is done by the users or potential users of the system in their own environments. User or customer testing is a stage in the testing process in which customers or potential users provide their inputs and advice on system testing. User testing is very essential in the software testing process, even the system and release testing has been carried out by the testers or developers.
The three basic types of user testing are:

(I) Alpha Testing
Alpha testing is done by the internal teams of the system within the developer’s site before the final software is released to the external customers. Alpha testing is performed without the involvement of the development teams.
Alpha testing can be done in two phases:
- During first phase of alpha testing, software testing can be done by in house developers in order to detect the bugs quickly.
- During second phase of alpha testing, software testing is performed by the QA team in the organisation for additional testing.
Alpha testing is performed as an internal form of acceptance testing, before the beta testing is performed.

(II) Beta Testing
A release of software is made available by the organization to one or more potential or beta users and allows them to install the package to raise any issues that they discover with the developers of the
software. Beta testing also called user testing is an important part of the software development cycle. It adds value to the software testing as opportunity for testing is provided to the real customers in order to provide inputs and advice for the functionality, design and usability of the product.

Advantage of running beta testing is:
Identification of unexpected errors can be made easily, because beta users use the software unexpectedly.

(III) Acceptance Testing
After system and release testing is performed, product is delivered to the customer and the customer runs black box tests depending on his/her expectation of the product functionality. Acceptance testing is a testing in which customer decides whether or not to accept the final product from the software developers.

Stages in the acceptance testing process:
- Define acceptance criteria
- Plan acceptance testing
- Derive acceptance tests
- Run acceptance tests
- Negotiate test results
- Accept/Reject product [17].

3.3 Need of Test Automation Framework
Test Automation Framework is required when huge number of developers work on different modules of the same application and also to avoid conflict of individual approach of developer towards an application testing.

Every organization needs to automate their testing efforts for the following reasons:

- Major focus of the developers will be on testing the software product rather than on the development of the testing environment.
- Unattended testing capability can be achieved by the software tester through automated testing framework.
- Using automated testing framework, software tester can test each module of the application independently without disturbing the entire application.
- Multiple or parallel testing can be done at once with the help of automation frame works.
- Automated testing can be exercised all the time of the day without any interruption.
- A basic set of software tools and services can be developed by testers for generating various automated test cases which are time-saving [18].
3.3.1 Components of Automation Framework

![Diagram of Automation Testing Framework]

- **Object Repository**: Object Repository abbreviated as OR consists of set of locator types associated with each web element.
- **Test Data**: Input data used to test a scenario and compares the expected values with the results obtained.
- **Configuration File/Constants/ Environment Settings**: The Configuration file stores the information regarding the application URL and its browser which remains constant throughout the framework.
- **Generics/ Program logics/ Readers**: These are the classes used to store the functions to have access across the entire framework.
- **Build tools and continuous integration**: This generates test reports, email notifications and logging information.

3.3.2 Types of Automation Framework

Test Automation frameworks differ from each other based on different factors to do automation like reusability, ease of maintenance etc. In this section the various types of Test Automation Frameworks widely employed in the organization are discussed along with their pros and cons.

**(I) Module Based Testing Framework**

Module based Testing Framework is based on OOPs concept i.e. Abstraction. This framework divides the application to be tested into number of isolated modules by creating individual test scripts for each of them. Combination of these individual test scripts produces a larger test script which represents more than one module. Abstraction layer separates these modules in such a way that any change performed in an individual section does not interrupt the module.
Figure 5: Module Based Testing Automation Framework

**Pros:** Easy to maintain, cost effective and scalable due to higher level of modularization. Change in one module does not affect the remaining modules of the application.

**Cons:** Individual implementation of test scripts for each module needs embedding of test data into larger test scripts. Manipulations must be made in test scripts to obtain different sets of test data.

(II) Library Architecture Testing Framework

The Library Architecture Testing Framework is based on Module Based Testing Framework. Instead of dividing the application under test into test scripts, the application is differentiated into various functions and common functions of different applications are grouped together, thereby a common library of all these functions can be created. Hence these libraries are called within the test scripts whenever required.

![Library Architecture Testing Framework](image)

**Pros:** This framework introduces higher level of modularization. It provides re-usability since common library functions can be called in test scripts. Highly cost efficient, easy to maintain and scalable.

**Cons:** Any change in test data requires changes in the entire test script.

(III) Data Driven Testing Framework

This framework is used to test the application for same functionality multiple times with different set of input data without embedding the test data into the test script. Here test data can be retained into external databases like property files, xml files, excel files, test files, CSV files, ODBC repositories etc. The data can be stored in “Key-Value” pairs and data can be accessed using the keys within the test scripts.
Pros: Due to this framework, number of scripts required to generate test scenarios are reduced, thereby minimizing the amount of code.

Cons: This process is complex to create different test data sources.

(IV) Keyword Driven Testing Framework

Other than differentiating the test data it inserts set of codes known as Keywords belonging to the test script into an external data file. These Keywords are self-guiding which are stored in table like structure that helps in determining the actions to be performed on the application under test.

