Hazard Analysis
TPG

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

Table 1: Revision History

Date	Developer(s)	Change
10/25/2024	All members	Revision 0 of Hazard Analysis
03/29/2025	Richard	Edited traceability and rewording of safety requirements
03/31/2025	Richard	Made FMEA table more coherent for TA Feedback
04/02/2025	All members	Revision 1

Contents

1 Introduction

This document describes the potential hazards of the Tangled Program Graphs (TPG) capstone project. Details include information such as the scope and purpose of the hazard analysis, system boundaries and components, critical assumptions, Failure Mode & Effects Analysis (FMEA) tables, newly discovered safety and security requirements, and a roadmap describing the timeline for the Capstone project's safety requirement implementation.

The definition of a hazard utilized in this document is based on the definition by Nancy Leveson:

A hazard is any potential condition within the system that could harm or damage the project. This includes unexpected conditions such as security risks and safety hazards (**?**).

For some terminology used within the Hazard Analysis, such as **agent** or **environment**, please refer to the SRS documentation (Tangle, 2024).

2 Scope and Purpose of Hazard Analysis

The document evaluates hazards that arise from (a) Tangle, (b) its integration with MuJoCo, and (c) the CI/CD pipeline during its development.

The purpose of the hazard analysis is to identify any hazards that may occur within the system's components. The hazards will be analyzed to find the reasons and causes of the failures, ultimately leading up to the creation of mitigation strategies in an attempt to reduce the hazard and its potential damage if such hazard arises. The analysis will result in the creation of new safety and security requirements for the project that had not been added in Revision 0 of the SRS.

3 System Boundaries and Components

The hazard analysis will be conducted on the following system which will have the following boundaries and components:

1. The TPG codebase includes the following components:
   (a) The reinforcement learning algorithm is responsible for training the agents.
   (b) The environment in which agents will be interacting with.
   (c) Scripts that are run to initialize the interaction between agents and environments.

(d) The OpenGL visualizer will allow for visualization of the reinforcement learning interaction.

2. The device in which the repository is being run from.
3. The interface integrated with TPG and MuJoCo.

4 Critical Assumptions

Here are the following critical assumptions that will be made:

- Dependent libraries and frameworks are stable and any current issues or bugs don't affect the product's functionality.
- The MuJoCo environment functions as expected, following the documentation's specifications.
- The operators and users of the systems are using the product in its intended form.

5 Failure Mode and Effect Analysis

Design Function	Failure Mode	Effect of Failure	Causes of Failure	Detection	Recommended Actions	SR	Ref.
GitHub Actions CI/CD Pipeline	Invalid Config File	Merged code fails integration	Improper syntax or misconfiguration in the file	Error messages during CI run	Validate config file syntax prior to merging and introduce automated config checks	SR-1	H1-1
GitHub Actions CI/CD Pipeline	Dependency Compatibility Issues	Build failure and uncompilable code	Mismatched or outdated dependency versions in the environment	CI error logs and build failure alerts	Enforce dependency version constraints and perform environment validation tests	SR-1	H1-2
GitHub Actions CI/CD Pipeline	Invalid Testing Setup	Test suite failures leading to undetected issues	Integration of breaking changes or misconfigured test scripts	Error messages and test reports in CI logs	Enhance test coverage and implement a pre-merge testing pipeline	SR-1	H1-3
Experiment Processing	Incomplete Experiment Data	Missing or partial output logs affecting analysis	OS-specific issues or failures in parsing input data	User reports and error handling messages	Validate input data, add redundancy in logging, and rerun experiments if needed	SR-2	H2-1
OpenGL Integration	Software Incompatibility with OS	Application failure to run and/or incorrect visual output	Incorrect bindings used or OS lacking required OpenGL support	Error messages on startup, user reports, and cross-platform testing outcomes	Provide alternative output (e.g., fallback to video rendering or text-based interface) and notify the user that the feature is unavailable on this platform	SR-3	H3-1
Open Source Contribution	Unauthorized Code Changes	Potential security vulnerabilities and instability	Merging of untrusted contributions or unreviewed code with known issues	CI/CD reviews, branch protection mechanisms, and monitoring for anomalous commits	Use strict code reviews, revert malicious changes via Git control, and enforce contributor trust policies	SR-1	H4-1

Table 2: Failure Mode and Effect Analysis

6 Safety and Security Requirements

SR-1:

The system shall have a CI/CD pipeline that goes through the essential steps (build, test, linting, etc.) whenever new code is committed.

