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

Software Configuration Management Plan

Version <1.0>

2/24/2020

Document Control

Approval

The Guidance Team and the customer shall approve this document.

Document Change Control

|  |  |
| --- | --- |
| Initial Release: | 0.1 |
| Current Release: | 1.0 |
| Indicator of Last Page in Document: | ♠ |
| Date of Last Review: | 2/24/2020 |
| Date of Next Review: | 2/28/2020 |
| Target Date for Next Update: | TBD |

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:

Steven Roach

Elsa Tai Ramirez

Customer: Dr. Oscar Perez

Vincent Fonseca

Herandy Denisse Vazquez

Baltazar Santaella

Florencia Larsen

Erick De Nava

Software Team Members:

Jose Antoine Leon Cordero

Elizabeth Barragan

Manuel Delgado

Yamel Hernandez

Abel Rodriguez

Change Summary

The following table details changes made between versions of this document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modifier | Description |
| 0.1 | 2/5/2020 | Abel Rodriguez | Create SCM draft document |
| 0.2 | 2/6/2020 | Abel Rodriguez | Finalize what I can and document the critical error of trying to create this on my own |
|  |  |  |  |
| 0.2 | 2/14/20 | Yamel Hernandez | Worked on Sec 3.1. |
| 0.3 | 2/14/20 | Elizabeth Barragan | Worked on Sec 3.2. and Finalized, started working on Sec 3.3. |
| 0.3 | 2/14/20 | Abel Rodriguez | Editing and finishing up Section 1 and 2 |
| 0.3 | 2/14/20 | Yamel Hernandez | Worked on section 3.1 and created figure 3.1.1 |
| 0.3 | 2/14/20 | Manuel Delgado | Worked on Section 4 |
| 0.4 | 2/15/20 | Yamel Hernandez | Finalized section 3.1, worked on intro for sec 3 |
| 0.4 | 2/15/20 | Abel Rodriguez | Contribute to Section 1 and 2 |
| 0.5 | 2/22/20 | Abel Rodriguez, Elizabeth Barragan, Manuel Delgado, Jose Leon Cordero | Reviewed and Finalized Sections 1, 2 |
| 0.6 | 2/23/20 | Abel Rodriguez, Manuel Delgado, Jose Leon Cordero | Reviewed Sections 3, 4 |
| 0.7 | 2/24/20 | Elizabeth Barragan | Reviewed Sec. 3.3, 3.4, 4. Added missing content in Sec 3.2 |
| 0.8 | 2/24/20 | Yamel Hernandez | Reviewed Sec 3.3,3.2, & 4. |
| 0.9 | 2/24/20 | Abel Rodriguez | Finish 3.3 and review section 4 |
| 1.0 | 2/24/20 | TEAM 5  Jose Leon Cordero, Elizabeth Barragan, Yamel Hernandez, Manuel Delgado, Abel Rodriguez | Finalized SCM 1.0 |

s

Table of Contents

[Document Control ii](#_Toc33472423)

[Approval ii](#_Toc33472424)

[Document Change Control ii](#_Toc33472425)

[Distribution List ii](#_Toc33472426)

[Change Summary ii](#_Toc33472427)

[1. Introduction 1](#_Toc33472428)

[1.1. References 1](#_Toc33472429)

[2. Software Configuration Identification 2](#_Toc33472430)

[2.1. Software Configuration Item Identification 2](#_Toc33472431)

[2.2. Software Configuration Item Organization 2](#_Toc33472432)

[2.2.1. Labeling Schemes 2](#_Toc33472433)

[2.2.2. Project Database 2](#_Toc33472434)

[2.2.3. Directory Structure 3](#_Toc33472435)

[2.2.4. Backup and Recovery Plan 3](#_Toc33472436)

[3. Software Configuration Control 4](#_Toc33472437)

[3.1. Documentation 4](#_Toc33472438)

[3.1.1. Change Proposal Document 4](#_Toc33472439)