Pros: Single Keyword can be used across multiple test scripts. This does not require knowledge of coding.

Cons: Highly complicated due to introduction of large number of keywords [19].

(V) Hybrid Testing Framework

Hybrid testing framework is a combination of Module based, Library architecture, Data-Driven, Keyword Driven testing frameworks. It provides benefits of all associated frameworks.
(VI) Behavior Driven Development Framework

Behavior Driven Development framework allows automation of functional validations to testers. It does not require the tester to be acquainted with programming language. With the help of various tools like Cucumber, Jbehave, automated testing is made easy.
4 METHODOLOGY

The automation of a cloud hosted application involves scrum methodology, an agile process for the successful delivery of the product with increased efficiency and quality. Various software development models are explained in this chapter.

4.1 Software Development Methodology

Software development methodology in software engineering also called software process, software development life cycle is intended for better planning and management activities within an organization. It is nothing but splitting of software development work into distinct phases of activities with the aim to improve planning and management activities of an organization.

4.1.1 Software Process

Software development methodologies are various software processes or software models that are being deployed for the development of a software product in order to meet the organization goals. A set of structured activities required to develop a system is nothing but software process. It involves various activities to be followed in order during the development of a system.

A software process involves the following in general:

- **Software Specification**: Includes definition of activities what the system should do.
- **Software Design and Implementation**: Defines the design of the software and implementation of the system.
- **Software Validation**: Validating whether the customer the developed software or not
- **Software Evolution**: Involves changing the entire system in response to the customer input and requirements.

During the development of a software product, activities involved in the software process are specifying a software model, designing a user interface etc. and also involves ordering of these activities.

Software process descriptions include the following:

- **Pre-and post-conditions**: Conditions to be given in prior and after the execution of the process.
- **Roles**: Defines roles or responsibilities of the people involved in the development of a process.
- **Products**: Defines the outcomes of the software process activities [20].

Two widely used software processes are:

(I) Plan-Driven Processes

Plan driven methodologies are traditional way to develop software process basing on system engineering and quality disciplines.

Plan driven process are processes where entire planning activities are planned in prior and growth of this process is measured against the plan. In general, verification and validation processes are included in this plan driven model.

(II) Agile Processes

In agile processes, planning can be done in incremental manner and the change of process is very easy according to the customer requirements. Agile process is the most practical approach in order to develop a software product.
Now a days, most practical software processes involves both elements of the plan driven and agile process. At the outset, there are no wrong or right software processes but depending on the time and requirements of the customer, one can decide whether the particular software process is suitable for the developing software product or not.

4.1.2 Software Process Models
Selecting a right software model is very important for an organization because entire development and testing processes in an organization are dependent on the right software model.

Widely used software models are:

(I) Waterfall Model
Water fall model is a plan driven model. It was the first driven software process model having distinct and separate phases of the product specification and the product development. Water fall model is a linear sequential flow and it is also referred to as linear sequential life cycle model. It is very simple to understand and use, in which progress appears like flowing steadily downwards like waterfall throughout the phases of software product implementation. In waterfall model, phases never overlap. So it is very easy to understand and implement. In the waterfall model, each phase in the implementation can be started only of the previous phase got completed. It does not define the software process to go back to the previous phase, if any change is required to be made in the previous phase according to the requirements. Waterfall model is the earliest approach that can be used for projects which are small and which does not require uncertain requirements. Review of the project takes place at the end of each phase by developers and they determine whether the project is in right path and whether or not to continue or discard the project [21].

Diagram of Waterfall Model

Drawbacks of Waterfall Model:
- The main drawback of waterfall model is, it is difficult to handle changes during ongoing of a software process
- In principle, a phase has to be complete before going to the next phase.
(II) Incremental Development Model
In incremental development model, entire requirement is divided into multiple builds. Various development cycles are introduced here making the life cycle as Multi Waterfall Cycle. Each cycle is implemented and divided into small and easily managed separate modules. Each module is passed through software process cycle i.e. software requirements, software design, software implementation and software testing. Each subsequent release of the module adds its functionality to the release of the previous modules. In incremental development model, each build or version get upgraded and integrated into the successive builds until the complete software product is ready to release. Thereby, the final software product got step by step.
Using incremental development model, cost of accommodating dynamic customer requirements is made easy. Because it is easy to analyze and document the work that has to be redone as compared to the waterfall model. Also, the implementation part can be made visible to the customers so that customers can get a clear idea regarding the extent of work that has been completed so far in an organization [22].

Diagram of Incremental Development Model

![Diagram of Incremental Development Model](image)

Figure 11: Incremental Development Model

Drawbacks of Incremental Development Model
- This process is invisible
- Whenever new increments are added, the system structure tends to be degraded and it is very complex to manage the system.
- Total cost and maintenance of the system is high.

