Reasoning about Programs

Jeremy Bradley and Francesca Toni

`Room 372.  Office hour - Tuesdays at noon.  Email: jb@doc.ic.ac.uk`

Department of Computing, Imperial College London

Produced with prosper and LaTeX

Course Details

- Course title: Reasoning about Programs
- Course code: 141
- Number of courseworks: 5
    - Hand-in dates: 25 Jan, 1st Feb, 8th Feb, 15th Feb, 15th Mar
- Syllabus
    - Induction for Haskell programs
    - Invariants in Java programs
    - Java algorithms

Why Reason about Programs?

’85–’87 **Therac-25 X-Ray machine**: program error results in radiation overdoses

- Cost: lives of several people

1994 **Intel Pentium chip**: FP error affecting 6th d.p.

- Cost: $0.5 billion

1996 **Ariane 5**: arithmetic overflow caused forced destruction of rocket and payload

- Cost: $1 billion

’80–’05 **Windows, Word, etc**: Data loss from crashes. Usually memory overflows

- Cost: Lost productivity – $ many trillions?

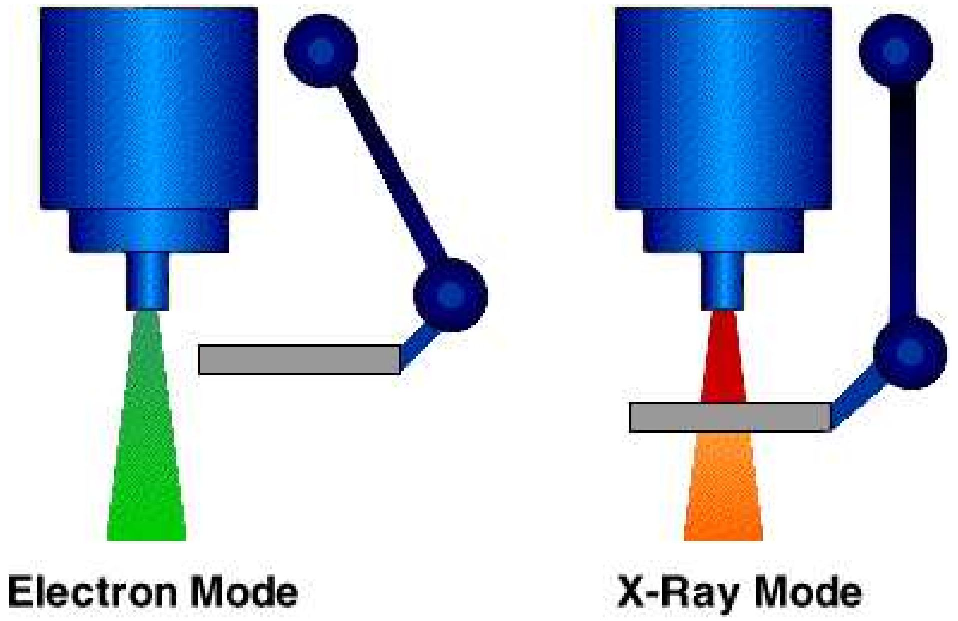
Ariane 5: Some details

- 64 bit number converted to 16 bits
- 64 bit number exceeded 16 bits in size causing memory overflow
- overflow caused main guidance system to crash
- backup guidance system was running the same software so it also crashed
- rocket veers off course
- self-destruct mechanism initiates

