Safety, privacy, and security

Projects

- Great work on milestone presentations!
  - I will try to read through the reports this weekend
- Next steps
  - Address peer review feedback on FSM (copy spreadsheets, mark each item as “fixed” or “will not fix” with the reason)
  - Keep working towards final demo
  - Keep fleshing out and updating documentation
  - Soon: modeling and verification

Cautionary Tale Presentations

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

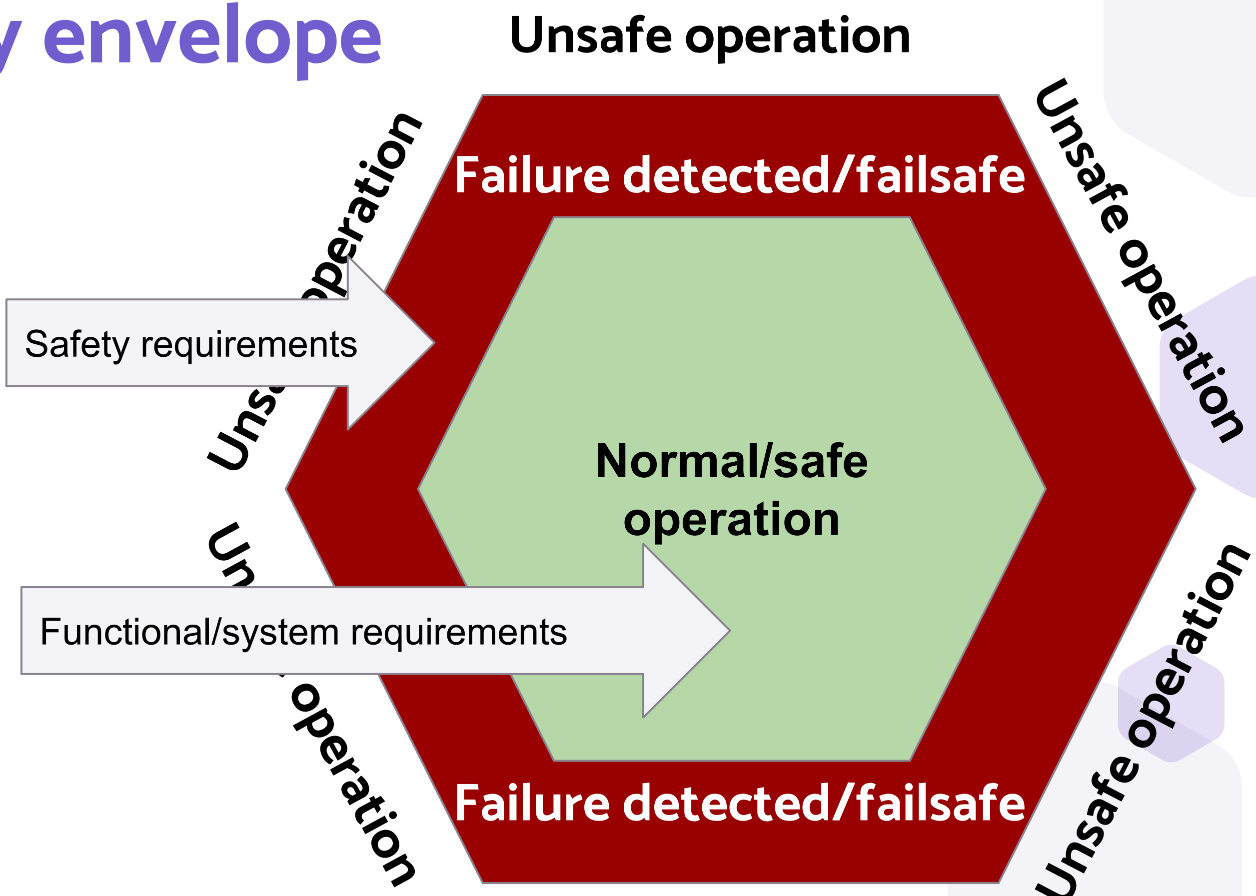
System is assumed unsafe unless engineered for safety

