

# How to Use Mathematics to Write Better Programs

Rafael Castro

27/10/2023

# Bugs are annoying



# Bugs are annoying



# Therac-25: The killing radiation therapy machine



# People got hurt

- At least 6 accidents between 1985 and 1987, 3 died later



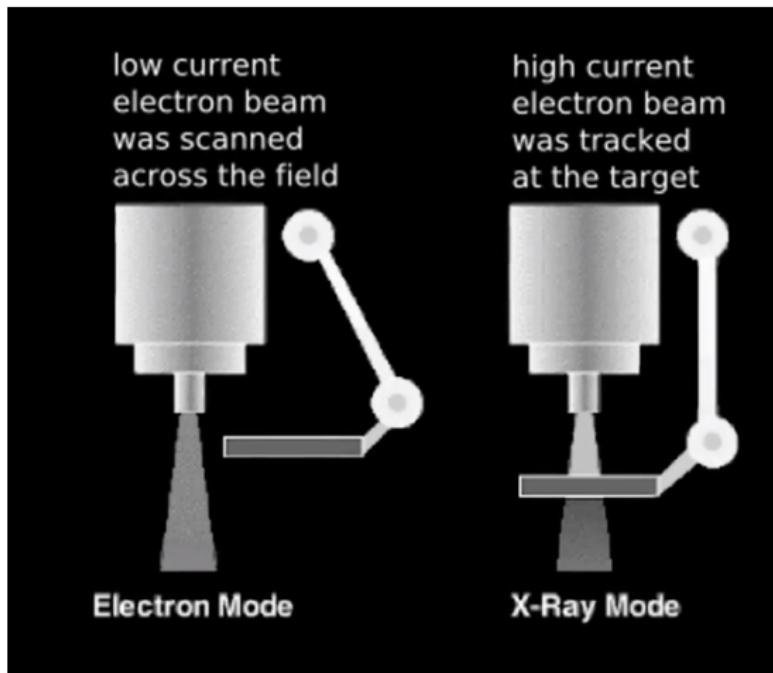
# Malfunction messages

- Malfunction messages from 1 to 64, with no extra information

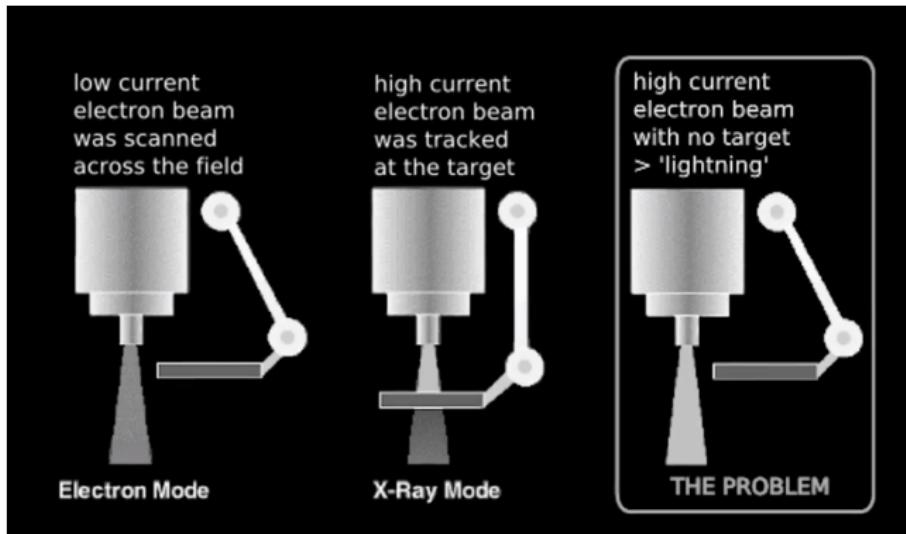


# The machine a bug

- The machine had three modes: field light, direct electron beam, x-ray



# The machine a bug



# How to cause the bug

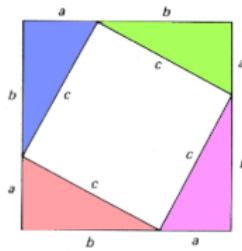
|                           |                    |                |          |
|---------------------------|--------------------|----------------|----------|
| PATIENT NAME: John        | BEAM TYPE: E       | ENERGY (KeV):  | 10       |
| TREATMENT MODE: FIX       |                    |                |          |
|                           | ACTUAL             | PRESCRIBED     |          |
| UNIT RATE/MINUTE          | 0.000000           | 0.000000       |          |
| MONITOR UNITS             | 200.000000         | 200.000000     |          |
| TIME (MIN)                | 0.270000           | 0.270000       |          |
|                           |                    |                |          |
| GANTRY ROTATION (DEG)     | 0.000000           | 0.000000       | VERIFIED |
| COLLIMATOR ROTATION (DEG) | 359.200000         | 359.200000     | VERIFIED |
| COLLIMATOR X (CM)         | 14.200000          | 14.200000      | VERIFIED |
| COLLIMATOR Y (CM)         | 27.200000          | 27.200000      | VERIFIED |
| WEDGE NUMBER              | 1.000000           | 1.000000       | VERIFIED |
| ACCESSORY NUMBER          | 0.000000           | 0.000000       | VERIFIED |
|                           |                    |                |          |
| DATE: 2012-04-16          | SYSTEM: BEAM READY | OP.MODE: TREAT | AUTO     |
| TIME: 11:48:58            | TREAT: TREAT PAUSE | X-RAY          | 173777   |
| OPR ID: 033-tfs3p         | REASON: OPERATOR   | COMMAND:       | █        |

# How to write software that don't cause catastrophic failures?

- Formal Methods!
- There are many techniques: static analysis, model checking and formal proving

# What is to prove something?

- Establishing mathematical facts that are universally true
- You know some: Pythagoras theorem:  $\forall abc. c^2 = a^2 + b^2$



Two ways to calculate this area:

- ①  $(a + b)(a + b) = a^2 + ab + ab + b^2$
- ②  $4(ab)/2 + c^2$

they should be the same!

$$\begin{aligned} a^2 + ab + ab + b^2 &= 4(ab)/2 + c^2 \\ a^2 + b^2 &= c^2 \end{aligned}$$

- ③ How can you be sure that there is no mistake in the proof?

# Proof assistants

- Proof assistants: programs that verify if your proof is correct!
  - We trust the verifier!
- I use a proof assistant called Isabelle/HOL to verify programs
- How my research works:
  - ① I write a specification of the program: what it should do, and what it should **not** do
  - ② I write the program
  - ③ I write a proof that the program respects the specification
    - Usually this requires writing many other auxiliary proofs

# Examples!

- Pythagoras
- Sorting

# This is a real job!

- TLA+/TLC: Amazon AWS, Intel, Microsoft...
- Coq (CompCert): Airbus
- Isabelle/HOL (seL4): Defense Advanced Research Projects Agency (DARPA)
- Formal methods companies: Absint, Trustworthy Systems
- Apple: [https://jobs.apple.com/en-us/details/200343072/  
formal-verification-engineer](https://jobs.apple.com/en-us/details/200343072/formal-verification-engineer)

# Thank you!

- Questions?