Ariane 5: Result

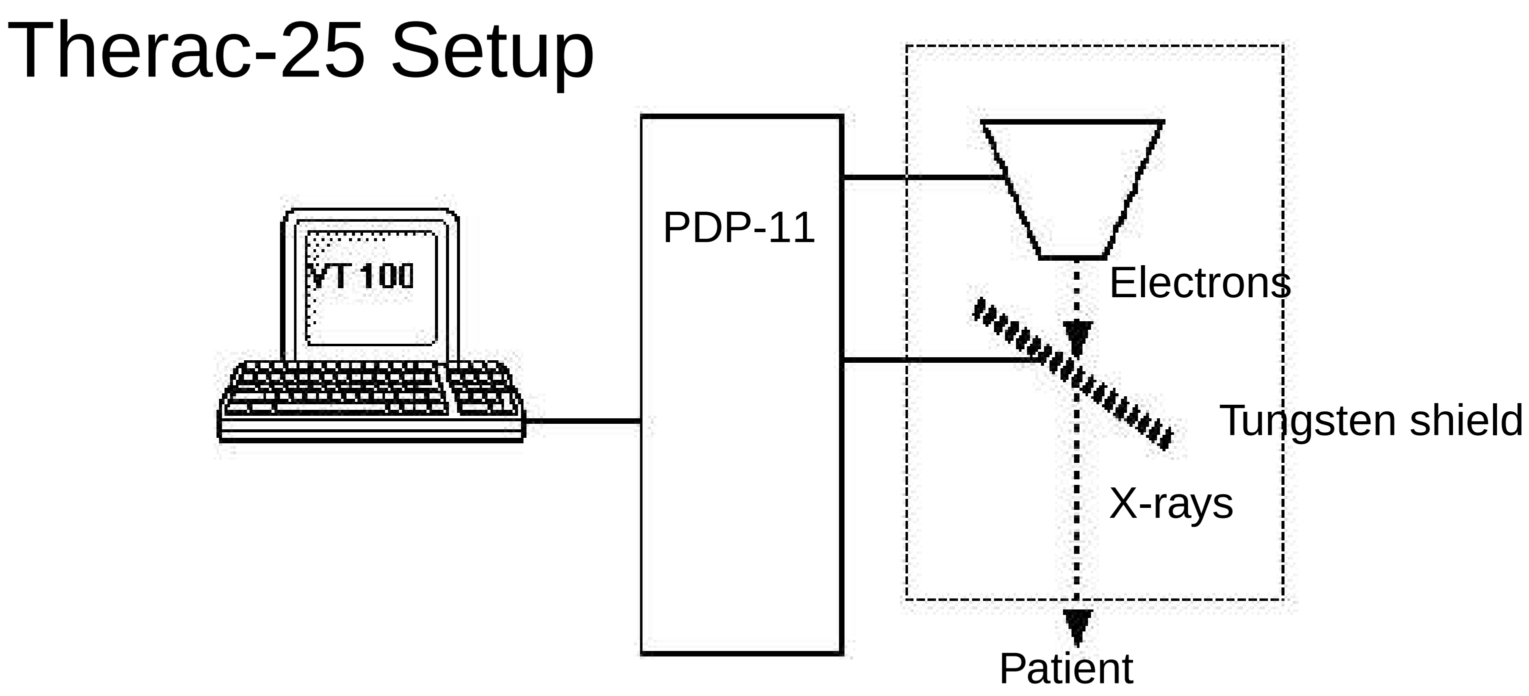
- Irony: software which contained overflow wasn’t needed during flight and could have been disabled before takeoff
- James Gleick, NY Times, Dec. 2006: `http://www.around.com/ariane.html`

Therac-25: Some details



- 6 people received overdoses of between 15,000 and 20,000 rads
- Typical treatment dose should have been 20-50 rads

Therac-25: Some details



- A real-time reactive system (hard!)
- Inherited legacy code from Therac-6/20

Therac-25: Some more details

- Many *fail-danger* errors in design, testing, code and interface
- Testing was exclusively at system level – not modular
- Interface erroneously reported no/low dosage received
- Poor documentation of error reports
- No validation
- Leveson and Turner, IEEE Computer 26(7), July 1993:

  `http://courses.cs.vt.edu/˜cs3604/lib/Therac_25/Therac_1.html`

Solutions?

- Scalable design
  - clarity
  - maintainability
- Verification and testing
  - design against specification
  - implementation against design
  - modular as well as system-level
- Quality and document control
  - check and document all of the above

Which bit do we look at?

- Low level design
- Use mathematical techniques for:
  - verification of functions
  - verification of methods
  - verification of loops
- Larger program-level verification comes later (2nd year)