Safety is built-in, not added

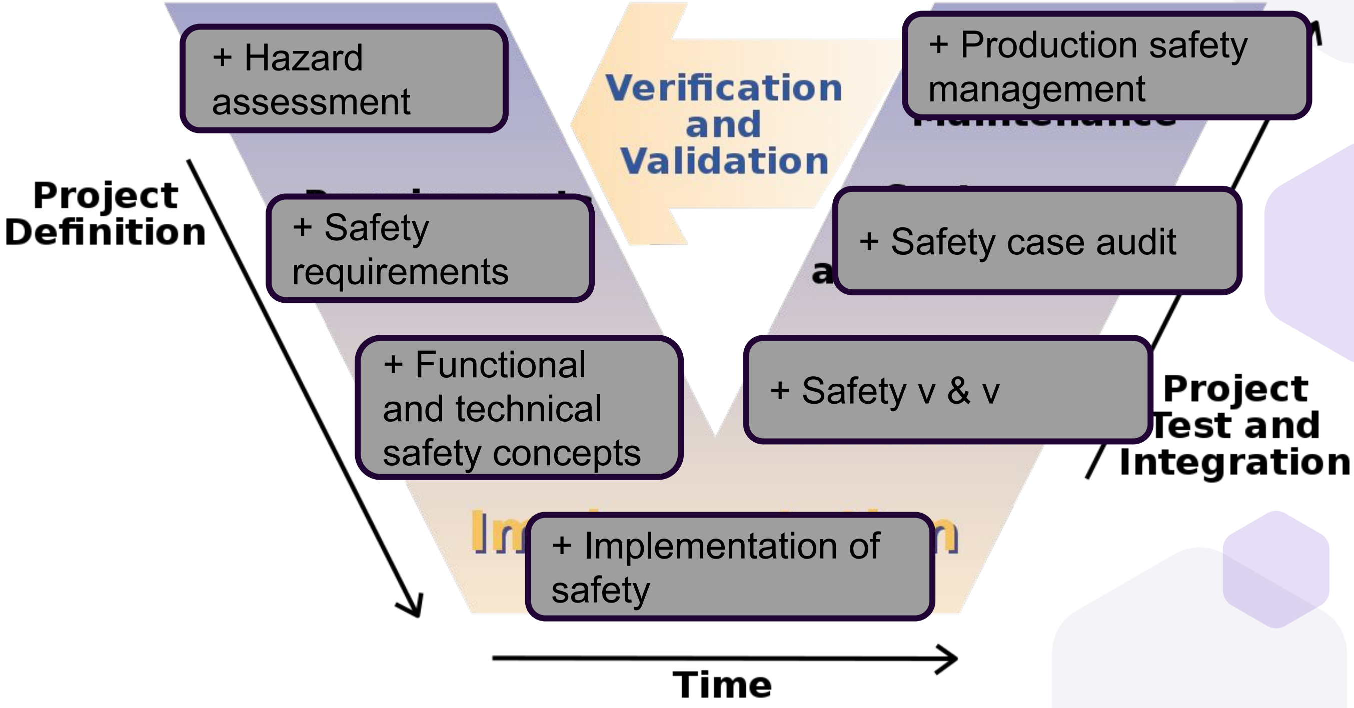
Safety requirements are not an afterthought

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

Safety envelope



Safety V model (applies to security as well)



Safety standards

Guide how to engineer for safety

- How to assess risk
- What SW processes to use
- What code standards to follow
- How much/what kinds of testing
- How much formal verification

Different standards for different domains

- Progression for automotive: MISRA -> IEC 61508 ⟶ ISO 26262 ⟶SOTIF/ISO21448 (⟶UL 4600?)

