DR. SARAH ABRAHAM

CS349

SOFTWARE LIABILITY

WHAT CAN GO WRONG IN SOFTWARE?

- Bugs
  - Improper specification
  - Improper implementation
  - Race conditions
  - Holes in security
  - Etc etc…

THERAC-25

- Radiation therapy machine
  - Designed to target cancerous cells with X-rays or electron beam
- Killed 6 patients in 1986 and 1987

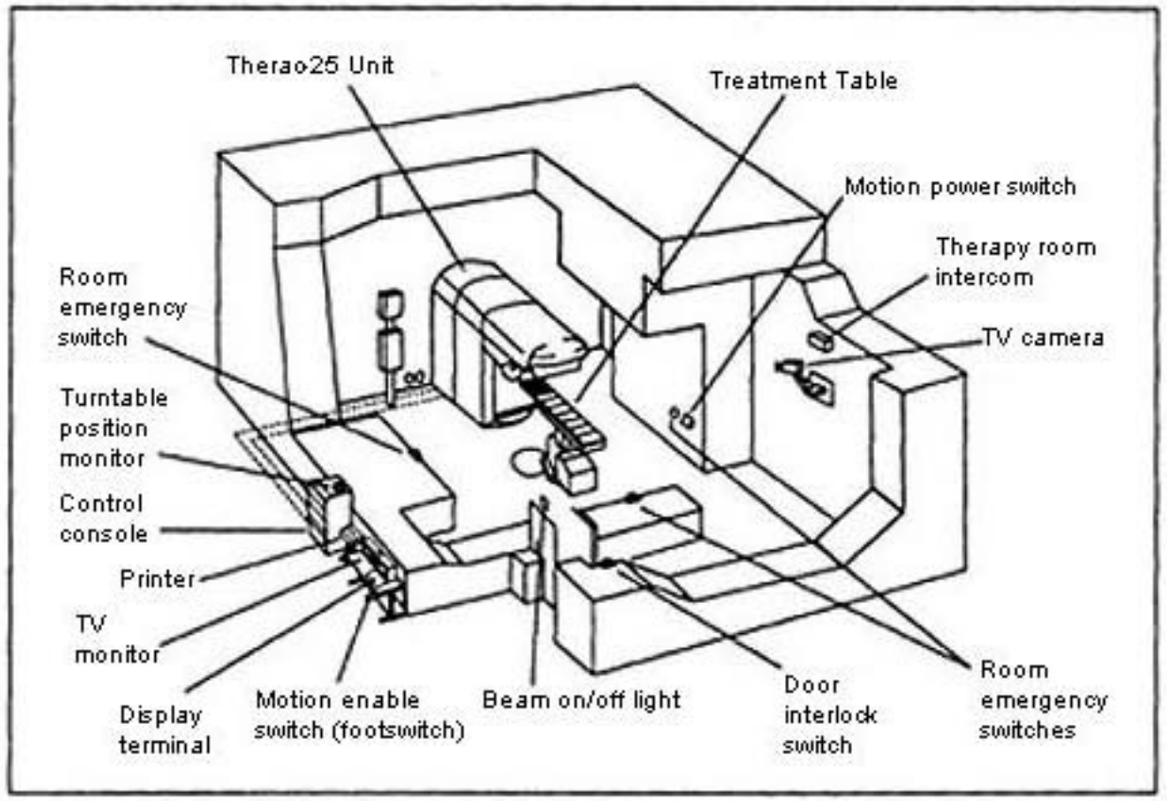


Figure 1. Typical Therac-25 facility

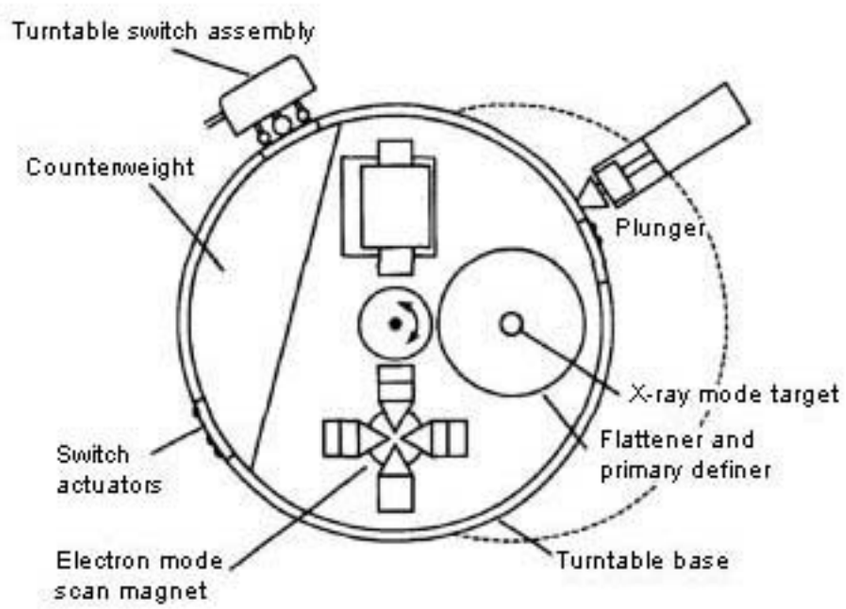


Figure B. Upper turntable assembly

WHAT WENT WRONG?

- Previous versions of the machine had hardware interlocks to prevent operating in an invalid mode
