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	Back End Developers	Initial documentation for Hazard Analysis
...	...	...

Contents

List of Figures

List of Tables

1 Introduction

In today's society, technology and engineering solutions are simply expected to work. The multitude of complex engineering designs and systems created to meet the needs of the world are expected to be infallible in the eyes of the public. When failures do occur, society considers them it shocking. When these failures happen in engineering systems critical to an aspect of safety, health, or some other critical role, people can die. In many cases, these failures are predictable and preventable in the design phase. The causes of such failures are known as hazards.

1.1 Purpose of Hazard Analysis

It is therefore necessary for engineers to perform extensive and thorough assessments of the systems they design in the aim to elimate as many failures as reasonably possible. This is process is called hazard analysis.

More formally, hazard analysis is a step in the process to assess risk within an engineering system. Its aim is to identify and assess the potential conditions which may cause failure. These hazards can exist and cause failures alone, or in combination with other hazards or conditions. Once completed, a hazard analysis should provide a comprehensive assessment of the hazards within a system according to the system's components and boundaries, the assumptions critical in performing judgements regarding hazards and the scenarios they may occur in, and the requirements necessary to ensure that these hazards will be mitigated within the realm of reasonable possibility.

2 Scope

The purpose of this document is to perform this hazard analysis on the system to be designed by the Back End Developers. This document will first provide a description of the system boundaries and components of the system on an abstract level (both the hardware and software components), and then will list and justify the assumptions made in order to perform this hazard analysis. These assumptions will be kept to a minimum, in the hope to reduce the number of potentially overlooked hazards. The document will then describe the Failure Mode and Effect Analysis (FMEA) done by the team of the Back End Developers, and then detail the specific safety and security requirements that have been discovered in the process of performing the hazard analysis. The document will end with a roadmap describing the steps which will be taken in order to implement the novel discovered requirements, the timeline in which they will be implemented, and what considerations must be made regarding said requirements.

3 System Boundaries and Components

The system consists of several components that make up the entire system

3.1 Battery/Power Management system

This component facilitates stepping down/up the source voltage from the battery to the necessary values required by different parts of the device. Moreover, it consists of a charge protection circuit for battery protection (Over-voltage and Discharge). Finally present is a battery level indicator that will generate an alert should the battery life fall below a certain threshold.

3.2 Sensor Array system

This component represents all the various sensors that will be used to collect the state information about the user. Also consists of various filters to facilitate smooth and accurate data collection.

3.3 Prompt generation system

This component handles all prompt generation, from the detection of when a prompt occurs, to its specific creation and finally its display on the screen.

3.4 Display System

This system manages all functionality of the device's display, such as prompt display, showing basic user feedback such as date, time, temperature, etc.

3.5 Data Storage system

This system handles all logging and storage of data collected by both the sensor array and the prompt generator. Data is stored along with an indication of which system it came from and all prompts will be stored with the data and time of entry.

3.6 Device Manager

This system handles all connection and communication between the device and the host software.

3.7 Error Handler - Hardware

This component constantly checks the states of every system present and ensures that if any of them fail or return an error, an alert is generated. Moreover the system will also try to fix the problem wherever possible.

3.8 Error Handler - Software

This component monitors the state of the host software and will attempt to solve any errors that arise and will alert the user should the attempts fail.

3.9 Host Software

This system is the primary interface for the researchers to analyze the data collected by the device. It consists of several features that allows the researcher to set different thresholds for activity tracking, calibrate all the sensors, update and create new records for participants, and finally interact with the data stored on the device.

3.10 Configuration

This system allows researchers to calibrate the sensors on the device , set and modify the different thresholds for activity tracking and create new prompts.

3.11 Records

This system stores all information about users present in the study. Moreover it provides functionality for researchers to create new records for new users.

3.12 Data View

This is where all data stored on the device can be viewed/ analyzed by the researchers in a graphical manner. The system also contains some functionality for data sorting/parsing and statistical analysis.

