Design Report for

Continues Integration Pipeline Implementation for Tech11Software

Submitted By

Aswin G Sugunan *(13)*

Jefin Jacob *(4)*

Nitin Suresh *(23)*

Vishnu Bose *(39)*

7th Semester

Under the guidance of

Mrs. Greeshma N Gopal

JULY 2016

Department of Computer Science and Engineering
College of Engineering,Cherthala
Pallippuram P O,Alappuzha-688541
Phone: 0478 2553416, Fax: 0478 2552714
http://www.cectl.ac.in

Contents

List of Figures

Chapter 1

INTRODUCTION

Software development, as we know it today, is a demanding area of business with its fast-changing customer requirements, pressures of an ever shorter timeto- market, and unpredictability of market. With the shift towards modern continuous deployment pipelines, releasing new software versions early and often has become a concrete option also for an ever growing number of practitioners.

Continuous delivery is a software development practice where new features are made available to end users as soon as they have been implemented and tested. In such a setting, a key technical piece of infrastructure is the development pipeline that consists of various tools and databases, where features ow from development to deployment and then further to use.

1.1 Purpose

The purpose of the design document is to describe the behavior of the proposed CI framework system. Requirements Specification defines and describes the operations, interfaces, performance, and quality assurance requirements of the proposed system. The document describes the design constraints that are to be considered when the system is to be designed, and other factors necessary to provide a complete. The Design report analyse the SRS report and converted to implementation form by means od necessary diagrams (DFD, sequence diagram,structure Diagram, ER diagrams) and also mention the system modules and briefly their operations

1.2 Solution Scope

The objective of the project is to put in place a Continues Integration framework for product development activities of Tech11 Software. This would enable the Tech11 team to rapidly bring a product change or feature to production gaining market advantage. This activities of this project will involve accessing different CI integration approaches and solutions available,identify the feasibility of those solution by doing POCs and demos, fine tune the final solution and set up the CI infrastructures, educate the developers on CI culture.

Here the solution describes the difference methods which can solve the problems caused by following the traditional methods, which are

- Decrease risk by uncovering deployment issues earlier,
- increase flexibility by giving the organization the option to release at any point with minimal added cost or risk,
- Less change in code loss, The modification can be re modified in the fraction of the time by any team members.

Chapter 2

OVERALL DESCRIPTION

2.1 Product Perspective

Continuous Integration is based on continuous performance of acts of integration of source code, testing, building and deployment in response to each change to the source code of the project submitted by the developer and for the use of tools for support of the development and testing by compliance with the established procedure automatically. The proposed system directs the user (developers) for time and cost effective production and solves majority of the Integration hell problem.

Integration Hell refers to the point in production when members on a delivery team integrate their individual code. In traditional software development environments, this integration process is rarely smooth and seamless, instead resulting in hours or perhaps days of fixing the code so that it can finally integrate. Continuous Integration (CI) aims to avoid this completely by enabling and encouraging team members to integrate frequently (e.g., hourly, or at least daily).

2.1.1 Proposed System

The Proposed system is to help software developers to ensure new features are made available as soon as the program has been implemented and tested. This product also helps in reducing the time needed to develop a software and also acts a guideline for future software developments.

2.2 solution Function

Continuous integration the practice of frequently integrating one's new or changed code with the existing code repository should occur frequently enough that no intervening window remains between commit and build, and such that no errors can arise without developers noticing them and correcting them immediately.[9] Normal practice is to trigger these builds by every commit to a repository, rather than a periodically scheduled build. The practicalities of doing this in a multi-developer environment of rapid commits are such that it is usual to trigger a short time after each commit, then to start a build when either this timer expires, or after a rather longer interval since the last build. Many automated tools offer this scheduling automatically.

- Version Control

  This practice advocates the use of a revision control system for the project's source code. All artefacts required to build the project should be placed in the repository. In this practice and in the revision control community, the convention is that the system should be buildable from a fresh checkout and not require additional dependencies. Extreme Programming mentions that where branching is supported by tools, its use should be minimised.[9] Instead, it is preferred for changes to be integrated rather than for multiple versions of the software to be maintained simultaneously.