- Hardware locks and much of the manual controls removed in favor of software control
- Software controls had a race condition
  - If X-ray mode was selected, machine would begin set up (a process taking 8 seconds)
  - If switched to Electron mode during this time, the turntable (which directs the radiation) would enter an unknown state

THERAC-25 SOFTWARE

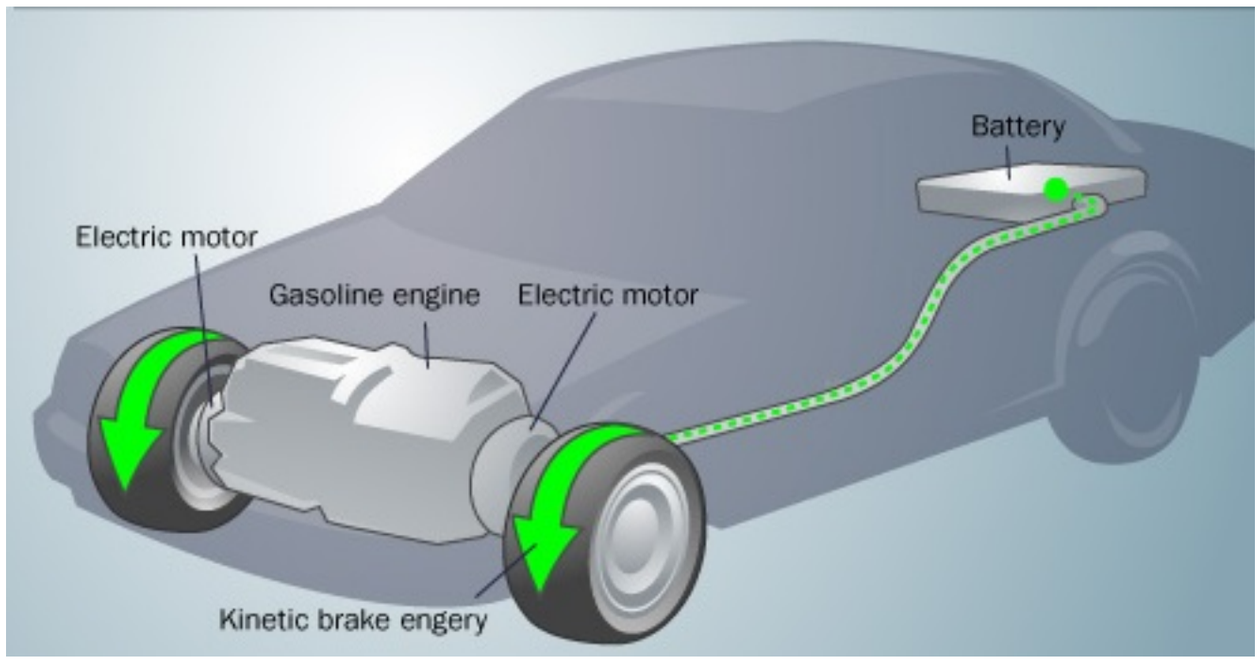
- Entire design and code base was flawed
- No timing analysis of a real-time system
- No unit tests
- No fault tree analysis
- Written by one software engineer and no independent review
- Never tested as software-hardware combination before reaching hospital

