MyCovidView

Software Configuration Management Plan

SCMP-1.0

Authors:

| Alif Fahmi bin Mazli | 17205512 |
| --- | --- |
| Muhammad Syaamil Fa’iq bin Zabisham | 17204215 |
| Daniel Wan Jonathan | 17203977 |
| Nima Razavi | S2000478 |
| Navid Razavi | S2110706 |

# Abstract

This document is the Software Configuration Management Plan (SCMP) of the MyCovidView project. This project is part of the assignments at Universiti Malaya. The document complies with the SCMP from the Software Engineering Standard, as set by the European Space Agency [[1].](#_heading=h.17dp8vu) This document contains information on the standards to be used for writing the documentation required for this project, as well as information about the processing and storage of these documents.

# Table of Content

[**Abstract 2**](#_heading=h.1i4ld5h6y2i)

[**Chapter 1 5**](#_heading=h.r0zk6p10j4j)

[Introduction 5](#_heading=h.9hmochsixwmj)

[1.1 Purpose 5](#_heading=h.3znysh7)

[1.2 Scope 5](#_heading=h.tyjcwt)

[1.3 List of definitions 5](#_heading=h.1t3h5sf)

[1.4 List of references 5](#_heading=h.2s8eyo1)

[**Chapter 2 6**](#_heading=h.1ksv4uv)

[Management 6](#_heading=h.vdwk57etgs5i)

[2.1 Organization 6](#_heading=h.2jxsxqh)

[2.2 Responsibilities 6](#_heading=h.3j2qqm3)

[2.3 Interface Management 7](#_heading=h.4i7ojhp)

[2.4 SCMP Implementation 7](#_heading=h.1ci93xb)

[2.5 Applicable Procedures 7](#_heading=h.2bn6wsx)

[**Chapter 3 8**](#_heading=h.1pxezwc)

[Configuration Identification 8](#_heading=h.x1vgppsk9mwn)

[3.1 Naming Conventions 8](#_heading=h.2p2csry)

[3.2 Baselines 8](#_heading=h.3o7alnk)

[**Chapter 4 9**](#_heading=h.ihv636)

[Configuration Control 9](#_heading=h.yo9t5tljqknz)

[4.1 Library Control 9](#_heading=h.1hmsyys)

[4.1.1 Web Development Library 9](#_heading=h.2grqrue)

[4.1.2 Mobile Development Library 9](#_heading=h.mw0lwinwck46)

[4.2 Media Control 9](#_heading=h.3tbugp1)

[4.3 Change Control 10](#_heading=h.nmf14n)

[4.3.1 Web Development Library 10](#_heading=h.37m2jsg)

[4.3.2 Mobile Development Library 10](#_heading=h.aki9xsxar26j)

[4.3.3 Document Library 10](#_heading=h.p2d1i7lhhlmv)

[**Chapter 5 12**](#_heading=h.111kx3o)

[Status Accounting 12](#_heading=h.pvvhxzu6xdxr)

[**Chapter 6 13**](#_heading=h.206ipza)

[Tools, Techniques and Methods 13](#_heading=h.abojywcd9eoq)

[6.1 Tools 13](#_heading=h.2zbgiuw)

[6.1.1 Git 13](#_heading=h.3ygebqi)

[6.1.2 GitHub 13](#_heading=h.sqyw64)

[6.1.3 Canva 13](#_heading=h.jugqi6g3kpjg)

[6.1.4 Figma 14](#_heading=h.saxeau791mzi)

[6.2 Techniques and Methods 14](#_heading=h.3hv69ve)

[6.2.1 Committing 14](#_heading=h.4h042r0)

[6.2.2 Tags and Branches 14](#_heading=h.3vac5uf)

[**Chapter 7 16**](#_heading=h.pkwqa1)

[Supplier Control 16](#_heading=h.321xv0q9xurq)

[**Chapter 8 17**](#_heading=h.1opuj5n)

[Records Collection and Retention 17](#_heading=h.an4law8lr1vd)

# Chapter 1

## Introduction

This chapter explains the purpose of this document as well as what the scope of this document is, that is, how it is related to other documents in the project.

### Purpose

The purpose of this document is to set rules and guidelines to which all project members should adhere. This concerns the versioning, identification and layout of all documents that are created for this project. All major documents should adhere to strict rules, while for other documents, such as files containing code, guidelines are set that are more loose.

