Functional Requirements Document

Prepared for

Mr Yegash Naidu (Director)

Project

Version Control

Prepared by

Mr. Sello Tsoka and Mr. Chris Ejike

Contributors

Mr. Tshepo Chuene (System Analyst)

|  |  |  |  |
| --- | --- | --- | --- |
| **Description of Deliverable: Functional Requirements Document**  The requirements specification for the application. | | | |
| **Team Lead Member Sign-Off** (To be signed off by team member upon completion) | | | |
| Name | Date | |  |
| Mr. Tshepo Chuene |  | |  |
| **PMO Quality Sign-Off** (To be signed off by Person(s) appointed for quality review) | | | |
| Name (Print or Type) | Date | | Signature |
|  |  | |  |
| **Team Sign-Off** (To be signed off by team leader only when deliverable has been satisfactorily completed and quality is acceptable) | | | |
| Name (Print or Type) | | Date | Signature |
|  | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |

# Executive Summary

The Requirement document describes in common terms:

* The problem summary including current business/environment issues
* The project goals and objectives
* User/Key Stakeholder needs and expectations
* Proposed technology to support the new or altered business processes
* How implementation of the proposed solution will benefit the users/stakeholders

The FRD (Requirement) is the starting point of the solution and system development and is a collaborative effort between all business and technology stakeholders

The purpose of the FRD is to communicate business needs in common terms to all project and technical team members to ensure the end product meets the business objectives. It is the first phase of the Systems Development Life Cycle. During later phases, the FRD will be used to:

* Design how the solution will be delivered
* Test that requirements are delivered correctly
* Measure the quality of the project deliverables and outcome

The following table provides a high-level description of the content in each section of this document:

|  |  |
| --- | --- |
| Section | Description |
| [Introduction](#_Introduction) | Provides a description of the problem summary, the business reasons for this initiative and the key stakeholders. Additionally, it includes a glossary of terms and acronyms |
| [Project Overview](#_Project_Overview) | Provides a description of the scope and assumptions around the initiative. |
| [Project Approach](#_Project_Approach) | Provides a description of the approach employed to gather requirements for the initiative |
| [Business Requirements Definition](#_Business_Requirements_Definition) | Provides a detailed listing of the Functional, non-Functional, and Integration- & Interface requirements as well as impacted business process flows. |

# Introduction

## Problem Summary

### Background

The use of version control is very important for any IT environment. It provides a fast and easy way to keep tracks of changes and encourage team collaboration. Currently, our IT Services is manually implementing their change tracking. This is as a result of their previous version control account (JAVA.NET) which was closed by the host.

### Problem Statement

No version control currently used

## Business Impact of problem

* Takes extra time to backup projects manually Vs using a version control.
* Hard to keep track of changes made to projects
* Audit requirements for software development

### Current situation

The current situation did not have a proper process structure for developers to follow. They mostly worked independently.

Collaboration is done on one machine one developer at a time. No simultaneous collaboration. Backup changes are done on USBs, external hard drive(s) or use cloud storage (Google Drive, One Drive etc.). Finally, documentations are done and saved on knowledge base.

**2.1.3 Root cause analysis**

**Why are the development team not using version control anymore?**

* The Java.net account they had was closed

**2.1.4. Envisioned solution/objective.**

Deploy a system that is able to keep track of changes on the projects and allows for simultaneous team collaboration and can be locally managed. This solution should include the following basic client requirements:

* Must be able to perform version control
* Must be flexible enough to support many platforms
* Must be user friendly
* Must be secured (permit access at different levels)
* Ability to work disconnected/remotely

Other added benefits are also allowed.

**What are the version control options proposed?**

* Java.net
* GIT - which is a preferable alternative based on its popularity and ease of use
* SVN
* SharePoint/Microsoft Team Foundation Version Control,
* Mercurial and

