23: Safety-critical systems

Therac-25

- Computerized radiation delivery system in the 1980s
- Well-documented example of how SW process failure can lead to serious consequences
- Great retrospective summary: http://sunnyday.mit.edu/papers/therac.pdf

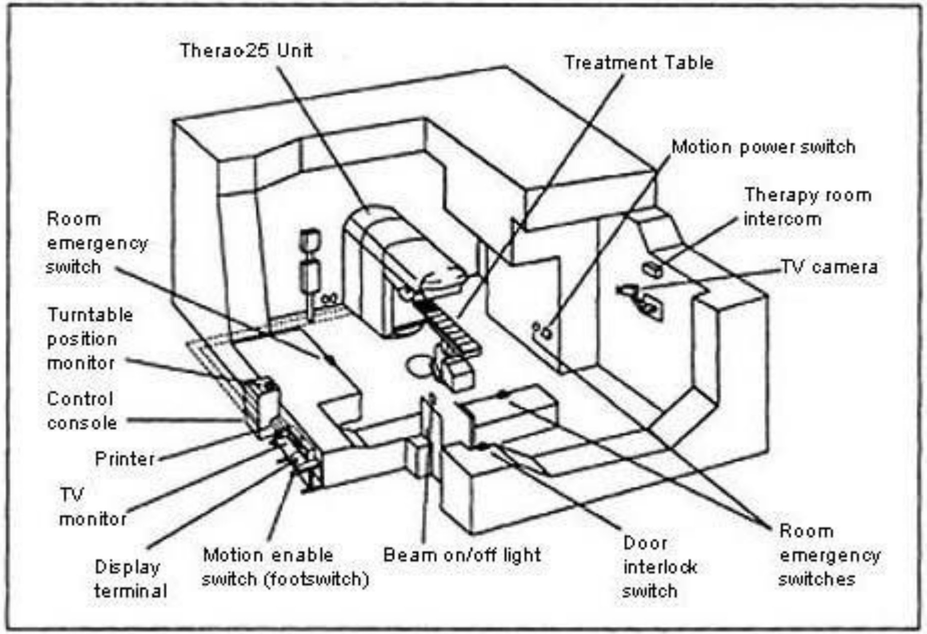


Figure 1. Typical Therac-25 facility

Image source

Therac-25 summary of mistakes

- Homebrew OS, non-atomic mutexes
- Reuse software from Therac-6
- No reviews in software process
- Cryptic error messages that were ignored by operators (false idea that frequent shutdown means system is safe)
- System testing only
- No hardware cross-check
- Overflow for flag variable
- Assumed one bug fix meant system was now safe

Safety-critical systems

Systems where failure of operation can cause serious harm or death

- Direct contact with humans (cars, robots, medical devices)
- Affect human well-being (power plants, HVAC systems)

Disclaimer: this lecture is a ***starting point*** *for reasoning about safety-critical software. For true safety-critical development,* ***apply a well-known standard*** *as part of a safety-focused development culture*

Safety plans and safety requirements

Safety is part of the lifecycle

- If you are only evaluating safety at the testing stage, you are not engineering for safety