This document should be read as a reference. It can be used when a developer or project member is not sure about how to do something as a mainstay.

### Scope

In this project, the following configuration items (CIs) will be produced:

* + - Software Configuration Management Plan (SCMP);
    - Software Project Plan (SPP);
    - Software Quality Assurance Plan (SQAP);
    - System Test Plan (STP)
    - Canva (Mobile app)
    - Code (Web app);
    - Product Backlog.

The ESA standard mentions a System Test Plan (STP) as well, but in our case this can be omitted.

### List of definitions

CI: Configuration Item

CM: Configuration Manager

SCMP: Software Configuration Management Plan

SPMP: Software Project Management Plan

SPP: Software Project Plan

SQAP: Software Quality Assurance Plan

STP: System Test Plan

BWF Bureau for Web Facilities

BMF Bureau for Mobile Facilities

### List of references

[1] ESA, ESA Software Engineering Standards. ESA, March 1995.

# Chapter 2

## Management

This chapter specifies which project members are involved in configuration management. Also, the responsibility a team member has in a function involved with configuration management is explained. Then, some general responsibilities are explained. Finally, a template for the creation of documents is given and some conventions are set about document creation, to ensure a consistent layout.

### Organization

The team members involved in configuration management are the configuration manager (CM) and vice CM. The project members that have volunteered to fulfill these roles are named in the SPMP[.](#_heading=h.3rdcrjn)

Other group members should always assist the CM and vice CM.

### Responsibilities

The CM and vice CM are responsible for copying documents to the master and archive library at the right moments, as mentioned in chapter [4.](#_heading=h.ihv636) They are in general responsible for the contents of the master and archive library. Another task for them is creating and updating document templates, although this task can be delegated.

The CM is primarily responsible for configuration management, although he or she can delegate tasks to the vice CM, in which case the vice CM is responsible. Whenever the CM is (temporarily) not available, the vice CM should take over the tasks of the CM, including the responsibility for these tasks.

Finally, all project members are responsible for the documents they work on. This means that they update the document status sheet and make sure the latest version of the document they work(ed) on is available in the development repository (refer to chapter [4,](#_heading=h.ihv636) section [4.1.1).](#_heading=h.2grqrue) When multiple group members work on the same document, they share the responsibility and additionally are responsible for the combined consistency of the document. Also, they should make sure that the repository remains in a “workable” state. That is, they should solve possible merge conflicts together.

### Interface Management

The deliverables will be developed using external tools and methods (refer to chapter 6), each respective web or mobile team will be in charge of providing external services and packages that ease the software development cycle. In the event of failure for any of the packages or services used by other group members, the CM will contact BWF or BMF in order to resolve the issue for the respective team.

In general, the CM can help other project members when they have trouble with some software that is used (refer to chapter [6).](#_heading=h.206ipza) However, the CM may delegate this task to other group members who have more expertise on the subject.

### SCMP Implementation

In this project, we will have only one SCMP document, contrary to what is described in the ESA standard [[1].](#_heading=h.17dp8vu) Thus, this document will not contain a plan for every phase of the project. Instead, refer to the SPMP for the planning of the phases.

### Applicable Procedures

1. Determine the Document Structure:
   * Identify the purpose and scope of the document.
   * Outline the sections, headings, and subheadings required.
   * Determine the desired formatting and styling guidelines.
2. Create the Document:
   * Open Microsoft Word or any word processing software that supports .doc files.
   * Begin creating the document based on the determined structure.
   * Follow the established formatting and styling guidelines.
   * Save the document in the .doc format.
3. Set Up a GitHub Repository (Only applicable for CM):
   * Create a new repository on GitHub or select an existing repository for the document.
   * Ensure that the repository is properly organized with appropriate folders and subfolders.
4. Add the Document to the Repository:
   * Navigate to the desired location within the repository where the document should be stored.
   * Upload the .doc file to the repository using the "Add file" or "Upload files" feature on GitHub.
   * Provide a meaningful and descriptive filename for the document.
5. Collaborate and Review:
   * Share the document's GitHub repository link with the team members involved.
   * Encourage team members to review and provide feedback on the document.
   * Utilize GitHub's collaboration features such as pull requests, comments, and issue tracking to facilitate discussions and revisions.
6. Document Version Control:
   * Consider using Git branches to manage different versions or variations of the document.
   * Encourage team members to make edits and updates on separate branches.
   * Merge approved changes into the main branch to maintain an up-to-date version.
7. Document Approval and Finalization:
   * Establish a process for document approval, such as requiring a certain number of team members to review and sign off on the document.
   * Once approved, make any necessary final edits or revisions to the document.
   * Update the document's version number and date to reflect the final version.
8. Document Access and Distribution:
   * Determine who should have access to the document repository.
   * Set appropriate permissions and access levels for team members.
   * Share the repository link or document file with relevant stakeholders or external parties as needed.
9. Document Maintenance:
   * Regularly review and update the document as required.
   * Utilize GitHub's version control capabilities to track changes and document the revision history.
   * Communicate any updates or changes to the team members who need to be aware of them.

