Development Plan
SE 4G06

Team 3, Tangle
Calvyn Siong
Cyruss Allen Amante
Edward Gao
Richard Li
Mark Angelo Cruz

Table 1: Revision History

Date	**Developer(s)**	**Change**
9/20/2024	Cyruss	POC Plan and Expected Technology
9/21/2024	Cyruss	Introduction, Confidential Info, IP, and Copyright License
9/21/2024	Mark	Team Meeting, Communication Plan, and Roles
9/22/2024	Edward	Coding Standard, Team Charter
9/23/2024	Mark	Workflow Plan and Project Decoposition
9/23/2024	Edward	Team Reflection
...	...	...

The following document contains all information regarding the development plan of the TPG project. The following includes, but is not limited to details such as confidentiality, IP and copyright, team communication guidelines, expected technology, and overall project workflow. Changes within the details may be revised at any point throughout the project lifecycle and are expected to be updated accordingly.

1 Confidential Information

The project will be an extension of the open-source TPG project by Dr. Stephen Kelly. As a result, there is no confidential information to protect.

2 IP to Protect

For this project, there is no intellectual property to protect.

3 Copyright License

This project is adopting the MIT License, in which more information can be found here.

4 Team Meeting Plan

The team will hold weekly meetings on Mondays at the H.G. Thode Library from to 3:30 PM to 4:30 PM. Additional meetings will be scheduled if necessary as agreed upon by the team members. It is each team member's responsibility to provide updates regarding their tasks and raise any concerns during these meetings.

The weekly recurring sync with the stakeholders will hold on Mondays from 2:30 PM to 3:30 PM in ABB 533 beginning October 7. These meetings will consist of project updates, requirements gathering and asking relevant questions.

5 Team Communication Plan

The team's official communication will take place in the "TPG Capstone" Discord server. This will include but will not be limited to general communication and questions, daily updates, meetings, design suggestions and code reviews.

For formal conversation with the stakeholders and the project's supervisor, all communications will take place in the "TPG Capstone" Microsoft Teams channel or via email if necessary.

6 Team Member Roles

The roles of each team member are designed to be flexible. For now, each student will be responsible for one specific role and the developer role.

Name and Role	**Responsibilities**
Richard Li - Project Lead and Developer	• Primary liaison to the supervisor and stakeholders. • Design, Planning, Implementing, Testing, Reviewing Code Changes, and Documentation.
Cyruss Allen Amante - Meeting Chair and Developer	• Leads all team meetings including with stakeholders and the supervisor. • Design, Planning, Implementing, Testing, Reviewing Code Changes, and Documentation.
Mark Cruz - Academic Communicator and Developer	• Primary liaison to the professor and TAs. • Design, Planning, Implementing, Testing, Reviewing Code Changes, and Documentation.
Edward Gao - Repository Manager and Developer	• Maintains the repository from merge conflicts, and the organization of commits, PR's, and merges. • Design, Planning, Implementing, Testing, Reviewing Code Changes, and Documentation.
Calvyn Siong - CI/CD Manager and Developer	• Oversees and ensures the CI/CD pipeline is running smoothly and bug-free. • Design, Planning, Implementing, Testing, Reviewing Code Changes, and Documentation.

7 Workflow Plan

7.1 Git Workflow

Git will be the main version control system for this project. The main code of the Tangled Program Graphs (TPG) framework is stored in GitLab. However, due to the nature of the Capstone project, the team will be cloning the repository as a subtree of the Github repository to implement the two main goals of this project: support software engineering best practices and the integration of the machine learning framework into another agent-environment.

The Gitlab and Github repositories will be frequently synchronized using Github subtree commands to keep the repositories' commit history consistent. To support standard software engineering principles, the Github repository will use Github Actions for running Continuous Integration checks, different kinds of comprehensive testing, and generative documentation.

The general outline of the workflow is as follows:

1. Pull latest changes from the **main** branch
2. Create a feature branch
3. Implement code changes
    (a) Add and perform unit and integration testing
    (b) If applicable, add comments and documentation
4. Commit changes with descriptive messages
5. Open a pull request with the **main** branch as base
    (a) Provide detailed description of changes and testing performed
    (b) CI pipeline will be triggered via GitHub Actions
        i. If CI pipeline fails, fix the issue and revert back to Step 3
    (c) Other team members check if changes satisfies the acceptance criteria
6. Merge changes using **Squash and Merge** method

7.2 Issues

Github Issues will be used to monitor tasks and their progress. New templates have been created within the project to better distinguish each type of issue. Issues will be linked in every pull request that provides a solution to the issue. Additionally, every issue will be assigned to one team member and will have status that can be tracked through Github projects' Kanban board template. It is every team member's responsibility to update the status of their issues throughout the development lifecycle.

7.3 Milestones

Github Milestones will be used to group issues and pull requests to track the overall progress of the project. This would allow the team to analyze small increments of their progress and determine if the project is on track to be completed. Since Github Milestones can have deadlines, the team will assess every milestone once the deadline has been reached to gather feedback for iterative improvement.