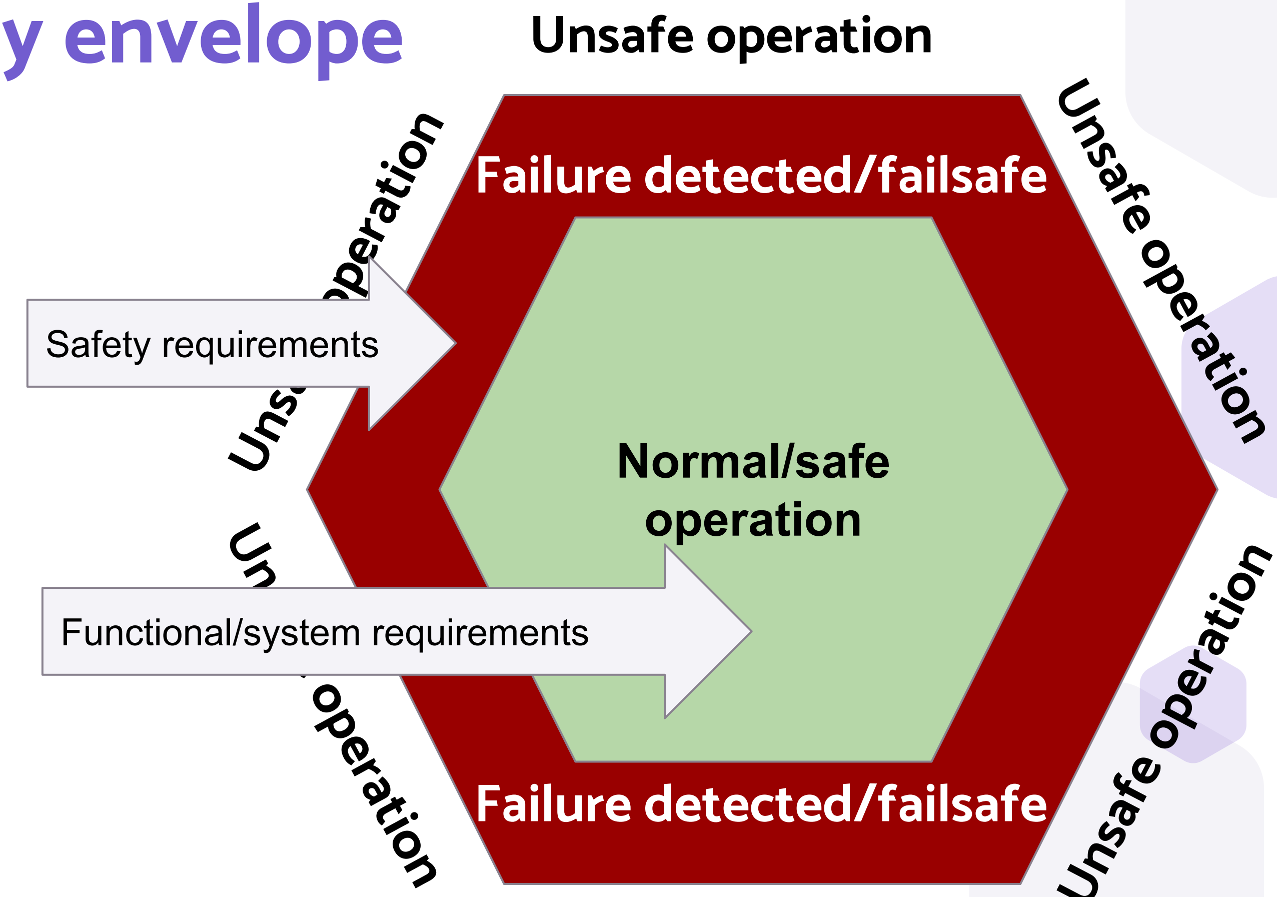
System is assumed unsafe unless engineered for safety