- Artifact Manager

  While many developers have adopted Maven as a build tool, most have yet to understand the importance of maintaining a repository manager both to proxy remote repositories and to manage and distribute software artifacts. A repository stores two types of artifacts: releases and snapshots. Release repositories are for stable, static release artifacts and snapshot repositories are frequently updated repositories that store binary software artifacts from projects under constant development. While it is possible to create a repository which serves both release and snapshot artifacts, repositories are usually segmented into release or snapshot repositories serving different consumers and maintaining different standards and procedures for deploying artifacts. Much like the difference between a production network and a staging network, a release repository is considered a production network and a snapshot repository is more like a development or a testing network. While there is a higher level of procedure and ceremony associated with deploying to a release repository, snapshot artifacts can be deployed and changed frequently without regard for stability and repeatability concerns.

- Continuous Integration Handler

  While there are many tools, I will focus on one of the most popular, Jenkins CI. This is one of the more popular (open source) tools available. Jenkins CI (the continuation of a product formerly called Hudson) allows continuous integration builds in the following ways:

  - It integrates with popular build tools (ant, maven, make) so that it can run the appropriate build scripts to compile, test and package within an environment that closely matches what will be the production environment
  - It integrates with version control tools, including Subversion, so that different projects can be set up depending on projection location within the trunk.
  - It can be configured to trigger builds automatically by time and/or changeset. (i.e., if a new changeset is detected in the Subversion repository for the project, a new build is triggered.)
  - It reports on build status. If the build is broken, it can be configured to alert individuals by email.

  Jenkins is an automation engine with an unparalleled plugin ecosystem to support all of your favorite tools in your delivery pipelines, whether your goal is continuous integration, automated testing, or continuous delivery.

- Test Automator

  In software testing, test automation is the use of special software (separate from the software being tested) to control the execution of tests and the comparison of actual outcomes with predicted outcomes.[1] Test automation can automate some repetitive but necessary tasks in a formalized testing process already in place, or perform additional testing that would be difficult to do manually. Test automation is critical for continuous delivery and continuous testing. There are many approaches to test automation, however below are the general approaches used widely:

  - Graphical user interface testing. A testing framework that generates user interface events such as keystrokes and mouse clicks, and observes the changes that result in the user interface, to validate that the observable behavior of the program is correct.
  - API driven testing. A testing framework that uses a programming interface to the application to validate the behaviour under test. Typically API driven testing bypasses application user interface altogether. It can also be testing public (usually) interfaces to classes, modules or libraries are tested with a variety of input arguments to validate that the results that are returned are correct.

- Continouos Deployment

  Continuous deployment is the next step of continuous delivery: Every change that passes the automated tests is deployed to production automatically. Continuous deployment should be the goal of most companies that are not constrained by regulatory or other requirements.

2.3 Operating Environment

The system is expected to be operated in Linux as well as in windows with the support of respective JRE (Java Runtime Environment). This system based project is completely platform independent. The most important requirement is the internet connection. This tool is coded using JDK 1.6.

Chapter 3

PROJECT REQUIREMENTS

This system based software can be used in any operating system such as Microsoft Windows, Linux or any kind of user application interface, since it is platform independent.

3.1 User Interface

Interface hardware shall be encapsulated in a set of classes that isolate hardware specifications from the rest of the software. In particular, interfaces for specific hardware boards shall be implemented as derived classes of an abstract class.

