Spice Girls

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

Version 1.5

February 21, /2020

Document Control

Approval

The Guidance Team and the customer shall approve this document.

Document Change Control

Initial Release:	1.0
Current Release:	1.5
Indicator of Last Page in Document:	$
Date of Last Review:	02/21/2020
Date of Next Review:	02/25/2020
Target Date for Next Update:	02/29/2020

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
Jake Lasley

Customer:

Mr. Vincent Fonseca
Mr. Baltazar Santaella
Ms. Herandy Vasquez
Ms. Florencia Larsen
Dr. Oscar Perez
Mr. Erick De Nava

Software Team Members:

Ana Zepada
Dima AbdelJaber
Ricardo Sanchez
Luis Ochoa
Scott Honaker

Change Summary

The following table details changes made between versions of this document

Version	Date	Modifier	Description
1.0	01/31/2020	Spice Girls	Document Creation
1.1	01/31/2020	Ana & Dima	Completed section 1 & 2
1.2	01/31/2020	Ricardo & Luis	Completed section 3

1.3	02/01/2020	Scott	Completed section 4
1.4	02/03/2020	Dima	Fixed formatting
1.5	02/21/2020	Ricardo	Feedback corrections

Table of Contents

1. Introduction

The purpose of this project is to design software that will be used by the Army Combat and Capabilities Development Command - Data Analysis Center. This center has analysts that currently must manually open hundreds of files before determining if they are valid or have information relevant to the current project. The software will be designed to automate the search through files for dates, times, and keywords and consequently reduce the time analysts spend on searching files. The software configuration management plan will detail the configuration of a system and outline the way changes will be controlled and the way in which the integrity and traceability will be maintained [1]. The intended audience is primarily the software team itself, who will use this document in order to guide changes to the system.

The second section of this document will discuss the configuration of the system and the way in which some of those components are named and organized.

The third section of this document will discuss the way in which access to

1.1. References

[1] S. Roach. Class Lecture, Topic: “Software Configuration Management (SCM)”. College of Engineering, The University of Texas at El Paso, El Paso, TX, Jan., 2020.

2. Software Configuration Identification

This section will discuss the configuration of the system as well as the naming convention for the software components.

2.1. Software Configuration Item Identification

The configuration will be made up of the following items:

1. Graphical user interface source code
2. Python source code
3. Use case diagram
4. Test suite
5. Requirements Reference Document
6. User guides

2.2. Software Configuration Item Organization

Each baseline shall be named as follows: /PICK v<version>.0. Updates to the baseline shall be named as follows: /PICK v<version>.<update>, where the update number starts at one and increases by one for each update. Within the main folder is stored the README file and the doc, src, resources, and ui folders. The doc folder will contain the documentation files, such as the SCM Plan, the CRC, SDD, Test Plan, etc. The src folder will contain the source code, python scripts of the project. The resources will contain miscellaneous files such as images, text files, etc. related to the project. The ui folder will contain the user interface files of the project, provided by Qt designer and with a *.ui extension.

The project source code will be stored on Github. Team members will work using different computers and will make a branch off of the most recent update to work on their local computer. These branches will be named as follows: /PICK v<version>.<update> <first initial><middle initial><last initial> <change>. Once each branch has been tested individually, merging will be completed by the entire team.

With the use of GitHub, each team member will have a copy of a recent version of the program on their local system. Changes should be uploaded to GitHub on a nightly basis after completing any work on the source code. GitHub is hosted on AWS servers which are considered extremely reliable, so no further backup is deemed necessary. In the unlikely case that one team member's local computer loses its data, the team member is responsible for pulling the most recent push on their own branch of the most recent update.

3. Software Configuration Control

This section will discuss the ways in which changes are prepared, evaluated, and approved. It will also discuss the mechanisms for controlling access so that collisions and unauthorized updates may be kept to a minimum despite simultaneous work being done.

3.1. Documentation

The software team will fill out a simple form on google docs in the folder “Change Documents” to answer the following questions:

1. Who is requesting the change?
2. What is the change which is being requested?
3. Why is this change important? What is the priority on this project?
4. How long (in hours) is this change anticipated to take?

3.2. Configuration Control Board

All changes will be discussed by the entire software team. Ultimately the decision to approve a change will be taken by the implementer and the lead for that particular deliverable.. Access to modifying different parts of the system will be delegated by the entire team. The most important factors when approving or disapproving a change are priority and the number of hours required to make the change (cost). V&V will conduct tests and upon finding an error will always report to the software team to discuss the possibility of approving a change.

3.3. Procedures

Suggested changes to the system can be initiated either by the client or the software team. In either case, after an informal discussion, the software team will send the clients a memo with concerns. A google sheet will also be updated to reflect the changes. The sheet will contain five columns labelled “baseline”, “change”, “approved by”, “modifications”, and “members responsible”. The entire team will have access to this sheet, and it is expected that the person in charge of completing the changes add their information to the next row.

Team members will have access to checkout and modify the items to which they have been assigned by the entire team. If throughout the process some modifications occur to the exact nature of the change, the team members responsible shall document it in the appropriate column after discussing the need for the changes with the entire team.

After answering the documentation questions outline above, the software team will discuss what team member(s) will be in charge of implementing the change and when the appropriate start and end dates for the modification would be, taking into special consideration the duration of the change and the priority of the change.

While completing the changes, if the actual time to complete exceeds the calculated time, the team will discuss the priority of the changes again and possibly agree upon further scheduling. Changes will be made on branches named as follows: /PICK v<version>.<update> <first initial><middle initial><last initial> <change>. If there exist modifications on the current baseline and the change is not urgent, all changes will be completed before pushing them as the next update. If other modifications exist but the modification is urgent, upon completion of the urgent change, a new baseline will be created; all the current branches should merge with the changes made

and should be renamed as branches of the new update. Otherwise, the completed change will be pushed immediately as the next update.

New baselines shall be created during the semester for each of the deliverables described by the class syllabus. New baselines will be created if certain changes are deemed urgent during the change approval process.

4. Software Configuration Auditing

Upon the release of each baseline and before the due date for each deliverable, the team will meet to discuss the completion of the objectives outlined when approving a change. This will require the review of the initial change documentation along with the change table which may reveal modifications to the original objective. The team will design integration, performance, and acceptance tests, in order to test the degree to which the changes meet the desired outcome. These tests, along with whether they are passed or failed shall be saved in a file named PICK v<version>.<update> tests within the team's github page.

$