Safety is built-in, not added

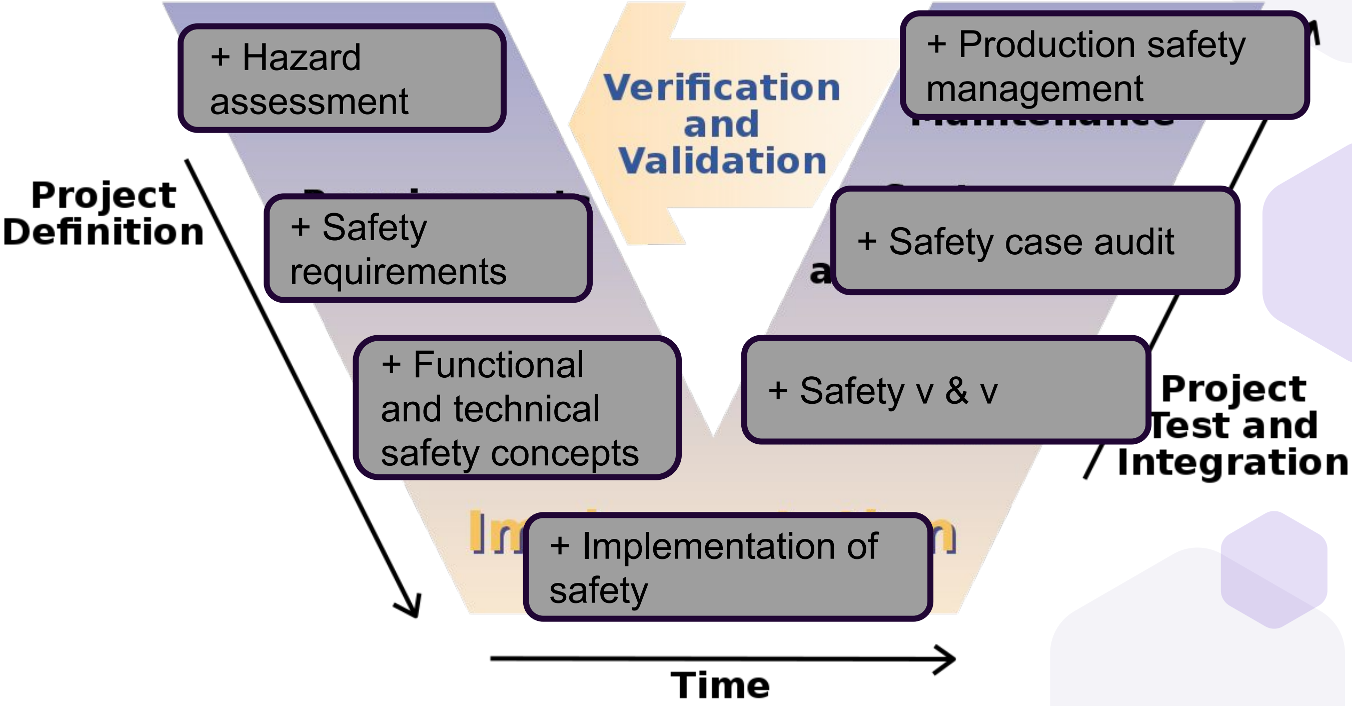
Safety requirements are not an afterthought

- “Working system” is not the same thing as a “safe” system

Safety envelope



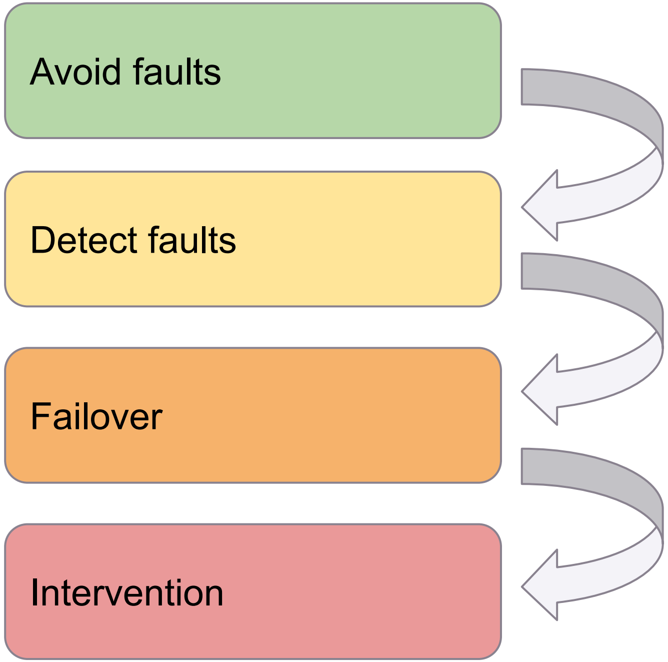
Safety V model (applies to security as well)


+ Hazard assessment
Project Definition
+ Safety requirements
+ Functional and technical safety concepts
Verification and Validation
+ Production safety management
+ Safety case audit
+ Safety v & v
Project Test and Integration
+ Implementation of safety
Time

What different ways can you think of that an e-scooter (hardware/software) might fail?