3.13 System Boundary Diagram

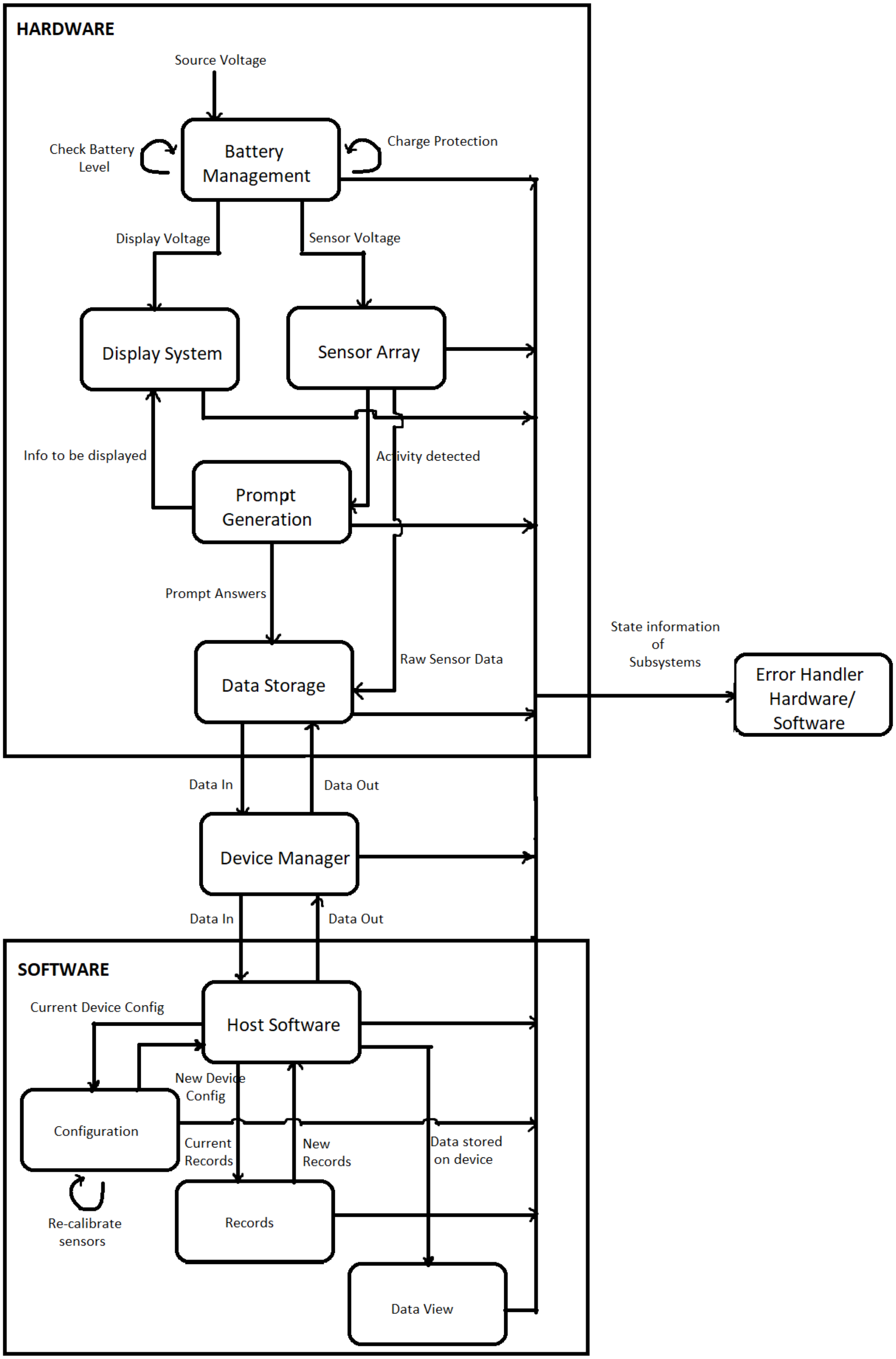

Figure 1: System Boundary Diagram

4 Critical Assumptions

- No wires will come loose during use.
- Batteries are pugged in correctly (the positive and negative ends are aligned as intended).
- All data are stored in the correct memory location.
- All subsystems work as intended.
- All off-the-shelf components work was intended.

5 Failure Mode and Effect Analysis

Design Component	**Failure Modes**	**Causes of Failure**	**Effects of Failure**	**Detection**	**Recommended Action**
Display System	Display not working	• Improper circuit connection • Battery Level too low • Code malfunction • Physical damage to display	• Users cannot answer prompts. • Users cannot view any information on the display	Display system returns an error code. Display Led is OFF	• Users cannot answer prompts. • Users cannot view any information on the display • Check wiring and run a diagnostic on the display system • Replace faulty hardware
	Incorrect information displayed	• Display Driver faulty • Improper interaction between Display system and Prompt generation	Users face unexpected outputs causing imporper use of device	Display system returns an error code.	• Let Error Handler try to solve issue. • Perform a manual overview of code. • Perform a system reboot.

