Hazard Analysis
Mechatronics Engineering

Team #1, Back End Developers
Jessica Bae
Oliver Foote
Jonathan Hai
Anish Rangarajan
Nish Shah
Labeeb Zaker

Table 1: Revision History

Date	**Developer(s)**	**Change**
October 19th, 2022	Jessica Bae Oliver Foote Jonathan Hai Anish Rangarajan Nish Shah Labeeb Zaker	Initial Documentation

Contents

[You are free to modify this template. —SS]

1 Introduction

[You can include your definition of what a hazard is here. —SS]

2 Scope and Purpose of Hazard Analysis

3 System Boundaries and Components

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]

5 Failure Mode and Effect Analysis

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]

- The device will not modify user data unnecessarily.
- Only authorized persons will be able to unlock locked-down user data.
- The device will have one channel to handle device logic, and a seperate channel to handle error handling.
- The error codes must be distinct and categorized so that operators may understand the importance of certain errors.
- The error handler must have safeguards in place to ensure it never returns a null value.

7 Roadmap

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

Appendix — More Than Just a Document

The Back End Developers would like to take a section of this document to describe the personal importance of hazard analyses (and other relevant documentation) to the team. A perfect example of this is the was the Therac-25.

The Therac-25 was a radiotherapy machine designed by Atomic Energy Canada Limited (AECL) in 1982. It was designed to treat cancer by sending beams of high-energy particles (either electrons or x-rays) through a patient's tumour, thereby killing it. This machine was one of the first of its kind to be computer controlled; using software to direct its functions rather than interdependent physical mechanisms. [2] Considered state-of-the-art at the time, the Therac-25 promised to revolutionize radiotherapy.

However, there were problems. Operators of the machine would find that they encountered many error messages from the computer console, of which none were explained by AECL. These errors were non-descript, and often simple read 'MALFUNCTION' followed by a number. The same operators stated that they had grown used to these messages, and often ignored them and proceeded with treatments.

In addition, none really knew how the Therac-25 was programmed. The original developer had developed the software for the previous Therac-6 and

Therac-20 models of radiotherapy machines, and had since left the company. AECL had directly ported the software into the Therac-25, and did not test the combination of the software and the new hardware properly.

Cancer patients treated using the Therac-25 began to report issues. They entered the machine thinking that the procedures would be painless (which is usually true for radiotherapy). However, they often felt powerful shocks and intense burning sensations around the areas that the Therac-25 would administer radation. Later on, these same patients would fall heavily sick with symptoms identical to those with acute radiation poisoning. Some patients had to have entire limbs removed due to intense radiation damage, and none knew why. The AECL assured the operators that the machine was incapable of administering deadly doses of radiation, and attested to its safety. [3] In reality, the machine was exposing patients to doses of radiation hundreds of times greater than what was prescribed. [1] Between 1985 and 1987, three people died of radiation exposure due to the Therac-25, and two others sustained lifelong injuries.

A later investigation revealed that when operators changed from the electron mode of the machine to the x-ray mode in a certain way, the Therac-25 would fail to move a dampening shield between the fully powered x-ray and the patient. This dampening shield served to lower intensity of the x-ray beam to safe levels. The software would correctly detect an error, but simply displayed a window with the text 'MALFUNCTION 54'. Having grown used to these messages, the operators would proceed with treatment without the shield in place. In addition, a flag variable which would instruct the machine to perform safety checks was designed to be set by incrementation rather than being set to a fixed value. This flag would often overflow from its maximum value of 255 to zero, which instructed the machine to bypass safety checks. A later review would reveal that the Therac-25 had eleven seperate critical engineering and institutional failures which would have put a patient's life in danger. The combination of these factors caused the Therac-25 to give unlucky patients massive radiation overdoses.

In response to this incident, the International Electrotechnical Commission (IEC) released standard IEC 62304; a standard which details development lifecycle standards specifically for software present on medical devices. It also describes the dangers and guidance regarding Software of Unknown Pedigree (SOUP).

The worst part is that **none of this had to happen.** Should a hazard analysis have been performed, the team at AECL who designed the Therac-25 would have known that unnaccounted arithmetic overflows could bypass safety features. They would have known that the software written by the original Therac-6 and Therac-20 developer could not be maintained and troubleshot even for the most basic errors. They would have known that non-descript errors in a system like the Therac-25 could potentially be life-threatening. They would

never have assured machine operators that overdoses were impossible.

Had they done so, they would have never decided that a direct port of the Therac-20 software would have been appropriate for the system they were designing. They may have thrown out the software all together and started from scratch. This would have taken much more time and many more resources than a simple port, but that team had a responsibility to ensure that the potential hazards of the Therac-25 were accounted for and mitigated. It would have been a good decision. As they did not make it, three patients died from treatments meant to save their lives. [4]

Events like these are tragedies, and are a sobering reminder of the responsibilities that engineers have to the products they design, the users of said products, and to the world that their systems exist in. **A hazard analysis is an essential document to ensure that catastrophically horrifying failures do not occur as a result of an engineer's work.** It has saved lives before, and will continue to save lives in the future. The overconfidence, lack of respect for procedure, and immature state of established engineering standards resulted in one of the worst disasters in medical history.

This section may seem depressing, overly-dramatic, or entirely unnecessary to some. But to the Back End Developers, the Therac-25 is an example of what can go terrifyingly wrong when engineers do not perform their due diligence. Cases like this remind the team members that they could be the next ones in a position to make safety-critical decisions in their engineering designs. This case also should remind all engineers that **there is a reason to all of this procedure.** It is common knowledge that engineers can be quite arrogant at times and dismiss documentation as frivolous and obstructionary to their 'real work' of coding and designing. However, cases like this prove the opposite. **Documentation is essential because the risks are real. The people who will use engineering systems and depend on them for their safety and very life are real.** It is the responsibility of every engineer to ensure cases such as the Therac-25 never happen again.

References

[1] Sara Baase and Timothy Henry. *A gift of fire: Social, legal, and ethical issues for Computing Technology.* Pearson, 2019.

[2] Nancy G. Leveson. *Appendix A: Medical Devices: The Therac-25.* Addison-Wesley, 1999.

[3] N.G. Leveson and C.S. Turner. An investigation of the therac-25 accidents. *Computer*, 26(7):18–41, Nov 1993.

[4] Barbara Wade Rose. Fatal dose, Jun 1994.

Table 2: FMEA Table

Design Component	Failure Modes	Causes of Failure	Effects of Failure	Detection	Recommended Action
Error Handler	Security Compromised	• Fail-open security check • Error data improperly protected • Malicious cyber attack	Participant data will be made vulnerable to exploitation	Error handler returns strange or incomplete results	Device enters data-lockdown mode, preventing data from being accessed until security issue is resolved.
	Errors are strange or incomplete	• Stack overflow • Memory leak • Error previously unaccounted for	Persons responsible for responding to errors will be unable to diagnose and address the underlying issue	Error comes in an unexpected form, or returns no value	• Use different channels to handle device logic and error handling • Ensure that strange errors either return as Optional or Empty List (i.e. any value but null)