Configuration Management Plan

1. Introduction

tl;dr Describe the plan's purpose, scope of application, key terms, and references.

- Overview description of the software project
- Relationship of SCM to the hardware or system configuration management activities for the project.
- The degree of formality, depth of control, and portion of the software life cycle for applying SCM on this project.

1.1 Identification

- Identification of the software configuration items (CIs) to which SCM will be applied.

- Identification of other software to be included as part of the plan (e.g., support or test software).

The SCM plan is for the PolyChord application's development, and the related testing, version control, continuous integration, and deployment software. For testing, Mocha, Chia, and Karma are used. For version control, Git and Github are used. For Continuous integration, Drone CI is used. For deployment, Docker and Docker Hub are used.

1.2 Purpose

The Configuration Management Plan lays out in detail how the DAJ team will manage the control of configuration items being developed in each phase. It defines both the policies and procedures for configuration management, and the infrastructure necessary to implement them throughout the phases of the project.

1.3 Scope

This version of the Configuration Management Plan is applicable to the Initial Design and Development phases of the PolyChord project. It may be modified for the following phases depending on the Configuration Management requirements for such phases. All PolyChord team members, while working on the PolyChord Project, will adhere to the approach outlined in this project.

1.4 Limitations and Assumptions

- Limitations, such as time constraints, that apply to the plan.
- Assumptions that might have an impact on the cost, schedule, or ability to perform defined SCM activities (e.g., assumptions of the degree of customer participation in SCM activities or the availability of automated aids).

2. Management

2.1 Organization

Organizational context (technical and managerial) within which the configuration management activities are implemented.

2.2 Responsibilities

For each board, list:

- Purpose and objectives
- Membership and affiliations
- Period of effectivity
- Scope of authority
- Operational procedures

2.3 Applicable policies, directives and procedures

External constraints placed on the SCMP Not sure what this would be in our case…

3. Activities

(What?) – Identify all activities to be performed in applying to the project.

3.1 Configuration identification

- Identify configuration items (events, items, procedures)
- Name configuration items (unique identifiers)
- Acquiring configuration items (physical procedures) - Not sure what this would be.

P100 PolyChord System

P110 PolyChord Frontend

P111 Tone.js (External Synthesizer Library)

P112 PapParse (External Parsing Library)

P113 Google Sign-on API

P120 PolyChord Backend

P121 Flask (Python Web Framework)

P122 Redis User Database

P200 PolyChord Development System

P210 Testing stack

P211 Karma (Javascript Test Runner)

P212 Mocha (Javascript Testing Framework)

P213 Chai (BDD/TDD Assertion Framework)

P220 Distribution System

P221 Docker Hub (Distribution Image Builder and Distribution Mechanism)

P222 Python Alpine Docker Image (Distribution Base Image)

P223 Gunicorn (Python WSGI HTTP Server)

P230 Version Control & Integration

P231 GitHub (Version Control)

P232 Drone CI (Continuous Integration Pipeline)

P300 Documentation System

P310 Drone CI (Continuous Integration Pipleine)

P311 Bash Scripts

P320 Markdown to PDF conversion

P321 Gotenberg API (Docker image thecodingmachine/gotenberg)

P330 PlantUML to PNG conversion

P331 Kroki API (Docker image yuzutech/kroki)

3.2. Configuration control

- Requesting changes
- Evaluating changes
- Approving or disapproving changes
- Implementing changes

Configuration control is taken care of within GitHub issues and pull requests: Change requests, evaluation, and approval or rejection all takes place within the GitHub issue

representing the proposed change, and once the change has been implemented, it's implementation is contained within a pull request (or series of pull requests), so that the changes' impact can be evaluated before it is merged.

3.3. Configuration status accounting

- Metrics to be tracked and reported and type of report.
- Storage and access control of status data.

3.4. Configuration evaluation and reviews

- At minimum an audit on a CI prior to its release.

- Defines objective, schedule, procedures, participants, approval criteria etc.

Configuration evaluation happens when the change is being discussed in it's related GitHub issue, and once an implementation is fleshed out, the review of the implementation is performed by the Product Owner on the Configuration change Pull Request. At least one of these steps (Evaluation and Review) occurs during each sprint per configuration change, though in some cases both may occur within the same sprint if the change is small in size.

3.5 Interface control

- Coordination of changes to CIs with changes to interfacing items outside of the scope of the Plan.

The majority of configuration items that are effected by changes outside the scope of the plan are the supporting software, which may receive updates and function changes over time. If possible, such updates should be taken care of within the normal software development cycle, though the software release reason should be taken into consideration when determining the urgency of the update: for example, feature updates should generally be considered low priority unless the new feature is needed for further software development, while security patches should be treated as high priority and incorporated in the very next development cycle, or even out of cycle if the security issue is major.

3.6. Subcontractor/vendor control

Incorporation of items developed outside the project environment into the project CIs.

3.7 Release Management and Delivery

Description of the formal control of build, release and delivery of software products.

4. Schedules

- Sequence and coordination of SCM activities.
- Relationship of key SCM activities to project milestones or events, such as:
    - Establishment of configuration baseline
    - Implementation of change control procedures

    - Start and completion dates for a configuration audit
- Schedule either as absolute dates, relative to SCM or project milestones or as sequence of events.
- Graphical representations can be used here.

5. Resources

- Identifies environment, infrastructure, software tools, techniques, equipment, personnel, and training.
- Key factors for infrastructure:
    - Functionality, performance, safety, security, availability, space requirements, equipment, costs, and time constraints.

Environment: Flask (web framework); Redis (database)

Personnel: Devin Christianson, Dawsin Blanchard, Jeremy Thiboutot, Alexander Revello, Alex Feren, Devin Merrow

- Identify which tools are used in which activity

6. Plan Maintenance

- Who is responsible for monitoring the plan?
- How frequently updates are to be performed?
- How changes to the Plan are to be evaluated and approved?
- How changes to the Plan are to be made and communicated?
- Also includes history of changes made to the plan.

The Configuration Manager is responsible for monitoring the plan, and updates to the plan should be performed as necessary, at a minimum once for every major Software release. Changes to the plan can be evaluated and approved through the same GitHub Issue and Pull Request mechanism as is used for Configuration Items, though these artifacts should be tagged as SCM, so that a history of the changes made to the plan can more easily be tracked.