***Graphing for Change***

***Configuration***

***Management Plan***

Graphing for Change

2018-03-22

v 1.2

\_\_\_\_\_\_\_\_\_\_\_\_Daniel Herman\_\_\_\_\_\_\_\_\_\_

[SCM Lead Signature Block]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[Project Lead Signature Block]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[Project Manager Signature Block]

**Revision Sheet**

|  |  |  |
| --- | --- | --- |
| Revision | Date | Brief Summary of Changes |
| Version 1.0 | 2018-02-22 | Baseline document draft |
| Version 1.1 | 2018-03-09 | Procedures largely specified |
| Version 1.2 | 2018-03-22 | Finished Document |
| Version 1.3 | 2018-04-04 | Corrected and polished documentation for submission |

[**INTRODUCTION**](#_ibx4jc4bnanp) **5**

[Purpose](#_46rfi2ahbtvj) 5

[Scope](#_t9xf4uvpl2jz) 5

[Definitions and Acronyms](#_5puc3m4eiptc) 5

[Key acronyms](#_y0yeal3kpbx1) 5

[Key terms](#_uwuq5nlagu0y) 6

[References](#_dzk3ray20bru) 8

[**Software Configuration Management (SCM)**](#_jntzdtg6nc2) **8**

[SCM Organization](#_2emj4gjmimks) 8

[SCM Responsibilities](#_xxo0y4s8sa4j) 8

[SCM Resources](#_slzfhn4iy9ij) 8

[**SCM ACTIVITIES**](#_mpq11w2p5vwm) **9**

[Configuration Identification](#_z337ya) 9

[Specification Identification](#_lnb077le3hz8) 9

[Change Control Form Identification](#_kxbb4c1utzue) 10

[Project Baselines](#_dz1n0ow3k4ov) 11

[Library](#_ektvswxj7zqx) 11

[Configuration Control](#_dxie6ffv4041) 11

[Procedures for changing baseline](#_srtoujs90yqm) 11

[Organizations assigned responsibilities for change control](#_fnvun0n1qya7) 12

[Interfaces](#_qhfrf9ri13v) 12

[Level of control](#_o786p66orw8h) 12

[Document revisions](#_pn7xv35dsa7y) 13

[Automated tools used to perform change control](#_xobu4x56ymhe) 13

[Configuration Status Accounting](#_7wzfawf3yh34) 13

[Storage, handling and release of project media](#_4unfr1f904tn) 13

[Information and Control](#_xa18eg7cg9l2) 13

[Reporting](#_imredfq68qto) 13

[Release process](#_88lmgbscghr7) 14

[Document Status Accounting](#_w2h0bwnboh08) 14

[Change Management Status Accounting](#_i7jmulprqffc) 14

[Configuration Audits and Reviews](#_v0iyd8hhhxd3) 14

[**CM Milestones**](#_55shrshf2x9c) **14**

[**Training**](#_dpi9e5zdah4t) **15**

[**Subcontractor/Vendor Support**](#_kkmai65u005e) **15**

# **INTRODUCTION**

## **Purpose**

The software configuration management plan (SCMP) is for use by developers that will be releasing working versions of the product. It is intended to both outline the dependencies among separate components, specify the requirements of hardware, and to guide developers in conforming to a process that promotes the stability of releases.

## **Scope**

This document is for use while developing and maintaining Graphing for Change: a project that is aimed at drawing meaningful graphical relations from data obtained from various entities involved in social services for the vulnerable. This document will contain the versioning compatibility information between components of the project. This includes database-server and web-server versioning, types, locations, and who has access to modify these. It also includes software component versions, compatibility between components, and what general functionality these components give to the project as a whole.

This document should be maintained throughout the development process and during maintenance. Until otherwise specified, only major separate components will be tracked in this document. Within a software component, if the outward facing interface does not change, then this document does not need to be changed.

