ZombieBytes

El Paso MPO Management System
Software Configuration Management Plan
Version <2.0>
2/8/2016

Document Control

Approval

The Guidance Team and the customer shall approve this document.

Document Change Control

Initial Release:	1/28/2016 V0.1
Current Release:	2/8/2016 V2.0
Indicator of Last Page in Document:	^-^
Date of Last Review:	2/8/2016
Date of Next Review:	3/8//2016
Target Date for Next Update:	3/8/2016

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. Salamah Salamah
Andres Olivas

Customer:
Mr. Roger Williams

Software Team Members:
Camel Heydarian
Christian Hughes
Chelsey Jurado
Jordan Cox

Change Summary

The following table details changes made between versions of this document

Version	Date	Modifier	Description
0.1	1/28/2016	ZombieBytes	Initial rough draft of the introduction.
0.2	1/29/2016	Chelsey Jurado& Carmel Heydarian	Completed and edit the introduction
0.3	1/30/2016	ZombieBytes	Initial rough draft of section 2.1
0.4	1/30/2016	Jordan Cox	Initial rough draft of section 3.2
0.5	1/30/2016	ZombieBytes	Complete and edit section 2.1
0.6	1/31/2016	Chelsey Jurado & Carmel Heydarian	Initial rough draft of section 2.2
0.7	1/31/2016	ZombieBytes	Complete and edit section 2.2
0.8	1/31/2016	Christian Hughes & Jordan Cox	Initial rough draft of section 3.1 and 3.3
0.9	1/31/2016	ZombieBytes	Complete and edit section 3.1 and 3.3
0.10	1/31/2016	ZombieBytes	Initial rough draft of section 4
0.11	1/31/2016	ZombieBytes	Complete and edit section 4
1.0	2/1/2016	ZombieBytes	Finalize the document for submission.
1.1	2/6/2016	ZombieBytes	Made revisions suggested by Dr. Salamah

1.2	2/7/2016	ZombieBytes	Review the changes made and created Auditing Form.
2.0	2/8/2016	ZombieBytes	Reviewed all changes made once more and approved document for final draft.

TABLE OF CONTENTS

1. Introduction

1.1. Project Description

The system we are developing is for the El Paso Metropolitan Planning Organization (MPO). The El Paso MPO currently uses Microsoft Access and Microsoft Excel to manage projects, create projects, and generate reports. The El Paso MPO Management System will be web-based. The system will improve the current process by having one workspace in which the users can create, update, and retrieve information on various projects and reports.

1.2. Purpose and Intended Audience

The purpose of the Software Configuration Management (SCM) plan, is to provide documentation that dictates firm guidelines that will be followed when making changes to artifacts. Having guidelines will uphold consistency when managing versioning control, providing backup, and documenting changes. The SCM will contain guidelines for establishing naming conventions and managing changes in the project. The SCM must be robust so that the changes and updates made to the system are consistent and the project is managed in an organized fashion. The intended audience for this document is the software development team, guidance team, and the representatives from the El Paso MPO.

1.3. Overview

This document will contain:

Software Configuration Item Identification:	This section identifies a list of elements from the project that are likely to change.
Software Configuration Item Organization:	This section identifies a method for managing versions via a version control policy, and the organization of the configuration items.
Documentation:	This section provides documentation for formal change requests.
Configuration Control Board:	This section provides information regarding the Software Configuration Board that will review proposed changes.
Procedure:	This section describes the process of managing items and explains the process for making changes to the various software configuration items.
Auditing:	This section serves as a tool for validating and verifying changes against the baseline version.

