Prevent, Mitigate, and Recover (PMR) Insight Collective Knowledge System (PICK)

Software Configuration Management Plan

Version 1.0

2/6/2020

**Document Control**

**Approval**

The Guidance Team and the customer shall approve this document.

**Document Change Control**

|  |  |
| --- | --- |
| Initial Release: | 1.0 |
| Current Release: | 1.1 |
| Indicator of Last Page in Document: | > |
| Date of Last Review: | 2/6/2020 |
| Date of Next Review: | 2/15/2020 |
| Target Date for Next Update: | 2/20/2020 |

**Distribution List**

This following list of people shall receive a copy of this document every time a new version of this document becomes available:

Guidance Team Members:

Dr. Gates

Dr. Salamah

Dr. Roach

Jake Lasley

Customer:

Dr. Oscar Perez

Vincent Fonseca

Herandy Denisse Vazquez

Baltazar Santaella

Florencia Larsen

Erick De Nava

Software Team Members:

Eduardo Lara

Gerardo Armenta

Hector Dozal

Irvin Bosquez

Victor Vargas

**Change Summary**

The following table details changes made between versions of this document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modifier | Description |
| 1.0 | 2/5/2020 | Irvin Bosquez | Section 1, 2 |
| 1.0 | 2/5/2020 | Eduardo Lara | Section 1, 2 |
| 1.0 | 2/5/2020 | Hector Dozal | Section 3, 4 |
| 1.0 | 2/5/2020 | Gerardo Armenta | Section 3, 4 |
| 1.0 | 2/5/2020 | Victor Vargas | Section 3, 4 |
| 1.1 | 2/10/2020 | Victor Vargas | Added details to section 3,4 |

**Table of Contents**

[**DOCUMENT CONTROL**](#_30j0zll) 2

[**Appro**](#_44sinio)val 2

[**Document Change Control**](#_2jxsxqh) 2

[**Distribution List**](#_z337ya) 2

[**Change Summary**](#_3j2qqm3) 2

[**1.**](#_3dy6vkm) **INTRODUCTION**  5

[**1.1.**](#_1t3h5sf) **References** 5

[**2.**](#_4d34og8) **SOFTWARE CONFIGURATION IDENTIFICATION**  6

[**2.1.**](#_2s8eyo1) **Software Configuration Item Identification**  6

[**2.2.**](#_17dp8vu) **Software Configuration Item Organization**  6

[**3.**](#_3rdcrjn) **SOFTWARE CONFIGURATION CONTROL**  7

[**3.1.**](#_26in1rg) **Documentation** 7

[**3.2.**](#_lnxbz9) **Configuration Control Board**  7

[**3.3.**](#_35nkun2) **Procedures**  7

[**4.**](#_1ksv4uv) **SOFTWARE CONFIGURATION AUDITING**

# Introduction

The software configuration management plan is intended to allow the team to create and maintain the lifecycle of a software system. This document will describe the three main concepts for our project “PICK” (PMR Insight Collective Knowledge), which includes Software Configuration Identification, Control, and Auditing. ”PICK” requires different people on the team to work on different areas of the project, which is why it is essential we follow the Software Configuration Management, SCM, plan within this document to ensure the process of creating the software from beginning to end flows fluently.

## References

[1] Tai Ramirez, E., Roach, S. Software Requirements Specification.docx

# Software Configuration Identification

This section will provide the information for baselines and updates for “PICK”.

## 2.1 Software Configuration Item Identification

This section will describe all the configuration items that will be included in “PICK”.

* Source Code: Code we will need to implement to make the application to succeed
* Design Document: We are going to design the Graphical User Interface, GUI, using qt Designer to meet the clients requirements on the Software Requirements Specification, SRS.
* Test Suites: Robot Frameworks, pytest, and pyunit
* Requirements Documents: Maltego, Python3, Splunk, and Pytesseract
* Project Plans: We are planning to follow the SCM plan. We are planning to follow a similar structure of code within the team. We will use testing techniques, and Security measurements.
* Project Standard: we will be going to use one class for each tab of the project; we will leave the working project in the master github all the time, and the new updates made by different members will be in their own branch
* The project will contain all the latest versions of all softwares.

## 2.2 Software Configuration Item Organization

This section will describe the labeling scheme for each baseline and update to the system. The first version of PICK will be v1.0, which will serve as the baseline to the system.

Any changes to the system will go through the team and when an update is to be provided for the baseline, the version will be updated to v1.1 for the first update, v1.2 for the second update and so on.

Additionally, any bug fixes done for a version of the baseline will be presented as v1.1.1 for the first update, v1.2.1 for the second update and so on. This number may increase as well if new bugs are identified and fixed and will be presented as v1.1.2 for the first update, v1.2.2 for the second update and so on.

In general the structure will be like this.

v1.0 --- Initial Version

v1.1, 1.2, 1.3… --- Following Baseline Versions

v1.1.1, 1.1.2, 1.1.3…. --- Bug Fixes with second number corresponding to baseline version.

