



Safety Requirements

Prof. Philip Koopman

“I cannot conceive of any vital disaster happening to this vessel. Modern shipbuilding has gone beyond that.”

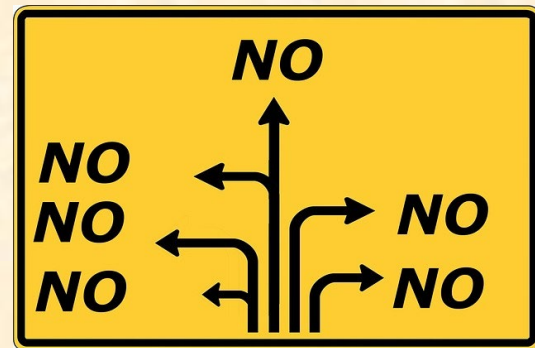
– *EJ Smith (Captain of the RMS Titanic)*

These tutorials are a simplified introduction, and are not sufficient on their own to achieve system safety. You are responsible for the safety of your system.

Safety Requirements

■ Anti-Patterns for Safety Requirements:

- No specifically identified safety requirements
- All functional requirements are safety critical
- Safety requirements can't be validated



■ Specifying safety:

- Safety goals: “working” is not the same as “safe”
 - How hazards are avoided at system level
 - Can involve correctness, backup systems, failsafes, ...
 - Often what the system does not do is as important as what it does
- Safety requirements:
 - More detailed safety-specific requirements allocated to subsystems

■ Overly-simplistic approach:

- Start with system requirements
- Annotate critical system requirements
- Then, annotate supporting requirements
- Problem:
Most requirements can become critical

■ Too many system components promoted to highest criticality level

- Allocating even one critical requirement to component makes whole thing critical

Requirement Annotation Approach:

- ☒ R01. Lorem ipsum dolor sit amet, consectetur adipiscing elit.
R02. Nam suscipit odio aliquam massa finibus, id imperdiet.
- ☒ R03. Quisque vehicula quam ut dui venenatis varius.
- ☒ R04. Nulla posuere diam ac augue bibendum, vitae laoreet.
R05. Pellentesque aliquam sem sit amet justo porttitor.
- ☒ R06. Vestibulum scelerisque lacus ac neque volutpat, dictum.
- ☒ R07. Ut venenatis ante in ligula efficitur, congue posuere.
- ☒ R08. Nam a nulla ultrices, tempor quam et, fringilla nisl.
- ☒ R09. Vestibulum a arcu interdum, placerat eros non, ultrices.
- ☒ R10. Ut commodo odio eu elit porttitor facilisis.
- ☒ R11. Etiam et sem eu eros congue sollicitudin.
- ☒ R12. Proin tincidunt arcu quis dui tristique volutpat.
R13. Fusce quis magna aliquet, venenatis sem ac, rhoncus.
- ☒ R14. Cras vel nulla eget orci semper varius scelerisque tellus.
- ☒ R15. Cras mollis lorem vitae libero sollicitudin lobortis.
R16. Vestibulum luctus nisi ac nibh varius congue.
- ☒ R17. Maecenas consequat augue eu venenatis euismod.
- ☒ R18. Quisque viverra felis in est ornare consectetur.
- ☒ R19. Cras pellentesque turpis sit amet justo scelerisque.