# Chapter 3

## Configuration Identification

In this chapter, a versioning scheme is set. All documents created for the project should adhere to this scheme.

### Naming Conventions

All documents have a unique identifier. This identifier is title abbreviation-version, for example URD-0.1. The initial version of every document is 0.0. Then, after every formal review, the version number is bumped up with 0.1. A document that has been reviewed three times thus has version number 0.3. Only when the client or management has approved a document, the version number is bumped up to 1.0. Basically, the version number will not change after that, but it is theoretically possible that after that, some more changes are required and versions 1.x are created. After a second final approval (note that the fact that there is a *secondary* approval that is *final* already indicates that this is an exceptional situation), the version will become 2.0, et cetera.

Changes noted in the document status sheet in every document only mention changes since the last version. Older versions of the document can be found in the master (and archive) library (which will be discussed in more detail in chapter [4),](#_heading=h.ihv636) so all changes leading to the current version of a document can at all times be retrieved. In practice, even between-version changes can be retrieved from the development library, but this functionality will probably not be needed.

### Baselines

A baseline is a document that has been reviewed and approved externally. Baselines are stored in the development library, as discussed in chapter [4.](#_heading=h.ihv636) As described in the ESA standard, the CM makes sure that any version of every document can be directly downloaded from or rebuilt from the various libraries.

The ESA standard prescribes that new versions of management documents need to be created for every phase of the project. However, as the MyCovidView project is a relatively small project, we will have only one version of every document, including management documents, for the complete project. Phase-specific information will be added in the form of appendices to documents if needed.

# Chapter 4

## Configuration Control

This chapter describes how we handle different versions of CIs and where they are stored: we introduce the concept of libraries. Moreover, we describe how different libraries interact and what the role of the CM is in the management of these libraries.

### Library Control

All CIs that are created for the MyCovidView project have to be stored somewhere. We call a place where CIs are stored a *library*. Since MyCovidView is a relatively small project, there are three libraries created to store the CIs.

#### Web Development Library

The web development library is the library where all CIs for the web development project are stored initially. Documents in this library are generally under construction and can thus change a lot. From the development library, all versions of a CI that are stored in it can be retrieved and thus, every modification to a CI can be undone at any time.

In practice, the web development library is a single Git repository. Git is discussed in more detail in chapter [6.](#_heading=h.206ipza) We have the following repository:

* + - * MyCovidView: This repository contains all code that makes up the MyCovidView application.

The repository is stored on and accessible through GitHub (discussed in section [4.2](#_heading=h.3tbugp1) and chapter [6).](#_heading=h.206ipza)

#### Mobile Development Library

The mobile development library has the same function as the web development library, to store all the CIs belonging to the mobile development project. However, the difference here is instead of using Git repository, we will be using Canva.

In practice, the mobile development library is a Canva project that will contain pages that represent each page in the final mobile application. Canva is discussed in more detail in chapter [6.](#_heading=h.206ipza) We have the following project:

* + - * SPQ Mobile App prototype: This project contains all the pages design and external plugins that make up the MyCovidView mobile application.

The project is stored on and accessible through Canva (discussed in section [4.2](#_heading=h.3tbugp1) and chapter [6).](#_heading=h.206ipza)

### Media Control

The libraries mentioned in section [4.1 are](#_heading=h.1hmsyys), as also mentioned there, stored on GitHub and Canva for web development and mobile development respectively. GitHub is a commercial service that can be used freely as long as the repositories hosted by them are publicly accessible. They have offline encrypted backups of all repositories that can be used in case of complete failure. Also, because of the distributed nature of Git, every project member has a local copy of every library on his/her computer. Canva is a cloud-based design and visual communication platform. Canva requires its users to have an internet connection to access its services but they automatically save all changes in real-time.