(III) Reuse-Oriented Software Engineering
The reuse-oriented model also called reuse-oriented development in software engineering is a method of software development in which a program is developed by producing sequence of models. Each model is derived from the preceding one in accordance to the certain prototypes or rules. Reuse-oriented model can reduce the overall costs of the software development as compared to the existing manual models. As each phase of the process builds on the previously refined phase, it is time consuming to develop the reuse-oriented model. It can also reduce the errors or bugs existing in the system compared to the existing manual models [23].

Drawbacks of Reuse-Oriented Software Model
- Reuse-Oriented Software Model is always not reliable because it is always not easy to find the stock of reusable components.
- In case of unavailability of reused components, new program components must be designed unexpectedly.
4.1.3 Agile Methodology

Agile methodology is introduced in order to meet the rapid software development. The primary goal of agile methodology is to reduce overhead in the software process models and it aims to react more quickly to the changing requirements of the customer with less documentation and with less rework. Agile methods focus on the code rather than the system design, so software developers always try to keep code as simple as possible.

Principles of agile methodology

The main aim of agile methodology is to deliver software quickly with the working functionality and with less complexity in it.

The principles followed in agile methodology are:

- **Customer Involvement**: Customer interaction with system development is highly essential. Customers should get involved in the whole development process by giving feedback and inputs on the respective iterations.

- **Incremental Delivery**: The software is developed and delivered incrementally satisfying the customer requirements in each incremental delivery. Also, changes made in the next increment can be delivered to the customer in succeeding increments.

- **People Not Process**: In agile methodology, each developer in the development team is given a high priority, and a developer is independent to develop the software according to his/her own way of working.

- **Embrace Change**: System requirements change regularly according to the changing requirements of the customer, so system must be made in a way to accommodate the change in requirements.

- ** Maintain Simplicity**: Agile methodology focuses on maintaining the system more simple and development process with less complexity. In order to achieve this simplicity, developers work hard and smart in order to eliminate complexity in the developing software product of an organisation.

Agile methodology is applicable to software domains developing a small or medium scale product for sale. In order to achieve this, tightly integrated teams are deployed in agile methodology. In agile development methodology software specification, software design, software implementation and software testing are interleaved and the outputs of the development process are decided by keeping negotiation developed during the software development process in mind. In agile development methodology, planning can be done in incremental manner and the change of process is very easy according to the customer requirements. It is the most practical approach to implement in an organization [24].

4.1.4 Agile Methods

Widely used and implemented agile methods in software development process of an organization are:

- Extreme Programming (XP)

- Scrum Methodology

The description of these two agile methodologies is given below:

(1) Extreme Programming

Extreme programming is best known and most widely used method in agile methodology. Extreme programming takes an extreme approach to incremental development in extreme programming. In extreme programming:

- Newer versions can be built various times in a day

- Increments can be delivered to customers for every 2-3 weeks
All tests run successfully for every build in order to consider the build successful.

**Principles of Extreme Programming**

- **Incremental Planning:** Software requirements need to be recorded on story cards and developers determine which story card to be released based on the availability of time and according to their priority.
- **Small Releases:** At the outset, release with minimal requirements is made available to the customers. Later on, software with developed functionality is released to the customers in an incremental way.
- **Simple Design:** Agile methodology aims to make the system simple with less complexity. In order to satisfy this rule, developers work to make the system design in a simple way according to the requirements of the customer.
- **Test First Development:** Automated test framework is used in order to write test cases for every new functionality before that functionality is ready to be implemented in the system.
- **Refactoring:** All software developers in an organization try to refactor the code whenever possible with the addition of any new functionality if required, according to the requirements and specifications of the customer to keep the code simple and with less complexity.

**Extreme Programming Release Cycle**

![Extreme Programming Release Cycle](image)

**Extreme Programming practices**

Extreme programming in agile methodology follows certain practices in order to develop a product. They are:

- **Pair Programming:** In pair programming, developers always try to work in pairs in order to exchange their views and to support themselves in a friendly work environment of an organization.
- **Collective Ownership:** Pairs of developers work on all aspects of the system, so that participation is equal for all software developers in the code. There is equal right for each developer to change anything in the code.
- **Continuous Integration:** As soon as work is completed by a developer pair for the given task, work should be continuously integrated into the system and the integrated work should run continuously and pass all the unit tests.
• **Sustainable Pase**: More amount of overtime by a developer is not encouraged and acceptable by an organisation because it might affect the quality of code and productivity might be reduced.

• **On-site Customer**: End-user (customer) of the system should be available in the organisation for full time for the use of extreme programming team. In extreme programming process, customer plays a crucial role and he/she is also considered as a member of development team to give feedback and inputs for the work done by the development team in an organisation.

### 4.2 Scrum Methodology

The main aim of software project managers in any organization is to manage the project team so that the software can be delivered to the customer on time and within the planned budget for the project. Agile methodology requires a different approach for project management in order to adapt it to incremental development of the project and also to the particular strengths of the agile methods.