Table 2: Display FMEA **Req:** HR1 ,SR1, SR2

Design Component	**Failure Modes**	**Causes of Failure**	**Effects of Failure**	**Detection**	**Recommended Action**
Prompt Generation System	Prompt not generated	• Prompt generation code faulty • System stuck in an idle state where no activity is detected.	• Display system will not produce an output. • Users will be unable to provide feedback regarding activity	Prompt Generation system returns an error code.	Let Error Handle try to solve the issue
	Incorrect Prompt Generated	• Prompt generation code faulty • Improper interaction between Prompt generation and Sensor Array	Prompt generated produces unexpected outputs causing imporper use of device	Prompt Generation system returns an error code. Test prompt produces unexpeted outputs	• Let Error Handler try to solve issue. • Check System Array State • Perform a system reboot.

Table 3: Prompt Generation FMEA **Req:** SR1, SR2, SR3

Design Component	**Failure Modes**	**Causes of Failure**	**Effects of Failure**	**Detection**	**Recommended Action**
Sensor Array	• Heart rate not detected • Surrounding pressure not being tracked • No motion detected	• Device not worn properly • Sensor breaks down, due to passing thresholds limits • Device surface is dusty or contaminated	EMA may not be triggered	Software check to see if any sensor data is being collected	• Ensure the device has been wrapped around properly for more accurate detection • Reboot the system • Let error handler try to solve the issue

Table 4: Sensor Array FMEA **Req:** HR2, HR5, SR1, SR2

Design Component	**Failure Modes**	**Causes of Failure**	**Effects of Failure**	**Detection**	**Recommended Action**
Battery management system	Device not starting	• Device is faulty • Battery has died • Device may have overheated • Water may have damaged the battery	Device is not turned on, resulting in no monitoring	Battery dead indicator appears when turning on the device	• Charge the device • Change batteries if needed

Table 5: Battery Management FMEA **Req:** HR1, HR3,HR4, SR1

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)

Table 6: Error Handler FMEA **Req:** SR1,SR2, DSRQ1

Design Component	Failure Modes	Causes of Failure	Effects of Failure	Detection	Recommended Action
Data Storage	Data stored at wrong memory location	• Incorrect software commands • Memory space doesn't exist (invalid memory selected) • Insufficient memory space • Physical damage to hardware memory chip	Lost and unsaved data	Set up error handler to check if each data point is successfully stored at the correct memory location each time	Replace faulty hardware or set up correct memory path
	Data Stored with incorrect type	• Wrong data type used for storing data	Analysis program can't interpret data	Failed data analysis	Convert data to correct type

Table 7: Data Storage FMEA **Req:** SR1, SR5, DSR2

Design Component	Failure Modes	Causes of Failure	Effects of Failure	Detection	Recommended Action
Device manager	Unable to establish connection	• Loose wires • Incorrect communication protocol • Incorrect parameters for serial packets (size, format, etc.)	• Data can't be transferred between hardware and software • Lost data	• Check list of connected devices on device manager • Visual inspection of wiring and circuitry • Attach an error detection LED on the device	• Make sure all necessary connections are made • Reboot device • Restart host software

Table 8: Device Manager FMEA **Req:** SR1,SR2

Design Component	**Failure Modes**	**Causes of Failure**	**Effects of Failure**	**Detection**	**Recommended Action**
Records	Records not created and deletion of records	• Code that writes to database is faulty. • Connections between software systems are incorrect.	• Present users lose data, new users cannot have data stored. • Time loss.	• Records system returns an error code using error checking. • Database trigger for stagnant data.	Create redundant data sets and storage units, or cloud storage of data.
	Corrupted records	• Code is faulty. • Database structure is faulty. • Wrong datatype used for storing data.	Data stored for various users is unusable.	• Detect if data records are returning unusable values, e.g. NaN, Inf, 0, garbage values. • View values stored in database to check for unexpected outputs.	• Prevent users from accessing database. • Ensure database is robust prior to deployment through testing.