[3.1.2. Change Proposal Procedure 4](#_Toc33472440)

[3.2. Configuration Control Board 5](#_Toc33472441)

[3.3. Procedures 5](#_Toc33472442)

[3.3.1. Summary of Change Implementation 5](#_Toc33472443)

[3.3.2. Detailed Change Implementation 6](#_Toc33472444)

[4. Software Configuration Audit 7](#_Toc33472445)

# Introduction

The project concerning this document is the PICK system. Our clients, the Lethality, Survivability, and HSI Directorate (LSH), conduct analyses of cyber-attacks on their systems. The information gathered during an analysis has to be sorted, correlated, and interpreted – a task that can take weeks. Thus, the main purpose of the PICK system is to aid the endeavors of the LSH to conduct their analyses with the goal of reducing the time it takes to conduct analyses.

The purpose of the Software Configuration Management (SCM) plan is to identify configuration items, establish a development process, and create an auditing process to check for compliance with the development process. In Section 2, we identify and organize software configuration items. The organization is done in terms of versioning, directory structure, and recovery procedures in the case of development errors. Section 3 focuses on establishing a procedure to enact changes to the system. This procedure consists of documentation of changes, a control mechanism to justify changes, and the development process to carry out proposed changes. Finally, section 4 defines an auditing process to check actual project progress against the software configuration control process outlined in section 3.

## References

* 03SCM\_Assignment.pdf which contained a set of questions and due dates, was used as a basis to include information pertaining to the questions found in this pdf file in order to mention key items within the SCM.
* 03SCM.pdf, which contained the lecture material that discussed each section of the SCM in detail, was used to assist the team in understanding how to structure and create an SCM for our team’s system and break it down in section 2, 3, and 4 of the SCM.

# Software Configuration Identification

This section identifies system configuration items and their organizational structure.

## Software Configuration Item Identification

This section identifies configuration items. A configuration item is an abstraction of one or more project components outlined for the purpose of tracking project development. Configuration items are subject to change according to future client feedback and developments in the system. The following is a list of configuration items for the PICK system:

1. Documentation
   1. User Guide
   2. READ-ME
   3. SCM
   4. Change List
2. Source Code (SRC)
3. Tests

## Software Configuration Item Organization

This section outlines labeling and documentation schemes for project versions, identifies where the project is stored and its directory structure, and defines a recovery plan for project files.

### Labeling Schemes

The labeling scheme our team will apply to each baseline will be N.x, where N.0 indicates the current baseline, and x indicates a component update (e.g. 5.2, where .2 is the second component update in the 5th baseline). The name that will follow each version release is based on overall system functionality (e.g. Executable GUI), whereas the name that will follow each component release is based on the component’s functionality (e.g. Splunk integration).

### Project Database

The project will be stored on GitHub in the classroom repository named “CS4311-spring-2020/pick-tool-team05-static-variables”. The use of GitHub allows for remote, simultaneous work. Moreover, GitHub employs a versioning system which can be used to implement versioning as described in section 2.2.1.

### Directory Structure

The directory structure for the project is per configuration item. That is, each configuration item will sit at root directory and will include sub-directories for its components. An exception to this hierarchy is the READ-ME file, which will also sit at root for ease of access to the user. The directory structure is as follows:

1. Documentation
   1. User Guide
   2. SCM
   3. Change List
2. Source Code
   1. Frontend
   2. Backend
3. Tests
   1. Frontend
   2. Backend
4. READ-ME

### Backup and Recovery Plan

Backup of the project will consist of storing the latest project baseline on a secondary database upon baseline release. This separate database will be hosted in a private OneDrive directory, which is managed by all members of the team.

In the case of loss of access or files, or human error in interaction with the main database described in section 2.2.2, this secondary database will provide us with the latest baseline to use as recovery. Upon exercising a transfer of files from the secondary database into the primary database, the recovery process will incorporate a review of the configuration control board as described in section 3.2. Additionally, the recovery process will include documentation of errors that led to exercising backup and recovery procedures, as well as a potential revision of change procedures to avoid future error.