**Scrum**:
Scrum approach in agile methodology is a common methodology. Scrum methodology concentrates on iterative development of a software product rather than on the particular agile methodology practices [25].

#### 4.2.1 Phases in Scrum

![Figure 13: Phases in Scrum Methodology](image)

Scrum methodology is divided into three phases as below:

- **Initial Phase**: This is the starting phase of the scrum methodology. Initial phase is nothing but outline planning of the software product, where general objective of the project is defined here and software architecture is designed, in order to start the software product of an organisation with specified requirements.

- **Sprint Cycles**: Initial phase of the scrum methodology is followed by the sprint cycles of fixed length in size, where each sprint cycle represents development of the increments of a system.

- **Project Closure**: This phase in scrum methodology is like a project wrap up, where project is completed with required final documentation which includes helper manuals, product manuals, and user manuals, etc. and assessing of the lessons learned from the project completed is made here.
4.2.2 Scrum Process
The process inside scrum methodology is classified as follows:

**Sprint Cycle:**
Sprint cycle is of fixed length, usually 2-4 weeks. Sprints correspond to the development of a release of a system in extreme programming process of the agile methodology. For the initial phase of scrum methodology, the starting point of product planning is product backlog which is nothing but summary or list of work needs to be done on the project.

After initial phase of scrum process, selection phase involves selection of the project team members who work with the customer to select the specific functionality and features of the software according to the changing requirements of the customer or end user. These selected features and functionalities by team members are decided to be implemented during the sprint cycle of the scrum methodology. Once initial phase and selection phase is completed, team members agree with the respective requirements and specific functionalities specified by customers. Then the team organize themselves to develop the software. At this stage, the team is isolated from the organization and customers and communications is channeled only through the so called scrum master.

The role of the scrum master is to protect the development team from any external distractions. After the completion of work by the development team within a sprint length, work done by the development team is reviewed, tested and presented to the stakeholders. The next sprint length continues with the same procedure.

4.2.3 Team Work in Scrum Methodology
Team work is an important aspect in scrum methodology. Scrum master is a facilitator who arranges daily meetings which is known as scrum meetings, where discussion is done among team members about the completed work and scrum master tracks all these records, portion of the work completed and backlog of work which needs to be done. Scrum master measures progress in the team according to the completed and backlog work. Scrum master informs the progress in the team to the project manager and manages outside of the team.

The whole team attends short daily meetings to share the details about the completed and backlog work and describes the progress since last meeting so that everyone in the team knows what is going on and plans according to their progress to come out with better benefits.

4.2.4 Advantages of Scrum Methodology
- The whole product is broken down into set of manageable chunks.
- Unstable requirements can be recorded since they don't go into progress and can be easily identified and recorded by scrum team.
- Whole team knows everything happening inside the team, thereby team communication can be improved.
- Customers can get benefit due to team work maintained by developers and can give feedback to the functionality developed by team members.
- Trust between customers and developers is achieved due to implementation of scrum methodology.
5 EXPERIMENT

5.1 Test Appliance Library

Two crucial applications of cloud hosted application are: Test Appliance Library and Test Data Library. These applications are designed to increase the productivity and business efficiency. Test Appliance Library centralize, manage and correlate test equipment and test data by enabling an operator to participate in test operations and allows taking immediate actions and rapidly fixing any issues if existing in the field.

Today, the main challenge for many test managers is to deal with the constant updates of software or file downloads and to manage testing equipment like testing devices or platforms. Also test managers should keep track of technicians in the field involving network construction, maintenance and troubleshooting ensuring that they are properly equipped. In order to deal with these challenges, responsibilities rely on individuals to take appropriate actions to keep the process running.

Test appliance library functionality centralizes the management of all testing instruments of company like testing device and its associated modules etc. It acts and functions as a repository for software loads, licenses and testing device or platform profiles which contains information like status of the device, owner of the device and the device location. Overall view of test appliance library functionality is shown in Figure 16. Test appliance library functionality ensures that all software configurations are up-to-date and are consistent. Also with a periodic connection to the hosted server, an automated process updates all the necessary testing device information and updates the test data by capturing the stored test data in the server and downloads of software or files is done automatically. Also, Options which are nothing but instances of modules can be downloaded automatically.
All these processes can be done without interfering technicians test routine. A technician nothing but owner of that respective test equipment gets an information in the form of text message or email that appliance is going to be modified. So technician can simply plug and go in one simple action, connecting his testing device to server and takes over. Test managers can get control over all issues using via testing device profiles. Testing device profiles are created with all customized configurations and are controlled by a test manager. Test appliance library profiles enables test managers to standardize team’s testing instruments, ensuring all test appliance is up-to-date according to their specifications.

![Figure 15: Applications of Cloud Hosted Application](image)

Test Appliance Library plays a crucial role in increasing the productivity of the cloud application. Test appliance library functionality provides the user, an opportunity to manage test appliance by presenting the user with an array of critical information which helps to manage the Appliance Library. For the very first time, manager will have all this information and later it is provided to the recommended users. This array list contains information about testing devices and modules, such as versions of the software, calibration date of the software, testing device status as well as ownership details of the testing device.