All our files will be in a single directory, but inside there will be separate folders that will each hold specific files. Some folders include, GUI elements, source code, test cases. In regard to backups of our code, one team member will do a full backup of all the mainline code to a separate repository every week on Sundays at 10:00pm. Additionally, a separate team member will be in charge of a local backup that will be done every week on Sundays at 10:00pm just like the one in the separate repository every week. Doing so will maintain both backups with the same versions of all the code used and have two different backup controls for the team project. Each file that is backed up will have the date of the most recent changes to it in order to keep track of all the recent files, along with comments about each backup that happens.

The repository that we will be using is going to be GitHub. GitHub allows our team to have a master branch, which will serve as the mainline to the source code of the system. Whenever an update to the code is to be provided, the new code will be merged onto the mainline and the mainline will be the newest version. The branching strategy that will be used is a simple GitHub Flow which requires branching off master every time a new feature or bug fix is required, allowing master to stay deployable and when a feature or fix is finished, it is merged back to master. Tags will be used to identify versions that a branch is working on or corresponds to. Prior to a branch being merged back to master, it will be discussed with the team to make sure all necessary changes for the new version or correct.

# Software Configuration Control

To make any change to the configuration items the proposed changes must first be communicated and discussed with all members of the team. After an agreement has been reached regarding the proposed changes an investigation will be performed regarding the potential benefits as well as the disadvantages regarding changes to the configuration Items. An investigation will require all members of the group to individually research different ways in which the change will impact productivity around the team. After the investigation is concluded and if the results indicate a positive the proposed change will be put into place.

## 3.1 Documentation

This section will focus on any control changes that will happen to the system and how it will be processed:

-Any proposed changes will be brought up during team meetings.

-Every team member will have a description on what changes need to happen in the system or what changed they suggest.

-Once all change proposals have been heard, the team in unison will decide which changes need to happen and which don't, and which changes should have a higher priority. This will be set based on deadlines and the SRS, Memos and Client Meetings, giving the SRS main features a higher priority over aesthetic changes for example.

-The results for the change proposals will be based on the feedback the team gets from the clients, and from the meetings that happen with the guidance team.

-Since the team is responsible for both proposing changes and writing the code, this documentation will serve a big part of the system to make sure we are doing the correct changes and not doing any unnecessary work.

-Once changes have been decided, the task will be set on the GitHub Project section and a xdescription will be documented on google docs.

## 3.2 Configuration Control Board

For every major point in the development of the system a lead will be assigned, the lead will be responsible for controlling the changes and approving changes proposed by other team members. Team leads will be assigned based on who has not been a lead before and whose role fits best for the assigned task (Lead programmer, Architect, etc).

Specifically, the teams V&V will be tasked with ensuring that any changes done to the system are both valid and verifiable, this will be done by auditing main changes and tracing such changes to the SRS manually. To ensure that each team member only directly contributes to a specific component without overlapping we will assign and divide tasks during a weekly personal meeting.

The branches generated by the changes proposed will keep track of changes and who made them, making it easy to search for changes and ask certain members to explain them.

## 3.3 Procedures

This section will describe the way the team is going to keep track of all changes and proposes to the system. The team will use GitHub Projects Section to keep track of activities and tasks that must be completed. The issues and pull request sections will provide the team with tasks, features that are being worked on and fixes.

For the configuration items and the modifications, GitHub automatically keeps track of all changes, commits, pushes and pull requests made, so that is taken care of automatically.

For team meetings and decisions made by the team, we will document those in a Google Docs with dates and time of the decision, which will usually be reflected on items in the projects section of GitHub.

In general, the procedure for a change in the code will be as follows:

-After a change has been proposed and accepted, a task will be created in GitHub, along with, as explained before, a lead.

-A branch will be created for the fix, or feature to be added.

-The team or a subset of the team will work on the branch.

-Once a feature is added or a bug is fixed, a merge request will be generated to try and rejoin with the main branch.

-If any conflicts exist, we will look at the conflicting modifications, trace the conflict back to where it came from to see if it needs to be fixed, or the feature is incompatible with the current main.

-If unsolvable conflict, it will be deemed as a necessary fix, and will get its own branch to solve that. Else, branched will be merged back and the main branch version will be updated according to the already mentioned numbering scheme.

In general, the procedure for a change in plan, or the change of a document or a design model will be as follows:

-After the change has been proposed and accepted (following the procedure described in previous sections) the document or artifact will be created (if already created, it will be modified where it was created) with a collaborative tool such as Google Docs, drawIO or any similar tool that automatically keeps track of changes and allows concurrent working.

-A lead, and the team or a subset of the team will work on the artifact.

-Once finished, the artifact will be compared to the current version of the system, to see if the current system requires a modification or an update.

-If no update or modification required at the moment, the artifact will be saved. Otherwise, we will initiate the procedure for code editing or the artifact will be redone without the conflicts.

# Software Configuration Auditing

The mechanism the team will use to determine the degree to which a configuration of the software system mirrors the intended software system will be the SRS. The SCM plan the team creates will have to follow the necessary requirements from the SRS and the clients needs. The team will follow the plan and make necessary changes to the SCM plan if they are needed, but making sure the team follows the plan will show positive results.

**>**