3.2 Hardware Requirements

//jefine fill cheytho

Processor	:	Min 2GHZ
RAM	:	Min 2GB
Hard disk	:	Min 200GB

3.3 Network Requirements

// jefinee fill cheytho

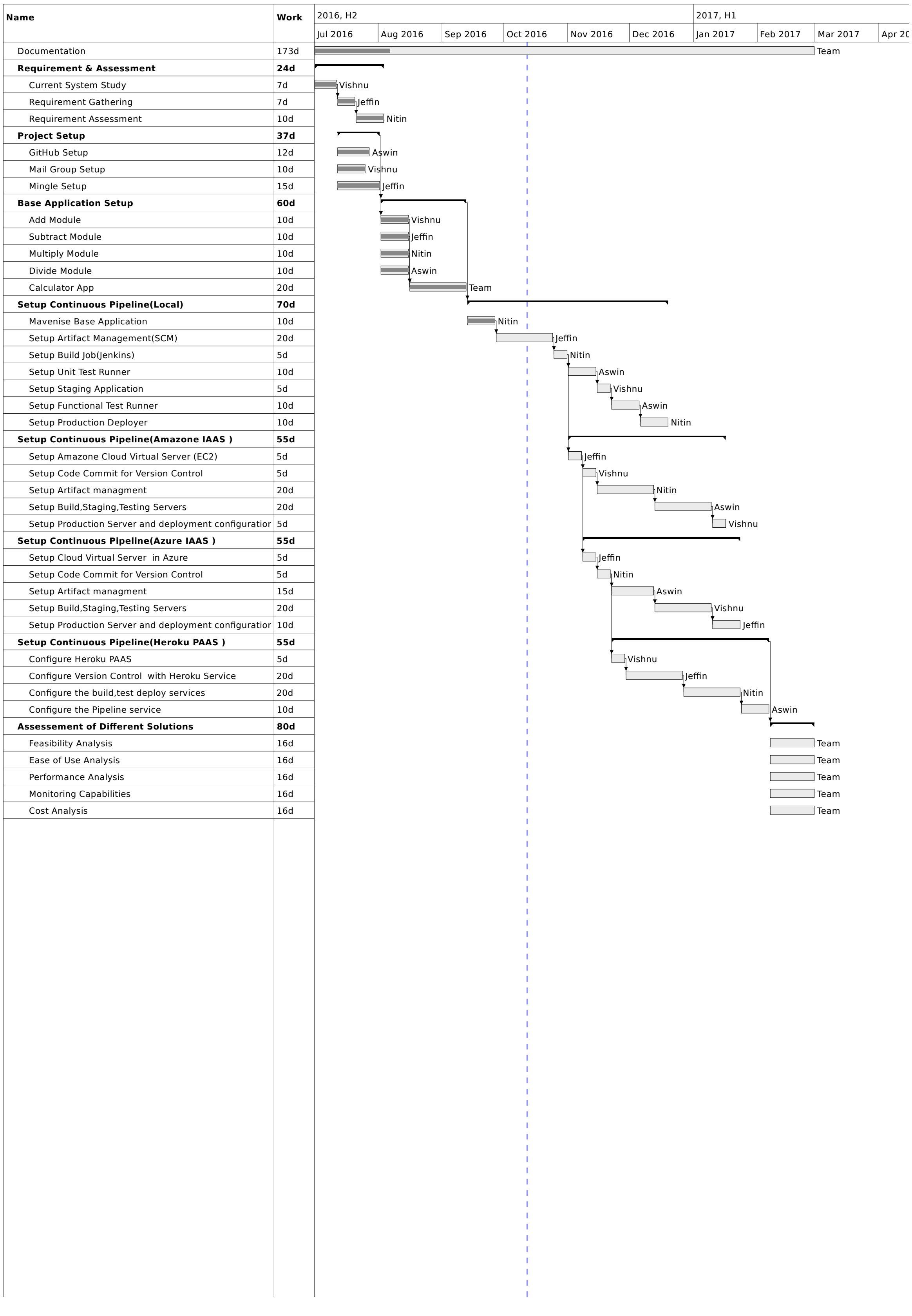
- Systems need high speed internet connection.

3.4 Software Requirements

- Java, Net Beans IDE
- Internet explorer or any other supported browsers

3.5 Gantt Chart

Gantt Chart provides an approximation about the completion of project.It gives a rough description of the completion time,testing time etc.



Name	Work
Documentation	173d
Requirement & Assessment	**24d**
Current System Study	7d
Requirement Gathering	7d
Requirement Assessment	10d
Project Setup	**37d**
GitHub Setup	12d
Mail Group Setup	10d
Mingle Setup	15d
Base Application Setup	**60d**
Add Module	10d
Subtract Module	10d
Multiply Module	10d
Divide Module	10d
Calculator App	20d
Setup Continuous Pipeline(Local)	**70d**
Mavenise Base Application	10d
Setup Artifact Management(SCM)	20d
Setup Build Job(Jenkins)	5d
Setup Unit Test Runner	10d
Setup Staging Application	5d
Setup Functional Test Runner	10d
Setup Production Deployer	10d
Setup Continuous Pipeline(Amazone IAAS)	**55d**
Setup Amazone Cloud Virtual Server (EC2)	5d
Setup Code Commit for Version Control	5d
Setup Artifact managment	20d
Setup Build,Staging,Testing Servers	20d
Setup Production Server and deployment configuratior	5d
Setup Continuous Pipeline(Azure IAAS)	**55d**
Setup Cloud Virtual Server in Azure	5d
Setup Code Commit for Version Control	5d
Setup Artifact managment	15d
Setup Build,Staging,Testing Servers	20d
Setup Production Server and deployment configuratior	10d
Setup Continuous Pipeline(Heroku PAAS)	**55d**
Configure Heroku PAAS	5d
Configure Version Control with Heroku Service	20d
Configure the build,test deploy services	20d
Configure the Pipeline service	10d
Assessement of Different Solutions	**80d**
Feasibility Analysis	16d
Ease of Use Analysis	16d
Performance Analysis	16d
Monitoring Capabilities	16d
Cost Analysis	16d