# Software Configuration Control

This section will detail the process for documenting, preparing, and executing changes to the project.

## Documentation

The documentation of software configuration control will outline a format to formally propose changes in the system. Additionally, this section will define the process to document any approved change to a configuration.

### Change Proposal Document

The change proposal document will consist of a .CSV file with the following fields:

* Actual end date of change
* Actual start date of change
* Date
* Description of change
* Expected end date of change
* Expected start date of change
* Feasibility level of change
* Individual in charge of executing change
* Individual requesting change
* Initials of individuals who approved a change
* Justification of change
* Priority level of change
* Problem domain of change

Figure 1 depicts a version of the change proposal document. This document will be stored in both databases. In the primary database, it will be updated per-change. In the secondary database, it will be updated per baseline release.

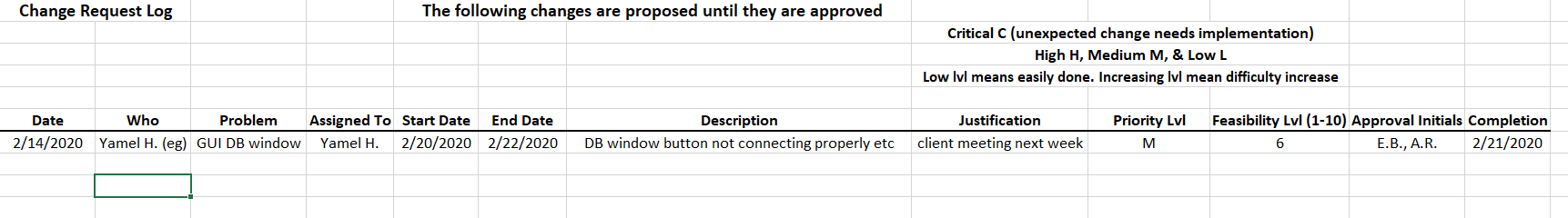


Figure 1: Excel Changes Document

### Change Proposal Procedure

Changes must be first proposed during team meetings and must have a majority rule, with at least three members approving of the change. Once the change is approved by three members, the change will be added to the change document for official documentation. In the change document, the proposer of the change must specify the problem they want to address and justification for the change. The team will then determine the time frame to work on the change, who is assigned to work on the change, and priority level. The priority level must be specified as “Critical” when an unexpected change is relevant to the entire entity of the system, “High” means necessary change to complete a baseline, “Medium” means necessary changes to complete a component, “Low” means fixing defects or improving minor aspects of a component. Once the entry is approved of, the assigned member of the task must fill in when the task was completed.

## Configuration Control Board

This section identifies the organizational body for formally evaluating and approving or disapproving a proposed change to a system. The Configuration Control Board (CCB) is composed of the V&V, System Analyst, Lead Programmer, Designer, and System Architect.

To ensure that changes are carried out correctly, the CCB will assign the following responsibilities and limitations with regards to modifying the system:

* All members of the CCB, except for the person requesting a change, will determine its approval or denial
* The following are limitations on modifying the system:
  + A contributor who is assigned a change can only work on that change until its completion

When a change is proposed, the change will be discussed and handled by the CCB in a face-to-face meeting. In this meeting, the CCB will use the following questions to approve or deny a change:

* Is the change feasible?
* Is the change necessary?
* Is there enough time to implement the change?
* What is the priority level of the change?
* When will the change be released?

Once a change has been approved by ¾ of the CCB committee, the distribution of changes will be considered using the following factors:

* Change complexity
* Feasibility
* Time constraint

Once a change has been assigned, the CCB will determine any extra responsibilities with regards to managing the change.

The V&V will report errors of a change in an error-tracking document. Each error will be assigned an ID for traceability and be mapped to the line number and file where it is found. The individual(s) responsible for a change are also responsible for refactoring the code to handle errors.