Refer to chapter [6](#_heading=h.206ipza) for more information about GitHub and Canva and how the libraries can be accessed through it.

### Change Control

In this section, we discuss who can change the contents of the various libraries.

#### Web Development Library

Since the web development team is comprised of several smaller teams, each team is only allowed to change any CI in their respective github branch

This means that any group member can create new files, edit existing files and delete files from the web development library. There are two reasons why we allow this: first of all, the size of this project is relatively small. The entire group is working in the same room, so consultation can be done efficiently. Secondly, when a project member makes an error, this can be restored at any time because we use Git. Git also handles conflicts that may occur when two team members change the same file.

The general structure of the development library can however not be changed by group members: this is something the CM is responsible for. This is fair, as the CM is chosen by all group members.

#### Mobile Development Library

The permission to change any CI in the mobile development library is only applicable to the members of the mobile development team. This means that any group member can add pages, edit existing designs and update the design system from the mobile development library.

#### Document Library

The Document Library is designated to store Project Plans, SQA Plans, Testing Plans, and other related documents.

Only authorized members of the project team, including the project manager, quality assurance lead, and designated document owners, have the permission to change any document in the Document Library. These authorized members can create new documents, make edits to existing documents, and delete outdated or irrelevant documents as necessary. This approach allows for efficient collaboration and ensures that the document content remains up to date.

The decision to restrict access to the Document Library is based on the importance of maintaining version control and traceability for critical project documentation. By limiting access to authorized members, we can minimize the risk of accidental changes or unauthorized modifications. Additionally, this control measure enables us to track and assign responsibility for any changes made to the documents, which is crucial for compliance and audit purposes.

To maintain version control and facilitate efficient collaboration, we will utilize a version control system, such as a document management platform or a revision control system. This will allow authorized members to track changes, review document history, and restore previous versions if necessary. The specific version control system chosen for the Document Library will follow the naming convention stated in section 3.1.

# Chapter 5

## Status Accounting

In this chapter, we discuss how the configuration status of CIs is documented and reported in a clear way to group members and management.

The main tool used in order to keep track of CIs related to the web development and mobile development libraries is Trello, all teams should participate in using Trello as their guide line and indicate the start and finish of any CIs that they are in charge of.

As described in chapter [4,](#_heading=h.ihv636) there are three libraries in which a CI can reside. To report the status of all CIs in relation to the document library, the CM will maintain an additional document which focuses on recording different versions of documents. This document will list all CIs and for every CI a table indicating its status. Such a status could look as follows:

| *Version* | *Library* | *Date into master* | *Comments* |
| --- | --- | --- | --- |
| current | Development | - | - |
| 0.3 | Master | 02-Jun-2023 | - |
| 0.2 | Archive | 03-May-2023 | - |
| 0.1 | Archive | 25-Apr-2023 | - |

Note here that the top line will always be present, even when for instance in the above example, the current version is version 0.3. This is because changes in the development library are not tracked, but we want to make it explicit that there is a version sitting there, for completeness.

# Chapter 6

## Tools, Techniques and Methods

In this chapter, the various tools are discussed that are used in the MyCovidView project. Also, some methods to keep the various libraries tidy are discussed.

### Tools

The MyCovidView application is a web application that runs locally in the browser and a Canva prototype that runs on Canva’s cloud platform. In practice, the web version of MyCovidView is implemented using React framework and also Vite for the frontend build tool. The charts used for data visualizations are powered by Carbon Charts, a dataviz framework package that supports React projects.

#### Git

All code and documentation is stored in Git repositories. Git is a distributed, lightweight version control system. It enables all team members to work efficiently in parallel on the same documents/code base, even to some extent on the same file. Apart from enabling working in parallel, it also provides an implicit backup system, as every team member has a local copy of a repository.