1.4. References

Preston-Werner, T. (n.d.). *Semantic Versioning 2.0.0*. Retrieved January 29, 2016, from http://semver.org/

Research National Institute of Dental and Craniofacial. (2014, March 6). *Version Control Guidelines*. Retrieved Janurary 30, 2016, from http://www.nidcr.nih.gov/Research/ToolsforResearchers/Toolkit/VersionControlGuidelines.htm

Ritchie, S. D. (2012, March 5). *Build Numbering and Versioning*. Retrieved January 30, 2016, from http://ruthlesslyhelpful.net/2012/03/05/build-numbering-and-versioning/

2. Software Configuration Identification

In this section we have identified elements and documents that are part of developing the El Paso MPO management system that are likely to change. We have also described the structure of the repository that we will use for version control. Additionally, we have described the naming conventions we will be using to organize updates to our documents and source code.

2.1. Software Configuration Item Identification

The focus of this document is managing change. In order to create a systematic process to manage and implement changes, we must first identify what items are most likely to change throughout the development process. The items that we have identified that will change are the following:

- Software Requirements Specification (SRS)
- Software Configuration Management (SCM) Plan
- Source Code
- Test suites
- Software Design Specification

2.1.1. Software Requirements Specification

The SRS contains all the requirements required to create the system. Any time a change is made to the requirements the SRS must be updated to reflect these changes.

2.1.2. Software Configuration Management

The SCM document contains the process for maintaining control of the software configuration items. The process may change during the life cycle of this project, therefore this document must be updated to correspond to the changes made.

2.1.3. Source Code

The source code is the implementation of the system. If there is a change in the requirements the source code is likely to change. The source code will also require change when an error is detected.

2.1.4. Test Suite

Test suites are meant to test the system to make sure it adheres to the requirements. Therefore any time a change is made to a part of the system, the test suites must be updated to reflect the changes made.

2.1.5. Software Design Specification

This document will outline the overall design of the MPO system. This document is being included because the design of the MPO system will be subject to change. This means that the changes must be reflected in the SCM document.

2.2. Software Configuration Item Organization

This section will specify how the items in our system are to be organized. The structure will be easily searchable and will include all items created as part of the development process.

2.2.1. Directory Structure and Naming Conventions

The directory structure is shown in Figure 1 below.

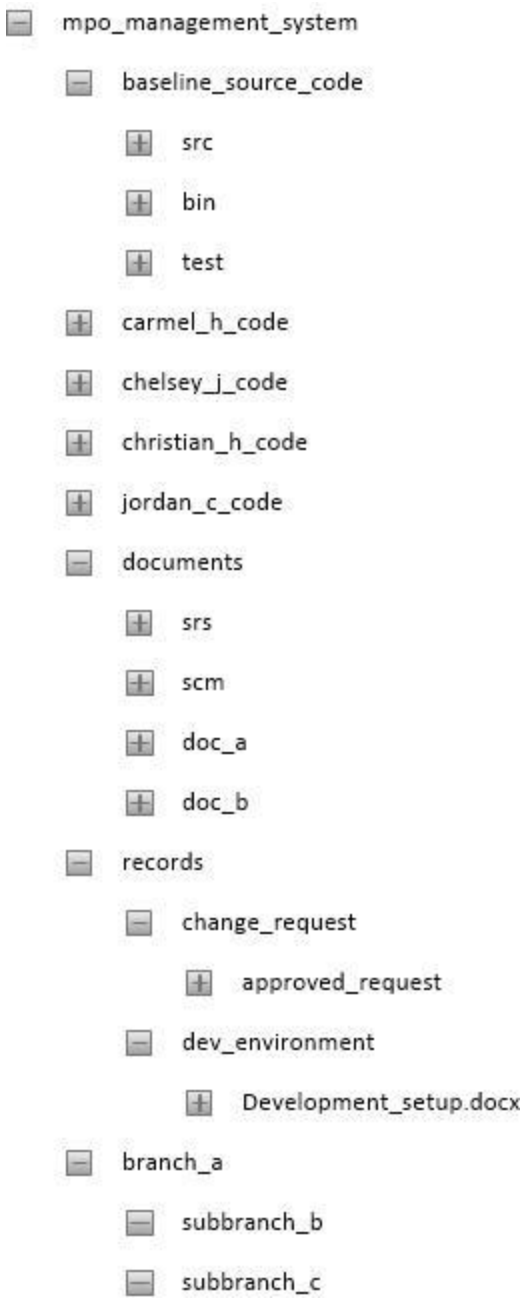


Figure 1. Directory structure of the system repository.