Rationale: Maintaining a large code base with numerous contributors needs to be standardized. Want to minimize time spent onboarding and debugging nagging issues. Running through this pipeline upon each commit/pull request ensures the code base is properly synchronized across contributors and a seamless developer experience.

Associated Hazards: H1, H4-1

SR-2:

The system shall inform contributors of errors that occur during the running of an experiment between the agent and the environment. It is the contributors responsibility to ensure the validity of the experiments they want to run between the agent and the environment.

Rationale: Since the contributor has to manually specify parameters to run an experiment, it is very possible an incompatible parameter is run. The system shall inform the contributor which parameter(s) are leading to an incomplete experiment and to highlight these in the input data.

Associated Hazards: H2-1

SR-3:

The system shall ensure consistent performance across different operating systems functions.

Rationale: Contributors use a variety of platforms to do development (e.g. Linux, Windows, MacOS). Ensuring each of those user groups is able to onboard quickly and use the framework lowers the friction to access this project and enables more development from a wider user base.

Associated Hazards: H3-1

7 Roadmap

This hazard analysis has introduced new safety and security requirements as seen in the sections above. The majority of these requirements will be attempted to be implemented into the design of the project by the Revision 0

demonstration in February 2025. At the moment, the team will prioritize implementing safety and security requirements SR-1 and SR-2.

Justification for SR-1 and SR-2 Prioritization:

- **SR-1 (CI/CD Pipeline):** Establishing a robust CI/CD pipeline is crucial for maintaining code quality, ensuring consistent builds, and streamlining the development workflow. By addressing SR-1 early, we can prevent integration issues, automate testing, and facilitate collaboration among developers. This proactive approach will minimize the risk of introducing errors into the codebase and improve overall project stability.
- **SR-2 (Experiment Error Reporting):** Providing contributors with clear and informative error messages during experiment execution is essential for accelerating the development and debugging process. By addressing SR-2 early, we empower contributors to identify and resolve issues more effectively, leading to faster iteration cycles and improved experimental outcomes. This proactive approach will minimize wasted time and effort and ensure the validity of experimental results.

The remaining requirements will be addressed in subsequent iterations of the project, with the specific sequence determined based on ongoing risk assessments, resource availability, and stakeholder feedback. However at any moment throughout the development of the project, some requirements may be ultimately decided not to be pursued due to potential time constraints.

Appendix — Reflection

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. **What went well while writing this deliverable?**

   While writing this deliverable, the team communicated well on the progress of the hazard analysis. As items were being completed, constant updates between members were given to give traceability and allow team members to review sections written by other members.

2. **What pain points did you experience during this deliverable, and how did you resolve them?**

   Some pain points that were experienced during this deliverable were the constraints of time management. Although reading week was in effect, there were many other items to consider such as midterms and assignments throughout the hazard analysis. This was somehow resolved by delegating tasks between members to balance the time managed on the project itself and the documentation. Another pain point that was experienced is that there was initially a lack of understanding of what hazards were between members. Some had initial thoughts and ideas of what should be considered, while others had different ones. In the end, this was resolved by communicating with the team to come to a consensus on what the definition of a hazard was and the overall scope of what this hazard analysis would become.

3. **Which of your listed risks had your team thought of before this deliverable, and which did you think of while doing this deliverable? For the latter ones (ones you thought of while doing the Hazard Analysis), how did they come about?**

   Before this deliverable, we definitely thought of the GitHub Actions related risks since many of us have had experience with making/using DevOps pipelines. The ones we thought of are related to the OpenGL Interface and Experimentation related risks because we did not have prior background to the development and needed our supervisor (Dr. Kelly) to explain how the TPG framework works and how he currently uses it.

4. **Other than the risk of physical harm (some projects may not have any appreciable risks of this form), list at least 2 other types of risk in software products. Why are they important to consider?**

   Two other types of software product risk include security and compliance risks. Security risks are important to consider to protect the product itself. If some malicious actors are looking to cause harm to the system, mitigation strategies and prevention are important to keep the system and any critical data such as user information, safe as well. Compliance risks are important to consider as if any crucial changes are made to the system, it is necessary to make sure all laws and guidelines such as PIPEDA, are still being followed.

References

Team 3 Tangle. System requirements specification. https://github.com/TPGEngine/tpg/blob/main/docs/SRS/SRS.pdf, 2024.