We Showed Up

Prevent, Mitigate, and Recover (PMR) Insight Collective Knowledge System (PICK) Software Configuration Management Plan

Version <1.7>
<2/6/2020>

© 2020 <Enter team name here>

Document Control

Approval

The Guidance Team and the customer shall approve this document.

Document Change Control

Initial Release:	1.0
Current Release:	1.7
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. Gates
- Dr. Salamah
- Dr. Roach
- Elsa Tai Ramirez
- Peter Hanson
- Jake Lasley

Customers:
- Dr. Oscar Perez
- Vincent Fonseca
- Herandy Denisse Vazquez
- Baltazar Santaella
- Florencia Larsen
- Erick De Nava

Software Team Members:
- Ricardo Alvarez
- Daniela Garcia
- Matthew Iglesias
- Jessica Redekop
- Diego Rincon

Change Summary

The following table details changes made between versions of this document

Version	Date	Modifier	Description
1.0	2/4/2020	Matthew Iglesias	Section 3.1: Documentation
1.1	2/5/2020	Jessica Redekop	Section 3: Introduction Section 3.2: Configuration Control Board Section 3.3: Procedures (Intro)
1.2	2/5/2020	Jessica Redekop	Section 1: Introduction
1.3	2/5/2020	Daniela Garcia	Section 2.2: Software Configuration Item

			Organization
1.4	2/5/2020	Daniela Garcia	Section 3.3: Procedures
1.5	2/5/2020	Ricardo Alvarez	Section 2.1: Software Configuration Item Identification Section 4: Software Configuration Auditing
1.6	2/5/2020	Matthew Iglesias	Section 1.1: References
1.7	2/5/2020	Diego Rincon	Revised Section 1-4 Added introduction and re-wrote Section 4

TABLE OF CONTENTS

1. Introduction

This document will describe the Software Configuration Management (SCM) Procedures for the PICK Software Tool. We are creating the SCM plan to follow a uniform set of rules that the Software Configuration Identification, Control, and Auditing that will be referred to throughout the PICK Tool development lifecycle for consistency and ease of modification management. Section 2, Software Configuration Identification, will provide labels for the code baselines and their updates. Section 3, Software Configuration Control, describes the detailed mechanism for preparing, evaluating, and approving or disapproving all change proposals to the PICK configuration items throughout the life cycle. Section 4, Software Configuration Auditing, will describe how auditing will be done for the code pushes, pulls, and testing.

1.1. References

[1] O. Perez et al, Requirements Definition Document, Lethality, Survivability and HSI Directorate, 2019.
[2] “Components and Containers in AWT”. Internet: https://www.cs.utexas.edu/~mitra/csSpring2009/cs313/lectures/GUIComponents.html, 2009 [Jan. 28, 2019]

PEP 8 -- Style Guide for Python Code. (n.d.). Retrieved from https://www.python.org/dev/peps/pep-0008/

2. Software Configuration Identification

This section provides labels for the baselines and their updates.

2.1. Software Configuration Item Identification

The identified elements relevant to the configurations of the PICK Tool software include:

1. Source Code: The source code being developed for the PICK Tool will comply with the following conventions:
    a. Due to the nature of the project, the design pattern to be followed will be the model-view-controller model (MVC).
    b. The naming conventions for the methods' variables will follow the snake-case convention such as in:
        i. process_image_logs(log_path)
    c. The naming convention for the classes will follow the camel-case convention such as in:
        i. class UserInterface(PickUI)
    d. The methods will be documented following the docstring conventions.
2. Software Requirements Specification (SRS): This document serves as the fundamental guideline of the elicited requirements from the clients and will help shape the development of the software in accordance to them.
3. Docstring Derived Documentation: In order to document the Python code being developed, each method will feature documentation featuring the Docstring convention, which will also be exported for quick reference to the team and clients.
4. Test Results: Relevant test results from the configuration documented will be used to inform the team and clients about possible bugs and future steps.
5. The files will be organized in the following folder structure:
    a. /doc: This folder will contain all the files relevant to documentation of the PICK Tool, such as SCM, test results, version control documentation, and method/function documentation.
    b. /src: This folder will contain all the files relevant to the source code and functionality.
    c. The README.md, changelog and the executable file will be in the main folder.
6. Test Plans and Procedures: For testing, the team will implement Travis CI integration with GitHub and take advantage of the included unit testing.
7. Tools:
    The following tools will be relevant to each configuration of the PICK Tool since they interact closely with the functionality to be provided to the clients:
    a. Optical Character Recognition (OCR): Due to the cruciality of the role of this tool in the software for transcribing image logs, the OCR will be tracked as a feature of the configurations.
    b. Transcriber: This tool plays a significant role in the system since it will be used to transcribe the audio and video logs into text.
    c. SPLUNK: Since this tool is used to convert the logs into manageable structures to be gathered from the Splunk API into the system.
    d. Maltego: This tool will be used by the system to create the graph of nodes.