Safety Integrity Levels

A (standards-based) target to attain for each safety function

Named SIL levels (IEC 61508/ISO 26262 has SIL-1, SIL-2, SIL-3, SIL-4)

- SIL-4 means least acceptable failures (in ISO26262, $< 10^{-9}$ per hour)

Each SIL may require:

- Maximum accepted risk of failure
- Minimum accepted software quality
- Minimum accepted redundancy architecture
- All hardware to be certified at or above that level

} Analysis and mitigation techniques

Different standards for different domains

Approximate cross-domain mapping of ASIL

Domain	Domain-Specific Safety Levels						
Automotive (ISO 26262)	QM	ASIL-A		ASIL-B	ASIL-C	ASIL-D	-
General (IEC 61508)	-	SIL-1		SIL-2		SIL-3	SIL-4
Railway (CENELEC 50126/128/129)	-	SIL-1		SIL-2		SIL-3	SIL-4
Space (ECSS-Q-ST-80)	Category E	Category D		Category C		Category B	Category A
Aviation: airborne (ED-12/DO-178/DO-254)	DAL-E	DAL-D		DAL-C		DAL-B	DAL-A
Aviation: ground (ED-109/DO-278)	AL6	AL5		AL4	AL3	AL2	AL1
Medical (IEC 62304)	Class A	Class B				Class C	-
Household (IEC 60730)	Class A	Class B				Class C	-
Machinery (ISO 13849)	PL a	PL b	PL c	PL d		PL e	-

Image source

Standards inform practice

ISO 26262

Table 3: 7.4.3 Principles for software architectural design		ASIL			
		A	B	C	D
1a	*Hierarchical structure of software components*	++	++	++	++
1b	*Restricted size of software components* [a]	++	++	++	++
1c	*Restricted size of interfaces* [a]	+	+	+	+
1d	*High cohesion within each software component* [b]	+	++	++	++
1e	*Restricted coupling between software components* [a, b, c]	+	++	++	++
1f	*Appropriate scheduling properties*	++	++	++	++
1g	*Restricted use of interrupts* [a, d]	+	+	+	++

Table 4: 7.4.14 Mechanisms for error detection at the software architectural level		ASIL			
		A	B	C	D
1a	*Range checks of input and output data*	++	++	++	++
1b	*Plausibility check* [a]	+	+	+	++
1c	*Detection of data errors* [a]	+	+	+	+
1d	*External monitoring facility* [c]	o	+	+	++
1e	*Control flow monitoring*	o	+	++	++
1f	*Diverse software design*	o	o	+	++

Image source

Risk Matrices

A way of reasoning about the amount of risk of a hazardous event

IEC 61508		Consequence			
Likelihood (failures per year)		Catastrophic	Critical	Marginal	Negligible
		Multiple loss of life	Single loss of life	Major injuries	Minor injuries at worst
Frequent	$> 10^{-3}$			I	II
Probable	10^{-3} - 10^{-4}			II	III
Occasional	10^{-4}-10^{-5}	I		III	III
Remote	10^{-5}-10^{-6}	II			IV
Improbable	10^{-6}-10^{-7}	III	III	IV	IV
Incredible	$< 10^{-7}$	III	IV	IV	

Unacceptable

Undesirable

Tolerable (cost tradeoff)

Acceptable

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

Image source

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

Causes, effects, probabilities, etc

Fault tree analysis

Use boolean logic to determine what low-level failures could cause an anticipated failure

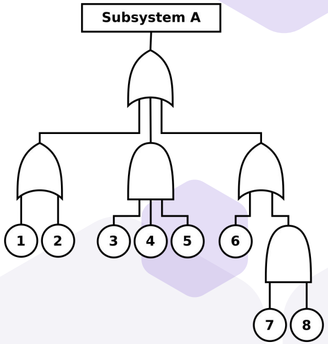
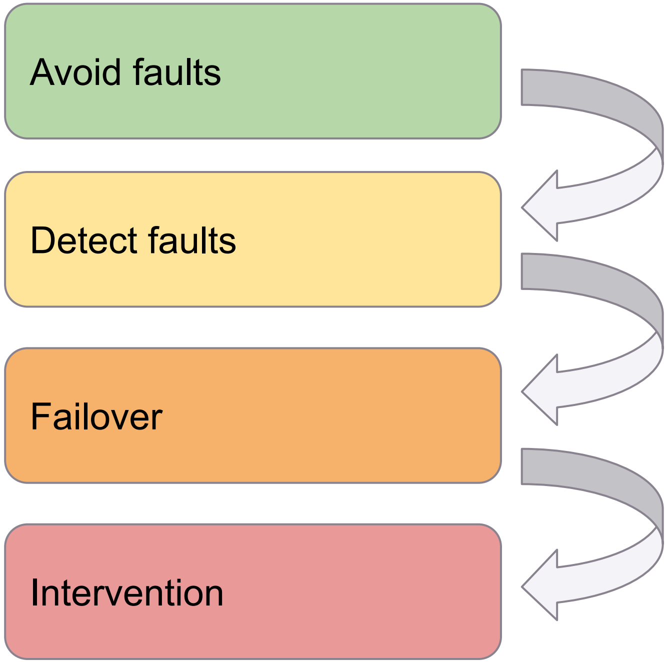


