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

Software Configuration Management Plan

Version 1.0

02/06/2020

**Document Control**

**Approval**

The Guidance Team and the customers shall approve this document.

**Document Change Control**

|  |  |
| --- | --- |
| Initial Release: | 1.0 |
| Current Release: | 1.0 |
| Indicator of Last Page in Document: | $ |
| Date of Last Review: | 2/5/2020 |
| Date of Next Review: | 2/6/2020 |
| Target Date for Next Update: | 2/6/2020 |

**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. Ann Gates

Dr. Salamah Salamah

Dr. Steven Roach

Ms. Elsa Tai Ramirez

Customer:

Dr. Oscar Perez

Mr. Vincent Fonseca

Ms. Herandy Denise Vazquez

Mr. Baltazar Santaella

Ms. Florencia Larsen

Mr. Erick De Nava

Software Team Members:

Mr. Valente Arellano

Mr. Tarek Haddad

Mr. Joseph Warren

Mr. Mark Williams

Mr. Ivan Torres

**Change Summary**

The following table details changes made between versions of this document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modifier | Description |
| 1.0 | 2/5/2020 | All team members | Completed first draft of the SCM plan |

**Table of Contents**

[**1. Introduction**](#_heading=h.vroi28flkonb) **1**

[1.1. References](#_heading=h.ijxu0set6roz) 1

[**2. Software Configuration Identification**](#_heading=h.ozftrqwaf7kq) **2**

[2.1. Software Configuration Item Identification](#_heading=h.dp3z83erm8y4) 2

[2.2. Software Configuration Item Organization](#_heading=h.ss0s16desbu0) 2

[**3. Software Configuration Control**](#_heading=h.170uq2v305ut) **3**

[3.1. Documentation](#_heading=h.g5ieupdkjaz) 3

[3.2. Configuration Control Board](#_heading=h.rj63z1kdawl7) 4

[3.3. Procedures](#_heading=h.xsxwuialj7cz) 5

[**4. Software Configuration Auditing**](#_heading=h.b0hbrx2dadrw) **6**

# 1. **Introduction**

The purpose of the Software Configuration Management (SCM) Plan is to document and brief the customers on the configuration management process the software team seeks to establish in order to meet the needs of the system in development - the Prevent, Mitigate, and Recover (PMR) Insight Collective Knowledge System (PICK). Essentially, PICK aims to provide the Lethality, Survivability and HSI directorate (LSH) with the capability of correlating and graphically representing events that have occurred in an Adversarial Assessment (AA).

The SCM Plan defines the procedures the software team will adhere to for:

* Software Configuration Identification
  + Defining and organizing Configuration Items (CIs)
* Software Configuration Control
  + Controlling modifications and releases of CIs
* Software Configuration Auditing
  + Verifying that defined processes are being followed and that delivered versions contain intended updates

The intended audience of the SCM Plan is Dr. Oscar Perez, Mr. Vincent Fonseca, Ms. Herandy Vazquez, Mr. Baltazar Santaella, Ms. Florencia Larsen, Mr. Erick De Nava and the software development team.

## 1.1. **References**

[1] “GitHub features: the right tools for the job,” GitHub. [Online]. Available: https://github.com/features. [Accessed: 27-Jan-2020].

[2] Hans Van Vliet, Software Engineering, Principles and Practice, 3rd edition, John Wiley & Sons, 2008. Chapter 4. [Accessed 05-February-2020]

# 2. **Software Configuration Identification**

The Software Configuration Identification section details all configuration items of the system that are necessary for its creation and installation. Additionally, how our team will ensure an appropriate and safe method of handling all items across current and past versions.

## 2.1. **Software Configuration Item Identification**

Throughout the lifecycle of software development, many iterative versions will need to be completed and reviewed. Even when a product meets all requirements and is ready for use by the client, new versions are always possible as there will always be room for improvement. Each of these versions consist of a configuration, which is the set of items needed to create and install the software. Across all of these versions, the set of items that make up the configuration may slightly change, but one thing that is certain is the modification of at least one of these items to justify a new version. The set of items that make up the configuration for the PICK tool are the following:

* Software Requirements Specification (SRS) document. A document constructed by the Guidance Team that reflects the requirements of the system to be built. However, as requirements change over the development process, those changes will need to be reflected in the SRS and updated in the system.
* Source code. The item that will experience the most change over time. Each new version, whether it is a baseline or not, will have changes made to the source code.
* Test Suite. The set of all test cases we will need to test functions of the system. This is likely to evolve and grow throughout the development of the source code.
* PyQt third party software. A plugin that will likely not see any changes during development but is necessary for construction of the system.
* SCM Plan. This document is the foundation for how new iterations of the system will be released, with the procedures found in section 3 affecting the pacing and organization of our modifications.
* User Guide. A document created by the development team for the clients with instructions for installing and maintaining the software. Likely to be created near the end of the system development.