7.4 Labels

Github Labels will be used to group different types of tasks and pull requests. These can be used to further categorize the priority and nature of each task, allowing the team members to distinguish their criticality throughout the duration of the project.

The following are labels that will be used in the future. These labels may be modified or replaced throughout the lifecycle of the project:

- high-priority - important and requires immediate attention
- medium-priority - important but not urgent
- low-priority - not important and lowest in priority
- feature - related to new features
- bug - related to software bug fixes
- documentation - related to documentation

8 Project Decomposition and Scheduling

8.1 Project Management

The Kanban board template in Github Projects will be used to organize and monitor existing issues. The GitHub project can be found in the TPG project's Github repository. This board will contain the opened issues that can be moved across different stages of the development lifecycle such as In Progress, In Review and Closed.

8.2 Project Decomposition

The team will be using Github Milestones to group issues based on their common goal to be used as feedback regarding future improvements. This can also be used to keep track of all the important deadlines in the course outline. The project can be decomposed through several milestones based on the major

phases of the development, for example Introduction and Research, and Development of CI/CD pipeline. The team must fully understand each large task or milestone to properly break them into smaller, manageable tasks and to set reasonable deadlines.

For example, a large task can be decompose into:

- Planning and Research - gathering of information and identifying the scope
- Implementation - designing, execution of actions and coding
- Verification - testing and validating of the implementation
- Review and Feedback - peer, stakeholder and supervisor review
- Final Approval and Publication - publishing of approved implementation

By having this approach of task decomposition, the team can be flexible with assigning roles to better manage the workloads that the project requires. This method can be applied to every aspect of the project to ensure that large tasks are consistently broken down into smaller ones.

8.3 Task Assignment

Each team member will have different strengths, weaknesses and interests that will be considered when assigning tasks. Issue size, which can be assigned within each Github issue, will be leveraged to ensure equal workload for every member. At the start of each milestone, the team must have a planning meeting to allocate tasks for the upcoming weeks. Each task must be evaluated one-by-one to fairly estimate the effort required per task. These tasks must also have a reasonable buffer time in the case of delays or unexpected challenges.

9 Proof of Concept Demonstration Plan

For the project's proof of concept demonstration plan, some significant risks may occur while in development and hinder the project's success. Here are the most significant risks:

- **GitHub / GitLab integration:** As of now, the TPG repository is currently on GitLab; however, GitHub is required to be utilized for this project. This TPG repository is still currently being contributed by both Dr. Kelly and graduate students. Failure to be able to integrate both changes could deal with some serious merge conflict issues.
- **Code Robustness:** Due to the large codebase and current unfamiliarity of the code, difficulties in testing may occur and the team may be unable to gather large code coverage within our testing suite. This will delay the goal of establishing a CI/CD environment for TPG and cause major hurdles in the remainder of the project.

- **Time-Consuming Refactoring:** A part of this project includes refactoring the code to follow standard software engineering principles, as this code is currently being modified throughout the project, refactoring may be a significant time-consuming task, preventing the completion of the other project goals.
- **Programming Language Learning Curve:** As most of the team for this project have not had experience with the C++ programming language, it will require some additional learning. This may slow down the progress of our project and prevent the proof of concept demonstration plan from being achieved.
- **Unable to create compatibility with the MuJoCo environment:** The project goal in the end is to allow the TPG code to be simulated within the MujoCo environment. If we are somehow unable to do this, we may need to reroute the focus of our project.

If these significant risks can be prevented or overcome for this project, then the proof of concept demonstration shall include the following to demonstrate the overcoming of the risk:

1. The overcoming of TPG integrated into a MuJoCo simulation environment by a basic demonstration of CartPole.
2. Walkthrough the CI/CD pipeline of the TPG project.

10 Expected Technology

These are the following expected technologies that the team expects to use for the project:

- **Git / GitHub / Gitlab:** For version control.
- **GitHub Projects:** Create an organized project space to track current progress.
- **LaTeX:** Allows for a professional-looking report, with easy-to-use syntax and commonly used in the industry.
- **Make:** Easily generates the LaTeX format into PDF for a report.
- **C++:** The existing codebase of TPG is written in this language.
- **VS Code:** Personal preference for members in this project and by the supervisor.
- **MuJoCo:** The main simulation environment the project aims to be compatible with.

- **Clang-tidy:** An open-source, widely used linter with an easy learning curve.
- **Bash:** For creating scripts.
- **Python:** For plotting analytical data for TPG.
- **Catch / Google Test:** For unit testing within the project.
- **Cppcheck / Gcov:** For static analysis and code coverage analysis.

11 Coding Standard

We aim for our project to be well documented and easy to understand. Git will be used for version control alongside GitHub for project management and CI/CD. For Git commit messages, we will follow the conventional commits specification at https://www.conventionalcommits.org/en/v1.0.0/. The provided issue templates will be used and additional templates will be added for bug reports and feature requests to ensure consistency in issue reporting. Since the TPG project is primarily written in C++, we will adopt the Google C++ Style Guide. For LaTeX documents, we will be following the file structure provided with the template and maintain consistent formatting conventions. Unit tests must be provided for all non-trivial functions and classes and integrated with the CI/CD pipeline.