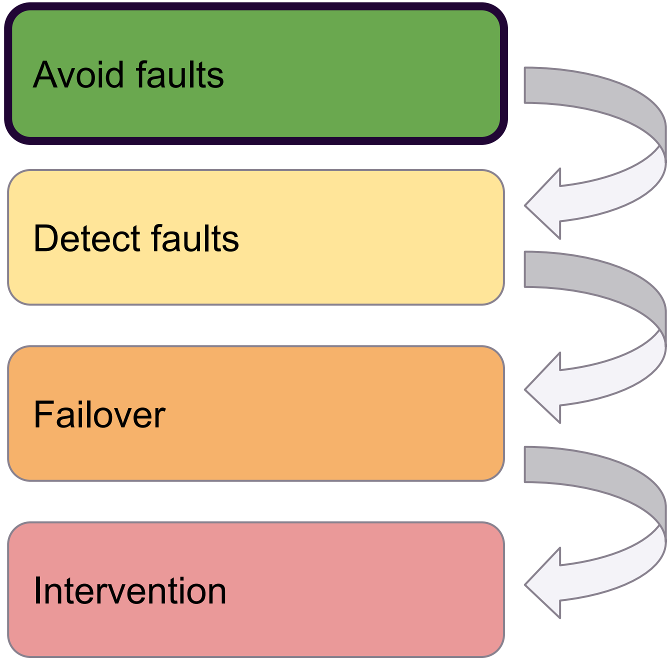
Image source

FTA for scooter

Escalation of safety



Escalation of safety



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

Code style

Style guides (MISRA C)