Fig. 3.1: Gantt Chart

Chapter 4

SYSTEM DESIGN

4.1 Modules

change cheyyanm .... :D jefine

Based on the project aspect the project is divided into 4 modules:

- Input
- Feature Extraction
- Recognition
- Output

4.1.1 Input

Input to the software can be of any form. It can either be a scanned image of a handwritten document or a printed text. We can also download the image using the Download option on the GUI. The input has a lower cut off for its quality.

4.1.2 Feature Extraction

Feature Extraction of input is the first phase of processing. Each and every character should be separated for its easy recognition. And the separated characters will be fit into a matrix of standard size.

4.1.3 Recognition

As soon as the characters are fit into the matrix, the occupied cells should be given value 1 and 0 otherwise. Each and every character is given a certain value range for matrix with respect to the occupied cells.

4.1.4 Output

Output can be obtained either as a text file or an audio file as we wish.

4.2 Data Flow Diagram

The data flow diagram (DFD) is used for classifying system requirements to major transformation that will become programs in system design. This is starting point of the design phase that functionally decomposes the required specifications down to the lower level of details. It consists of a series of bubbles joint together by lines. Bubbles: Represent the data transformations. Lines: Represent the logic flow of data. Data can trigger events and can be processed to useful information. Systems analysis recognizes the central goal of data in organizations. This dataflow analysis tells a great deal about how organization objectives are accomplished. Dataflow analysis studies the use of data in each activity. It documents these finding in the DFDs. Dataflow analysis give the activities of a system from the view point of data where it originates , how they are used or hanged or where they go, including the stops along the way from their destination. The components of dataflow strategy span both the requirements and systems design. The first part is called dataflow analysis.

4.2.1 Level 0 DFD

Level 0 DFD gives a simple information about the overall structure.

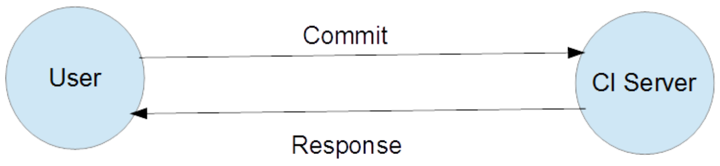


Fig. 4.1: Level 0 DFD

4.2.2 Level 1 DFD

The level 1 DFD gives a basic structure of the project indicating the five modules needed to execute the project.

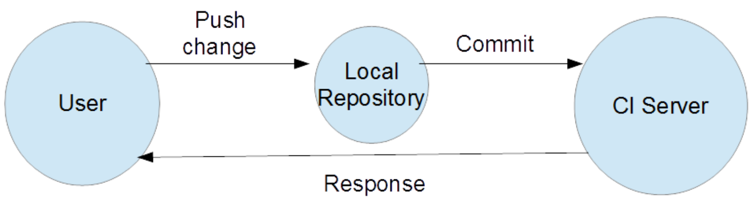


Fig. 4.2: Level 1.1 DFD

4.2.3 Level 2 DFD

The level 2 DFD gives a much more advanced idea about the execution. Each of these sections perform unique functions and these are combined together to yield the final product.