## 2.2. **Software Configuration Item Organization**

We, the Oroware team, will be using distributed version control with GitHub, a software product that hosts source code control tool “Git” as well as some other features [1]. Each developer will check out a full mirror of the system [2] onto their own machine and, after making necessary changes, merge with the GitHub repository to create a new version. This method of version control will allow us to have multiple copies of the system, albeit with slight changes, in case we lose the repository on GitHub. In the event the repository is lost, our recovery procedure will amount to uploading the most updated and complete mirror amongst our team member’s machines to a new repository. For this reason, each team member will ensure to merge with the master version at least once a week to keep a backup master close at hand. It is important to note that it is highly unlikely that we will lose our source code files because the source control tool we are using, Git, is very reliable.

The initial release of the system will have a version of 1.0. Every update after that will increment the digit after the decimal point ( e.g. v1.2 will be the second update after the initial release). When a new baseline is established, the digit before the decimal point will be incremented (e.g. v2.0 will be released after major stable update has been released).

# 3. **Software Configuration Control**

In this section, we describe the software configuration control mechanisms we will use in this project 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. At a high level, software configuration control defines the process by which a new version is created in a software project. Specifically, software configuration control provides a detailed mechanism for preparing, evaluating, and approving or disapproving all change proposals to the configuration items throughout the life cycle.

## 3.1. **Documentation**

In this section, we review the documentation we will use to formally define a proposed change to the software system being developed. We also explain how changes to the software configuration will be documented. Essentially, we will use a straightforward change request form to define proposed changes. A sample change request form with the same format we will use is as follows:

|  |  |
| --- | --- |
| Change Request ID: | 3 |
| Change Request Description: | Add and integrate popup window when user clicks ‘View Graph’ button in the pushed vector configuration |
| Configuration Item: | ui.py |
| Current Version Number: | 0.4 |
| Next Version Number: | 0.5 |
| Assigner: | Mark Williams |
| Assigner Email: | mawilliams7@miners.utep.edu |
| Assignees: | Joseph Warren,  TJ Haddad |
| Assignee Emails: | jswarren@miners.utep.edu, tjhaddad@miners.utep.edu |
| Request Date: | 2/2/2020 |
| Desired Delivery Date: | 2/16/2020 |
| Actual Delivery Date: | Unknown |
| Priority: | 2 (Out of priority list from 1 to 5 where 1 is the highest priority and 5 is the lowest) |
| Estimated Time To Complete: | 2 hours |
| Actual Time To Complete: | Unknown |
| Approver: | Mark Williams |
| Approver Email: | mawilliams7@miners.utep.edu |
| Status: | OPEN (Out of status list of “OPEN”, “ACCEPTED”, “COMPLETED”, “APPROVED”, “DECLINED”) |

We will keep a database of all change requests created and this will allow us to document all changes to the configuration that occur during the project, as well as analyze metrics related to the progress of the project. It is important to note that we will use the same change request form for changes to all configuration items (i.e. the same change request form will be used for source code *and* documentation as well as any other pertinent artifacts.)

## 3.2. **Configuration Control Board**

The Configuration Control Board (CCB) will be responsible for the control and approval of changes made to the system in development, PICK. The CCB consists of all members of the Oroware software team, namely: Valente Arellano, Tarek Haddad, Joseph Warren, Mark Williams and Ivan Torres. Only the aforementioned members of the software team will have access to modify any part of the system, without restriction, provided that the correct procedures are followed when doing so. Approval or disapproval of changes will be discussed amongst the CCB and ultimately decided by considering a number of factors regarding the proposed alteration to the system.