Spaghetti code

Special topics: global variables, floating point

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

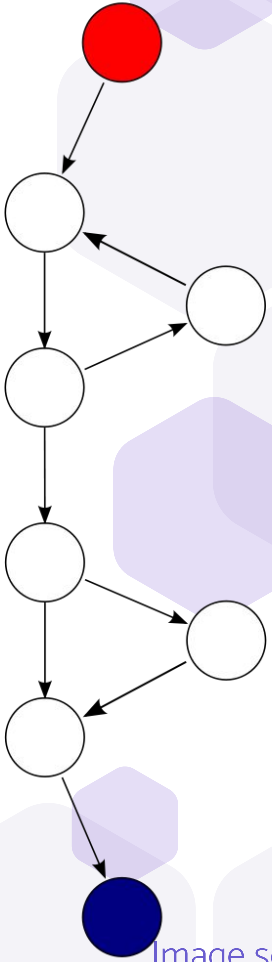
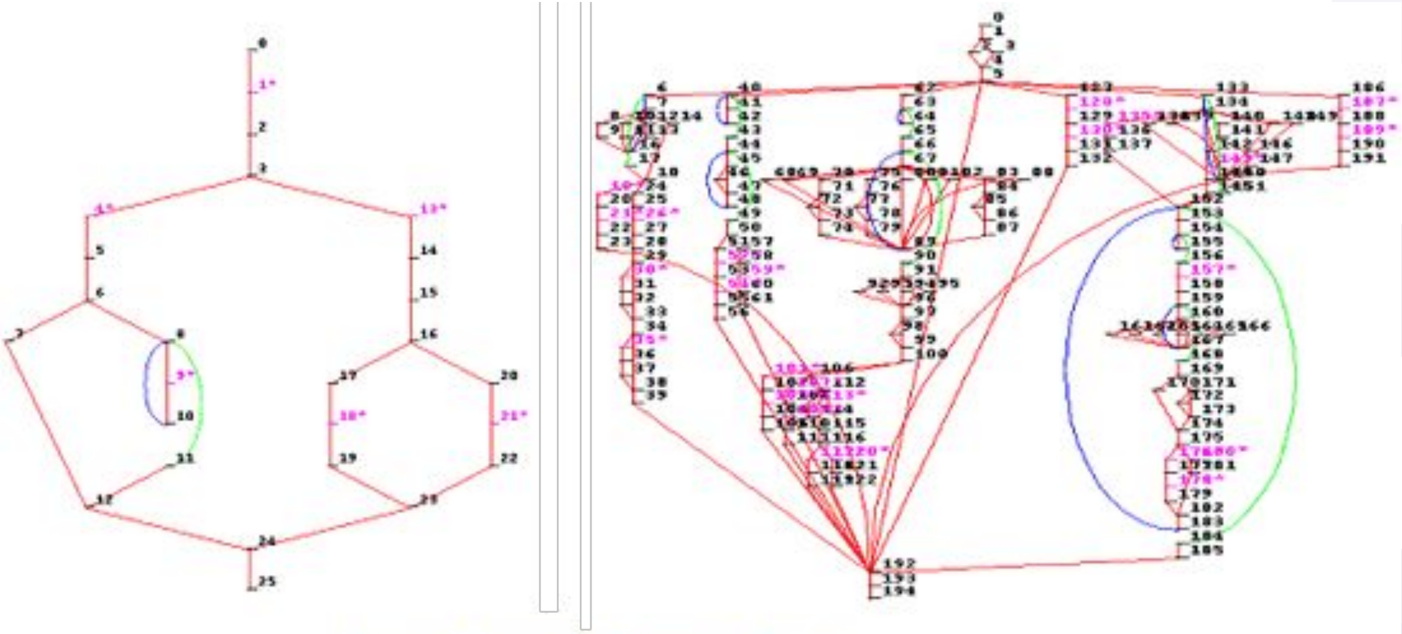


Image source

Which would you rather test/maintain?



Simple Code vs. Complex Code

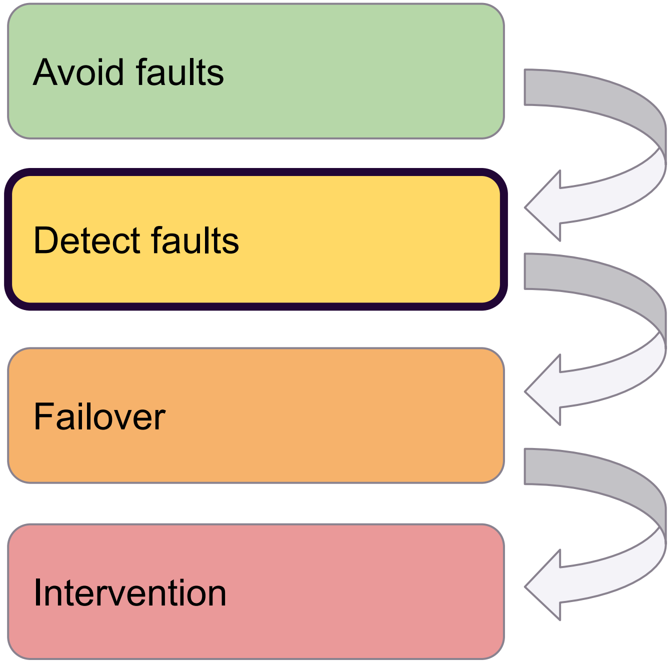
Image source

Why would global variables be considered harmful?

Why would floating point be considered harmful?