BUT EVEN AFTER THE SOFTWARE WAS PATCHED . . .

- Another patient died due to a counter overflow that resulted in the mirror not moving correctly

PRIUS BRAKING SYSTEM

- 2010 Prius models and Lexus hybrids recalled
- Issue with software of regenerative braking system
- Brakes temporarily stopped working when going over a bump
- Fixed with software patch



PRIUS ACCELERATION SYSTEM

- In 2009-2011 three Prius recalls because of claims that vehicle would uncontrollably accelerate
- Likely it was mechanical rather than electronic
  - Sticky accelerator pedals and floor mats that caught accelerator pedals
- In 2013 Toyota still found liable because Toyota did not follow best practices for a real-time operating system
  - No protection from cosmic ray bit flips
  - Run-time stack not large enough so possibility of data overwrite

INTEL PENTIUM FLOATING POINT DIVIDE

- Pentium floating point error discovered in 1994
- Missing entries in table for SRT division in FPU control logic leads to incorrect look up
- Only two paths to reach bug
- Intel received backlash from mainstream news sources despite claiming effects were small and unlikely

HOW BAD WAS THIS BUG?

- Hard to say since it was caught quite quickly
- Discovered by a mathematician working with distribution of primes
- In 1996 Ariane 5 rocket exploded 37 seconds after liftoff due to conversion of 64-bit floating point to a 16-bit integer
  - Overflow wasn’t handled so computer cleared memory
  - Memory dump interpreted as an instruction to rocket nozzles

KNIGHT CAPITAL GROUP

- In 2012 trading company lost $440M in 30 minutes
- Revenue from previous quarter was $288M
- Trading glitch sold overvalued shares back to market at a lower price
- Affected 148 companies in the New York Stock Exchange

WHAT WENT WRONG?

- Technician did not copy new trading code (RLP) to one of eight servers that routed equity orders
- RLP code repurposed flag formerly used to activate an old function “Power Peg”
- Power Peg function for testing trading algorithms by moving stock prices higher and lower
- Orders sent to eighth server with repurposed flag triggered Power Peg

WHO IS RESPONSIBLE FOR:

- Therac-25?
- Prius acceleration/braking systems?
- Pentium FDIV?
- Knight trading glitch?
- What should happen to the responsible parties?