Naming conventions are important to maintain organization. It will allow for quick access to documents associated with the project. The naming convention that the software team has decided on is as follows:

ZombieBytes_<Doc Name>_<Version>.<ext>

The convention can be broken down like so: the name of the software team followed by the document name followed by the version and finally the file extension. Each of these elements is separated by an underscore.

2.2.2. Version Numbers

In order to maintain item versions, we created version number schema. For source code the software team has decided on the following format::

X.Y.Z

X represents a major change in the system and is the main version number. A major change is a new feature that alters the system so that it is no longer backwards compatible with the previous release. Y represents a minor change in the system. A minor change is the addition of a new feature, this feature adds functionality to the previous release. Z represents revisions made to the code and bug fixes.

For documents the software team has decided on the following format:

X.Y

X will represent when a document reaches its final version, e.g. the final draft (1.0), final paper (2.0). Y will represent any changes made such as adding/removing/editing sections, rewording, grammatical errors, etc.

2.2.3. Version Control System

The software team will be using Git for the primary version control system, and will be using GitHub to publish the Git repositories online. This will allow all members of the software team to have a working copy of the project on their system at all times. Members will be required to update their repository before working on the project to ensure that everyone is working on the latest version of the project. The project, being located on all team members' systems, and on GitHub, will serve as the main form of backup. Furthermore, all files in the repository will be backed up on a weekly basis to an external hard drive by the teams lead programmer.

