Software Configuration Management Plan for RaiderNAV

Team Getana

Version 4.0

(17 April 2018)

Revision History

Version	Date	Summary
4.0	17 April 2018	Updated for deliverable 4
3.1	6 April 2018	Updated table of contents.
3.0	1 April 2018	Sections 1.2.4, 2.2, and 3.1.3 updated to reflect missing information. Sections 4 and 5 completed. Minor wording changes throughout.
2.0	9 March 2018	Initial document

Table of Contents

1.1 Purpose

The purpose of this document (henceforth variably referred to as “the plan,” “this plan,” “this document,” “SCM plan,” and “Software Configuration Management plan”) is to document the configuration management elements and procedures tied to development of the *RaiderNAV* application. It describes what Software Configuration Management (SCM) activities are to be done, how they are to be done, who is responsible for doing specific activities, when those activities are to happen, and what resources are required. It covers all portions of the software development lifecycle for the *RaiderNAV* application. This plan exists to ensure orderly, controlled management of changes to the *RaiderNAV* application.

The primary users of this document are assumed to be those participating in SCM activities as part of the development of the *RaiderNAV* application.

1.2 Scope

1.2.1 Overview description of the software project.

The *RaiderNAV* software application is designed to provide college students at Texas Tech University with a better way to find their classes, navigate through campus, and schedule their day.

1.2.2 Applicability

SCM will be applied to the following software configuration items (CIs): Project Plan, Software Configuration Management Plan, Requirements Specification, Use Cases, User Stories, Hosting Solution, Code, Test Plan, System Design, Release plan, and Image.

1.2.3 Identification of other software to be included as part of the plan

Other software subject to this plan include choice of IDE (Android Studio), choice of APIs (Google Maps), choice of repositories (GitHub, ZenHub, Google Drive), choice of continuous integration tools (Travis CI, CodeClimate), choice of build system (Gradle), and choice of platform (Android).

1.2.4 Relationship of SCM to the hardware or SCM activities for the project

No relationship between the SCM and hardware activities has been identified.

1.2.5 The degree of formality, depth of control, and portion of the software life cycle for applying SCM on this project

This SCM will be applied informally to the *RaiderNAV* project. Flexibility and loose adherence is expected. This SCM applies to all portions of the software lifecycle for the *RaiderNAV* application.

1.2.6 Limitations

Limitations which apply to this plan include:

- Time constraints - In the event that a change is inconsistent with this plan and a short deadline requires the change, implementation of the change takes precedence over adherence to this plan.

1.2.7 Assumptions that might have an impact on the cost, schedule, or ability to perform defined SCM activities

This plan assumes that customer participation in SCM activities will be limited to interactions between the product owner and a conservative scrum master. Consequently, this plan is designed with the expectation that a minimal number of changes will be required during development.

1.3 Key Terms

Key terms used in this plan include:

- Agile - A software development methodology
- API - Application Programming Interface
- Baseline - A specific system; list of all specific component versions included in the system plus a specification of the libraries and configuration files involved
- CI - Configuration Item (except in the case of TravisCI, where it stands for Continuous Integration)
- Codeline - Sequence of versions of code with later versions derived from earlier versions; applies to each component separately
- IDE - Integrated Development Environment
- Mainline - A sequence of baselines representing different versions of a system
- Product owner - Customer requesting software in Scrum method
- SCM - Software Configuration Management
- Scrum - An Agile methodology
- Scrum Master - Project manager of Scrum method
- SDLC - Software Development Lifecycle
- “The Plan”/“this plan” - This document

1.4 References

This section includes all references to policies, directives, procedures, standards, terminology, and related documents.

This document is meant to conform to the requirements of IEEE Std 828-2005.

2. SCM Management

This section is intended to be used by organizations and individuals within the *RaiderNAV* project structure. It identifies the responsibilities (and allocation thereof) and authorities for managing and accomplishing planned SCM activities.

2.1 Organization

SCM as outlined in this plan applies to Team Getana. Team Getana is responsible for any and all SCM activity on the *RaiderNAV* project, as well as the problem resolution process.

Team Getana members include Arpit Desai, Brody Williams, Michael Acosta, Vasilis Vloutis, Sakshyam Silwal, and Yong Wu.

Functional roles include Scrum Master, Product Owner, and Development Team. Functional roles are rotated on a semi-weekly basis and assignment is decided at the end of each sprint first by volition of team members and subsequently by drawing straws if any roles remain unfilled. Efforts are made to ensure that each member assumes the roles of Scrum Master and Product Owner at least once.

Officially, the Scrum Master interfaces between the Development Team and the Product Owner in most circumstances. Due to the rotating nature of the roles, interfacing is casually observed. All team members, including those assigned as Scrum Master or Product Owner, are also assigned to Development Team tasks. Additionally, due to tight deadlines and high variability in team member scheduling and availability, any team member may augment a role that they are not assigned to during a given sprint.