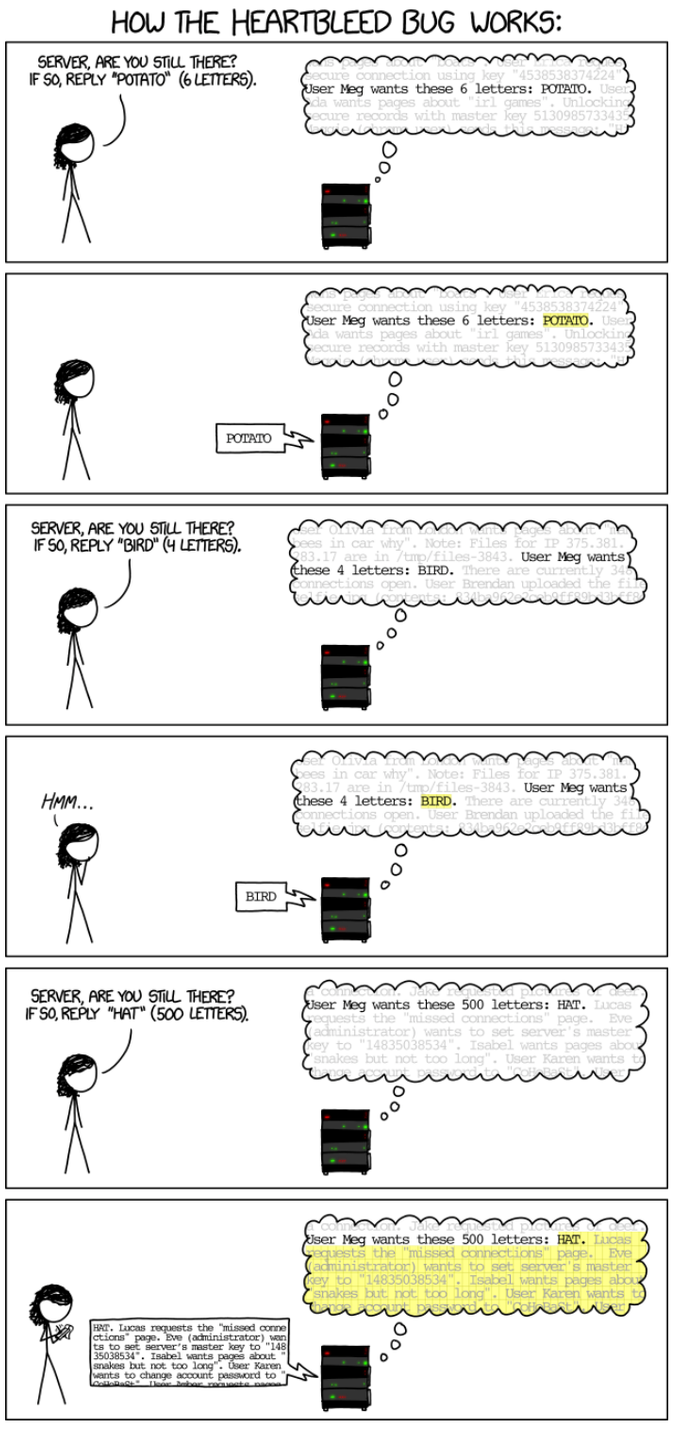
CASE STUDY

- You have just been hired onto a drone delivery startup company. You thought you were being hired for backend web development, but it becomes clear the company is short-handed, so you are expected to help out the embedded systems team. Although you have very little experience in this area, you are eager to learn more. Between the pressing deadlines and constantly shifting milestones, you have very little mentorship or oversight. What do you do?

ENGINEERING VERSUS SOFTWARE DEVELOPMENT

- Engineers are often regulated at the state or national level
  - Must have a license to practice
  - License can be revoked or suspended if engineer violates safety or ethical practices
- Engineers can also be sued as an individual (usually in conjunction with a law suit against the firm or company) for work done for the firm/company
- Software developers do no need to be licensed and are usually not held individually responsible for negligence
  - May result in company blacklisting though

WHAT GOES WRONG IN SOFTWARE DEVELOPMENT?



- Consider the 2014 Heartbleed vulnerability on OpenSSL
- Computers verify a valid connection by requesting data (and size of data) be sent back from connected machine
  - memcpy(bp, pl, payload);
- Programming mistake did not verify that pl is of size payload
  - Not an algorithmic issue
  - Code is open source so anyone could have reviewed it

COMPLEXITY

- Large scale systems are hard to reason about algorithmically
  - States in a software program far exceed states in mechanical (or even electrical) systems
  - Algorithms may be correct based on untrue (or unenforced) assumptions
- Large scale systems are hard to reason about programmatically
  - Many different programmers working on different parts of the system
  - The correctness of the small pieces does not ensure the correctness of the larger pieces
  - Legacy code remains long after original programmers leave

WHAT GOES WRONG IN SOFTWARE DEVELOPMENT?