Factors to be considered are as follows:

* Sensibility
  + Does the reasoning behind the proposed change make sense?
  + Will the change benefit the system? How?
  + Is the change based on a need?
* Required Effort
  + How long will it take for the team to implement the change?
  + Does the team have the time required to work on integrating the change?
* Scope
  + Is the change within the scope of the project?
  + Will new requirements be introduced to the project by implementing the change?
* Impact
  + Will other aspects of the project be affected by implementing the change? If so, how?
  + What are the possible risks of adopting the change?
* Priority
  + Is the change urgent? Are there more pressing tasks at hand to accomplish?

No changes should be merged into the baseline of the project nor should any changes override the work of another team member without approval being granted. In most cases, the one who proposes a change to the system will be tasked with implementing the change themselves and will need the granted approval of one other member before integration (this process is discussed in further detail in the following subsection - 3.3 Procedures). If the implementer requires assistance with building a change, they will distribute tasks to other members of the team accordingly. When a working version of the change is completed, the code is to be tested and read through by V & V who will formally and thoroughly report errors found in code. Error reports should include details of actions performed by the system that do not produce the expected result. Details on the error should be documented in the most logical and organized manner possible.

The following can be considered a general outline of an error report:

1. Summary of test scenario.
2. Description of the error encountered.
3. Build or version of the system the error was found in.
4. Series of steps to reproduce the error.
5. Actual results produced by the system.
6. Expected results.

## 3.3. P**rocedures**

In this section, we describe the procedures we will use to control changes to the software system being developed. Specifically, we will describe in detail what steps each team member will take to modify configuration items which include the steps necessary to define a new baseline and the approval necessary from the configuration control board throughout any given change process. It is important to note that we will be using established version control tools (essentially just Github for source code and Google Drive for all other configuration items) to aid in software configuration control for this project. We will be using, evaluating, and improving on a software configuration control system we have developed in conjunction with these tools to control changes in configuration items.

Specifically, the following steps will be taken when a configuration item must be changed:

1. A team member will create and fill out a change request form for the configuration item that needs to be changed. The team member must specify at least one approver who is not themselves for the change request. Additionally, the team member must specify at least one assignee for the change request.
2. The team member who created the change request form will upload the change request form to the change request form database
3. The team member will notify the team members who have been assigned the change request form of their assignment
4. The team members who were assigned the change request will accept it and begin working on it in a sandbox environment. If the team members who were assigned the change request do not decide to accept it, they must inform the approver(s) of the change request from why they will not accept it. The approver(s) must decide whether or not the team members must complete the change and inform the team members who were assigned the change request of their decision. If the team members who were assigned the change request do not agree with the decision of the approver(s), they can appeal the decision to the complete configuration control board who will make a final decision on acceptance based on a majority vote.
5. The team members who were assigned the change request will complete the change request as described by the form
6. The team members who were assigned the change request form will change the status of the change request form to complete
7. The team members who were assigned the change request form will inform the approver(s) of the completion of the change request
8. The approver(s) will approve the change request form and change the status of the change request form to approved.
9. The approver(s) will inform the team members assigned the change request that the change request form has been approved. If the approver(s) do not improve the change request form, they must inform the team members assigned to the change request of why the change request was not improved and allow the team members who were assigned the change request to address the concerns and resubmit the change request for approval. If the team members who were assigned the change request do not agree with the approver(s) reasoning, they can appeal the decision to the complete configuration control board who will make a final decision on approval based on majority vote.
10. The team members who were assigned the change request will integrate their changes with the baseline and the baseline will be updated to a new version.

It is important to note that the analyst will be in charge of administering the change request database as well as monitoring the software configuration control process to ensure that all team members are following the described procedures.

# 4. **Software Configuration Auditing**

After implementing a change in a version, we intend to ensure that the change reflects the desired requirements before releasing the new version to the client. To verify that all intended modifications and implementations were made correctly, we will use the following process:

1. Make a list of all requirements affected from the SRS
2. Go through each affected requirement and ensure the system is fulfilling the requirement
3. If the requirement is not fully met, take some time to plan out how other changes can be made to fulfill the requirement
4. Ensure all team members are following the correct processes to complete any approved changes by monitoring the change request database
5. Document any comments regarding areas of the system that may need further inspection and the overall success of the audit

It is important to note that we will also monitor configuration metrics (like average time to make a change, severity of changes, etc.) using the change request database.

**$**