## Procedures

This section outlines procedures to implement approved changes and update the primary and secondary project repositories.

### Summary of Change Implementation

At every meeting, team members shall dedicate 15 minutes to propose changes to the system. Upon presentation, the CCB will adhere to the discussion questions outlined in section 3.2 to approve and assign a change. After a change has been assigned, the individual(s) assigned will fork a branch for the change from its parent development branch. In this branch, only work for that specific change can be carried out.

Once new changes are added to the change list and assigned, team members shall review existing changes in the change list. This review serves as an audit in which the progress of changes is compared against its impact – a process that stands to verify or update its expected finish time.

For changes that are completed, the individual(s) responsible for the change will test compatibility of the change before the meeting. Testing is done by merging the parent branch into the change’s branch and implementing any relevant testing code. Once testing is completed, the individual(s) responsible for the change shall present testing results during the change review. If tests do not pass, the V&V shall conduct an error analysis of the change per the procedure outlined in section 3.2. Depending on the magnitude of error, the V&V shall propose an extension to the expected delivery time of the change. If results are sound, the CCB shall approve the lead programmer to merge the change branch into the change’s parent development branch.

Once merged, the team shall decide whether the priority level of a change or set of changes in the current development branch justifies their release as a component update or complete baseline. When decided, documentation will be updated by CCB to formally account for the change. Additionally, the secondary database will be updated with the current copy of the project repository post merging of a change.

### Detailed Change Implementation

1. Propose Changes
   1. Present argument/reasoning for change
   2. CCB approve or disapprove change based on results of leading questions outlined in section 3.2
   3. When approved, the change will be added to the change list document per the outline in section 3.1
   4. The assigned member(s) will fork a change branch from the development branch and identify it based on the current baseline and component the change is associated to
2. Testing Changes
   1. The responsible individual(s) shall perform a git pull before any testing
      1. Any merge conflicts will need to be resolved by responsible individuals
   2. The responsible individual(s) shall merge the development branch into the change branch
   3. Tests for performance, style, and logic shall be created in the testing directory
   4. Test results shall be recorded into a document once finalized
3. Review Changes
   1. Pending changes in the change list shall be reviewed
      1. If a change is staged for completion and testing has been conducted, results will be verified by the CCB.
         1. If a change is verified, the procedure in Step 4 shall be conducted.
         2. If a change is not verified, the V&V will conduct an error analysis per the process outlined in section 3.2
      2. If a change is not staged to completion, the individual(s) responsible for the change shall update the CCB on progress.
   2. The CCB shall decide whether the priority level of merged change(s) justifies the release of a component or baseline
      1. If a component or baseline is approved, the procedure in Step 5 shall be conducted.
4. Implementing Changes
   1. The CCB shall approve the lead programmer to merge the change into the development branch
      1. The lead programmer shall merge files
      2. The lead programmer shall resolve any merge conflicts
   2. Formal documentation of change(s) shall be conducted by the CCB
5. Component or Baseline Update
   1. The CCB shall review all configuration items’ completion per the intended state of the system
   2. The lead programmer shall merge the development branch into the master branch
   3. The lead programmer shall update the secondary repository with the new component or release

# Software Configuration Audit

This section defines the Software Configuration Audit process. The Software Configuration Audit process includes conducting the audit of completed components of the baseline in question. This audit reflects the degree of compliance to which the current configuration mirrors the intended configuration.

The audit will check every accepted change against a specific requirement in the SRS. If a change cannot be mapped to a specific requirement, it will be checked against other changes to be justified. If at any point there is a change that does not map to anything, it will be counted as a deviation from the intended configuration. Additionally, if a change is not completed, that change will also be counted as a deviation from the intended configuration. At the end of the audit, the deviation ratio will be computed to determine the degree to which the current configuration reflects the intended.

If the deviation ratio is lower than 80%, the CCB will refactor pending changes in terms of priority and expected completion dates. Changes that cannot be justified shall be factored out of the change list.

♠