PMR Insight Collective Knowledge (PICK)

Software Configuration Management Plan

Version 2.2

2/24/2020

Document Control

Approval

The Guidance Team and the customer shall approve this document.

Document Change Control

|  |  |
| --- | --- |
| Initial Release: | 1.0 |
| Current Release: | 2.2 |
| Indicator of Last Page in Document: | $$$ |
| Date of Last Review: | 2/24/2020 |
| Date of Next Review: | 2/25/2020 |
| Target Date for Next Update: | 2/27/2020 |

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. Steven Roach

Jake Lasley

Customer: Mr. Baltazar Santaella

Ms. Herandy Vasquez

Mr. Vincent Fonseca

Ms. Florencia Larsen

Mr. Eric de Nava

Software Team Members: Itzel Rivas

Eder Rodriguez

Jose Gallardo

Gilbert Alvarez

Nusrat Sarmin

Change Summary

The following table details changes made between versions of this document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modifier | Description |
| 1.0 | 2/2/2020 | Itzel Rivas | Started the document |
| 1.1 | 2/4/2020 | Itzel Rivas | Added section 1 (introduction) |
| 1.2 | 2/5/2020 | Gilbert Alvarez | Answered: Describe how errors will be documented, how will change requests be made. |
| 1.3 | 2/5/2020 | Jose Gallardo | Answered: What source code control tool will you be using? Where is it hosted? |
| 1.4 | 2/5/2020 | Nusrat Sarmin | Answered: How will you back-up files? |
| 1.5 | 2/5/2020 | Eder Rodriguez | Defined the directory structure that we are currently using. |
| 1.6 | 2/5/2020 | Itzel Rivas | Added questions on how to keep track of the versions of our code- files, and storage of documents and code |
| 1.7 | 2/16/2020 | Itzel Rivas | Added section 2.1 |
| 1.8 | 2/17/2020 | Nusrat Sarmin | Added software configuration auditing to section 4 |
| 1.9 | 2/23/2020 | Eder Rodriguez | Added the workflow to section 3.3 procedures |
| 2.0 | 2/23/2020 | Gilbert Alvarez | Added section 3.3 “Configuration Control Board” |
| 2.1 | 2/24/2020 | Jose Gallardo | Added the versioning to section 3.3 procedures  Revised document and corrected some grammatical errors |
| 2.2 | 2/24/2020 | Itzel Rivas | Added to section 2.2, 3.1, revised document. |

Table of Contents

[Document Control ii](#_Toc61860422)

[Approval ii](#_Toc61860423)

[Document Change Control ii](#_Toc61860424)

[Distribution List ii](#_Toc61860425)

[Change Summary ii](#_Toc61860426)

[1. Introduction 1](#_Toc61860427)

[1.1. References 1](#_Toc61860428)

[2. Software Configuration Identification 1](#_Toc61860429)

[2.1. Software Configuration Item Identification 1](#_Toc61860430)

[2.2. Software Configuration Item Organization 1](#_Toc61860431)

[3. Software Configuration Control 1](#_Toc61860432)

[3.1. Documentation 1](#_Toc61860433)

[3.2. Configuration Control Board 2](#_Toc61860434)

[3.3. Procedures 2](#_Toc61860435)

[4. Software Configuration Auditing 2](#_Toc61860436)

# Introduction

The purpose of the Software Configuration Management Plan (SCM) is to give the customer and guidance team a clear and precise description of tracking and controlling changes in the PMR Insight Collective Knowledge (PICK) software. The SCM divides itself into three further sections: Software Configuration Identification (SCI), Software Configuration Control (SCC), and Software Configuration Auditing (SCA). The SCI is used to identify any items that can be controlled as well as establish their versions and baselines. The SCC is used to identify the procedures that will be used to control access to items in the configuration in order to prevent unauthorized updates and collisions between team members working on the system simultaneously. The SCA is used to provide a procedure for determining if the current configuration of the software system mirrors the software system pictured in the baseline and the requirements documentation. The intended audience of the SCMP are our customers and guidance team.

## References

[1] CD\_UP\_Change\_Request\_Form\_Example.doc

[2] GitHub. (2020, February 23). Retrieved from https://en.wikipedia.org/wiki/GitHub

# Software Configuration Identification

This section provides an insight into the configuration items in our system and how they will be organized.

## Software Configuration Item Identification

This list includes the items that are likely to change throughout the life of the project:

1. Source Code
2. Design Documents
3. Test suites
4. Requirements Documents
5. Project Plans
6. User Guides
7. Interface descriptions

## Software Configuration Item Organization

The source code control tool we will be using is Git. The source code will be maintained on the repository hosting service GitHub, which is a “US-based global company that provides hosting for software development version control using Git” [2]. The directory structure that we will be using to manage the documents and program code in GitHub is as follows:

PICK/

|

|— src/

|

|— views/

|

|— ui/

|— models/

|— managers/

|— database/

|— conn/

|

|— server/

|— client/

|— tests/

|— documentation/

|— design-models/

The version of the system updated will change by unit increments e.g. 1.0 to 2.0. Where 1.0 is the version of the first deliverable. Between deliveries, the version will be incremented by its decimal number e.g. 1.0 to 1.1 for every approved build of the system with new features. There will also be smaller builds for bug fixes which would add an extra second decimal point for new fix added. i.e. 1.0.1 to 1.0.2, this way it would be easier to identify which changes were made between each deliverable.