The two main features of test appliance library functionality are:

(I) **Centralize and Automate Inventory Management**
- Ensures whether testing devices/modules are allocated properly and can be identified across several departments.
- Acts as a repository for software downloads, licenses, testing device profiles.
- Configuration settings in respective testing device profiles are validated by testing device and the module.
- Consistent configuration settings are ensured by testing device profiles.

(II) **Generate Records**
- Inventory Records
- License renewal records
- Calibration due records
Example for Functionality of Test Appliance Library

Figure 17 gives an example for functionality of Test Appliance library.

5.1.1 Benefits of Test Appliance Library

- Ensures all the testing equipment is up-to-date.
- Current status of the test tools including testing device and the respective modules can be analyzed easily.

5.2 Cloud Platform or Testing device

The testing device can be connected to the Cloud application to download required equipment at anytime, anywhere and from any place. After connecting to the internet, the options of different applications are available, shown on the testing device through which it can be connected to the application.

The company uses the testing device for testing functionality of cloud hosted application known as test appliance. Since the company operates the testing device which is shown in the figure, it displays Cloud server. If the customer operates the same testing device from any remote location Cloud client is displayed.

Cloud client is available on the testing device that is used to connect to the Cloud application that is
launched on the Cloud application. Once that option is selected, the testing device enables the page of authentication that contains the URL address to which the testing device is to be connected and the corresponding password box of the user who wish to connect to the Cloud application.

After the necessary authentication details are filled correctly, the connect option must be enabled by the user. Then it takes a few seconds for the testing device getting connected to the Cloud application. Once the testing device gets connected to the Cloud application, a page with success message with summary displaying “Connected to the server” will be seen on the testing device. After successful connection is established with the cloud, the summary of the Cloud client application on the testing device is shown which contains Connection to the server, Inventory data exchange, Test data uploads, File uploads, Software downloads, File downloads and License downloads along with their status such as successful(if prescribed) and none(if not prescribed) are shown. After the synchronization is completed, OK button can be pressed to continue with further available properties.

Various options on the testing device on the right of the panel are available: Configuration, File Transfer, Logs, Options Manager, Testing device Info, Software downloads, Test data uploads are available.

Configuration: Synchronization of the testing device with the Cloud application comes under this option. The types of modules, type of connection to the cloud (either using Wi-Fi or LAN cables) and other pre settings information is displayed.

Options Manager: It displays the information of the product, i.e. what cables, fibers have to be tested. Type of options: static or floating, Status: whether yes or no, Options: Name of the options displayed, Description: It displays the test data parameters such as optical link speed, bit rate of tested optical cables.

To activate the static or floating options a key has to be assigned to the option which is brought from the cloud with which the connection is established to the testing device. Once the required options are enabled, a green tick appears besides the type of option and an update message will be displayed saying the option is assigned to the particular testing device. To activate the option the module must be restarted.

Testing device Info: This option gives information about the modules and other software packages available on the testing device. Also it gives information about the number and types of test records and types of testing that takes place between the testing devices from the operator and the customer.

Test data Upload: This option gives information about the number of tests that have been uploaded to the cloud through the testing device. This is a part of the Test Record Analyzer.

### 5.3  Automation of Test Appliance Library

This thesis involves automation of test appliance library feature of Cloud application. The major steps Involve in automating test appliance library are:

- Adding test appliance like testing devices or platforms, modules and other devices from third party manufactures
- Assigning operator nothing but owner for the appliance added
- Creating appliance group for software download i.e. adding testing devices and modules.
• Creating appliance profile i.e. selecting appliance groups added above, in order to verify test appliance library management functionality in large number.

All the above mentioned test cases are written using JScript and executed using a tool called Test Complete.

The amount of automation done, their results and effect of automation on the Cloud application Cloud application is further explained in the results section.


6 **RESULTS**

This thesis focuses on automating a feature of a cloud hosted application called test appliance library.

The test cases are written for each sub feature to automate the functionality. Nearly 80% of the test cases can be automated, out of which 30% automation is done by executing the test cases using a tool called Test Complete 11.0. Verification and Validation is done by setting various input parameters and calling these parameters from excel files, .csv files, .txt files.

Balance part of automation is planned in incremental way through upcoming releases. Certain areas cannot be automated due to the need of mandatory manual intervention.

6.1 **Common Results between both theses:**

The main aim of automation is to reduce the manual effort which directly improves the release cycles of the application, reduction in number of defects. As part of common results the metrics captured are 1. How much effort reduction in Manual Testing achieved due to automation release by release and 2. How much defects leakage to Production is arrested due to automation testing?

6.1.1 **How automation is reducing manual effort of testing**

For each release cycle certain amount of Automation is planned and executed. The functionality that is already in production is targeted for automation and the list of Test Cases automated is described in previous section.