2.2. Responsibilities

The Scrum Master will oversee project development, generally manage the project, ensure distraction-free adherence to scrum methodology of the development team, and negotiate changes with the Product Owner.

The Product Owner will prioritize features to be implemented in the project and be responsible for generating new user stories for the project.

The Development Team will be responsible for implementation and documentation.

Role rotation schedules follow.

Deliverable 0: No team member was assigned any role.

Deliverable 1:

Name	Role
Vasilis	Scrum Master and Development Team
Michael	Product Owner and Development Team
Arpit	Development Team
Yong	Development Team
Brody	Development Team
Sakshyam	Development Team

Deliverable 2 (Sprints 2 and 3):

Name	Role
Brody	Scrum Master and Development Team
Sakshyam	Product Owner and Development Team
Arpit	Development Team
Yong	Development Team
Michael	Development Team
Vasilis	Development Team

Deliverable 3 (Sprint 4):

Name	Role
Sakshyam	Scrum Master and Development Team
Vasilis	Product Owner and Development Team
Arpit	Development Team
Yong	Development Team
Brody	Development Team
Michael	Development Team

Deliverable 3 (Sprint 5):

Name	Role
Yong	Scrum Master and Development Team
Brody	Product Owner and Development Team
Arpit	Development Team
Vasilis	Development Team
Sakshyam	Development Team
Michael	Development Team

Deliverable 4:

Name	Role
Michael	Scrum Master and Development Team
Yong	Product Owner and Development Team
Vasilis	Development Team
Brody	Development Team
Sakshyam	Development Team
Arpit	Development Team

2.3. Applicable policies, directives, and procedures

No SCM policies, directives, or external procedures have been identified to apply to this project.

2.4 Management of the SCM process

Team Getana is responsible for the overall SCM process.

The anticipated cost of the SCM process is between 20 minutes and 6 hours of the share of time that would otherwise be directed at other functions and responsibilities. If detailed tasking information is provided during development, periodic monitoring of planned versus actual costs (in time) can be facilitated by observance of the Kanban board and Scrum Backlog. No periodic monitoring task has been assigned to any functional team member role at this time.

Independent surveillance of SCM activities to ensure compliance with the SCM plan will not take place, and no organization will be assigned this role.

Risks, including technical, economic, schedule, and managerial risks associated with the performance of SCM activities include:

- SCM activities may take more time than expected and interfere with project completion. In the worst case, a project may be turned in incomplete. The mitigation strategy associated with this risk is to prioritize project completion over any strict adherence to SCM activities.

3. SCM Activities

This section identifies all functions and tasks required to manage the configuration of the software system as specified in the scope of this document. Both technical and managerial SCM activities shall be identified. General project activities that have SCM implications shall be described from the SCM perspective.

3.1 Configuration identification

3.1.1 Identify configuration items (events, items, procedures)

A baseline is triggered by deliverable dates. Each deliverable will be regarded as the next iteration of the baseline.

Items controlled in the baseline include:

- File format for application file storage
- External API versions
- Domain Model & Detailed Design document
- Software Architecture & Design Pattern document
- Platform document
- Product Backlog
- Sprint Backlog
- Software Requirements Specification
- URN document
- Team Member report
- Sprint review
- Focus Group document
- Usability Study document
- Daily Scrum report
- Test cases
- Test plans
- APK files
- Support environments
- This document

Baseline controls are established when more than one team member has code dependent on a specific configuration. Changes to the baseline require review of all affected code and should be performed only at the beginning of sprints.

Consensus between team members is sufficient to approve changes to the baseline.

In order to identify CIs which should be controlled by this plan, care must be taken by team members to observe product breakages. In the event that a product breakage occurs, a review and casual cost-benefit analysis must be performed to determine whether the changed item will create more overhead as a controlled CI than will result from fixing future breakages. If the CI would likely create less overhead while controlled, it should be added to this plan.

3.1.2 Name configuration items (unique identifiers)

Documentation will be identified by unique file names consistent with the role of the document, the deliverable number, and the sprint number. Documentation file names will take the form of "Getana_Deliverable_i_{documentname}_j" where i is the deliverable number and j is the sprint number. Documentation will be kept under the "docs" folder of the configuration management repository within a folder designating the deliverable number. The sprint number should be prominently displayed within the documents themselves. Uncontrolled items should be placed within the "artifacts" folder within the "docs" folder.

Baselines will be referred to by deliverable number, but code will follow no such file naming convention.

3.1.3 Acquiring configuration items (physical procedures)

CIs will be maintained on GitHub at:

https://www.github.com/aluminiumi/RaiderNAV

No physical documents will be controlled.