## **Definitions and Acronyms**

### **Key acronyms**

BCR - Baseline Change Request

CCB - Configuration Control Board

CER - Change/Enhancement Request

CM - Configuration Management

CRC - Change Request Coordinator

DT&E - Developmental Test & Evaluation

FOT&E - Final Operational Test and Evaluation

ITR - Internal Test Report

PL - Project Lead

PM - Program Manager

PCVS - PC based source code version control software

SCM - Software Configuration Manager

SCMP - Software Configuration Management Plan

SCR - Software Change Request

SE - Software Engineer

SEPG - Software Engineering Process Group

SPR - Software Problem Report

SQA - Software Quality Assurance

SRS - Software Requirements Specification

TPR - Test Problem Report (DT&E)

VDD - Version Description Document

### **Key terms**

The following key terms are used as defined in the IEEE Guide for the Use of IEEE Standard Dictionary.

|  |  |
| --- | --- |
| **TERMS:** | **Definitions:** |
| configuration management | “A discipline applying technical and administrative direction and  surveillance to: identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics of a configuration item, control changes to those characteristics, record and report change processing implementation status, and verify compliance with specified requirements.” [IEEE90]. |
| customer | “The individual or organization that specifies and accepts the project deliverables. The customer may be internal or external to  the parent organization of the project, and may or may not be the  end user of the software product. A financial transaction between  the customer and developer is not necessarily implied.” [IEEE87] |
| database | “A collection of data fundamental to a system.” [IEEE91] |
| installation | “The period of time in the software life cycle during which a  software product is integrated into its operational environment and  tested in this environment to ensure that it performs as required.”  [IEEE91] |
| plan | “A detailed scheme, program, or method worked out beforehand for the accomplishment of an objective.” [Heritage85] |
| process | “A sequence of steps performed for a given purpose.” [IEEE90] |
| project | A unit of work to meet a specific customer requirement. Includes all tasks, activities, and functions necessary to meet the requirements. |
| project deliverables | “The work product(s) to be delivered to the customer. The quantities, delivery dates, and delivery locations are specified in the project agreement.” [IEEE87] |
| pull-review | A member of the development team responsible for approving changes to the baseline of the project. |
| quality assurance | “(1) A planned and systematic pattern of all actions necessary to  provide adequate confidence that an item or product conforms to established technical requirements.” [IEEE90]  “(2) A set of activities designed to evaluate the process by which  products are developed or manufactured.” [IEEE90] |
| software | “Computer programs, procedures, and associated documentation  and data pertaining to the operation of a computer system.” [IEEE90] |
| software life cycle | “A project-specific, sequenced mapping of activities.” [IEEE91] |
| specification | “A document that specifies, in a complete, precise, verifiable  manner, the requirements, design, behavior, or other characteristics  of a system or component, and, often, the procedures for  determining whether these provisions have been satisfied.” [IEEE90] |
| work product | --Any tangible item that results from a project function, activity, or  task. Examples of work products include customer requirements,  project plan, design documents, source and object code, user’s manuals. |

## **References**

IEEE87 IEEE Std 1058.1-1987, IEEE Standard for Software Project

Management Plans

IEEE88 IEEE Std 982.2-1988, IEEE Guide for the Use of IEEE Standard

Dictionary.

IEEE90 IEEE Std 610.12-1990, IEEE Standard Glossary of Software

Engineering Terminology (ANSI).

IEEE91 IEEE Std 1074-1991, IEEE Standard for Developing Software Life

Cycle Processes.

Heritage85 The American Heritage Dictionary, Houghton Mifflin Publishers,

1985.

# **Software Configuration Management (SCM)**

## **SCM Organization**

Reece Walsh (pull-reviewer)

Jeff Thompson (Product Owner Representative)

Daniel Herman (pull-reviewer)

Thomas Reid (Software-Developer)

Daniel Schwab (Software-Developer)

Justin Habetz (Software-Developer)

Zachary Bentsen (Software-Developer)

## **SCM Responsibilities**

It is the responsibility of anyone modifying a software configuration item to ensure that the modifications have been recorded. Once all changes have been made, and the documentation updated to reflect those changes, a pull request should be submitted for review. A pull-reviewer will then look over the changes and pull them into the mainline of the project (if accepted).

## **SCM Resources**

SCM members should be familiar with a few resources to ensure consistency over the duration of the project. SCM members should download PHPStorm version 2017.3 so that editor configuration errors can be mitigated. Members should be familiarized with github, since it is our main SCM tool.