Figure 19 represents the manual effort saved by automating Test Cases for each release. The percentage of automation and corresponding effort savings are captured.
Table 2: Table Showing Automation % & Corresponding Reduction in Manual Effort %

<table>
<thead>
<tr>
<th>Release number</th>
<th>Automation %</th>
<th>Effort Reduction %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>23</td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>25</td>
</tr>
</tbody>
</table>

The main aim of automation is to reduce manual effort and automate Testing as much as possible. The Automation strategy is dependent on priority of functionality and also on quantum of regression Testing to be done for each release. Data on automation for last six releases were collected and the effort savings achieved through each release and there by overall savings achieved for last six releases is depicted in the above Table 2.

While the current stage of automation is at ~26%, in the above table the last value related to 30% automation and corresponding effort reduction is an extrapolation of values based on current trend.

For example in release-1 at the stage of 5% of automation the corresponding reduction in manual effort is 8%. Similarly in release-2 at the cumulative percentage of automation at 10% and cumulative percentage of effort reduction observed is 12%.

Likewise in release-6, when the cumulative automation reaches 30% then the effort reduction cumulative is calculated to be at 25%.

The new functionality that is gone live in a given release is targeted for automation in next release that is how each new release is having incremental automation.

6.1.2 How Automation is Reducing Leakage of Defects to Production

![Figure 19: Graph Showing Analysis of Automation and Defects Reduction](image)

Automation not only helps in effort reduction but also in reducing defects getting leaked to production. One of the important metrics that is captured in any Testing Project is how many
defects are leaked to Production which could have been caught in different stages of Testing cycles.

Due to automation the critical aspect of Defects getting Leaked to Production is addressed to the extent of Automation done. Also Due to increase in automation for each release cycle the defects getting leaked to production are in downward trend.

Figure 20 is a pictorial representation of how the defects that are leaking into Production are getting reduced due to continuous automation that is being achieved. This data is collected for last six releases. Same data is represented in Tabular form below.

For example for release-1, at the stage of 5% of automation, the corresponding reduction in defects leakage to production is at 10%.

Similarly for Release-2, at the stage of 10% of cumulative automation, the corresponding cumulative reduction in defects leakage to production is at 12%.

Likewise by sixth release at the stage of cumulative automation is 30%, (right now automation is at ~26%) the corresponding value cumulative reduction in defects leakage to production is calculated to be at 32%.

<table>
<thead>
<tr>
<th>Release Number</th>
<th>Automation %</th>
<th>Defects Reduction %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>22</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>32</td>
</tr>
</tbody>
</table>

Table 3: Table showing Automation % & Corresponding Reduction in Defects

6.2 Individual Results: Results for Automation of Test Appliance Library

As part of thesis work, various test cases mentioned in the previous section 5.3 are written and automated by defining automation framework. Automating test compliance library is a crucial task as it involves the major part i.e. updating the test equipment and keeping track of software downloads into the platform.

Automation of Test Appliance Library is divided in to four different parts:

- Automation of software download test cases
- Automation of operator test cases
- Automation of test appliance test cases
- Automation of file download test cases

As part of thesis works following tasks are performed for Automation of Test cases in the areas mentioned above.

- Defining automation strategy
- Identifying what test cases to be automated
• Develop automation scripts
• Execution of these Test cases successfully by capturing expected results.

6.2.1 Automation Status of Software Download Test Cases

![Automation Status of Software Download Test Cases for Ongoing Phase](image)

Figure 20: Automation Status of Test Data Library

Figure 21 is a pictorial representation of current status of automation for ongoing release describing how many test cases are targeted to be automated in software download of Test appliance library and how many are automated so far, how many are yet to be automated and how many can’t be automated due to dynamic dependencies.

Same values are shown in Tabular format below in absolute numbers.

<table>
<thead>
<tr>
<th>Total Test Case count for the current Release</th>
<th>Test Cases Automated until now</th>
<th>Test Cases Pending to be automated</th>
<th>Test Cases cannot be automated</th>
</tr>
</thead>
<tbody>
<tr>
<td>72</td>
<td>42</td>
<td>18</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 4: Current Status of Automation on Software Download of Test Appliance Library

Table 4 provides current status of automation on software download of test appliance library

- There are total 72 test cases for automation in software download. This is total TC count to be automated for this release.
- In 72 test cases, so far 42 test cases are automated, 18 test cases are yet to be automated and 12 test cases can’t be automated due to dynamic dependencies.
6.2.2 Automation Status of Operator Test Cases

Figure 21: Automation Status of Report Analyzer

Figure 22 shows pictorial representation of current status of Automation for ongoing release describing how many Test Cases are Targeted to be automated in operator scenario of test appliance library and how many are automated so far, how many are yet to be automated and how many can’t be automated due to dynamic dependencies.

Same values are shown in Tabular format below in absolute numbers.