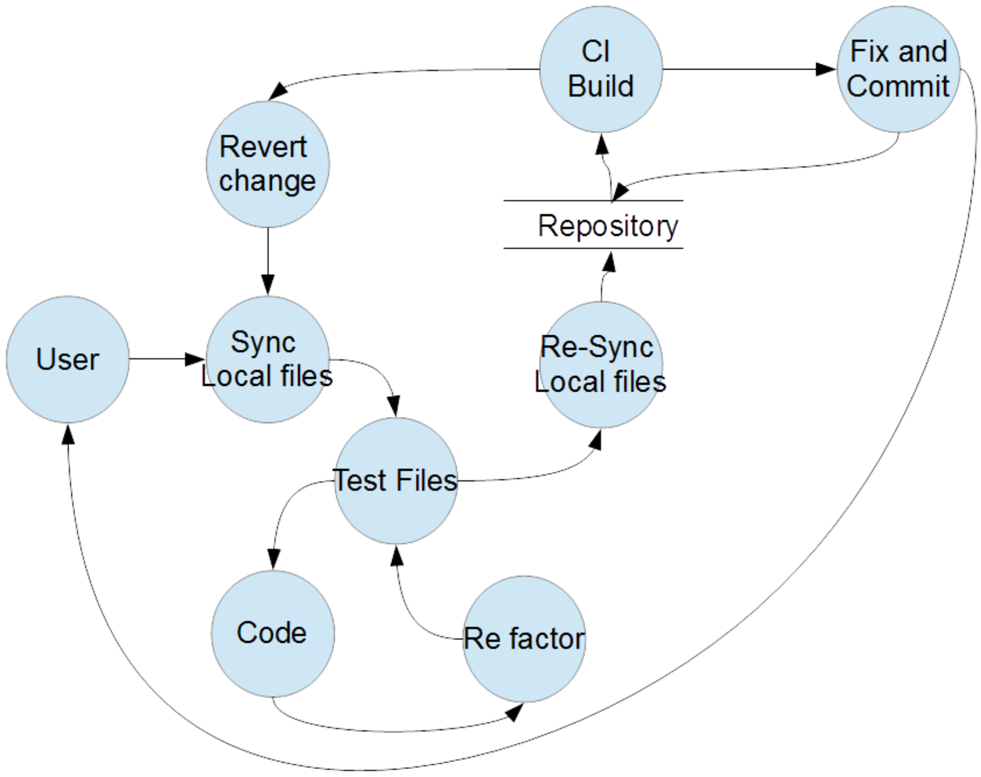


Fig. 4.3: Level 2.1.1 DFD

4.3 Flow Diagram

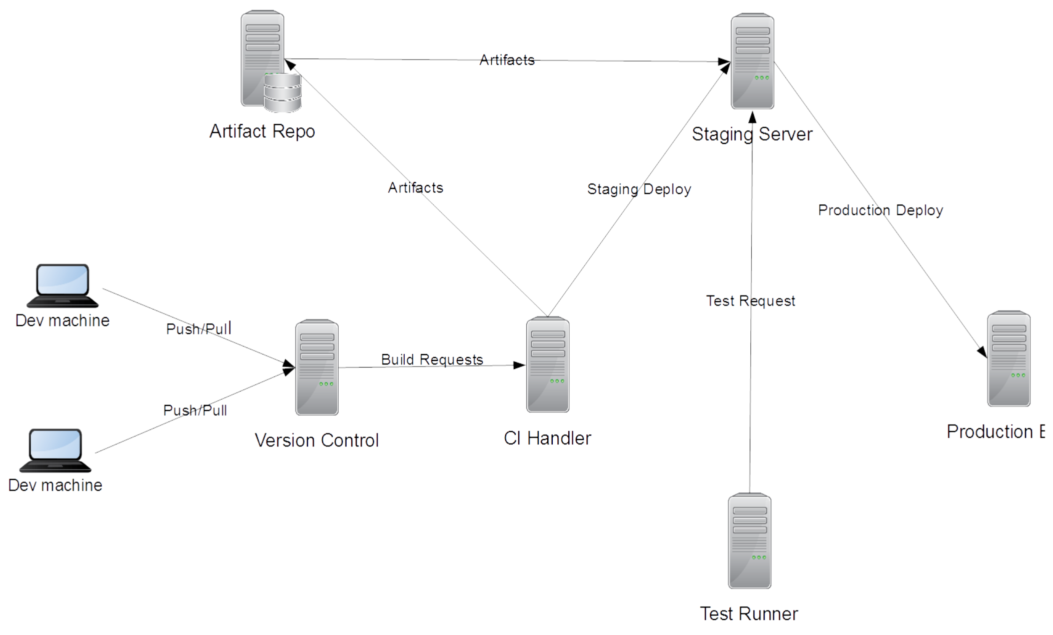


Fig. 4.4: Flow diagram

4.4 C.I Pipeline Diagram

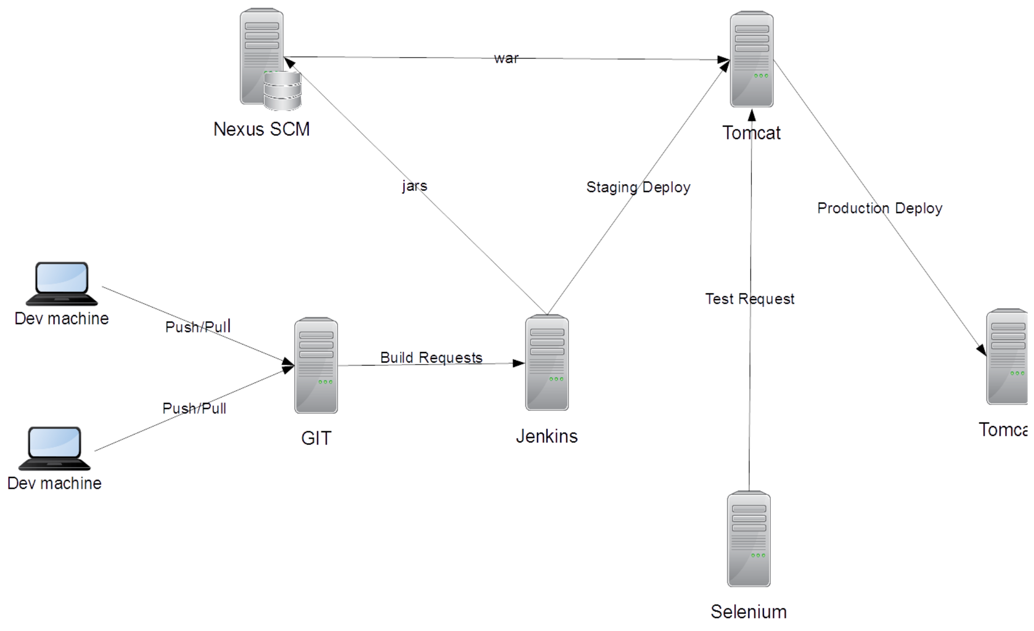


Fig. 4.5: C.I Pipeline

4.5 Sequence Diagram

A Sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

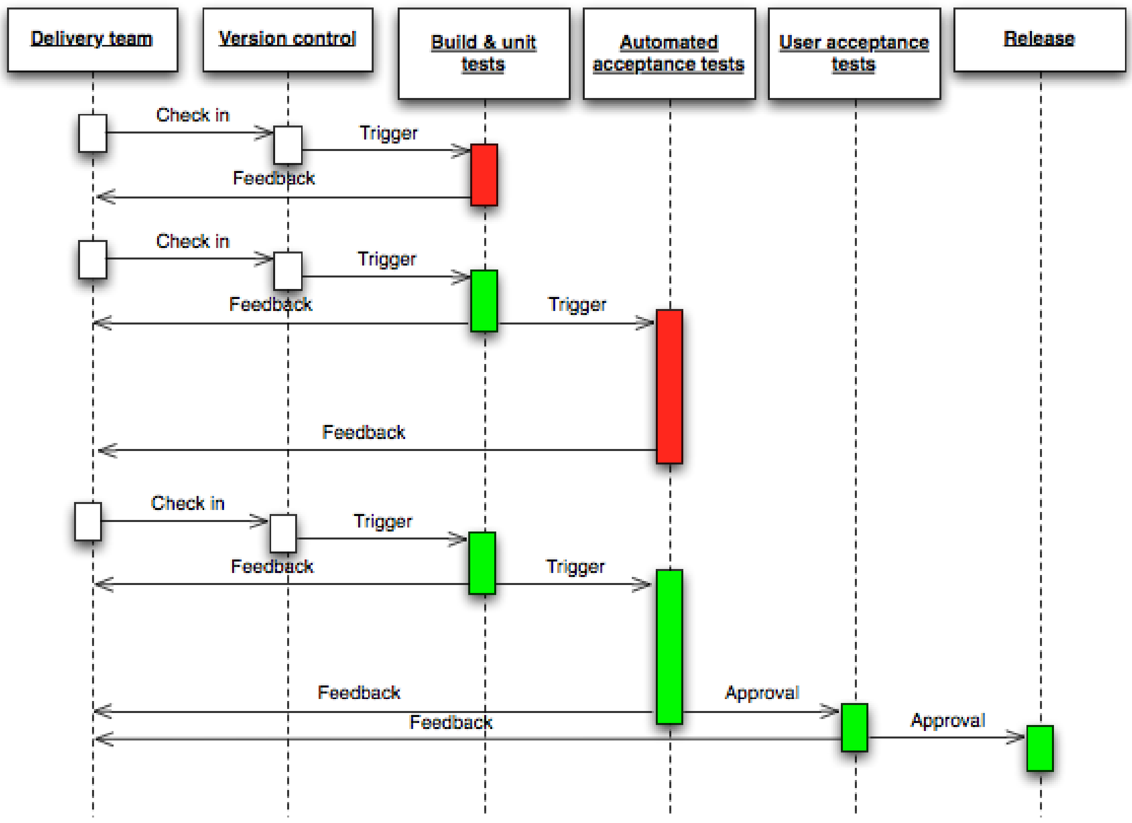


Fig. 4.6: sequence

Chapter 5

METHODOLOGIES

The product is completely been developed in Java environment in Windows platform. This chapter may explain our implementation details.

5.1 Languages And platforms Used

The product is developing in Java as well as C++ environment. We are looking forward on Netbeans IDE 7.1 for networking and web page development. For interface we use swing and awt packages in Java8.0. The development will be carried out in LINUX platform and since we are depending only on Open Source softwares we may claim that we are technically feasible. The reason for selecting C++ as the language is been listed below. C++ is a powerful and flexible Open Source language.We use a library called **FANN**(Fast Artificial Neural Networks) to implement our neural network. Reason for using Java for developing the interface is given below. Java is developed in the name of Green project, its Write Once Read Anywhere (WORA) feature makes it a powerful language packed with cross-platform dependence. Closely linked with the Object Oriented concept