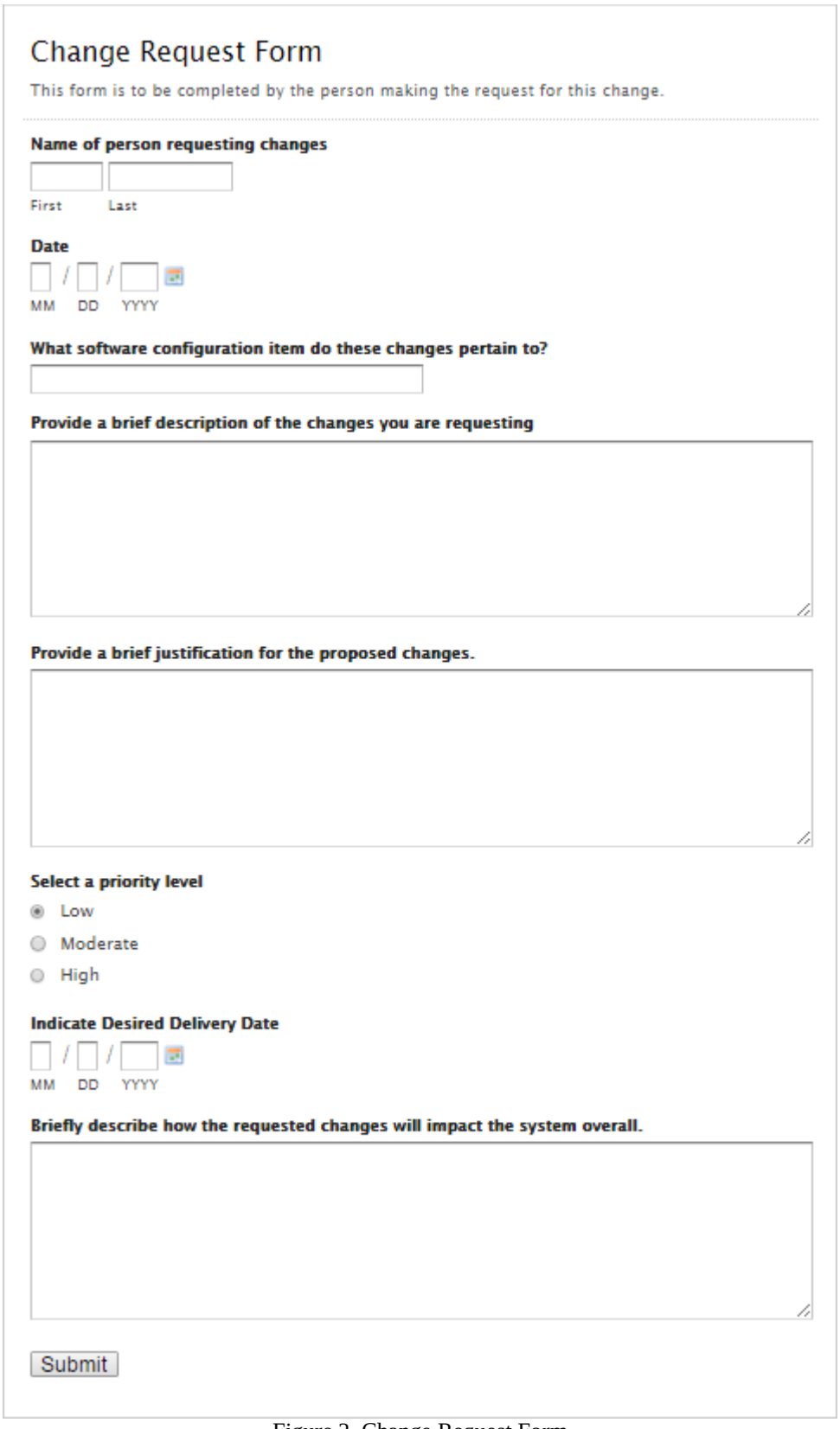
3. Software Configuration Control

This section contains the change request forms a team member must submit in order to notify the other team members that a change is needed. This section contains the change approval form that the team will need to approve in order for the change to be implemented. This section also establishes the structure of the configuration control board. Lastly, this section also provides the procedures the team must adhere to while making changes to the software system.

3.1. Documentation

The software development team needs to maintain a record of changes made to the system. Two forms were created to assist in this process. The first form required in this process is the Change Request Form (https://purpledinosaurs.wufoo.com/forms/ztgvt1715brohk/).

The Change Request Form must be completed by the person requesting a change to initiate the process of making changes to any of the SCM items. This form and the information that must be submitted can be found below in figure 2. The first form (Change Request Form) asks for a priority level which consists of 3 different classifications - low, moderate, and high. These priority levels describe the level of importance of the changes being proposed. The highest level of priority suggests that the changes being proposed are of a critical nature. For example, The high level of priority indicates that the change needs to be made immediately. The change will need to be made before progressing further on the document, or system. The moderate level of priority indicates a change that would affect multiple sections of a document or the system, but does not require immediate attention because it does not affect the entire system or document. The low level priority indicates a change that will not affect significant parts of the system. This change can even be a suggestion and may not be a required modification.



Change Request Form

This form is to be completed by the person making the request for this change.

Name of person requesting changes

First Last

Date

MM / DD / YYYY

What software configuration item do these changes pertain to?

Provide a brief description of the changes you are requesting

Provide a brief justification for the proposed changes.