<table>
<thead>
<tr>
<th>Total Test Case count for the current Release</th>
<th>Test Cases Automated until now</th>
<th>Test Cases Pending to be automated</th>
<th>Test cases cannot be automated</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5: Current Status of Automation on Operator Test Cases

Table 5 provides current status of automation on operator test cases

- There are total 4 test cases that are planned for automation in operator scenario of test appliance library for the current release.
- In 4 test cases, 4 are automated and there are no pending test cases for this scenario of test appliance library.
6.2.3 Automation Status of Test Appliance Test Cases

![Diagram of Test Case Automation Status]

Figure 22: Automation Status of Test Appliance Test Cases for Ongoing Phase

Figure 22 shows pictorial representation of current status of Automation for ongoing release describing how many Test Cases are Targeted to be automated in test appliance scenario of test appliance library and how many are automated so far, how many are yet to be automated and how many can’t be automated due to dynamic dependencies.

Same values are shown in Tabular format below in absolute numbers.

<table>
<thead>
<tr>
<th>Total Test Case count for the current Release</th>
<th>Test Cases Automated until now</th>
<th>Test Cases Pending to be automated</th>
<th>Test cases cannot be automated</th>
</tr>
</thead>
<tbody>
<tr>
<td>88</td>
<td>50</td>
<td>20</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 6: Current Status of Automation on Operator Test Cases

Table 6 provides current status of automation on operator test cases

- There are total 88 test cases that are planned for automation in test appliance scenario of test appliance library for the current release.
- In 88 test cases, so far 50 test cases are automated, 20 test cases are yet to be automated and 10 test cases can’t be automated due to dynamic dependencies.
6.2.4 Automation Status of File Download Test Cases

Figure 23: Automation Status of File Download Test Cases

Figure 24 shows pictorial representation of current status of Automation for ongoing release describing how many Test Cases are Targeted to be automated in file download scenario of test appliance library and how many are automated so far, how many are yet to be automated and how many can’t be automated due to dynamic dependencies.

Same values are shown in Tabular format below in absolute numbers.

<table>
<thead>
<tr>
<th>Total Test Case count for the current Release</th>
<th>Test Cases Automated until now</th>
<th>Test Cases Pending to be automated</th>
<th>Test cases cannot be automated</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>4</td>
<td>0</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 7: Current Status of Automation of File Download Test Cases

Table 7 provides current status of automation on file download test cases

- There are total 88 test cases that are planned for automation in file download scenario of test appliance library for the current release.
- In 12 test cases, so far 4 test cases are automated and 10 test cases cannot be automated due to dynamic dependencies.

In all the above mentioned scenarios of test appliance library, whatever test cases that are pending, those can be automated with the support of third party teams, by the company’s permanent employees.
7 CONCLUSION AND FUTURE WORK

7.1 Conclusion

The following conclusions are derived from the research work done:

- Detailed study of various automation tools like TestComplete for automating web based application is made (Refer to section 2).
- Analysis of various automation test cases and execution of those test cases using TestComplete automation tool is made using programming language known as JScript (Refer to section 2).
- Detailed study on the concepts of cloud computing is made (Refer to section 3.1).
- Detailed study on Amazon Web Services (AWS) which is a service vendor for cloud hosted application is made (Refer to section 3.1).
- Detailed study on the concepts of software testing and various types of software testing is obtained (Refer to section 3.2).
- Detailed study on the design of proper automation framework for software testing cloud hosted application is made (Refer to section3.3).
- Detailed study on software development methodologies is made (Refer to section 4.1).
- Company follows scrum methodology which is an agile approach for its product delivery. Detailed study on scrum methodology is made (Refer to section 4.1 and 4.2).
- Detailed study on test appliance library, a functionality of cloud hosted application is made (Refer to section 5.1).
- Detailed study on testing device for testing the functionality of test appliance library is made (Refer to section 5.2).
- Automation of test appliance library functionality of cloud hosted application is made by writing test cases in JScript programming language (Refer to section 5.3).
- Detailed study on regression test cases is made and the status of test cases automated, test cases pending and test cases cannot be automated are explained clearly with tables and graphs (Refer to section 6).
- At each stage of automation, the effort reduction and the defect reduction percentages were clearly explained numerically and graphically for every consecutive release showing the benefits of automation through manual effort reduction and quality improvement (Refer to section 6.1).
- Detailed analysis on the automation of Test Appliance Library, a functionality of cloud hosted application is obtained using the software testing tool Test Complete tool and the automated test cases are observed and graphically analysed (Refer to section 6.2).
- Graphs, tables and numerical explanation of the test cases automated are drawn and analysed for test appliance library functionality of cloud hosted application (Refer to section 6.2).
- Designed a test data-driven automation framework with necessary features for automating the test cases of test appliance library functionality of cloud hosted application (Refer to section 8).
- Evaluation of the TestComplete tool is made based on management goals, testing requirements and the technology (Refer to section 8).
To summarize the numerical results shown above, the overall Automation targeted for this cloud hosted application is 81% in ongoing phase. In this release 33.3% of automation is achieved in automating the feature called "Test Appliance Library" of cloud hosted application (Refer to section 8).

