Fill out milestone demo availability form!

24: Safety best-practices
Escalation of safety

- Avoid faults
- Detect faults
- Failover
- Intervention
Code style

Spaghetti code

Special topics: global variables, floating point

Style guides
Which would you rather test/maintain?

Image source
Spaghetti Code

Code whose structure is impossible to untangle

MCC (McCabe’s cyclomatic complexity)

Measure of branching logic in code

Easy way to compute: #1 of closed loops + 1

Some standards impose limits on MCC
Why would global variables be considered harmful?
Why would floating point be considered harmful (beyond floating point error)?
Floating point

Floating point error/imprecision

Portability

Not equally precise for representing all numbers

All comparisons with NaN return false (includes NaN == NaN)

NaNs propagate
Code style: MISRA C

- [link] embedded.com article on MISRA C
- [link] JPL C coding standard
- [link] TI C coding standard
What, besides coding, should be part of a safety-oriented project culture?
Reasoning about hazards/possible failures

**Hazop**
Hazard and operability analysis
Break system into nodes
Examine wording of system requirements to reason about potential failures

*Brake within 2s -> what happens if we brake after 2s?*

**FMEA**
Failure mode and effects analysis
Worksheets to reason about potential failures from bottom-up
Causes, effects, probabilities, etc

**Fault tree analysis**
Use boolean logic to determine what low-level failures could cause an anticipated failure
Escalation of safety-critical fault management

- 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:

- **Software**: doer/checker with failover
- **Hardware**: failure detection with redundancy

Components must truly be separate for true redundancy.

Hidden sources of correlation: shared libraries, shared power, shared connections, shared defective requirements....
**Doer/checker models**

- **Doer**
  - Behaving within safety envelope/functional requirement

- **Checker**
  - Emergency operation
  - Failover: switch to different component
  - Intervention: alert/switch to human operator
  - Shutoff: turn off system

---

Safety properties

**You will see runtime monitoring in lab!**
Redundancy

Entire system fails

Series System

System can still operate in reduced capacity

Parallel System
$p_A = 0.01$
$p_B = 0.2$
$p_C = 0.1$
$p_D = 0.03$
$p_E = 0.5$
$p_F = 0.001$