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.

3 System Boundaries and Components

This system is divided into three main components:

User Interface Component
This component is responsible for taking user input, providing the output to the user, and displaying the simulation of the 3D models.

Backend (Math) Component
This component handles the mathematical calculations required for the program. The calculations will be done using Python and specifically the libraries NumPy and SciPy.

Simulation Component
This component displays various models that are part of the CVT system, moving in accordance with the simulation.

4 Critical Assumptions

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.

A4: Users will not leak information received from the application to other teams or organizations.

A5: The software will not be connecting to the internet or any other network.

A6: The application will not be collecting any user data or personal information.

5 Failure Mode and Effect Analysis

Design Function	Failure Modes	Effects of Failure	Causes of Failure	Detection	Recommended Action	SR	Ref
Simulates the CVT system	Improper Load Balancing	• Increased wear over time leading to premature failure. • Suboptimal power transfer between the engine and the wheels.	• Incorrect distribution of forces across the pulleys due to inaccurate mass or load models. • Errors in simulating the torque requirements based on the load. • Inaccurate assumptions about load transfer between the pulleys.	• Erratic changes in acceleration of the vehicle. • Excessive load on one component compared to another. • Validate with torque data.	• Adjust load model to more accurately represent the vehicle. • Add dynamic load balancing checks to the simulation.	1	R18
	Inaccurate power transfer between pulleys (belt dynamics)	• Increased wear over time leading to premature failure. • Suboptimal power transfer between the engine and the wheels.	• Inaccurate modelling of belt tension and friction. • Incorrect pulley geometry, taking advantage of belt assumptions to lead to impossible configurations.	• Sudden drops in power transfer, or incredibly high torque differences. • Unreasonably high belt tension or slack.	• Dynamically check if belt tension is within reasonable limits.	2	R19
	Poor numerical precision	• Inaccurate simulation results, leading to incorrect design decisions • Inconsistent behaviour of the system, leading to confusion and potential damage.	• Insufficient precision in the ODE solver. • Inappropriate time-step size.	• Erratic in times of high acceleration or deceleration. • Results that are inconsistent with the expected behaviour of the system.	• Manually verify precision is sufficient by a factor of safety. • Validate the ODE library's precision and accuracy.	3	
	Insufficient frictional forces	• Excessive belt slippage or tension, leading to wear and poor efficiency. • Simulations provide overly generous results.	• Incorrect assumptions about the frictional forces between the belt and the pulleys. • Incorrect friction models for the pulleys due to false geometry or temperature assumptions.	• Consistently high forces that would be reduced by friction. • Slip conditions met excessively while not being accounted for. • Compare to belt slip and RPM data.	• Investigate friction assumptions and models. • Validate simulation in simple cases to ensure friction is accounted for.		R20
	Software Crash	• Loss of data and work. • High frustration and loss of trust in the software.	• Memory leaks or poor memory management. • Incorrect handling of edge cases, such as a division by 0. • Poor error handling, leading to system crashes rather than graceful failures.	• Application crashes during normal use. • Program does not output data. • Unusually high memory or CPU usage. • Logs showing unhandled exceptions or errors.	• Implement robust error handling. • Conduct memory profiling and avoid loading large or unnecessary data. • Test the software in edge cases to ensure it fails gracefully.	4	R21

Table 2: Failure Modes, Effects, and Recommended Actions for Insulin Delivery

6 Safety and Security Requirements

SR 1 The simulation will dynamically check load balancing and adjust the computations accordingly.
SR 2 The simulation will dynamically check that the belt tension is within reasonable limits.
SR 3 Ensure that the precision exceeds the standard 32-bit level by using 64-bit to ensure greater accuracy.
SR 4 The software shall implement robust error handling mechanisms that detect and handle at least 95% of all expected and unexpected errors, ensuring that no more than 5% of critical system failures result in unhandled exceptions or system crashes.

7 Roadmap

We will implement SR1, SR2, SR3 and SR4 as part of the capstone timeline. We have chosen to implement all of these is that we believe that they are all relevant. We also believe that they are all attainable within the project window.

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?

   One strategy that went was dividing the roles for the initial sections of this deliverable. The sections Introduction, Scope and Purpose of Hazard Analysis, System Boundaries and Components and Critical Assumptions were divided among team members. This allowed each team member to tackle a portion independently before reconvening to review everyone's contributions. This approach made completing this document more efficient. For the FMEA table, we decided to work together as this was a large part of the document it was important to make sure all team members were on the same page. This collaborative approach was proved effective, especially for generating causes of failure, detection methods, and recommended actions for each failure mode. Ultimately, this strategy ensured a thorough analysis but also helped us derive new failure modes, and requirements, which improved our understanding of the system.

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

   One pain point we had while completing this document was debating on what we would define as a failure mode. Initially, we had two broad categories, but there was some debate about breaking these down further. Some team members felt that broad failure modes would be more manageable, while others advocated for more specific ones for clarity. Our team debated what was considered a failure mode and what was considered a cause of a failure. After discussing the pros and cons, we agreed to break down these failure modes, as it would enhance system clarity and provide a more accurate description. This compromise allowed us to move forward cohesively and improved our system documentation as well as improve the hazard analysis document.

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 starting this deliverable, we had previously identified the risks of simulation failures and software crashes. While completing this deliverable, we realized that simulation failures was too broad and included multiple potential failure modes. Breaking it down allowed us to better identify and address specific risks, leading to a clearer and more detailed analysis. Our system now has a total of 5 failure modes improper load balancing, inaccurate power transfer between transfer between pulleys (belt dynamics), poor numerical precision, insufficient friction forces and software crashes. These various failure modes were the result of the pain points discussed above. By completing this document and sparking conversation about a failure mode vs a cause of a failure we were able to further derive failure modes thus making our system more accurate and further defined.

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 risk that should be considered when designing software products are data security risks and slow response times. These are very important to consider as they directly affect the users of the system and the system functionality. Data security risks such as data breaches compromise sensitive user information leading to legal and financial damages. Slow response times lead to negative user experience and an unreliable system. If a system is slow or unreliable, users are more likely to abandon it for alternatives. By addressing both security and performance ensures a reliable, user-friendly product that can maintain trust and meet operational demands.