Select a priority level

- Low
- Moderate
- High

Indicate Desired Delivery Date

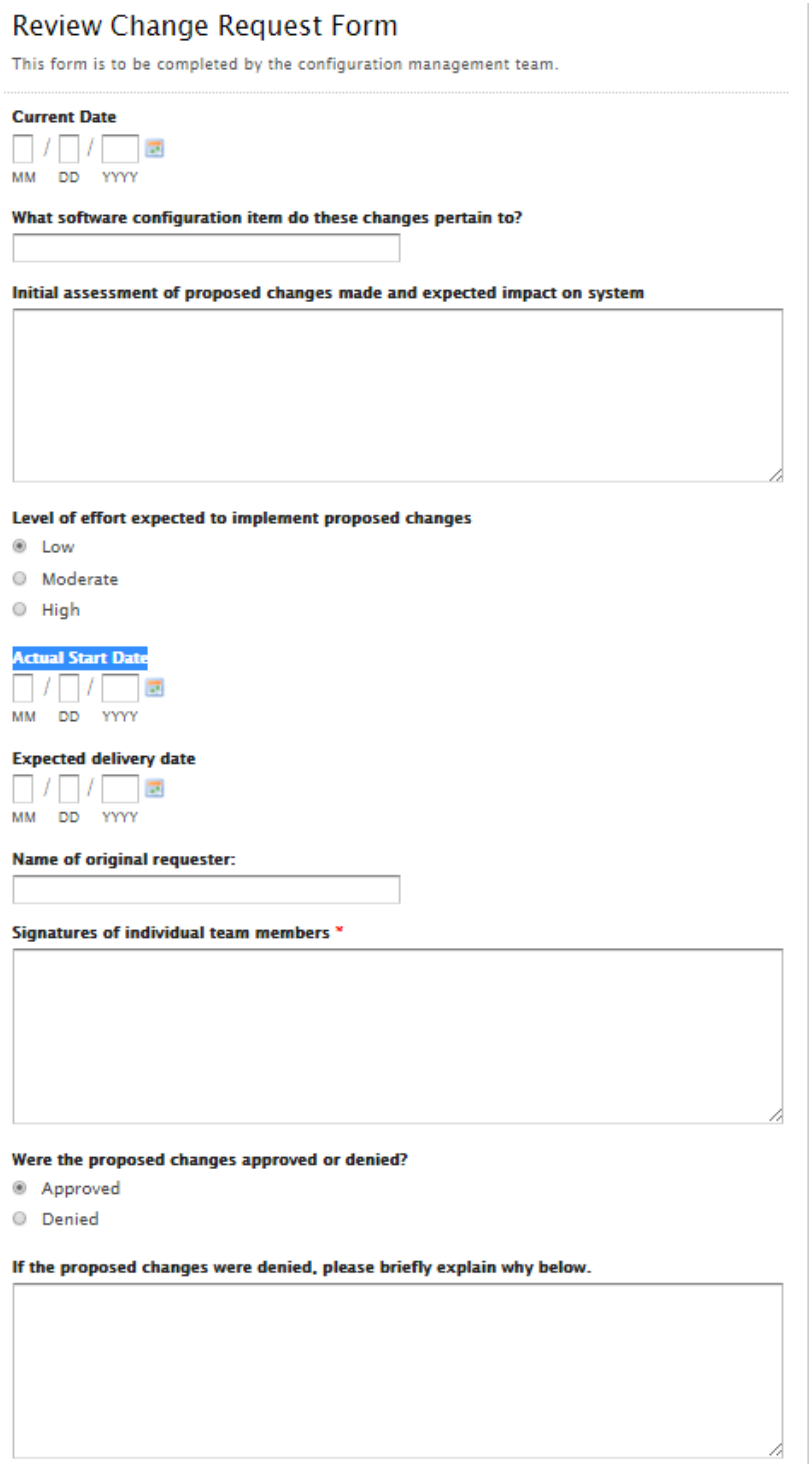
MM / DD / YYYY

Briefly describe how the requested changes will impact the system overall.

Submit

Figure 2. Change Request Form

The second form is the Review Change Request Form (https://purpledinosaurs.wufoo.com/forms/m1qyd3va0itu35u/) filled out by the Configuration Control Board when reviewing the original change request. It is in this form that the original request is either approved or disapproved. This form and the information that must be submitted can be found below in figure 3.



Review Change Request Form

This form is to be completed by the configuration management team.

Current Date

MM DD YYYY

What software configuration item do these changes pertain to?

Initial assessment of proposed changes made and expected impact on system

Level of effort expected to implement proposed changes

- Low
- Moderate
- High

Actual Start Date

MM DD YYYY

Expected delivery date

MM DD YYYY

Name of original requester:

Signatures of individual team members *

Were the proposed changes approved or denied?