Image source

Escalation of safety



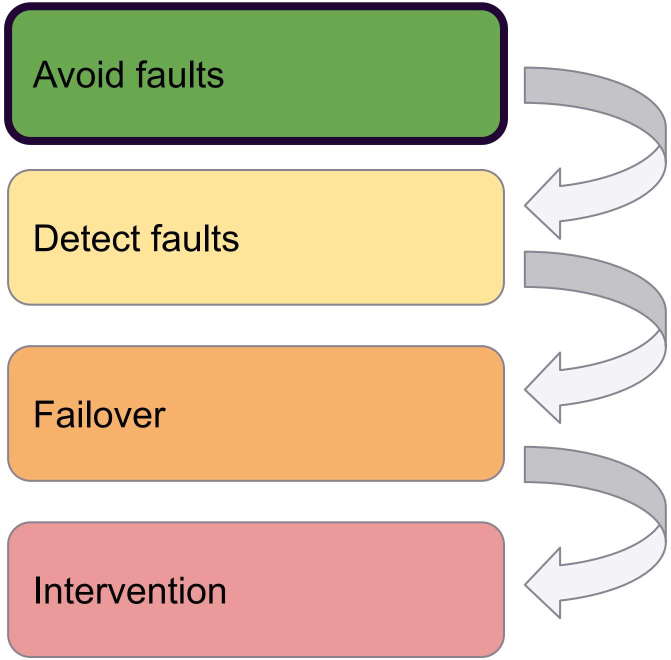
Avoid faults

Detect faults

Failover

Intervention

Escalation of safety



Avoid faults

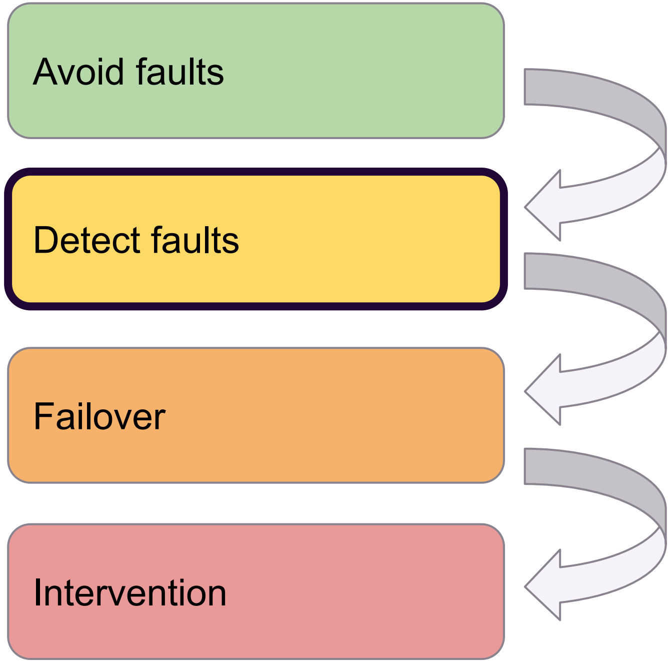
Detect faults

Failover

Intervention

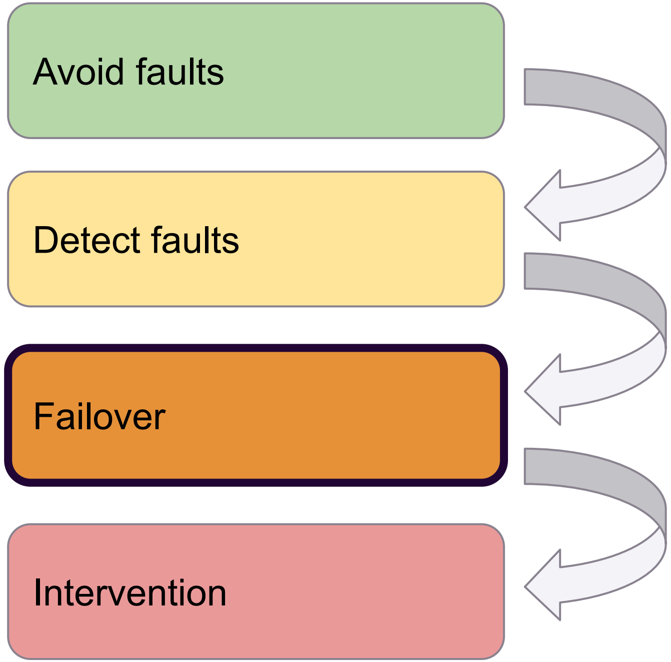
Pick a scooter software failure. How would you avoid it?

Escalation of safety



What are ways you can think of detecting one of the scooter faults?

Escalation of safety



Avoid faults

Detect faults

Failover

Intervention

Single points of failure

A single point of failure happens when a failure of one component renders the entire system unsafe

Avoid single points of failure by using redundancy (later this week)

- Hidden sources of correlation: shared libraries, shared power, shared connections, shared defective requirements....