Team Standards

SSDynamics

Oct 4, 2024

Carter K. Chas D. Connor A. Charles D. Brian Donnelly

Chris Ortiz
Senior Technologist, SSD Validation
Western Digital Corp., Flash Business Unit

John Lee
Senior Director, SSD Validation
Western Digital Corp., Flash Business Unit

Introduction

The Purpose of this document is to initialize and put into place Standards and Expectations for each team member. These expectations include, the team roles, team meetings, tools and document standards, and issue tracking. It is important to have coherent and strict rules in place to keep every team member accountable.

Team members and roles:

This section should introduce each team member and the role(s) you envision (based on your internal discussion of skills and desires) for that team member, as well as a clear description of the duties involved in fulfilling each identified role. Keep in mind that these roles outline lead responsibilities, but not exclusive ones -- it is expected that all team members will be deeply involved in every aspect of your project. Roles may include, but are not limited to (feel free to identify and discuss other roles you think would be helpful -- the following are meant to be helpful examples):

- Team Leader: The team member who coordinates task assignments, runs meetings, and reviews final documents to verify requirements as well as ensures work is progressing, and makes efforts to resolve conflicts.
    - Chas Diaz
- Customer Communicator: The team member who coordinates and conducts customer communications.
    - Charles Descamps
- Recorder: This team member maintains detailed meeting minutes.
    - Carter Kaess
- Architect: This team member is primarily responsible for ensuring that core architectural decisions are followed during implementation.
    - Charles Descamps
- Release Manager: This team member coordinates project versioning and branching, reviews and cleans up commit logs for accuracy, readability, and understandability, and ensures that any build tools can quickly generate a working release.
    - Connor Aiton
- Coder: It is expected that everyone will have a role in producing code. If possible at this early stage, you might specify *what parts* of the coding (backend, front-end, node.js, MSP430 programming, etc.) that individuals will lead on.
        - Code Sections:
            - TLA+ to PlusPy
            - PlusPy Modification and error traceback
            - Python to NVMe CLI
            - Logging
    - Charles Descamps
    - Carter Kaess
    - Chas Diaz
    - Connor Aiton

Team Meeting Expectations

- Group Meeting Times: We are to meet weekly at 6pm in the engineering building or online(if the need arises; additional meetings may be arranged).
- Client Meetings: We are to meet during Friday mornings at 9am on Teams, weekly.
- Mentor Meeting: We are to meet weekly on Wednesdays at 9am on the 3rd floor of the SICCS building.
- Meeting times can and will most likely change as the project progresses, every member should be informed and participate in the meetings.
- Agenda Structure:
  - All meetings will begin with a small overview of the subjects to be discussed.
  - All of the topics will be written in the agenda template that will be used to both keep the meeting on track and to write down the meeting minutes.
  - Then we will go through each topic in order.
  - After completing all topics we can discuss any other topics or questions that we thought of during the meeting.
- Minutes:
  - We have a meeting minutes agenda template which will be copied and used for each meeting.
  - https://docs.google.com/document/d/1rPFs4sM29iKi7hRhJ1UDYk-q7Wipy7W2g8TG8K5nvaM/edit?usp=sharing
- Decision-Making Process:
  - If the design chosen is small the decision making power will go towards however is in control over the task. The design decisions will usually be done by whoever is the most experienced or knowledgeable about the specific function, task, or system. However in the case of major disagreements, a 3/4 majority will be ideal and discussion should occur to attempt to achieve this. If there is still an issue or turmoil within the team, we will ask for outside guidance to get a set of fresh eyes on the problem.
- Attendance:
  - A member with permission is allowed to miss three meetings per semester.
  - Two unexcused absences will be allowed for the semester, however a reason must be given in the following two days for the absence, if no communication is made this will be brought up to the mentor, and a punishment will be decided based on the time and how much it negatively impacted the group.
  - Team members will only be reprimanded for tardiness after the first 10 minutes of a group meeting, and will be reprimanded if a member is not on time for a client or mentor meeting. Each team member will have three chances to be tardy to a group meeting without repercussions per semester.
  - There is the caveat that exceptions can be made for absences and tardiness if there is an emergency for the member under suspicion.
- Conduct:
  - The structure for team conduct in meetings is to have the meetings evolve around the agenda, off-task talk or discussion can be completed before or after the meeting has

started and the agenda has been completed. Members may need to complete work after or before the meeting but it should be not completed while during the meeting.
    - If there have been issues with a design or divided team, the first course of action is for a simple heads up and conversation to the person or people who the issue is with.
    - If the issue persists the topic will be brought up at a team meeting.
    - If the issue has become even larger than the scope of what can be solved in a team meeting, outside help will be brought in, this could still be in the scope of with the mentor and getting more clarifying questions of design choice from the client.
    - If there are issues with non-participating team members the course of action will go
        1. Give the team member a heads up.
        2. Talk about the topic at a team meeting, re-evaluate the workload, see if there is an issue with what the member has been tasked with, and give support.
        3. If the issue can not be resolved and there have been multiple interventions at team meetings, or repeated non-finished work, the topic will be brought up to the team mentor then with the CS Capstone Organizer.
        4. If there are still issues then the process of kicking the team member off the team will start.