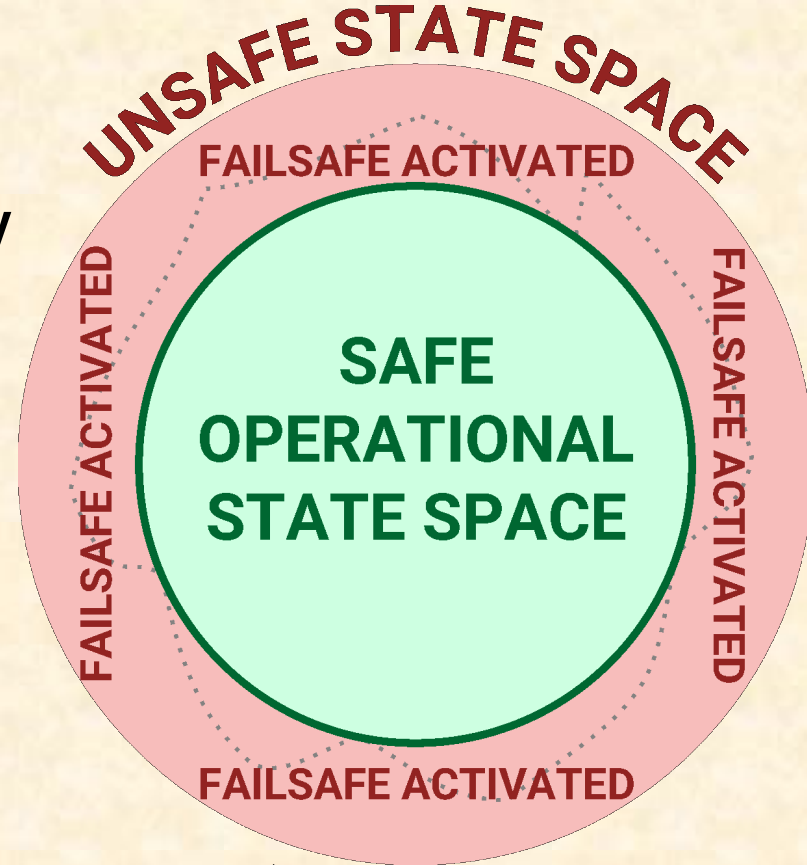
Safety Envelope Requirements Approach

■ Safety Envelope:

- Specify unsafe regions for safety
- Specify safe regions for functionality
 - Deal with complex boundary via:
 - » Under-approximate safe region (reduces **permissiveness**)
 - » Over-approximate unsafe region
- Trigger system safety response upon transition to unsafe region

■ Partition the requirements:

- Operation: functional requirements
- Failsafe: safety requirements (safety functions)



Architecting A Safety Envelope System

■ “Doer” subsystem

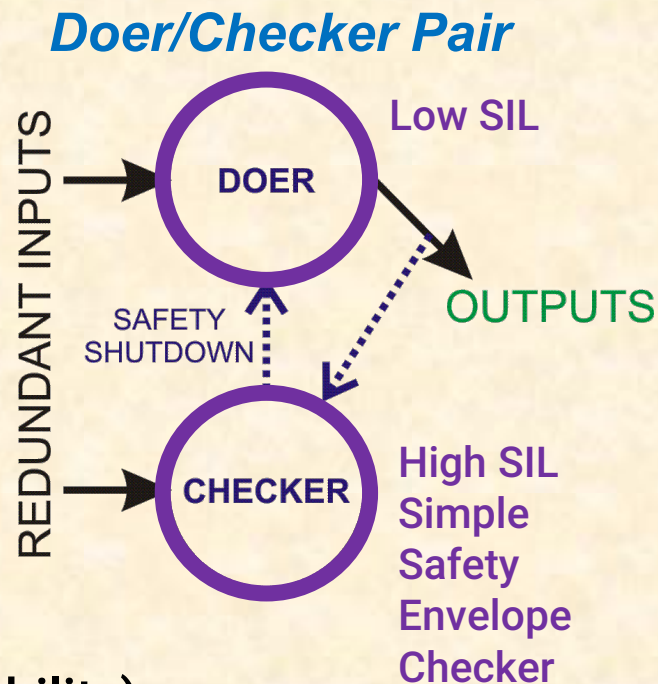
- Implements normal functionality
- Allocate functional requirements to Doer

■ “Checker” subsystem

- Implements failsafes (safety functions)
- Allocate safety requirements to Checker

■ Checker is entirely responsible for safety

- Doer can be at low SIL (failure is lack of availability)
- Checker must be at high SIL (failure is unsafe)
 - Often, Checker can be much simpler than Doer



Safety Requirements Best Practices

■ Doer/Checker pattern

- Functional requirements allocated to low-SIL Doer
- Safety requirements allocated to high-SIL Checker

■ Good safety requirements

- Trace to system-level safety goals
 - Orthogonal to normal functional operation if possible
- Make safety simple to validate (test, peer review)
 - Safety testing mostly exercises the Checker box

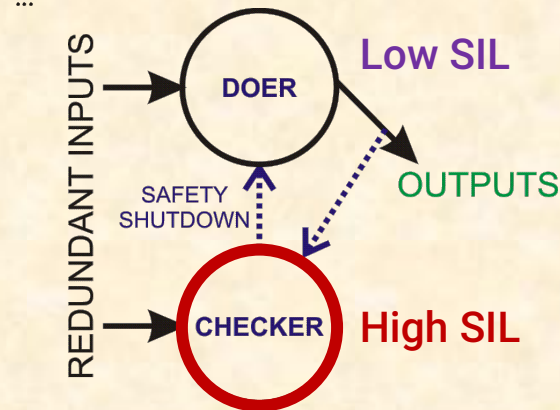
■ Pitfalls:

- Tradeoff between simplicity and permissiveness
 - Doer optimality costs Checker validation effort
- Fail-operational functions may require multiple Doer/Checker pairs