Appendix — Reflection

Why is it important to create a development plan prior to starting the project?

Creating a development plan prior to starting the project is important because it helps in outlining the project's scope, objectives, and deliverables. It ensures that all team members are aligned with the project's goals and provides a roadmap for the project's execution. This planning phase helps in identifying potential risks and challenges early on, allowing the team to devise strategies to mitigate them. Additionally, a well-structured development plan facilitates better resource management and timeline estimation, which are crucial for the successful completion of the project.

In your opinion, what are the advantages and disadvantages of using CI/CD?

The advantages of using CI/CD include automated testing, which helps in identifying bugs early in the development process, and ensuring that the codebase remains stable and functional. CI/CD also promotes repeatable and consistent testing environments, reducing the chances of discrepancies caused by different development setups. For example, our team members own devices with different operating systems, so having a CI/CD pipeline ensures that the project can be tested consistently across all devices. Moreover, it can block pull requests until tests pass, ensuring code quality. However, the disadvantages include the potential for CI/CD configurations to become complex and time-consuming to set up and maintain. Additionally, if not managed properly, CI/CD processes can consume significant computational resources, leading to increased costs and longer build times.

What disagreements did your group have in this deliverable, if any, and how did you resolve them?

For this deliverable, we were in agreement on the selection of intellectual property and projet management policies. However, there was some back and forth regarding the expected technologies and coding standards to be used. Since, some of us are using Mac and others using Windows, we had to decide on which technologies to use so that our progress is platform-agnostic. Tools such as Docker, WSL, and Linux Virtual Machine was considered. But in order to come to a consensus, we determined that such a decision was out of the scope of this delieverable, and will be decided as more of us begin working on the project. Overall, we had productive discussions and reached a consensus on the important aspects of our project.

Appendix — Team Charter

External Goals

Our team has several external goals for this project. Firstly, we aim to present a standout project at the Capstone EXPO that showcases our technical abilities and innovative thinking. Winning a prize at the EXPO would be a significant achievement and a testament to our hard work. Secondly, we want to develop a comprehensive project that serves as a strong talking point in future job interviews, demonstrating our skills and experience to potential employers. Lastly, we aim to achieve an A+ in this course, reflecting our dedication and the quality of our work.

Attendance

Expectations

Our team expects all members to arrive on time for meetings, with a grace period of up to 5 minutes. Consistent lateness should be explained, and team members are expected to attend at least 80 percent of all meetings. Attendance will be tracked on GitHub, including team meetings, meetings with our supervisor, and TA. Active participation in discussions and decision-making processes is more important than mere attendance.

Acceptable Excuse

Acceptable excuses for missing a meeting or a deadline include medical emergencies, pre-approved commitments (with prior notification to the team), and technical difficulties such as internet outages or device failures. Unacceptable excuses include lack of interest in the project, poor time management, and procrastination.

In Case of Emergency

In case of an emergency, the team member must inform the team as soon as possible and provide an acceptable excuse. They should also give an estimated time frame for resolving the emergency and propose a contingency plan for covering their responsibilities.

Accountability and Teamwork

Quality

Our team expects high-quality preparation for meetings and deliverables. Each member should come prepared with their assigned tasks completed to the best of their ability, ensuring that all contributions are thorough and well-researched. Moreover, we will check each other's work through assigning reviewers to each

other's pull requests. The review process will be based on the provided rubric to ensure that all contributions meet the required standards.

Attitude

Team members are expected to contribute positively to the team environment, respecting each other's ideas and cooperating fully. We will adopt McMaster's student code of conduct to ensure respectful interactions. For example, Appendix A in the Code of Student Rights and Responsibilities requires students to refrain from using derogatory or offensive language, maintain a positive and respectful attitude, and refrain from engaging in any form of harassment or discrimination. If a team member violates these rules, the team will address the issue through a formal appeal process. We expect all team members to adhere to these standards at a bare minimum. What we hope to foster is an environment where team members feel inspired to contribute their best work and support each other in doing so.

Stay on Track

To keep the team on track, we will use project management metrics such as attendance, commits, and issue tracking. These metrics will be used as a rough indicator of team member's contributions. We recognize that these metrics are not perfect. For example, deleting 100 lines of code is often more productive than writing 100 lines of code, but this is not reflected in the commit count. We will also consider other factors such as the complexity of the changes made and the number of reviews a team member has conducted. Nonetheless, members who consistently meet or exceed their targets will be recognized and rewarded, while those who fall short will face consequences such as bringing coffee to the next meeting or scheduling a meeting with the TA or instructor. We will set clear targets for attendance and commits, and ensure that everyone is aware of their responsibilities.

Team Building

To build team cohesion, we will schedule regular fun activities and group rituals, such as team lunches or game nights. These activities will help strengthen our bond and improve our collaboration.

Decision Making

Our team will make decisions through consensus whenever possible. If consensus cannot be reached, we will hold a vote. In case of disagreements, we will follow our conflict resolution plan, which includes discussing the issue openly and seeking input from our supervisor or TA before making a final decision.