Whenever a change is made, it must be stored on the “Changes” document for the system. There will be a folder on Microsoft Teams, where Team 7 members will be constantly uploading the documentation for their changes. Every two weeks all changes are collected and a member that will be assigned will upload the changes to the repository. The file would be located under the Documentation directory with the name “Changes.doc”.

To ensure file and folder security, prudence dictates we take a backup. There is a single python file which does the actual backup. We pushed our files to git. So, if mistakenly we delete or loose the file then we can get the it from git. If we delete the git repository then we should have the files in our laptop. If someone pushed wrong files to the repository then we can easily get back the previous version of the files in git. That’s how we ensure our files back-up. However, we know another two kinds of backup: Copying files to another disk or folder and Compressing files and folders and saving the zip file.

# Software Configuration Control

This section involves the procedures and actions the team will take when managing configuration items.

## Documentation

In order to make changes to the software system, a meeting will be prepared to discuss the changes that are to be done. In the meeting all members (team 7) will be present, and the member that requested the meeting will prepare a form for submitting a change request for the Control Configuration Board to review. After the form has been submitted and reviewed then the Configuration Control Board will decide whether the changes requested will or will not be accepted. Then, the team will decide who will make the changes and the changes will be pushed to the developer branch were the team will put all the features. The changes will be properly document and tested to ensure that implementation doesn’t affect the system at large. Upon completion, and when everything is running properly the team will make a “release” to the master codebase.

The following is a sample form which contains 2 sections, the first section will contain the individual change request and the second one will contain the Configuration Control Board (CCB) final decision regarding the change request [1].

1. **Submitter- General Information**

Requestee:

Brief description of request:

Type of CR (change request):

Enhancement

Defect

Date Submitted:

Date Required:

Reasons for change:

Other Artifacts Impacted

Features

Version:

Justification:

Bug reports

Version:

Description:

How to replicate:

Priority:

Low (is not intended but system will continue working as intended)

Mild (may cause features to fail)

High (The systems stopped, crashed or doesn’t allow correct usage)

Picture(optional)

Comments(optional):

Approval Signature and date

1. **Configuration Control Board – Decision**

Decision:

Approved

Approved with conditions

Rejected

Decision date:

Decision explanation:

Conditions(optional)

Approval Signature and date

## Configuration Control Board

Team members will document any modification (this involves adding features, fixing errors) on a doc file on Microsoft teams. This allows all team members to view teammates work. On the doc file teammates will provide information as seen in section 1.4, Documentation, and highlight the whole block red to indicate to the team that the section requires attention. More blocks of changes will be added below. All team members on team 7 will have access to the git repository to modify parts of the system, however changes team members make, must be made on their own branches.

Approving and denying changes is consulted by the entire team and decisions are influenced by

* Whether the change is necessary. (does it fix a problem, does it correlate to a system requirement, is the change needed to fix and urgent problem)
* Does the change improve the quality of the system (less code to write, does it improve the ability to maintain the system in the future)?
* Is the change feasible (can it be done on time; does it require large amounts of research)?
* Will the change satisfy a new or existing requirement by the clients?

The V & V will document this change approval/denial by changing the highlighted color to yellow for work in progress. Generally, the teammate who proposed the change will work on it. They will perform modifications on their branch and submit a pull request that is approved or denied by the lead programmer. Another person may make changes to changed component this can be done by specifying their name and reason below the block they want to modify on the doc file in teams. Approved changes will be highlighted green on the change file and appended to another doc file named ChangesMade.md in the documentation subdirectory. This will be done by the V&V.

## Procedures

Every developer will Fork the PICK git repository to their own repository, when contributing to the project they will have to create a new branch from the develop branch, not master. The naming conventions of branches will be <git username>/<feature>-<to>-<implement>. Once the feature/fix branch is created the developer will start working in their newly created branch. When a working implementation is complete and is ready to be merged to develop, they will have to go thought a Pull Request. The lead programmer and V&V will review the Pull Request (PR) and will give the final call if to merge it with the develop branch or advise for changes to be done to the PR before merging it. With regards to versioning, we will use git tag when merging to the master branch starting at version 0.0. In terms of updating the version we will increment the number by 0.1 for each merge. In the case of a hot fix or emergency fix, we will update the version by 0.0.1.

# Software Configuration Auditing

This section covers the steps the team will take when reviewing the system.

The goal of configuration auditing is to verify that all software products have been produced, correctly identified and described, and change requests have been resolved in accordance with the Software Configuration Management Plan (SCM). All documentation is present and accurately described the system to be commissioned. All source code is maintained in a secure repository and reflect the executable. In our PICK system we assured these steps.

* The baselines and standards match the actual components in the IT environment
* We verified that services and products are built and documented, according to documented requirements.
* We verified that the correct and authorized versions of any CI exist and is correctly identified and described
* We confirmed that the current environment is as expected and documented in the CMS, and that any change requests are resolved
* We checked that configuration modifications are implemented through authorized changes
* We verified that CI specifications are compliant with defined configuration policies and baselines
* We validated that all required documentation for each CI is available
* We have checked data quality for accuracy and completeness
* We initiated an incident for discovered unauthorized changes
* We are confident that configuration information is accurate, controlled, and visible

$$$