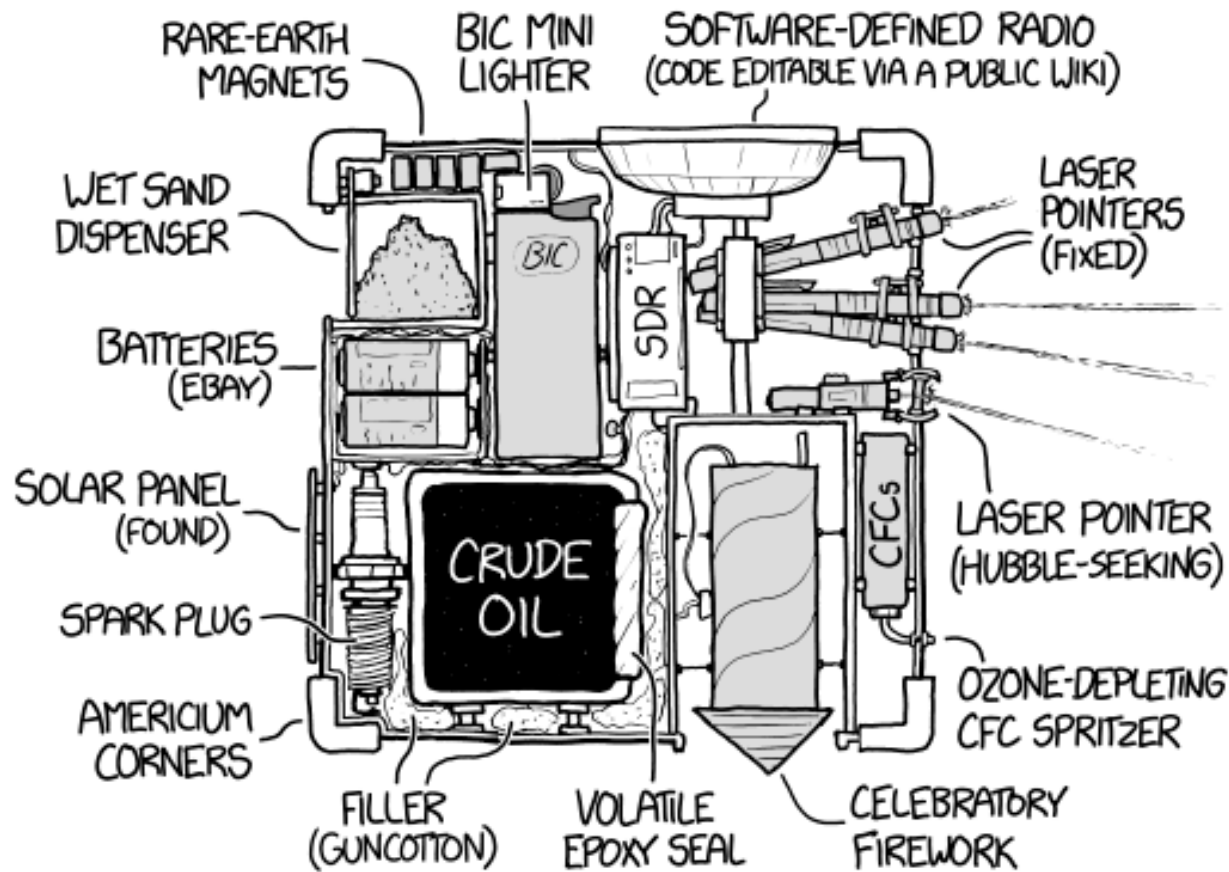
Doer Requirements

D-R01. Lorem ipsum dolor sit amet, consectetur adipiscing elit.
D-R02. Nam suscipit odio aliquam massa finibus, id imperdiet.
D-R03. Quisque vehicula quam ut dui venenatis varius.
D-R04. Nulla posuere diam ac augue bibendum, vitae laoreet.
D-R05. Pellentesque aliquam sem sit amet justo porttitor.
D-R06. Vestibulum scelerisque lacus ac neque volutpat, dictum
D-R07. Ut venenatis ante in ligula efficitur, congue posuere.
D-R08. Nam a nulla ultrices, tempor quam et, fringilla nisl.
D-R09. Vestibulum a arcu interdum, placerat eros non, ultrices.
D-R10. Ut commodo odio eu elit porttitor facilisis.
...

Checker Requirements



C-R01. Et sem eu eros congue sollicitudin.
C-R02. Tincidunt arcu quis dui tristique volutpat.
C-R03. Quis magna aliquet, venenatis sem ac, rhoncus.



MY CUBESAT PROPOSAL WAS THE FIRST TO BE REJECTED FOR VIOLATING EVERY DESIGN AND SAFETY REQUIREMENT SIMULTANEOUSLY.