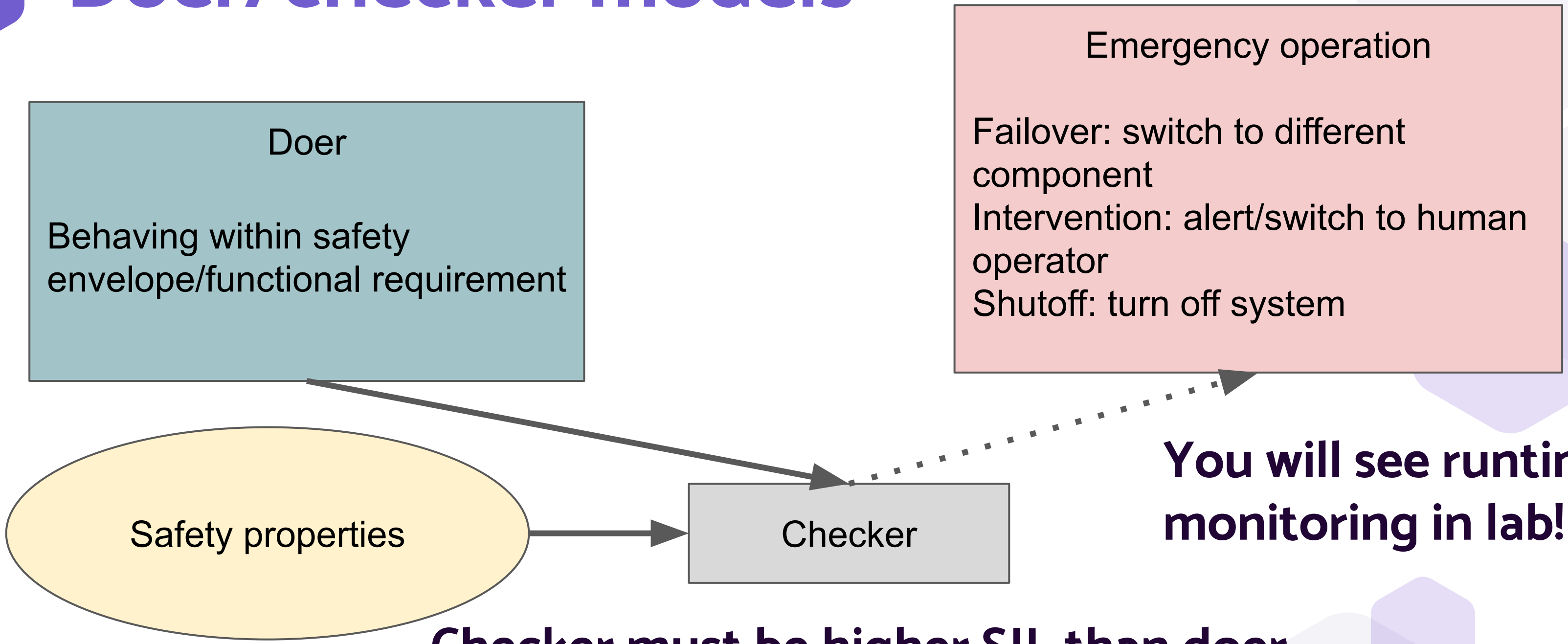
What, besides coding, should be part of a safety-oriented project culture?

Escalation of safety



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

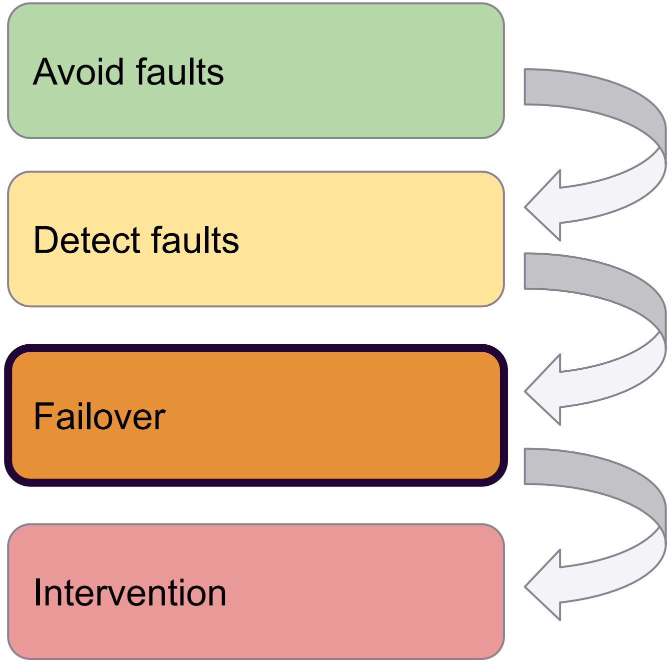
Doer/checker models



You will see runtime monitoring in lab!