Table 9: Record System FMEA **Req:** SR1,SR2, SR5, DSR1,DSRQ1,DSRQ2

Design Component	Failure Modes	Causes of Failure	Effects of Failure	Detection	Recommended Action
Data View	Data not visible	• Data collection failure. • Poorly aggregated data. • Incorrect software implementation. • Statistical analysis of data is faulty.	Information provided is not useful to researchers or participants.	• Error handling to detect if the software is not performing as it should be. • Check error codes of device manager.	• Visual inspection to make sure device is working. • Ensure database is robust prior to deployment through testing
	Graphical data representation incorrect.	• Improper communication with device manager. • Device malfunction, no correct data.	Incorrect observations could be made based on the data.	• Error handling to detect if the software is not performing as it should be. • Check error codes of device manager	• Visual inspection to make sure device is working. • Use error handler to solve issue.

Table 10: Data View System FMEA **Req:** SR1,SR2

Design Component	Failure Modes	Causes of Failure	Effects of Failure	Detection	Recommended Action
Host Software	Calculation and floating point error	• Improper units used (metric vs Imperial) • Rounding error • Floating point conversion error	• Wrong information can be reported. • Rounding errors can propagate through different formulae resulting in data inaccuracies. • Data calculated will be completely wrong due to binary overflow.	• Unit and integration testing for checking proper metrics and calculation. • Boundary condition testing for detecting binary overflow/floating point conversion errors.	• Avoid/fix type conversion and export raw data collected through hardware in case calculation of other metrics for tracker fails. • Use error handler to solve issue.
	Memory allocation failed	• Memory fragmentation or overflow due to Dynamic memory allocation.	• This can cause data to be overwritten and also failure of storing data.	• Host software stops running and system fails.	• Reboot Software and make sure to free memory if not using it during memory allocation.
	Security threats and data loss	• Malware injected. • SQL injection.	• Software could have catastrophic failure and device could stop running. • SQLi cyberattack can let attackers view or modify the database causing data leaks.	Unrecognizable patterns and processes running in the host software.	• Run a closed-loop network to avoid backend database manipulation. • Limit access of tracker to Internet/IoT devices to avoid hacking and data leakage. • Perform a manual overview of system and processing running.

Table 11: Host Software FMEA **Req:** SR1,SR2

Design Component	Failure Modes	Causes of Failure	Effects of Failure	Detection	Recommended Action
Configuration system	Calibration fails	• Sensors not wired correctly with system. • System stuck in idle state and does not record reference for calibration. • Not enough power for sensors. • Code to calibrate sensors and system fails.	• Wrong Calibration of system will result in incorrect activity detection. • Data will be false/will not be measured correctly.	Display message to check if all calibration is completed correctly.	Reboot system and check wiring to sensors.

Table 12: Calibration System FMEA

6 Safety and Security Requirements

6.1 Hardware Requirements

1. **HR1**: All Hardware systems will have an associated indicator LED.

2. **HR2**: Device components will have sufficient electrical tolerances for Current, Voltage, Temperature etc.

3. **HR3**: Fuses will be connected in the circuitry to shut down the device in cases of high current draw.

4. **HR4**: Charge protection circuity will be integrated to avoid overcharging/discharging of the battery.

5. **HR5**: The intensity of emitted frequencies will be limited to safe levels.

6.2 Software Requirements

1. **SR1**: The Error Handler will initially try to solve all Non-Hardware issues.

2. **SR2**: Tests will be conducted for all Software before deployment to the device.

3. **SR3**: All users are required to complete the test prompt.

4. **SR4**: Configuration setting will fall within accepted boundaries.

5. **SR5**: Manipulation of database shall be done with standard DBMS practices.

6.3 Data Storage Requirements

1. **DSR1**: Error reports and incorrect data will be logged into a separate database.

2. **DSR2**: All data to be backed up, in case of power failure.

6.4 Data Security Requirements

1. **DSRQ1**: Access to records will be restricted to administrative users with encrypted ssh access key.

2. **DSRQ2**: The device will not modify user data unnecessarily.

7 Roadmap

Table 13: Road Map

SNo.	Requirement	Expected date of implementation
1	Name(s)	Description of changes
2	Name(s)	Description of changes
3	...	...
4	...	...
5	...	...

Appendix

Therac-25 case study

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.

Error Codes

To facilitate error handling, every system will return an error code that represents the status of that system. These error codes follow the following format, BED_ERR_ERRORTYPE For example BED_ERR_NONE represent that the system is working correctly. Some more examples of error codes are:

- BED_ERR_PARAM_ERR : Represents an error with the paramaters of the function.
- BED_ERR_GENERAL : Represents a generic error that hasnt been catalogued yet.
- BED_ERR_INVALID_DATA_SIZE : Represents an error related to a size mismatch in what is stored within a block of memory or variable.

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.