# **SCM ACTIVITIES**

## **Configuration Identification**

We identify a configuration by a baseline that consists of:

* source files
* libraries
* executable files
* data files
* Version Description Documents (VDD)
* User's Manuals

It is required that this configuration passes all relevant tests before it is accepted as a baseline. A configuration will give be given a unique versions number written vX.M.P:

* X is the main version number, which will be inherited from the baseline from which this baseline was derived.
* M is the major patch number, and corresponds directly to the sequence of baselines, where if a baseline is more recent than another, then it will have a high M
* P is the minor patch number, indicating relatively unimportant change

A VDD is a document associated with another element of the baseline.

The VDD should contain:

- a list of other run-time relevant dependencies and their versions.

- a description of the component the VDD is associated with

- any known bugs and their cause

- the name of any unit tests files associated with this file

### **Specification Identification**

**Testing:**

Code that has not been thoroughly tested (as deemed by the pull request reviewer) will not be accepted into the code base until testing has reached an acceptable level.

Our testing framework for our PHP code relies on PHPUnit version 5.3. Only this version, or compatible versions should be used to run the tests. The tests associated with a class are located within the same folder and have the same name with ‘Test’ appended to the file name.

**Repositories:**

Git:

* Our source code and documentation repository
* <https://github.com/conscientiousCode/mapFC>
* Currently it is a public repository, so avoid posting server information.

JIRA:

* We have a JIRA board that is the repository for our user-stories, use cases and tasks.
* It is located at <https://jira.ins.ok.ubc.ca>.
* To connect, you must either use the VPN (*myvpn.ubco.ca*) or be on UBCO’s campus.

**Documentation:**

Our documentation is written collaboratively through google docs and is managed by Reece Walsh.

**Database:**

The database that we use is MariaDB v5.5.56, an SQL database. It is hosted at cosc304.ok.ubc.ca:3306/db\_rwalsh . A VPN through UBCO or to be on campus is required to connect to the database.

### **Change Control Form Identification**

A change request to the configuration for source code is only necessary when we would revert a component(s) to a previous version. Regular development is geared towards releasing updated code and should be considered the norm. Any user story in JIRA on a sprint is considered approved.

Any activity that would require the team to re-tool, code to be rewritten, the direction of a sprint, or the overall architecture of the project to be changed should be passed through the change request process.

Procedure:

* Fill-out the change request form in the repository under “documentation/SCM/Change Request Form”
* Discuss the change with the dev team.
* Indicate on the form why it was approved or denied
* Implement the changes, and update configurations as necessary.
* Add the completed form to the repository in the change request folder
* It should be titled with its date in yyyy-mm-dd - <your text>

### **Project Baselines**

#### 3.1.3.1 Changing a Baseline

When a change is requested, it is discussed with the development team. The development team must reach an 80% consensus on accepting the change, with 100% agreement being strongly encouraged.

When possible, the following workflow is strongly encouraged when implementing a change:

* Develop acceptance tests for the change request
* Compile a list of documents/components/resources that will be modified as a result of implementing the change.
* Implement the change
* Update the documents from the list
* submit a pull request.

#### 3.1.3.2 Merging Procedures:

The following procedure must be used when merging back to the baseline the branch originated from (to protect that baseline):

* Pull the branch from the distributed repo to ensure the baseline is up to date
* Create a new branch from the baseline called <branchToBeMerged>Merger
* Change to this newly created branch
* Merge the branch that you want to be a part of the baseline into this branch
* Resolve any merge conflicts
* Test the merger branch after merging
* Change to the base line
* Merge the merger branch into the baseline (should have no conflicts)
* Test the baseline
* Submit a pull-request

### **Library**

* This project will not use any configuration management repository.
* This project relies on the Google Charts API, the Google Maps API, and D3.
* The current project relies on Reece’s license to the Google Maps API, which has a limit of 1000 calls per day.

## **Configuration Control**

### **Procedures for changing baseline**

As part of the Agile process, we will always consider the most recently pushed version of master branch of the git repository at the end of each sprint as the new baseline. This baseline is then used to define what must be accomplished during the next sprint.

If the content of a sprint is going to be changed part of the way through, then a change request form should be filled out and submitted to one of the two pull-reviewers.

When a change is requested, it is discussed by the development team. The development team must reach an 80% consensus on accepting the change, with the ideal being 100%.

The following activities must happen in order once a change has been approved:

* update the JIRA task list
* Develop an acceptance test for the change request
* Compile a list of documents/components/resources that will be modified as a result of implementing the change.
* Implement the changes
* Update the documents from the list
* submit a pull request. (if applicable)

### **Organizations assigned responsibilities for change control**

All organizations/developers are responsible for maintaining documentation and ensure that the code that they write is well tested. When an organization/developer has completed an assigned task, they must submit a pull request for a second review of the work that has been done.