External libraries will be pulled in by Gradle, and must be specified in the *build.gradle* file within the *app* folder on the GitHub repository. Software libraries, including documentation and data, will not be placed under physical control except insofar as specific versions will be identified for use. No documentation requirements or inspection requirements have been identified at this time. The *build.gradle* file in the root project folder (not the *app* folder) must include the following lines in the repositories block to ensure import of currently-identified external libraries:

```
google()
jcenter()
```

No retention periods or disaster prevention and recovery procedures are implemented.

The following external libraries are controlled:

3.1.3.1 Google Play Services Location

This library must be kept at version 9.2.0 or lower, as newer versions require newer versions are incompatible with Android 4.0.3.

This library is included by placing the line

```
implementation 'com.google.android.gms:play-services-location:9.2.0'
```

within the *dependencies* block of *app/build.gradle*.

3.1.3.2 Google Play Services Maps

This library must be kept at version 9.2.0 or lower, as newer versions require newer versions are incompatible with Android 4.0.3.

This library is included by placing the line

implementation 'com.google.android.gms:play-services-maps:9.2.0'

within the *dependencies* block of *app/build.gradle*.

3.1.3.3 Google Gson

At this time, no specific version of this library has been identified for control. The latest available version may be used.

This library is included by placing the line

implementation 'com.google.code.gson:gson:2.8.2'

within the *dependencies* block of *app/build.gradle*.

3.1.3.4 Google Direction Library

This library must be kept at version 1.0.4 or lower, as newer versions require newer versions of Google Play Services which are unavailable to Android 4.0.3.

This library is included by placing the line

implementation 'com.akexorcist:googledirectionlibrary:1.0.4'

within the *dependencies* block of *app/build.gradle*.

3.2. Configuration control

The following change controls are imposed on all baselined CIs identified in 3.1.1.

3.2.1 Requesting changes

To request a change to a baselined CI, the need for the change must be made known at a scrum meeting or by broadcast text messaging to the team. No record of the change request need be maintained.

3.2.2 Evaluating changes

Self-identified stakeholders in the change will evaluate the change request and either approve or disapprove it. Changes should be evaluated according to their effect on the deliverable and their impact on project resources.

3.2.3 Approving or disapproving changes

Consensus must be reached among stakeholders to either approve or disapprove a change request.

3.2.4 Implementing changes

The development team is responsible for implementing changes. Verification, implementation, and release of change will proceed in accordance with regular development steps. No record need be maintained beyond regular development records.

3.3. Configuration status accounting

Configuration statuses as tracked in GitHub include Open, Closed, and Merged. The information is publicly accessible on the internet.

3.4. Configuration evaluation and reviews

A configuration audit shall be performed on CIs prior to finalizing each deliverable. In the configuration audit, all CIs will be reviewed. The procedure involves the Scrum Master (at a minimum) and any additional team members looking over each of the required deliverable items and ensuring that their contents match requirements. No documents other than the CI documentation under review and the initial project requirement documents are needed to perform the review. No records of configuration audits will be maintained. In the event of a failed audit, a change must be implemented immediately by the development team to rectify the failure.

3.5 Interface control

The *RaiderNAV* software interfaces with Android OS 4.0.3 and with the Google Maps API 9.2.0 in accordance with the APIs published by Google.

3.6 Subcontractor/vendor control

The *RaiderNAV* project does not incorporate subcontracted software and hence requires no procedure to monitor subcontractors.

3.7. Release management and delivery

Release management will be performed by the Scrum Master while finalizing each deliverable. Delivery of software is fixed by product deliverable requirements and includes submitting a compressed package (ZIP archive) of all relevant project work to Blackboard. A note should be provided in the comments field on Blackboard to indicate where the documents for the deliverable may be found (e.g., “The documents for this deliverable may be found in the docs/deliverable3 folder.”).

4. SCM Schedules

This section establishes the sequence and coordination for the identified SCM activities in section 3 and for all events affecting the implementation of this document. It states the sequence and dependencies among all SCM activities and the relationship of key SCM activities to project milestones or events.

This section is in effect as of 1 April 2018.

4.1 Milestone End Dates

Deliverable 1 (Sprint 1) – 26 January 2018

Deliverable 2 (Sprint 2) – 23 February 2018

Deliverable 2 (Sprint 3) – 9 March 2018

Deliverable 3 (Sprint 4) – 2 April 2018

Deliverable 3 (Sprint 5) – 14 April 2018

Deliverable 4 (Sprint 6) – 10 May 2018

4.2 Activity Schedules

Beginning of Each Milestone

1. Establish configuration baselines
    - Initialize deliverable documents with appropriate filenames and content which conforms to past structure.
    - Product Owner sets priority of tasks for deliverable. Scrum Master and Product Owner negotiate on final priority of tasks for the deliverable.
    - Scrum Master and development team hold scrum meeting to establish deliverable goals within Scrum Backlog and Kanban board. The assignments of those goals to development team members are also established and recorded.
2. Implement change control procedures (optional)
    - Perform this function in accordance with 3.1.

