We Showed Up

Prevent, Mitigate, and Recover (PMR) Insight Collective Knowledge System (PICK) Software Configuration Management (SCM) 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:	2/23/2020
Date of Next Review:	3/1/2020
Target Date for Next Update:	3/3/2020

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
1.8	2/17/2020	Jessica Redekop	Revised Introduction 1.1: Grammar Revised Section 2.2: Added folder organization for configuration items and removed detritus material
1.9	2/22/2020	Matthew Iglesias	Section 3.1: Revised changes upon Teacher Assistant requests
2.0	2/23/2020	Ricardo Alvarez	Fixed Software Configuration Item Summary
2.1	2/24/2020	Daniela Garcia	Revised section 3.3 and made corrections as specified by the Teaching Assistant
2.2	2/24/2020	Matthew Iglesias	Compiled Team's work into final product

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 process will be referred to throughout the PICK Tool development lifecycle for consistency and ease of modification management. This section provides an introduction and overview to the plan and provides references that will be used throughout the document. Section 2, the Software Configuration Identification, will describe the identification (labels) and organization for the configuration items. Section 3, the 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. It will also describe the naming conventions for version control branches. Section 4, the 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]
[3]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 a description of the configuration items identified to be relevant to the configurations of the PICK Tool project as well as the directory structure to be followed for each of them. The purpose of this is to keep track of the relevant files of the system and provide an organizational plan to maintain consistency between each configuration.

2.1. Software Configuration Item Identification

The software configuration items (CIs) are the items relevant to each configuration of the PICK Tool being developed, those identified as necessary include:

1. Source Code: This refers to the source code developed by team We Showed Up for the PICK Tool.
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 following the Docstring convention which will be also exported for quick reference to the team, clients and future maintenance.
4. Test Plan Document: This document includes the test objectives, procedures, scope, results, schedules, features to be tested, features not to be tested, assumptions and tools to be used for testing.
5. Tools Documentation: Due to the functionality being implemented to the PICK Tool, there shall be documentation derived from the current releases of the tools that the software shall interact with in order to ensure consistence between configurations. The tools being considered include:
   a. Optical Character Recognition (OCR): Due to the cruciality of the role of this tool in the software for transcribing image logs, the OCR shall be tracked as a feature of the configurations.
   b. Transcriber: This tool takes a big 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 in order to create the graph of nodes.
6. SCM: This document will be prone to change throughout configurations if there is a procedure found to be better than the ones expressed at the time of writing.

2.2. Software Configuration Item Organization

The Software Configuration Item Organization will be organized in folders on the version control system, git. We will have the source code directory, namely *src/* where all source code files will reside. As we are modelling our software structure through responsibility-driven design, we will populate the *src/* folder by subdividing the Software Configuration Items into their respective components into respective folders. For example:

```
src/  >
        splunk/
        montalgo/
        popups/
        main.py
```

All other files will reside in the base *src/* directory. As the software development progresses and the SRS is not complete, we reserve the right to manipulate, add, and remove folders deemed, by the team, as necessary to the project structure. Ultimately the structure the Software Configuration Item Organization will be inside of our source code folder in git.

3. Software Configuration Control

This 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 in order 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, using Github's pull requests and manually including the changes in a word document. In addition, 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. This ensures 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 a code he or she wishes, using the (+) button on using Github's pull request. 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 deliverable branch.

The factors taken into consideration to begin a merge will be described in Section 4. 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. This 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 correction 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 done 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 which 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 the changes will be controlled by pull request approvals.

The procedure for making changes will be as follows:

1. Proposed changes will be discussed in person during team meetings and distributed among team members.
2. A change log will be kept with all proposed changes, marking the ones that were accepted and the ones that were denied.
3. Once the change has been approved and assigned the programmer will update the log file with their name on it.
4. The programmer will pull the most recent stable version depending on the change and implement it.
5. After it is implemented, the programmer will upload the changes to the repository in their branch.
6. When the working version is uploaded and tested, the programmer will update the change log to reflect that the change has been completed and where the change is located.
7. Lastly, the programmer will notify another member of the team the changes have been uploaded so they are able to review 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 oversee making sure the branch creation, commits, and pull requests are done correctly. The baseline branches will be created by the lead 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 check that all user interfaces and their components are included in the system, and the correct components and data types are implemented in all classes. The naming convention for the classes will follow the camel-case convention (i.e., *UserInterface(PickUI)*) and the naming conventions for the methods' variables will follow the snake-case convention (i.e., *process_image_logs(log_path)*). Finally, the SRS will be used as a guideline to test the behavior of each UI component in the system to ensure that the system is functional in accordance with the clients' specifications and requirements.

Audits will be carried out twice a week, with the first audit covering the user interfaces and class inspection, and the second audit testing the behavior of the software to ensure it complies with the specifications of section 3.2.3 (Stimulus) of the SRS. Once all elements in the audits are addressed, they will be validated, which will ensure 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 updates will be measured by the state of TravisCI unit tests. Once critical Travis CI unit tests focused on the functionality of previous and current features are completed satisfactorily, and in their entirety, the current features will be fully implemented in the updates.

Reports on updates will be traced back to the SRS and other documents specifying the client's requirements, such as recordings and notes taken during demos to ensure the features being implemented in the software comply with the client's specifications. The executable file in the main repository will be undergoing black box and sanity testing to ensure, from the perspective of the end-user, that the features on the update reports are included and implemented in the main executable file, as well as considering the basic usability and accessibility of the application.

*