A pull-reviewer is responsible for maintaining the baseline and ensuring that the changes being pulled in pass their tests and match the design and component descriptions of their associated documentation.

A product-owner representative may be asked by a pull-reviewer to validate the changes that are being represented.

### **Interfaces**

Our project relies on the following 3rd party interfaces:

* Google Charts API:
  + <https://developers.google.com/chart/glossary>
* Google Maps API:
  + <https://developers.google.com/maps/documentation/javascript/tutorial>
* D3:
  + <https://d3js.org/>
* MariaDB SQL:
  + <https://mariadb.org/>
* GitHub:
  + <https://git-scm.com/docs>

### **Level of control**

**Approving Changes:** All members of the development team are responsible for understanding the impact of a change request. The change is then decided by an 80% majority threshold vote.

**Validating Changes:** Once a change has been made, it is sent for review to a pull-reviewer. If accepted, the change will be merged into the mainline. If denied, then an explanation of the problems with the change will be sent back to the developer who submitted the pull-request.

**Requesting Changes:** Any member of the development team, or an individual representing the client, can fill out a change request form for review.

### **Document revisions**

Document revisions should be made and added in with any development changes that would cause the documents to become out of date with the components in the workspace. For major documentation changes, a pull request is sent for approval of the revisions to ensure completeness and accuracy.

### **Automated tools used to perform change control**

Our github repository is used to share up-to-date components and documentation.

JIRA is being used to manage and give context to changes that are to be made in the future. Tasks that are on JIRA are already considered to be approved.

## **Configuration Status Accounting**

### **Storage, handling and release of project media**

When a project is going to be released for use outside of development:

* Assign a unique release version number (RVN) as specified under 3.1 of this document
* Record the RVN in the release documentation as well as the date
* Assemble the components in the manner specified in the building configuration document
* Test and release

### **Information and Control**

### **Reporting**

Before a sprint is ended, a pull-reviewer will run all automated tests of the mainline to support the correctness of the program. If any failures are found, the component that failed will be recorded, and the team member who submitted the faulty component will be notified. The rest of the team members will be notified of any end of sprint anomalies of the mainline during our weekly meetings.

### **Release process**

A release occurs when the members of the development team agree that significant enough changes have occurred within the project to provide significant benefit to those who would use it: an agreement of 80% of the team members is required.

Once in agreement:

* the current mainline is tested thoroughly (acceptance, unit and integration).
* components necessary for release are gathered and checked off against.
* The software is compiled and tested in an environment independent of the software project workspace.
* If all tests pass, then the release is versioned, and it is released with notes about what is different from the previous version.
* The project is then released (the files on the webserver are changed/ the database has its tables updated)

### **Document Status Accounting**

Changes to this document must be accompanied by a team email indicating the sections changed and a version number increase. The revision table at the beginning of this table should be updated, and the version number and footer should be changed. This email should specify the sections that were changed.

### **Change Management Status Accounting**

This document should be changed whenever the procedures change. Procedures should only change at the end of a sprint. When changes are made, follow section 3.3.5

## **Configuration Audits and Reviews**

As this project is very short term, there will be no formal audits of the baseline unless:

* There is a major persistent bug.
* We are about to begin a refactoring sprint (so we can find particularly poorly architected sections)

If an audit is to be performed, each developer will be given a target section of code to refactor. These insufficiency will be documented and then placed in JIRA as a user story/task.

# **CM Milestones**

Documentation maintenance is of the utmost importance as once some documentation is out of date, then it may cause misinformation to spread to other developers and magnify the effects. When a sprint closes, documentation must be complete and up to date.

Testing of all component interfaces is required to ensure proper behaviour of components. Unit tests, as a general rule, tell you what a problem cannot be, which is sometimes just as useful as telling you what a problem is. All interfaces should be completely tested before being submitted in a pull request.

# **Training**

SCM training will be a collaborative effort among the team members, and will not be allocated any specific resources during the process of this short sprint.

# **Subcontractor/Vendor Support**

This project will not subcontracted. If this project is going to be subcontracted, a plan and set of requirements must be discussed at that time.