Tools and Document Standards

Tools used, expectations for how they will be used, and related processes.

Version Control

GitHub (Code/Code Documentation):

ALL capstone related code (including forks of open source projects) will be under the SSDynamics GitHub organization. The main code base will have 3 main branches: dev/test/main, other branches for specific and notable features/experiments can be created as needed (stale branches will be closed). To improve and enforce code quality, GitHub actions/workflows will be used. Capstone related deliverables will be stored collaboratively on Google Drive.

- Each member will have a fork of the main code base to work on individually.
    - Keep private forked repositories private
    - Update fork by pulling the main repository before attempting to merge up to remote
    - Merge requests will not be accepted or reviewed if it fails automated tests (unless a flaw in automated testing is found)
        - If there is a flaw to fix in automated testing, track on task tracker with high priority
    - “Pull often, Commit often” for each subtask of a feature and at minimum submit one pull request back to main for each feature.
        - Unless highly related, do not submit a pull request for multiple features
- Branching strategy
    - New development is done in dev branch (automated linting may be added)
        - A sub strategy that is commonly used:

1. Someone may create a new branch locally for a feature being developed,
2. If other member pushes to organization repo first, member pulls dev branch from organization repo to local repo's dev branch
3. Then, member merges dev into feature branch, resolving merges (thus keeping dev branch history and conflicts more clean)

- Complete features should be pushed back to main repo for review and basic testing
- Testing branch may be used for unit/integration testing and ensuring that new features work as intended rigorously
- If everything is working and passes, Test may be merged into main tagged as a stable version
  - To further add to trackable stable version, use GitHub releases and packaging
- General workflow: local Dev→ PR to remote Dev→Test→ Main
- Improper use of branches and repos may result in rejected PRs and administrative repo actions by release manager to maintain order

Google Drive (Deliverables and Capstone Documentation):

- Use for tracking and storing all capstone related deliverables that will be used for grading
- Drive will have at minimum three folders under a shared root folder: assignments, meeting agendas and misc.
  - Assignments: Any deliverables that are being turned in
  - Meeting agendas: Team, Mentor and Client meeting goals and schedule
  - Misc: Any other files that facilitate team productivity and collaboration

Issue Tracking

Team Task Tracker (Larger trackable tasks) and GitHub Issues (Smaller programming subtasks from task tracker)

To track issues that arise or are foreseen in this project, we will be using a mix of two trackers in this project. For larger more complex tasks or non-coding related tasks, they will be tracked using the task tracker sheet. The smaller subtasks that may appear from breaking down tasks from team task tracker, even if small, will be in GitHub Issues.

- Use the organization's repository for GitHub Issues in all coding and code documentation issues
  - Issues will remain short and concise. These should be understandable from a glance.
  - Start broad then go narrow if a longer description is needed, each issue should contain the following:
    - Issue
    - Replication
    - Suspected cause if any
  - Smaller coding subtasks that may be broken down from the task tracker will appear here

    - As features or patches are pushed back to organization's dev branch, describe fix using a GitHub closing term referencing the issue number
    - For each coding issue, assign at least one member to the issue (most likely whoever is assigned to the parent task on the task tracker sheet)
    - When there are multiple GitHub Issues for each person, GitHub Project board should be used
        - Utilize to do, in progress and completed board effectively
        - Prioritize tasks, tasks depended on by other tasks will inherit priority
        - Estimate small/medium/large
    - Bad/Non-descriptive issues will be closed

- For deliverables and capstone documentation, use Google Sheets task tracker
    - Break larger complex tasks into smaller tasks that can be completed by an individual person
        - Tasks that are larger than what is possible completing in one sitting should be broken into further subtasks
    - Failure to accurately use use task report may result in inaccurate contribution and time estimates
        - Responsibility of proper use ultimately falls upon the person doing the task
    - Each tasks will be descriptive, the type of task needs to be specified and hours will be estimated
    - If multiple people are working on the same task, mention percentage of contribution in the comments is required on completion
    - Set a default internal due date due one week before the actual due date
        - With a grace period of ~3 weeks on team initialization, regular delays on internal due dates will be discussed as a group
    - Remember to update the status of your tasks
    - Should contribution representation conflicts arise, team as a whole will review the revision and commit history together

Word Processing and Presentation

- Google Docs, Slides and Sheets completion will be a PDF unless otherwise stated
    - Particularly for simple plaintext documents that require high collaboration
- Other forms of word processing like Docs, Overleaf, Sheets and Powerpoint may be used as more formatting is needed.
- For project related documentation, use markdown.

Composition and Review

- Split up the document into the distinct smaller parts in the task tracker to uphold responsibility

- On a word processor with no live collaboration, assign one person to handle compiling and tracking everyone's contributions as one file
- Assign proofreading in task tracker
- On draft ready for final review, use Internal deadline, which will be one week ahead of actual deadline
    - When the draft is ready, submit to lead editor: *Charles Descamps (tentative)*

Team Self Review

For self reviews, we will follow the form of a Biweekly time during regular meetings where each group member will be given at least 5 minutes to give their impressions of their performance. There is no written outline required but arriving to the meeting with an idea of what they wish to discuss about their own performance. After each group member discusses their own performance, other members will be able to chime in with agreement, comments, or other productive feedback.