2.2. Software Configuration Item Organization

The labeling scheme for team three will be as follows:

The baseline will be identified by a development branch created from the master branch, labeled "d1_baseline_0.1" where the d1 signifies each modular deliverable, and the 0.1 the current version of the development process. For example, d2_baseline_0.3 is the baseline branch of the second deliverable with three updates. For the directory structure, we intend to have folders for icons, documentation, executable files, and source code, all of these folders will be included in the master branch.

Each team member is responsible for documenting and storing a current version of the project files and upload them to the team project repository. Each team member will be working on their branch, which will be named after the part of the project they are working on and created from the baseline branch. For example, "d1_graph_0.1" where the branch belongs to the team member working on the graph section of the deliverable and is working on the first update for the first deliverable. For the backup of files, the team member in charge of the current deliverable will continuously upload a working version of the repository after each major update, weekly, to ensure that the most current version is available. These procedures will be followed from the beginning of the semester to the submission of the final deliverable marked by version 1.0.

As every team member is responsible for their section of the work, the lead for the current deliverable will enforce the correct use of the team-agreed procedures for making changes to the team database.

3. Software Configuration Control

The following section describes the detailed mechanism for preparing, evaluating, and approving or disapproving all change proposals to the PICK configuration items throughout the life cycle. The purpose of this section is to identify what mechanisms will be used to control access to items in the configuration to prevent unauthorized updates and collisions between team members working on the system simultaneously.

3.1. Documentation

All team members are subject to documenting pull requests, commits, and changes to the software program, this includes providing adequate, detailed information each time a change is made. Additionally, providing each team member with a subsequent branch in which he or she will contribute, will be carefully documented and revised before merging the changes/additions to the master branch. It is the responsibility of each team member to commit their changes/additions made to the software program on the team repository. Change requests are to be descriptively documented by the team member initiating the pull request, ensuring any errors that may arise to be fixed accordingly.

Change requests will be made through a user who seeks to make a change request. The user can add a comment on the line of code he or she wishes, using the (+) button. The comment shall be descriptive, informative, and relevant to the line of code. The user is also responsible for ensuring the change request contains start and delivery dates, priority level, and required approval signatures.

3.2. Configuration Control Board

The Configuration Control Board is an organizational body for formally evaluating and approving or disapproving a proposed change to the PICK software system. Each branch created from the deliverable branch will be assigned to one team member, meaning that each item in a deliverable will be assigned to one team member who will create a branch from the deliverable branch to configure the item. Only the specific team member assigned to the configuration item will have access to the branch before the push into the deliverable branch. Changes will be approved or disapproved thorough Travis CI; once all team members audit the section and accept the merge onto the deliverable branch, the branch will be pushed onto the deliverable branch.

The factors taken into consideration to begin a merge will be described in Section 4. The team will document errors in the code (Section 3.1) found through the process described in Section 4. We will use every team member to accept the change to keep everyone up to date with the code and the changes implemented in it. Once the code is audited, the team will approve the code, and the branch will be merged onto the deliverable branch. Merges to the deliverable branch will be tested through Travis CI unit testing. If any other changes need to be made after the merge, a new branch will be pulled, modified, and pushed with the corrections using the labeling scheme described in Section 2.2. Pull requests for a new deliverable branch from the baseline and a configuration item modification/addition from a deliverable branch will be made through Travis CI, which will advise of any collisions in code.

3.3. Procedures

This section will describe the procedures for controlling changes to the PICK software system. The configuration items will be managed with the continuous integration software tool Travis CI and through manual pull requests on GitHub. The Travis CI tool for managing test procedures will help us avoid merge conflicts. For each deliverable, we will create a branch from the baseline; the branch labeling schemes have been described in Section 2.2 and pull request approvals will control the changes.

The procedures defined in this section must be consistent with the considerations and procedures defined in other sections of this document. In order to maintain consistency in the repository, the lead of the current deliverable will be in charge of making sure the branch creation, commits, and pull requests are made correctly. The lead will create the baseline branches by making sure the master branch is updated, creating the branch from the master branch, and updating the baseline branch using the master branch as the origin.

4. Software Configuration Auditing

The following section will be describing the processes through which the configuration procedures mentioned in Section 3 are tracked, measured, and evaluated to ensure that the same are being followed to comply with the client's requirements.

The primary tool to track SCC will be the SRS, which will be used to validate audits and ensure that the progress made on the deliverables of the project do not deviate from the client's requirements unless explicitly specified by them on later stages of the project's development. Progress on the implementation of features on intended updates will be measured by the state of Travis CI unit tests; features will be fully implemented on updates once Travis CI unit test are completed satisfactorily in their entirety. Finally, reports on updates will be traced back to the pertaining documentation and the executable file in the main repository will be tested to ensure the features on the update reports are included and implemented in the executable.

.

*