**Comparison of different well known version controls**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FACTORS** | **Java.net** | **Git** | **SVN** | **SharePoint/** [Team Foundation Server](https://en.wikipedia.org/wiki/Visual_Studio_Team_Foundation_Server) **(TFS)** | **Mercurial** |
| **Supported platforms** | Java IDE, windows | [POSIX](https://en.wikipedia.org/wiki/POSIX), [Windows](https://en.wikipedia.org/wiki/Microsoft_Windows), [OS X](https://en.wikipedia.org/wiki/OS_X), cross-platform via [Visual Studio Team Services](https://en.wikipedia.org/wiki/Visual_Studio_Team_Services), Linux, Java IDE, | [Unix-like](https://en.wikipedia.org/wiki/Unix-like), [Windows](https://en.wikipedia.org/wiki/Microsoft_Windows), [OS X](https://en.wikipedia.org/wiki/OS_X) | [Windows](https://en.wikipedia.org/wiki/Microsoft_Windows), cross-platform via [Visual Studio Team Services](https://en.wikipedia.org/wiki/Visual_Studio_Team_Services) | [Unix-like](https://en.wikipedia.org/wiki/Unix-like), [Windows](https://en.wikipedia.org/wiki/Microsoft_Windows), [OS X](https://en.wikipedia.org/wiki/OS_X) |
| **User friendly?** | yes | yes | yes | yes | yes |
| **security** |  | [SHA-1](https://en.wikipedia.org/wiki/SHA-1) hashes | Numbers | Numbers | Numbers, [SHA-1](https://en.wikipedia.org/wiki/SHA-1) hashes |
| **Usage** | Was used by Oracle, VUT | Used by Apple, Oracle, Google, Facebook, BMW,IBM, Yahoo, Intel | Used by Apple, Yahoo, Intel | Used by Microsoft | Used by Apple, Facebook, |
| **Language** | Only used for Java platforms. | Cross-compatible to C, Perl, Shell scripts, Ruby, PHP, Perl, python, XML and also used for HTML with inline CSS and JavaScript, Css, json, apache, sql, java, Delphi, applescript, cpp - C++ | C, C#, PHP, Python, Perl, Ruby, and Java, SQL | Only for Microsoft | [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), [C](https://en.wikipedia.org/wiki/C_(programming_language)) |
| **Cost** | Was free but closed now | Free | free | Need Microsoft license for deployment unless used for Open source projects. | free |
| **Storage methods** | Change-set and Snapshot | Snapshot | Change-set and Snapshot | Change-set | Change set |
| **Support access permission levels** | yes | yes | yes | yes | yes |

**PROs and CONs of selected version controls**

1. **Git**

**Pros:**

* User friendly
* Preferable alternative for developers who are not comfortable using CVS/SVN
* Larger community than other version controls. Thus more possible help available
* Faster operation speed than SVN (takes Git 1/3 of a second to copy and 13 seconds to do the same on SVN)
* Cheap branch operations – thus merging is easy
* Full history tree available offline
* Wide variety of plug-ins for IDEs
* Distributed, peer-to-peer model (means collaborating developers on the same local network can share their commits before pushing it to the master)

**Cons:**

* Learning curve for those used to SVN
* Not optimal for single developers
* Limited Windows support compared to Linux

1. **SVN**

**Pros:**

* Newer system based on CVS
* Includes atomic operations (when you do a commit to a project, either everything you want to commit goes in, OR nothing does. Thus avoids partial commits of files).
* Cheaper branch operations
* Wide variety of plug-ins for IDEs
* Does not use peer-to-peer model

**Cons:**

* Still contains bugs relating to renaming files and directories
* Insufficient repository management commands
* Slower comparative speed

1. **Mercurial**

**Pros:**

* Easier to learn than Git
* Better documentation
* Distributed model

**Cons:**

* No merging of two parents
* Less out of the box power
* It has a smaller community than Git.

1. **TFS**

**Pros:**

* Full integration with visual studio
* Integration with work items & projects
* Flexibility

**Cons:**

* Flexibility (In order to configure it “just right” it can be time-consuming!)
* Heavyweight back-end requirements (Two server setup recommended, backup/restore costlier & time-consuming)

**Recommendation**

**We recommend Git for the following reasons:**

* Very easy to use. You do not need programming expertise to use it.
* Git uses a ***Distributed repository model***. – Git is locally enabled. Thus work continues even offline.
* Open source
* Git tracks state of changes, history of entire project and also integrates the source tree
* Git will not use much bandwidth as you can work offline after the entire cloning of master repos to local branches.
* No loss of committed data. Even if they are deleted from recycle bin. Can still be recovered
* Developers have full control on commands that they want. They can even customize their files and project to suit them. For example, you can merge selected segment of different works done by different people instead of merging the entire versions of different people to make a new version.