- Consider the Equifax breach in 2017
  - Database of credit reporting agency breached by malicious hackers resulting in the personal data of over 140M people being compromised
- Point of failure was a vulnerability in the Apache Struts implementation
  - Vulnerability disclosed by Apache in March, breach happened in May
  - Breach continued through July
- Equifax set up a free credit monitoring system that was also vulnerable to attacks

HUMAN FACTORS

- Poor design and coding practices
  - Much software developed in unreasonable time frames that neglects proper consideration of design or time for testing
  - Demand for high volumes of software and code leads to under-qualified programmers working on sensitive systems
    - Emphasis is on budget and internal operations rather than user experience
- Complex operations and interests
  - 3rd party developers and tools can introduce issues with design specification and/or implementation
  - Chain of management may distance high-level view of company from workers focusing on daily tasks

INSTAPOLL QUESTION

- Have you worked on a large scale software system at a company or lab? If so, have you experienced any issues due to software complexity/human factors? If not, how have your class experiences given you insight into challenges of software development?

HOW DO WE COMBAT BUGS?

FORMAL VERIFICATION

- Formal verification can prove correctness of algorithms in certain circumstances
  - Model checking explores all states and transitions in the model
- Theorem provers and satisfiability solvers can determine system guarantees based on its specifications
- Dependently-typed programming (such as Agda) uses highly specific and expressive types to ensure code that can compile must run according to its specification

SOFTWARE SIMULATION

- Formal verification doesn't necessarily scale to large systems and can be slow to test
  - Representing a large-scale system as a verifiable model may be difficult or impossible
- System simulation allows for testing on models that may be harder to verify mathematically
  - Helps creators reason about system's model before and during the creation of that system

UNDERSTANDING RACE CONDITIONS

- Occur when order-dependent events happen in an unexpected order
- Nondeterminism makes them difficult to reproduce
- Changes to program (such as debug mode) can introduce or eliminate unexpected program output
  - Does not mean race condition is fixed!

TESTING AND EXPERIMENTATION

- Unit tests help reason about functionality and expected behavior
- Dynamic analysis can check unusual paths and states
- Quality assurance (QA) allows additional eyes to assess weaknesses in software
- But there will still be bugs!

ETHICAL DATA MANAGEMENT

- Privacy and security cannot be sidelined for the sake of rapid development and prototyping
- Welfare of consumers and the general public must be taken into consideration
- Even software for “non-critical” systems can impact public
  - e.g. Facebook, Uber, Sony, etc

SOFTWARE ENGINEERING CODE OF ETHICS

1. PUBLIC - Software engineers shall act consistently with the public interest.
2. CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
3. PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.
5. MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
7. COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.
8. SELF - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

IS THIS PROBLEM GETTING WORSE OR BETTER?

REFERENCES

- <https://hackaday.com/2015/10/26/killed-by-a-machine-the-therac-25/>
- <https://www.computingcases.org/case_materials/therac/therac_case_intro.html>
- <https://www.npr.org/templates/story/story.php?storyId=123537419>
- <https://www.eetimes.com/document.asp?doc_id=1319897>
- <https://web.archive.org/web/20081210133154/https://www.maa.org/mathland/mathland_5_12.html>
- <https://dealbook.nytimes.com/2012/08/02/knight-capital-says-trading-mishap-cost-it-440-million/>
- <https://www.nspe.org/resources/professional-liability/liability-employed-engineers>

REFERENCES

- <https://www.digitaltrends.com/computing/the-heartbleed-bug-explained-by-a-web-comic-xkcd/>
- <https://gizmodo.com/how-heartbleed-works-the-code-behind-the-internets-se-1561341209>
- <https://www.wired.com/story/equifax-breach-no-excuse/>
- <https://www.forbes.com/sites/forbestechcouncil/2018/09/20/move-fast-and-dont-break-things/>
- <https://www.nytimes.com/2018/09/26/technology/uber-data-breach.html>
- <https://www.computer.org/web/education/code-of-ethics>