#### GitHub

Git is a distributed system in which a central server is needed. This is the “main” repository, to which a number of project members can connect (this is called *cloning*, every project member has a *clone* of the “main” repository on his/her machine). GitHub is a (commercial) service that can host the “main” repository for us. As long as the repository is public, the service is free to use. The client has expressed that the project can be public, so we can use this free service from GitHub. Apart from providing “simple” hosting, GitHub also comes with a nice-looking web interface with which anyone can browse the repository. Also, there is a Wiki and bug-tracking system. Finally, GitHub provides a mechanism for hosting a website by means of creating a repository with a special name. We use this to build a website on which we can present some progress information to the client and maybe documentation or other project-related things.

#### Canva

Canva, a versatile online graphic design platform, has emerged as a popular tool for designers seeking to create visually captivating prototypes for mobile app designs. With its intuitive interface and vast array of design elements, Canva provides an accessible and user-friendly environment for crafting prototypes that effectively communicate the look and feel of a mobile application. Designers can leverage Canva's extensive library of pre-designed templates, diverse font selections, and a wide range of icons and illustrations to create interactive and engaging prototypes. Additionally, Canva allows for seamless collaboration and feedback sharing, enabling teams to iterate and refine their designs efficiently. By harnessing the power of Canva, designers can streamline the prototyping process and bring their mobile app ideas to life with ease and creativity.

### Techniques and Methods

In this section, we will discuss some methods we apply to keep the Git repositories tidy and the project manageable.

#### Committing

Committing changes is something that developers do a lot, so the conventions are simple, because a developer does not want to do something complex a lot. With this in mind, we have come to the following list of recommendations:

* + - * Make every relevant change to a repository a single commit. Do not combine multiple changes in a commit. This makes reverting changes easier.

**Example:** A commit wherein both a document and the general layout are changed, is not allowed. These should be two separate commits.

* + - * Always write a concise yet descriptive commit message for every commit. This makes it easier to read through the commit history and find relevant commits. Note that “concise” does not mean “at most two sentences”. You can definitely explain in some detail what you did. Just do not repeat the code you added, because it will be visible what you changed in your commit when looking at it in detail.
      * Refrain from committing binary files. These files will change a lot (probably) with every change, which does not work well in general.

#### Tags and Branches

In Git, it is possible to *tag* a repository at any moment. A tag is simply a reference to the repository at a certain point in time, with a label. You can list all tags that are present in a repository and easily revert to the point in time the tag was made. GitHub even shows all tags in a drop down menu on the website, so browsing the repository at the time a tag was created is easy. We use this feature to tag the repository whenever a new version of a CI is created. The tag should then have the identifier of that CI as a label, if the CI has a label. As an example for document repository: URD-0.3. The code in the repositories does not have an explicit identifier. When the code reaches a stable state, which should be at the end of each sprint, a tag with label v[version] should be created, for example v0.1. After each sprint, the version will be bumped with 0.1. After the last sprint of the project, the version will be bumped to v1.0.

Since we have separate repositories, the above strategy will result in the following:

* + - * The web repository will only contain code, no documentation and due to multiple teams the branches will be in the form of <group-number-<your title>>;
      * The repository project-docs will only contain documentation and thus only tags of the form [title abbreviation]-[version];

# Chapter 7

## Supplier Control

The tools listed in [6.1](#_heading=h.2zbgiuw) are all supplied by external suppliers, in some way or another. Tech- nically, Git is an open source project and thus not really supplied by anyone, but let us call that external as well, as no project member worked on Git.

We can then make the following overview:

| Tool | Discussed in Section | Supplier |
| --- | --- | --- |
| Git | 6.1.1 | Public Domain |
| GitHub | 6.1.2 | GitHub |
| Canva | 6.1.3 | Canva |
| Figma | 6.1.4 | Figma |

The above suppliers are all trusted by us, either because we have previous experience with software from the supplier or because we have tested the software in the research phase of the project and found that the software does what we want.

In general, when we consider using software from a supplier we trust, we just use the software. When we do not know the supplier, we test the software and if the software satisfies our tests and needs, only then we will use the software.

# Chapter 8

## Records Collection and Retention

In the development version, files can be deleted by any group member. Of course, all members need to agree on this decision, but theoretically, anyone can delete any file. Git retains files even after they are deleted, so files can be recovered at any time if needed.

Files in the master library can only be replaced with a newer version. At the same time, the old version will move to the archive library, where files cannot be removed. So, files placed there will be retained for the entire project.