In summary the automation is yielding the expected results and whatever effort that is getting saved out of this is being deployed for quality assurance, integration testing and further application development by the production teams. Also Automation not only helps in reducing defects but also guarantees consistent way of testing thereby increasing quality and client satisfaction. By using the Frame work the company’s clients are getting benefited with speed to market.

7.2 Future Work

Future scope of work includes:

- Continue to write automation scripts until targeted Percentage is achieved.
- Dynamically generating Log reports that show the required analysis if any defects found
- Roll-Back the features of the error detected application.
8 ANSWERING RESEARCH QUESTIONS

1. How to identify and analyse a right automation test data driven framework for automating cloud hosted application?

In automating the cloud hosted application, a right automation data framework is very essential for successful automation. There are various types of automation framework where company has selected test data driven automation framework, in which input values can be driven through Excel files and .CSV files. Test data driven framework allows a developer to combine the test script logic and data, storing it in an external database. This database can be Excel files, XML files, CSV files etc. A right automation test data driven framework is selected on the basis of high scalability, ease of scripting, modularity, reusability and low maintenance requirements. Other advantages considered in the usage of test data driven framework is more coverage, less manual intervention and ease of reporting. This framework has been using by company because it reduces number of scripts by covering all test scenarios. So, team requires less code to cover all the test scenarios.

Based on type of test data the configurable parameters can be changed dynamically in automation script which is the unique advantage in test data driven framework, hence it is selected as the automation framework for the cloud hosted application (Refer to related works in section:2)

2. What level of release cycles can be reduced to maximize the efficiency of automated testing of cloud hosted application?

In automation of a cloud hosted application, there are 658 test cases in total, where 176 test cases are from test appliance library. These total test cases are distributed across six release cycles. Out of 658 test cases, 74% can be automated and rest 26% cannot be automated due to necessary manual intervention (till 6th release). Considering all the automated, pending and non-automated test cases, the automation of cloud application is completed and achieved so far is 81.7% for ongoing phase, where 33.3% of automation for release 6 is achieved in ”Test Appliance Library functionality” of cloud hosted application.

![Figure 24: Distribution of Automated Test Cases over Total Test Cases Across Release Cycles](image-url)
Same values are shown in Table 8. Total number of test cases and total number of automated test cases for each release cycle is represented in absolute numbers in table 8.

<table>
<thead>
<tr>
<th>Release Number</th>
<th>Total Number of Test Cases</th>
<th>Number of Automated Test Cases</th>
<th>Automation Achieved in %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>45</td>
<td>26</td>
<td>57.8</td>
</tr>
<tr>
<td>2</td>
<td>58</td>
<td>35</td>
<td>60.34</td>
</tr>
<tr>
<td>3</td>
<td>76</td>
<td>50</td>
<td>65.8</td>
</tr>
<tr>
<td>4</td>
<td>83</td>
<td>59</td>
<td>71.1</td>
</tr>
<tr>
<td>5</td>
<td>96</td>
<td>73</td>
<td>76.1</td>
</tr>
<tr>
<td>6</td>
<td>300</td>
<td>245</td>
<td>81.7</td>
</tr>
</tbody>
</table>

Table 8: Table Showing Distribution of Automated Test Cases over Total Test Cases across Release Cycles

Automation of cloud hosted application achieved in six release cycles is represented in Figure 26. Also, it is observed that there is 24% change in percentage of automation achieved from Release1 to Release6. Considering all these results, the level of release cycles reduced in this phase is 24% compared to automation achieved in last six release cycles.

3. How to evaluate the tool for the automation testing of cloud hosted application?

This research question can be answered by evaluating the performance of TestComplete tool based on features like licensing cost, application it supports, language it supports etc. The tool considered for automation of cloud hosted application is TestComplete. TestComplete is an automated testing platform developed by Smart Bear Software for testing desktop, mobile and web applications. The tool is evaluated on the following criteria:

- **Licensing Cost**: TestComplete offers low cost compared to other automation tools like Selenium, Jbehave etc.
- **Application Support:** TestComplete can be installed very easily once it is purchased and tests can be created against any application.
- **Language Support:** TestComplete supports scripting languages like VBScript, JScript, Delphi Script, C++Script and C#Script.
- **Operating Systems:** TestComplete supports operating systems like Windows Vista, Windows 7, Windows 2008 and later operating systems.
- **Application Support:** TestComplete supports web based and desktop applications and it works very well for database applications.
- **Usage:** TestComplete supports well for both 32 bit and 64 bit operating systems.

Thus below are the main reasons to consider TestComplete as company's Automation tool:

- Ext JS Support which is a pure JavaScript application framework for building cross platform web applications.
- TestComplete is Cost effective compared to other automation tools like Selenium and Quick Test Professional.
- TestComplete supports web as well as desktop application testing (In future platform applications which are built on .Net framework are also well supported by TestComplete) (refer to related work in section 2).
REFERENCES