- Approved
- Denied

If the proposed changes were denied, please briefly explain why below.

Figure 3. Review Change Request Form

Once both forms are filled out, they will be entered into an Excel document that will serve as the Change Log. The links to the Change Request Form and the Review Change Request Form will be found in the repository, along with the Change Log.

3.2. Configuration Control Board

The Configuration Control Board will be made up of all the members of the software team. The team is made up of: Carmel Heydarian the Systems Analyst , Chelsey Jurado the Lead Programmer , Jordan Cox the V&V, and Christian Hughes the Designer. All the members of this team will be responsible for evaluating and approving or disapproving a proposed change to the system. All the team members will have access to modifying different parts of the system. Team members will only be able to modify any part of the system after the change is approved. Chelsey Jurado will be responsible for distributing the changes. Approving or disapproving changes will be decided with a vote from all the members of the team. In the event of a tie, the project lead for the software configuration item that is to be changed will have their vote count as two votes instead of one. An example of this would be: "The team has voted to change something in the source code. The lead programmer's vote will count for two votes instead of one." Changes will be approved using the documentation provided in section 3.1 of this document. The factors that will be evaluated to approve or disapprove a change will also be included in the document provided in section 3.1. Errors in the code will be reported by the V&V through the change request form indicating the file that is to be changed, the line number where the change will take place, and a description of the change that will take place. The artifacts that are changed will be placed in a version directory, and a previous version of the artifact will also be saved as a baseline.

3.3. Procedures

This section provides an overview for the various procedures involving managing changes to the Software Configuration Items.

3.3.1. Procedure for Checking-out Artifacts

If a team member wishes to checkout an artifact from the repository, they will first need to fill out and submit a Change Request Form to the Configuration Control Board. The Configuration Control Board will then review the changes requested and fill out a Review Change Request Form. Once the review is done, the board members will notify the requester if their request was approved. If approved the requester may check out the artifact from the repository; if denied the board will will provide an explanation as to why the request was denied.

3.3.2. Procedure for Committing Artifacts

When a team member wishes to commit an artifact to the repository, they must first notify the Configuration Control Board that the changes requested have been completed. The Board will review the changes made as described in section 3.2. If the changes are approved the artifact may be committed; if not it is the job of the team member to make sure the artifact is corrected as advised by the Board.

3.3.3. Procedure for Establishing New Baseline Configuration Artifacts

When establishing a new baseline for a configuration artifact, the Configuration Board will meed to discuss the changes made and to discuss if these changes will be approved. The Board will then discuss these changes with the Guidance Team, and their opinion on the changes made. If both the Configuration Board and the Guidance team agree with the changes made, the artifact will become the new baseline.

3.3.4. Administration of the Software Configuration Repository

Administration of the Software Configuration Repository will be managed by the administrators Chelsey Jurado the Lead Programmer of the team and Christian Hughes the Designer. Every team member will have access to the repository, but the administrators will be responsible for ensuring the repository is up to date, and that the naming conventions, and guidelines established in this document are being followed.

4. Software Configuration Auditing

It is important to follow the procedures that we have established in this document to ensure the thorough completion of the intended system. In order to accomplish this we have set in place an auditing process to do so. The auditing process is necessary because it produces documentation that eliminates confusion as to what changes were proposed, how they were handled, how long the changes took to implement, and how the changes affected the overall design of the system. To document the auditing process the Configuration Board must fill out a SCM Auditing Form (https://purpledinosaurs.wufoo.com/forms/q15y0yza123al82/), see figure 4.

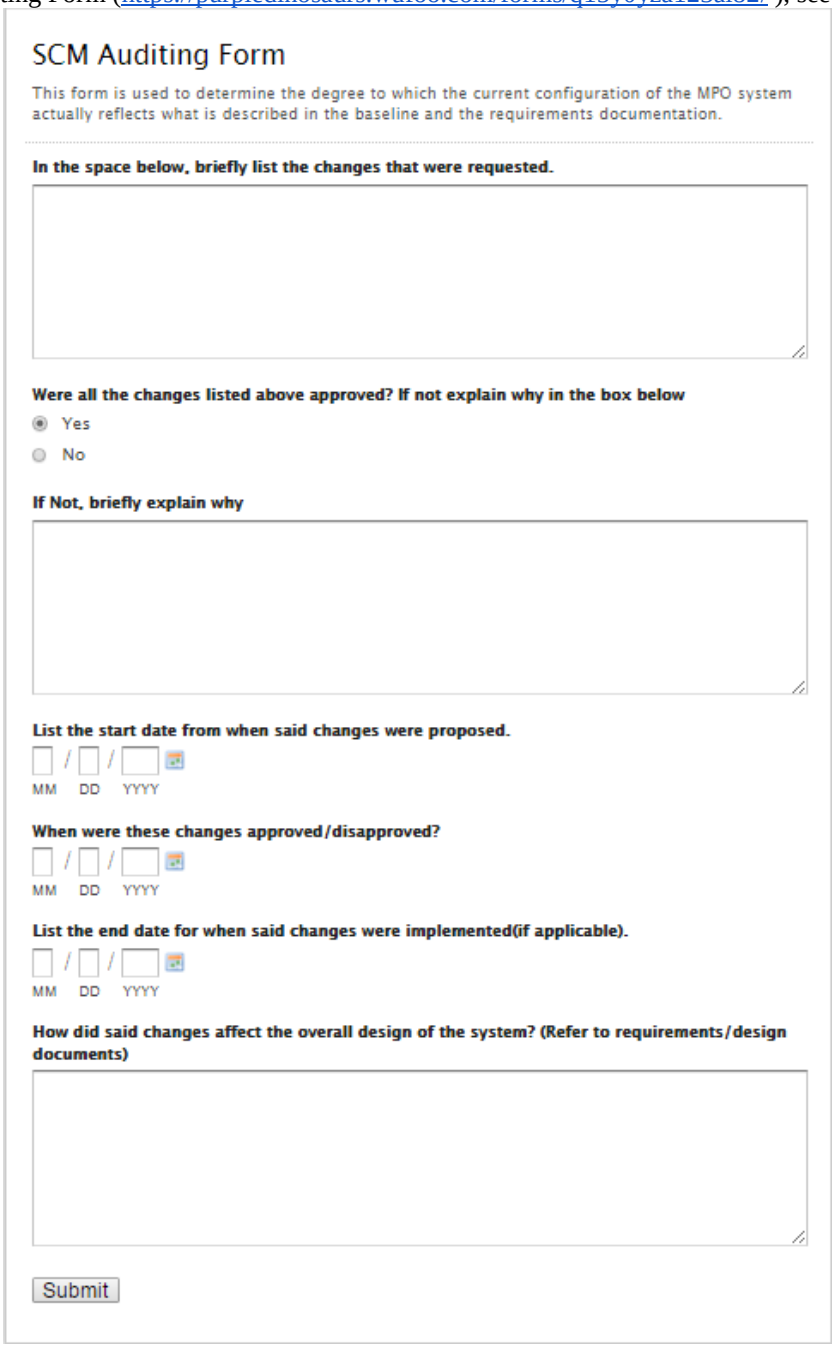


Figure 4. Software Configuration Management Auditing Form

It is important that when managing changes the team always takes the requirements into consideration. Any change made to a Software Configuration Item must be verified and validated against the requirements specified in the SRS. This verification and validation will be ensured by Jordan Cox the team's V&V. The method that will be used to document this verification and validation will be a traceability matrix. The requirements will be mapped to the various Software Configuration Items that fulfill them using forward tracing. Each Software Configuration Item will also be mapped to the requirement it fulfills using backward tracing. This will allow the team to easily determine which Software Configuration Items will most likely change whenever a requirement changes or vice-versa.

^_^