End of Each Milestone

1. Start and end configuration audit
    - Perform this function in accordance with 3.4.
2. Establish team member roles for next deliverable
3. Release deliverable
    - Perform this function in accordance with 3.7.

5. SCM Resources

This section identifies the environment, infrastructure, software tools, techniques, equipment, personnel, and training necessary for the implementation of the specified SCM activities.

5.1 Environment, infrastructure, tools, techniques, equipment, personnel, and training

Development Environment: Android Studio 3.1

Version Control: Git

Configuration Management Repository: GitHub

Issue Tracking Repository: ZenHub, Sprint Backlog

Build Management: Gradle

Documentation: LibreOffice, Google Docs, and Microsoft Office word processing and spreadsheet software

5.1.1 Configuration identification

This activity primarily depends on the merge conflict detection features of Git and GitHub. Development team members should identify conflicts and breakages based on the output of these tools, and may subsequently trigger the activation of this activity. All team members should be trained in the proper use of Git and GitHub, and on how to identify and respond to merge conflicts. All team members are expected to have an internet connection. Other tools may be used to facilitate communication during this activity, such as text messaging or email.

5.1.2 Configuration control

Standard communication (text messaging, email) and development tools (text editor, IDE) are used to perform this activity. No special training is presumed necessary.

5.1.3 Configuration status accounting

This activity requires the use of GitHub's issue tracking features and an internet connection. Team members should be trained in the proper identification of the statuses of issues within GitHub. Team members should also have the ZenHub browser extension and be familiar with the use of Kanban boards.

5.1.4 Configuration evaluation and reviews

Team members must review initial requirements as specified in the project requirements documents which have been provided to all team members. These documents are in PDF format and require a PDF reader. Team members must also have word processing software capable of opening .docx files in order to perform review and correction on editable documents. Team members must have a Git client capable of pulling the latest revisions of documents from the GitHub repository. Team members should be trained in proper use of Git, including pulling, committing, and pushing to repositories.

5.1.5 Interface Control

Team members are expected to use the Android Studio IDE in order to ensure compliance with the Android and Google Maps APIs provided by Google for the versions used in this product. No special training is expected to perform this activity.

5.1.6 Subcontractor/vendor control

No tools, techniques, equipment, personnel, or training are required for this SCM activity, as this activity will not be performed.

5.1.7 Release Management and Delivery

This activity requires Gradle (or Android Studio) in order to perform APK generation. This activity should be performed by a member familiar with key signing procedures. Delivery involves pushes to the GitHub repository and requires familiarity with Git. Delivery also involves archiving the repository for project submission, and thus should be performed with a tool capable of creating ZIP archives by a member familiar with how to archive folders. Submission is performed on BlackBoard and requires a modern web browser. All activities presume an internet connection.

5.2 Key factors for infrastructure:

The infrastructure for SCM is planned and documented with consideration for functionality, performance, safety, security, availability, space requirements, equipment, costs, and time constraints. The infrastructure shall be maintained, monitored, and modified as necessary to ensure that it continues to satisfy the requirements of the SCM process.

Functionality: Tools identified in 5.1 must provide at least the minimum functionality to perform the activities identified in Section 3.

Performance: Tools identified in 5.1 must be conservative enough with regard to hardware requirements so as to run on all team members' development devices. It is presumed that all team members will develop on a computer no older than 8 years and have at least DSL-class internet connections.

Safety: No special safety requirements have been identified for infrastructure at this time.

Security: No special security requirements have been identified for infrastructure at this time.

Availability: All tools must either be installed to development team devices or be persistent online. Availability must be 99% to facilitate round-the-clock development.

Space Requirements: All activities occur in virtual space and are presumed to not require any physical space. Team members must carry development devices small enough that six team members may comfortably share a table during face-to-face meetings.

Equipment Costs: Infrastructure must be accessible to team members within the budget of an average college student in the United States. Tools and resources which are free are preferred.

Time Constraints: All implementation and documentation must be completed by Deliverable 4 deadline. Tools must be available in order to facilitate compliance with this deadline.

6. SCM Plan Maintenance

The Scrum Master shall be responsible for any and all information contained within the Software Configuration Management Plan, including monitoring the plan. Updates will be performed infrequently. The Scrum Master shall ensure that the Software Configuration Management Plan is updated (if any changes are necessary) at the both the beginning and end of each sprint. A revision history of this document shall be maintained within this document at all times, and shall reflect any changes made from previous versions.

Any proposed changes to the Software Configuration Management Plan shall be proposed during a Scrum meeting and must be approved by all members of the development team before integration. Changes to the plan will be executed by the Scrum Master and subsequently communicated to all team members either electronically or in person. This document will remain available to all team members within the configuration management repository.