Hazard Analysis
CVT Simulator

Team #17, Baja Dynamics
Grace McKenna
Travis Wing
Cameron Dunn
Kai Arseneau

Table 1: Revision History

Date	**Developer(s)**	**Change**
Oct 23	All	First Draft

Contents

1 Introduction

A hazard refers to any circumstance or occurence that has the potential to endanger, damage or cause a system to fail. Hazards may arise from errors in design, implementation or operation and may affect both the system and the user.

In this document, we will examine the possible risks related to the creation and application of a software system intended to replicate the functionality of a Continuous Variable Transmission (CVT) for the car of the McMaster Baja Racing team. By systematically examining the components and assumptions of the software, this hazard analysis attempts to make sure that the finished solution satisfies performance objectives and is robust enough to manage the intricacies inherent in CVT modeling.

2 Scope and Purpose of Hazard Analysis

The purpose of conducting a hazard analysis when designing software that aims to improve the McMaster Baja team's Continuous Variable Transmission(CVT), is to identify and become aware of potential risks that could negatively affect the project. A hazard such as producing flawed simulations could delay the tuning process and create unreliable and incorrect outputs thus negatively impacting the vehicle performance and efficiency. These performance flaws could then lead to losses during Baja competitions. Through exploring the possible hazards involved, the project's code quality can be enhanced, potential risks can be mitigated and it ensures that the system will function as intended.

This document assumes that the user has access to a hard drive with sufficient storage space to download the software, an appropriate computer that meets the system requirements and a stable internet connection when downloading the software. [You should say what **loss** could be incurred because of the hazards. —SS]

3 System Boundaries and Components

[Dividing the system into components will help you brainstorm the hazards. You shouldn't do a full design of the components, just get a feel for the major ones. For projects that involve hardware, the components will typically include each individual piece of hardware. If your software will have a database, or an important library, these are also potential components. —SS]

4 Critical Assumptions

[These assumptions that are made about the software or system. You should minimize the number of assumptions that remove potential hazards. For instance, you could assume a part will never fail, but it is generally better to include this potential failure mode. —SS]

A1: Users will not intentionally misuse the software or purposefully try to find unsafe settings.

A2: The system that the software is running on will be not compromised and will function in a standard manner.

A3: Third party software and libraries used are trusted to be safe and free of vulnerabilities.

5 Failure Mode and Effect Analysis

[Include your FMEA table here. This is the most important part of this document. —SS] [The safety requirements in the table do not have to have the prefix SR. The most important thing is to show traceability to your SRS. You might trace to requirements you have already written, or you might need to add new requirements. —SS] [If no safety requirement can be devised, other mitigation strategies can be entered in the table, including strategies involving providing additional documentation, and/or test cases. —SS]

6 Safety and Security Requirements

[Newly discovered requirements. These should also be added to the SRS. (A rationale design process how and why to fake it.) —SS]

7 Roadmap

[Which safety requirements will be implemented as part of the capstone timeline? Which requirements will be implemented in the future? —SS]

Appendix — Reflection

[Not required for CAS 741 —SS]

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