but not completely Object Oriented Language due to its Interface libraries, this language offers more security than any other languages do. It is the same platform independence feature of JAVA that tempts programmers and developers to build the networking application interfaces in J2EE environment.

Chapter 6

CONCLUSION

Optical Character Recognition has it own significance nowadays. It has wide range of applications in the domain of postal automation, automatic number plate recognition, preservation of handwritten historical documents,bank check processing,reading aid for the blind people etc. Handwritten character recognition is traditionally divided into on-line and online recognition.We are doing an on-line HCR system.

Even though HCR systems are well advanced in foreign language scripts like Chinese and Japanese, only few works exist in Indian scripts especially in the South Indian scripts. This is mainly due to its large character set, high degree of similarity between these characters and the presence of compound characters in these scripts. Also, variations in the writing styles of different people make the recognition process more difficult. Among the Indian languages, most of the work has been reported in Devanagari and Bangla. The research on South Indian scripts namely Malayalam, Tamil, Telugu and Kannada have gained much popularity recently as many agencies like the Ministry of Communications and Information Technology and Government of India are providing aid and financial support for many of these projects.

Chapter 7
REFERENCES

[1] Yalniz, I.Z.; Manmatha. R, “A Fast Alignment Scheme for Automatic OCR Evaluation of Books” International Conference Document Analysis and Recognition (ICDAR), 2011.

[2] Jeff Heaton, “Introduction to neural networks in Java, Heaton Research Inc., 2005,0-9773206-0-X. [10]. Raman Maini and himanshuAggarwal, A comprehensive review of image enhancement techniques”, Journal of computing, volume 2, issue 3, 2010, ISSN 2151-9617.

[3] Brijmohan Singh, Mridula, Vivek Chand, Ankush Mittal and D.Ghosh, “A comparative study of different approaches of noise removal for document images”, International conference on soft computing for problem solving, volume 130/2102, p 847-854, 2011.

[4] ErginaKavallieratou and FotisDaskas,“Text line detection and segmentation Uneven skew angles and kill-and-dale writing”, Journal of Universal computer science, Vol 17, p 16-29, 2010.

[5] Vatsal H. Shah“Machine Learning Techniques for Stock Prediction”

[6] Robert Burbidge, Bernard Buxton, “An Introduction to Support Vector Machines for Data Mining” *Computer Science Dept., UCL, Gower Street, WC1E 6BT, UK*

[7] C N Aganthopolous, “A License Plate Recognition algorithm for Intelligent Transportation System applications”. University of the Aegean and National Technical University of Athens, p 377- 392, 2006.