Note: latest Git 2.13 improves security by:

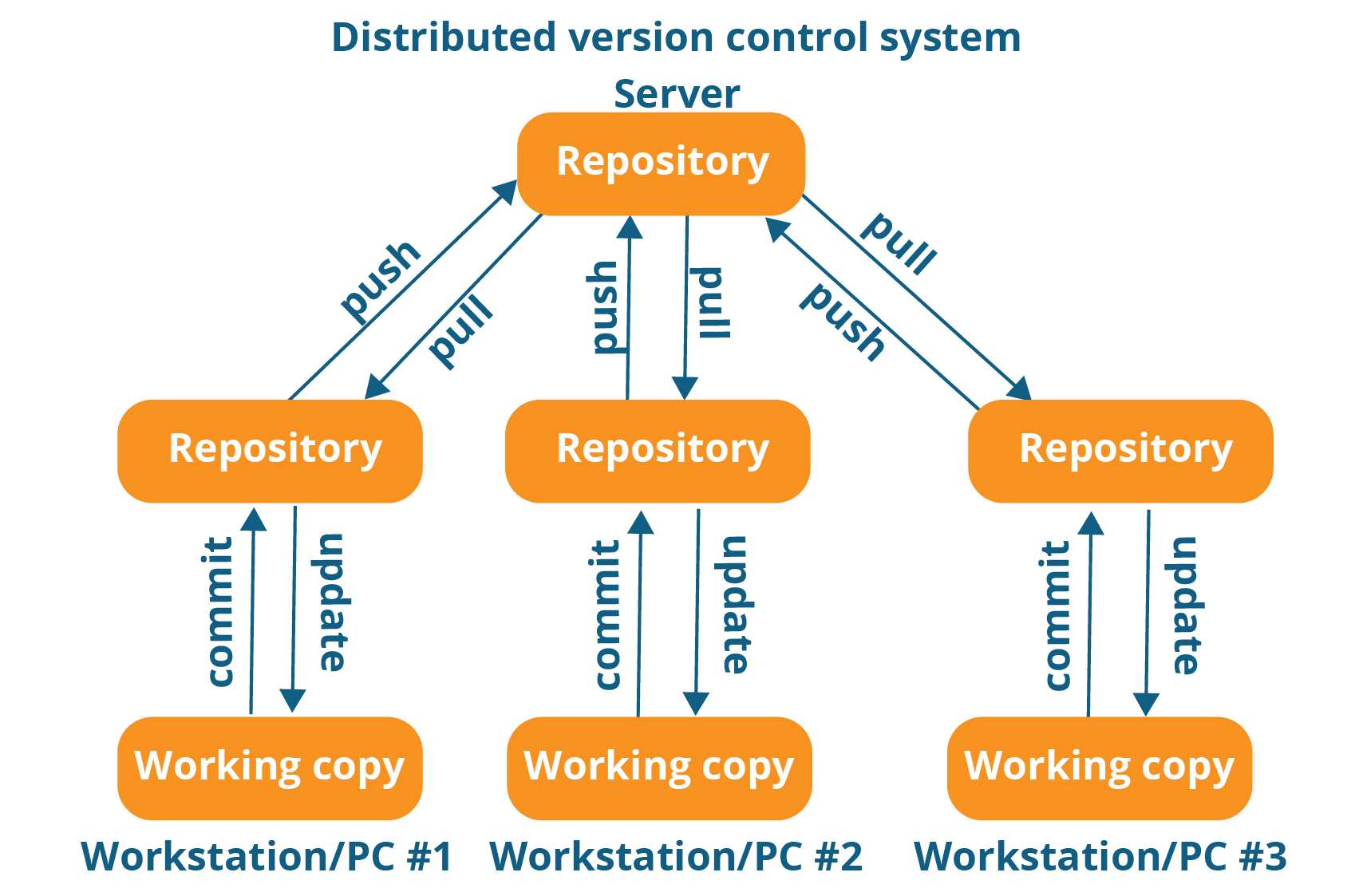
* Provide One Drive support for cloud storage of about 60GB repo thus no need to pay at GitHub for a personal repo. Git uses Git-annex plugin to achieve this.
* Providing the ability to handle multiple identities through conditional configurations. ***Conditional configuration*** provides a way to include a Git config file based on a set of conditions.

## User Profiles

The following table provides a description of the system key stakeholders and users:

| Stakeholder/User | Description |
| --- | --- |
| Mr. Yegash Naidu | Sponsor |
| Developers | System users who will keep track of the changes |
| Support | System users who will maintain the system |

## Envisioned Process



**Basic Git command terminologies**

* **git commit** – used to save and record changes to the repository. For easy reference, each commit has a unique ID.
* **git push –** used to send changes to the master branch of your remote repository
* **git pull –** used to get the latest version of a repository run *git pull*. This pulls the changes from the remote repository to the local computer.

Business Requirements

| Category | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| Process Id | Process Name | Requirement ID | Requirement Description | Business Rules / Examples | Priority | Owner |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |