Improving Visualization Knowledge (VisKo)

Software Configuration Management Plan

Version 1.2

2/26/14

Document Control

Approval

The Guidance Team and the customer shall approve this document.

Document Change Control

|  |  |
| --- | --- |
| Initial Release: | February 4, 2014 |
| Current Release: | V1.1 |
| Indicator of Last Page in Document: |  |
| Date of Last Review: | February 25, 2014 |
| Date of Next Review: | February 27. 2014 |
| Target Date for Next Update: | February 27, 2014 |

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. Yoonsik Cheon

Bhanukiran Gurijala

Customer: Dr. Nicholas Del Rio

Software Team Members:

Maria Cortes – V&V Lead

Manuel Dosal – Systems Designer

Mario Flores – Lead Programmer 1

Rebekah Gruver – Lead Programmer 2

Hector Quintana – Systems Architect

Marcela Vazquez – Systems Analyst

Change Summary

The following table details changes made between versions of this document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modifier | Description |
| 1.0 | 2/2/2014 | Leaf Development | Version 1.0 of completed SCM compiled by the entire team |
| 1.1 | 2/4/2014 | Leaf Development | Refined and detailed all sections of SCM |
| 1.2 | 2/21/2014 | Maria Cortes and Rebekah Gruver | Corrected and redefined the entire SCM document based on TA feedback |

Table of Contents

Document Control 1

Approval 1

Document Change Control 1

Distribution List 1

Change Summary 1

1. Introduction 3

1.1. Purpose and Intended Audience 3

1.2. Scope of Product 3

1.3. Overview 3

1.4. References 4

2. Software Configuration Identification 5

2.1. Software Configuration Item Identification 5

2.2. Software Configuration Item Organization 5

2.2.1. Naming Convention 5

2.2.2. Labeling Scheme 6

2.2.3. Management 7

2.2.4. Detailed Directory Structure 8

2.2.5. Back-Up 8

3. Software Configuration Control 9

3.1. Documentation 9

3.2. Configuration Control Board 9

3.3. Procedures 9

4. Software Configuration Auditing 11

5. Appendix A 14

6. Appendix B 15

7. Appendix C 16

# Introduction

## Purpose and Intended Audience

The purpose of the Software Configuration Management Plan (SCM) is to systematically control changes to the configuration and maintain the integrity and traceability of the configuration throughout the system’s life cycle. The Software Configuration Management (SCM) will help the team with quality assurance by tracking and controlling changes in the software. Some of the techniques to control the changes include revision control and the establishment of baselines. The primary focus of this document is to define methods that will describe and identify configuration items, configuration control, and configuration auditing. By documenting these methods it will allow the team to manage change throughout the software lifecycle. It will make it easier to know what was changed in the software and who made the changes, this way if anything goes wrong it can be fixed without causing any further damage to the whole system. Following, the SCM will detail the exact change that occurred and the person responsible for it [3].

The intended audience of this document is the guidance team, and the Leaf Development team, as a guide through all the process of implementation, and design of the improved VisKo System.

## Scope of Product

The VisKo system is a framework that supports the answering of visualization queries that describe what visualizations users want generated rather than specifying how they should be generated [1][2]. Relying on a knowledge base of visualization services, VisKo can translate user queries into equivalent visualization pipelines that generate the visualization specified in the query. Therefore, users of the VisKo system can generate visualizations while remaining hidden from the complexities associated with visualization pipeline assembly, a practice that encompasses the themes: visualization transformation theory, computer graphics, semiotics, and performance. Currently, VisKo is only accessible through an application-programming interface and requires a more appropriate user interface [2]. Therefore, it is desirable that VisKo be improved by adding a set of perspectives that can support different interfaces and related functionality that would greatly assist in the widespread adoption of VisKo [1][2].

## Overview

The SCM is divided into five main sections: Introduction, Software Configuration Identification, Software Configuration Control, Software Configuration Auditing and Appendix. The **Introduction**: describes the project and the audience for the document. It includes a brief description of the purpose of the project, the scope, and an overview of the SCM document. The **Software Configuration Identification:** provides a list of items likely to be changed throughout the life of the project. These items include source code, test cases, design documents and requirements. It also describes the labeling convention and how the team will be collaborating in the updates. It includes a visual description of the convention used to label the updates, as well as how changes will be requested. At the same time, this section describes how the team backup process in order to avoid total or partial loss of documents or source code. The **Software Configuration Control:** describes the mechanism to prepare and evaluate the proposal for changes to the configuration items throughout the life cycle. The control board will approve or disapprove all change proposals when a team member fills out a change request form and justifies the means for doing it. It also describes the process to be followed to control access to source code and documents in order to avoid unauthorized updates and collisions between versions if more than one team member are working on the same task simultaneously. The **Software Configuration Auditing**: describes two types of auditing, Functional Configuration Audit (FCA) and Physical Configuration Audit (PCA). Configuration audits provide the status of the project by evaluating the configuration items. Finally, the **Appendix:** includes several forms used for the SCM process, such as change request forms, audit reports and test suites.

## References

[1] Leaf Development, “Interview Report,” El Paso, TX, 2013.

[2] Guidance Team, “VisKo Software Requirements Specification,” El Paso, TX, 2014

[3] <http://en.wikipedia.org/wiki/Software_configuration_management>

[4] Cheon, Yoonsik,*Software Configuration Management (SCM).* [Microsoft Power Point] El Paso, TX, 2014.

[5] Guidance Team, “Software Configuration Management Plan Template,” El Paso, TX, 2014.

[6][http://www.westfallteam.com/sites/default/files/papers/Software\_Configuration\_Management\_Audits.pdf](https://bl2prd0511.outlook.com/owa/redir.aspx?C=ArAxK95bJEKTGM-diZ-MgpXUoNmDhNAIc22EG2EPMF8B3Ina264x83WnRN5aJf8ZatipKO8QmFk.&URL=http%3a%2f%2fwww.westfallteam.com%2fsites%2fdefault%2ffiles%2fpapers%2fSoftware_Configuration_Management_Audits.pdf" \t "_blank), “scm-audit.pdf”

# Software Configuration Identification

This section provides labels for the baselines and their updates. Provides a list of items likely to be changed throughout the life cycle of the project. It also includes a description of the naming or labeling convention to be used and how the team will be collaborating throughout updates. A back up method is described that will be used to avoid total or partial loss of documents or source code.

## Software Configuration Item Identification

The Software Configuration Item Identification consists of the elements the software team will develop and modify throughout the software life cycle [4]. These items shall be kept under maintenance and if changes occur, they shall be documented. The following list consists of – elements that are likely to change throughout the “Improving Visualization Knowledge” process:

1. Documents
   1. Software Configuration Management (SCM)
   2. Classes, Responsibilities, and Collaborations (CRC)
   3. Software Requirements Specification (SRS)
   4. Software Design Documentation (SDD)
2. Source Code
3. Test Suites
4. Audit Reports
   1. Functional Configuration Auditing (FCA)
   2. Physical Configuration Auditing (PCA)

## Software Configuration Item Organization

This section describes the labeling scheme and naming convention of the software configuration items. All software team members will be using a specific labeling scheme to identify each different section of the process.

### Naming Convention

The naming convention is as follows:

|  |  |  |
| --- | --- | --- |
| Configuration Elements | Baseline Format | Example |
| Documents | *documentTitle\_leadPosition\_V#*  ***Submitted versions:*** *Final\_documentTitle\_TeamName* | SCM\_Analyst\_V1.0.docx  Final\_SCM\_LeafDevelopment.docx |
| Source Code | *className\_leadPosition\_V#* | GUIResgister\_Designer\_V1.2.php |
| Test Suite | *testName\_date\_V#* | dbtestOne\_Jan30\_V1.xlsx |
| Audit Report | *ConfigurationItemName\_AuditType\_V#* | SCM\_PCA\_V1.1.docx  SCM\_FCA\_V1.1.docx |

Figure 1.0 – Elements Naming Convention

Figure 1.0 identifies the naming convention of each configuration element. Source code and documents will follow a similar format. Test suites and audit reports part are used to help the team organize, and test each element. Therefore, they have different formats, which will facilitate its use among the team members.

Additionally, to clarify the documents format, the team members may split parts of the document, work on them individually, and then get together to put together a baseline that has formally been reviewed by each member. When this occurs, they will follow the following format: *documentTitle\_leadPosition\_V#, example:* **SCM\_Analyst\_V1.0.docx**, as the baseline is updated, the version number will change. However, because documents are often submitted as team reports, these submitted reports would follow a special format: same as in Figure 1.0. The submitted documents will be named as follows: The word “Final” followed by the document title, followed by the team’s name: *Final\_documentTitle\_TeamName,* example **Final\_SCM\_LeafDevelopment.docx.**

### Labeling Scheme

The labeling scheme is as follows:

**Documentation**

v3.0

v2.0

v1.0

v2.2

v2.1

v1.2

v1.1

* Baseline: V1.0, V2.0, V3.0, etc.
* Updates: V1.1, V1.2, V1.3, etc.

Documentation will follow evolution graphs to control the version number of the documents.

Initially a baseline document will be put together, once all team members have compiled their assigned parts. A review will be given to the document as a whole, and updates will be made. Baselines will follow *documentTitle\_leadPosition\_[V1.0, V1.2, V1.3, etc.].* Each time a baseline has been formally reviewed, every team member must agree that the current document is a baseline; in addition, approval process for any new baseline will depend upon guidance team approval. No document will be considered a new baseline until the guidance team has reviewed it and approved it.

Regarding updates, each time a baseline is updated, the following format will be used: *documentTitle\_leadPosition\_[V1.1, V1.2, V1.3, etc.].* Updates will be made every time a team member makes a major change in the document.

**Source Code**

v2.0

v1.0

v1.2

v1.1

v1.1.1

v1.1.2

v1.1.3

* Baseline: v1.0.0, v2.0.0, v3.0.0, etc.
* Group Updates: Updates: v1.1.0, v1.2.0, v1.3.0, etc.
* Individual Updates: v1.0.1, v1.0.2, v1.0.3, etc.

The version numbers for the source code will be grouped in two categories: baseline, and the initial version of the code. As the initial version is updated, this will use a different version format, depending on how it is updated. If only one individual updates the baseline, then the version number will follow the following format: Individual Updates: v1.0.1, v1.0.2, v1.0.3, etc. However, if a group, meaning two members agree on changing the baseline, the version number will follow the following format: Group Updates: Updates: v1.1.0, v1.2.0, v1.3.0, etc.

**Test Suites**

|  |
| --- |
| **Labeling** |
| testData\_Jan30\_1.jtt |
| testData\_Jan30\_2.jtt |
| testDB\_Jan31\_1.jtt |
| testTwo\_Jan31\_2.jtt |
| testTwo\_Feb1\_1.jtt |

* Labeled by date and numbered in order by that date (as defined in naming convention).

**Audit Report**

* Baseline: V1.0, V2.0, V3.0, etc.
* Updates: V1.1, V1.2, V1.3, etc.
* The initial Audit Report is the baseline, as changes are made, this will follow the “update” version format.
* Every Audit Report is labeled with the version number as each update is made.

### Management

All team members will manage the configuration elements:

* All the team members will manage documents, in order to ensure that the documents are placed in the correct folders.
* Program/source code will be managed by the systems architect, and programmers: Hector Quintana, Rebekah Gruver, and Mario Flores
* The test suites and the audit reports are part of documents, however this will be managed by the V & V: Maria Cortes

The configuration elements will be managed as follows:

**Documents**

All documentation will be saved in the Leaf Development folder located in Google Drive. Documents will primarily be stored in the appropriate assignment folder (i.e. SCM folder) that the team creates in Google Drive. The assignment folder will contain each baseline version/subversion of the document in that folder, as well as the final submitted assignment document.

**Program/Source Code**

There will be only one program folder that contains multiple versions of source code in that folder.

**Test suites**: specified to each version of the code

Each test suite specific to its appropriate source code will be stored with that source code version and only that version. As mentioned before, the program folder will contain each version of source code, each with its appropriate test case.

**Audit Report:** specified to each version of the code

Each Audit Report will be stored in each update (v1.1..v1.3 etc.) in order to guide the team with their duties. Also, each Audit Report will be stored in the folder where the source code in progress is saved.

### Detailed Directory Structure

* Documents
  + Document Name
* Baseline
  + - v1.0
    - v1.1
    - v1.2
* Final Version
* Source Code
  + v1.0
    - v1.1
      1. Test Suites
      2. Audit Reports
    - v1.2
      1. Test Suites
      2. Audit Reports
  + v2.0
    - v2.1
      1. Test Suits
      2. Audit Reports

### Back-Up

Primary control of source code backup will be the team’s repository on GitHub. Back up of source code within the team’s repository will take place after every upload of fully functional code into the repository. However each member is responsible for maintaining each of their individual documents, source code, tests, etc. on his or her personal computers. Each member is also responsible for maintaining their branch on GitHub, then following the process of merging with others.

Google Drive will be used as a back up for all documents/files that have been worked on, and/or are being modified. These documents/files may be uploaded by individuals, as well as by the team. Google Drive is an online source; therefore there is no determined date of backing up the documents.

# Software Configuration Control

This section provides the detailed mechanism for evaluating and approving or disapproving all change proposals to the configuration items throughout the life cycle. The purpose of this section is to identify what mechanism 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.

## Documentation

Changes to the configuration items will be managed as follows:

* Any member of the team will be able to make changes to the section of the document or code they have been assigned until the due date arrives.
* Each member will have “control” over the document until the due date.
* In order to perform any change, a group member who has no control over a section in the document or source code, will be required to fill out the form attached in Appendix A
* After the form has been filled out, this needs to be sent to the control board.

## Configuration Control Board

The configuration control board will be managed as follows:

* + The control board will be composed of the programmers, and V&V.
  + The documents will be managed by the control group composed of the team member or team members lead of that document (i.e. SCM document, member in charge is the analyst).
  + The control group composed of Programmer 1 and Programmer 2 will manage the source code.
  + The control group composed by the V&V will manage the test suites, and audit reports.
  + The approval process will be as follows:
    1. Any group member who wishes to make a change will fill out a change request form attached in Appendix A.
    2. This form will then be emailed to the control board.
    3. The control board will receive the email and review the change request. They will be required to make a decision in less than 12 hours and let the requester know of their decision.
    4. If the change request was approved, then the requester will need to complete the changes by the anticipated completion date specified in the request form.
    5. After the changes are done, the requester sends these changes to the control group in order to review and if approved by the control group, the changes will be uploaded into a new version.
  + Unauthorized approvals include changing an item before the control group approves and after a team agreement.
  + In order to authorize any change in these items, it must pass the Software Configuration Auditing guidelines.
  + The configuration manager group will supervise that the guidelines in this document are upheld. At the same time, this group will maintain the product development activity by providing appropriate workspace and resources to build and test the project.
  + The configuration manager group will consist of the System Analyst and the System Architect.

## Procedures

The procedures are described:

* + The approval process for any new baseline will depend upon guidance team approval.
  + No document will be considered a baseline until the guidance team has reviewed it and approved it.
  + The team will be using a GitHub repository to store code.
  + Each member will have a separate branch that they will be able to freely edit and commit to.
  + The branches will be reset each time the code changes group version to ensure members work on the correct version of the code.
  + When a team member turns in any code they will be in charge of merging their own code with the main branch.
  + If a team member performs an unauthorized merge the code will be reset to the previous version.
  + The programmers and the V&V will be in charge of testing after changes are made.
  + Any other configuration item will only be directly accessible by the control board.
  + Team members will have to work on their part locally and submit it to the control board through email when done.
  + The control board will then merge this work with the current version of the configuration item.
  + Any work submitted would need to follow the configuration item format, which we will receive from the guidance team.
  + The configuration items will be stored on a Google drive folder in order to have version control.
  + After work has been submitted and merged the control board will need to follow the following guidelines.
    1. If the document is less than 10 pages, the control group will then email it to everyone for review.
    2. If it is greater than or equal to 10 pages, the control group will be able to submit it if the document is acceptable.
    3. If the control group finds an error in one part of the document or code, they have 12 hours to notify the responsible person. The responsible person will need to check their email within 12 hours in order to check if any errors were found. Then, he or she has to resubmit a corrected version of the item in the next 12 hours unless specified otherwise. Therefore, the initial submission of each member has to be done one day before the final document is submitted.
       1. An error is only expected to be any item that does not comply with the baseline documents.

# Software Configuration Auditing

This section explains the two types of auditing that will be taking place.  The importance of auditing is that it provides a mechanism for determining the degree to which the current state of the system is consistent with the latest baseline and documentation [3]. At the same time configuration audits provide greater visibility into the status of a project by evaluating the status of the configuration items. Finally, they also determine the traceability from requirements, and change requests to the implementation by investigating the baselines and changes to the baselines [4][5].

The two types of auditing are defined as follows:

* **Functional Configuration Audit** (FCA), which verifies that a configuration item is in accordance with its software requirements.
* **Physical Configuration Audit** (PCA), which ensures that items in the baseline are of the right version.

Each type of auditing procedure will be explained in greater detail in the tables below.

**Functional Configuration Audit (FCA)**

A FCA is performed by a group of three people assigned by the lead of the project, which includes the V & V Lead. Typically the process is performed no more than 1 day after a new configuration item has been identified and named.  The FCA team will look over the configuration item and answer the questions in [Table 1](#table1) along with any other question they feel should be addressed in evaluating the item. The FCA members will list major issues, and inconsistencies in the Audit Report included in Appendix B. At the same time the members may write comments, questions, and concerns regarding the issues found on the configuration items. As for small issues found, the members will solve the issues immediately as they go through the checklist on [Table 1](#table1).

**Table 1** Functional Configuration Audit

|  |  |
| --- | --- |
| **Checklist Item** | **Suggestions for addressing checklist item** |
| List all the behavioral requirements from the SRS that are met. | * This checklist item specifically asks if there are any requirements from the SRS that **have** been met in the configuration items. |
| List all behavioral requirements from the SRS that are addressed | * This checklist item specifically asks if there are any requirements that **can** be met by referencing this configuration item. |
| List all the non-behavioral requirements from the SRS that are met. | * This checklist item specifically asks if there are any non-behavioral requirements in the SRS that **have** been met in the configuration items. |
| List all non-behavioral requirements from the SRS that are addressed | * This checklist item specifically asks if there are any requirements that **can** be met by referencing this configuration item. |
| Have all change requests been resolved? | * For un-resolved change requests, the change request may simply need to be closed, or it may need further testing, or deferred to a later baseline. |
| Is the Configuration Item complete? | * Does the item explain everything it is trying to explain, or does it need to be used with another configuration item.  If so, list the configuration items in the Audit Report. * Was there anything you felt was missing from the report? If so list anything missing from the report, in the Audit Report. * Were there any inconsistencies regarding the software requirements? If so list inconsistencies in the Audit Report. |
| List any questions or concerns. | * List any questions or concerns you may have of the item or the requirements in the Audit Report. * FCA team will identify and list all issues for which action is required in the Audit Report. |
| Were all questions and concerns addressed | * Was there a follow up for the audit? * Did the team members address the issues found in the FCA process? * Team members must resolve all issues found in the FCA process. |

Additionally, in order to ensure all requirements have been met in the implementation, a traceability matrix will be used, and a copy of the traceability matrix will be added as an appendix to the SRS. This method will allow the teams ensure that all requirements are met in the implementation. As well as allow us to review requirements that still need to be implemented.

The traceability matrix will be updated a week after new features have been added to the implementation. The V&V Lead will be in charge of managing these updates. The updates will be completed in the traceability matrix using an Excel Spreadsheet from Microsoft as shown in Appendix C. Only the functional requirements that process information are expected to have test cases.

**Physical Configuration Audit (PCA)**

A PCA is assigned by the lead of the project. The PCA is performed by a group of two people who are not assigned to do the FCA; this is because it’s important to have all the team members work on different parts of the audit processes. Doing this will also allow the PCA group to be objective in their assessment of the FCA process. The PCA process typically takes place no more than 2 days after a new configuration item has been identified and named. The PCA team will look over the configuration item and answer the questions in [Table 2](#table2), which states a checklist that should be addressed when conducting a PCA. The PCA members will list major issues, and inconsistencies in the Audit Report included in Appendix B. At the same time the members may write comments, questions, and concerns regarding the issues found on the configuration items. As for small issues found, the members will solve the issues immediately as they go through the checklist on [Table 2](#table2).

**Table 2** Physical Configuration Audit

|  |  |
| --- | --- |
| **Checklist item** | **Suggestions for addressing checklist item** |
| All configuration items have been updated to the correct version | * This checklist item specifically asks if each configuration item is named accordingly to the **correct** version as specified in the software configuration identification. * **Example**: the SCM report has been updated to the correct version, after major changes made by the analyst. * List any inconsistencies in the Audit Report. |
| All configuration items named correctly according to the naming convention | * This checklist item specifically asks if each configuration item is **named correctly** according to the naming convention. * **Example**: Documents use the following format: *documentTitle\_leadPosition* * List any inconsistencies in the Audit Report. |
| All configuration items properly placed in the correct physical location | * This checklist item specifically asks if each configuration item **has been placed** in the correct physical location in the repository. * **Example:** all document configuration items are only placed in the documents, not with source code. * List any inconsistencies in the Audit Report. |
| List any questions or concerns. | * List any questions or concerns you may have of the item or the requirements in the Audit Report. * FCA team will identify and list all issues for which action is required in the Audit Report. |
| Were all questions and concerns addressed | * Was there a follow up for the audit? * Were issues resolved? * Team members must resolve all issues found in the PCA process. |

# Appendix A

**Leaf Development Team**

**Request Change Form**

|  |  |
| --- | --- |
| **Requester Name:** |  |
| **Date:** |  |
|  |  |
| **Code/Document name:** |  |
| **Code/Document Version:** |  |
| **Requested Change:** |  |
| **Reason for Change:** |  |
| **Due Date:** |  |
| **Text/Code file(s):** |  |

|  |  |
| --- | --- |
| **Control Group** |  |
| **Name:** |  |
| **Approved:** | * **Yes** * **No** |
| **Reason:** |  |
| **Signature:** |  |

**Fig. 1 Request Change Form**

# Appendix B

**Leaf Development**

**Audit Report**

**Software Configuration Item Name/Version:**

**Type of Audit Process:**

**Audit Members:**

**Date:**

|  |  |
| --- | --- |
| **List of Inconsistencies**  **Missing Items, Other Issues** | **Comments**  **Questions**  **Concerns** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Appendix C

**Leaf Development**

**Test Suite Matrix**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Req #** | **Requirement** | **Tested/Not Tested** | **Pass/Fail** | **Test Case** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Date:**

**Implementation Date:**

**Comments:**