Checker must be higher SIL than doer
Must be confident detection/emergency operation won't fail

Escalation of safety



Single points of failure

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

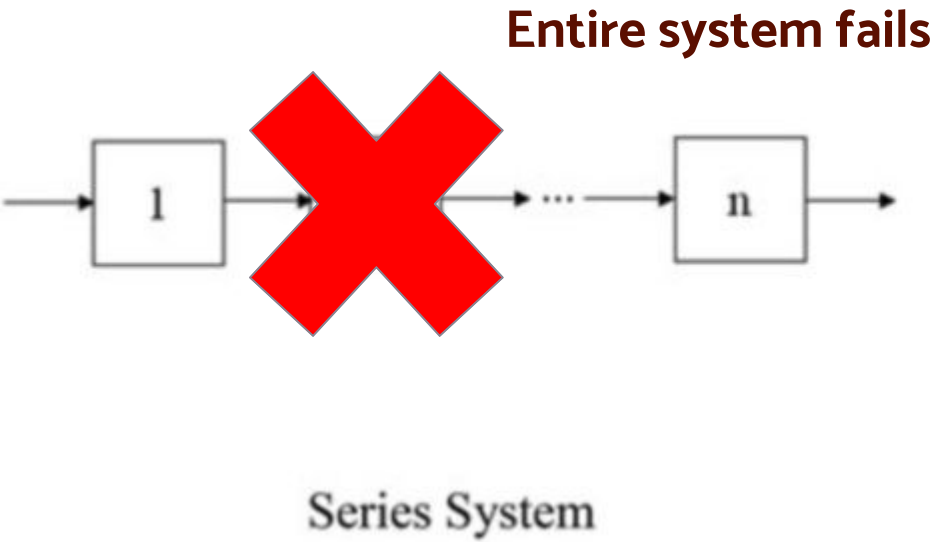
Avoid single points of failure by:

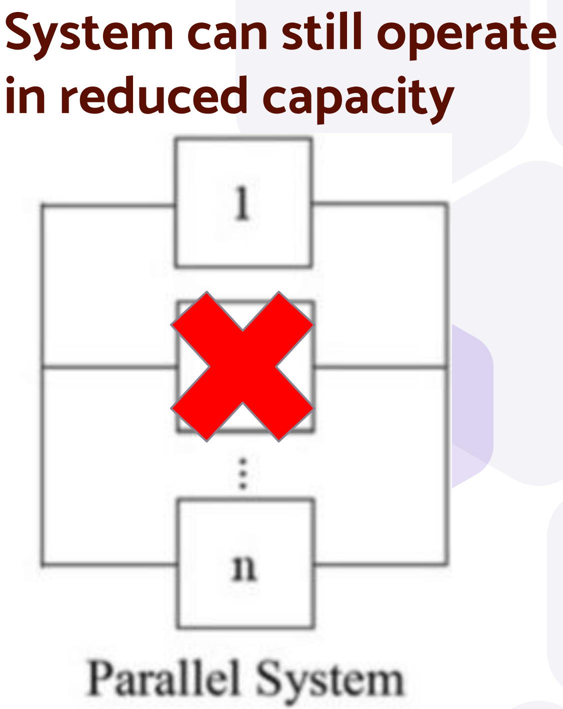
- **Software**: doer/checker with failover
- **Hardware**: failure detection with redundancy

Components must truly be separate for true redundancy

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

Redundancy


Entire system fails
1
n
Series System


System can still operate in reduced capacity
1
n
Parallel System

Redundancy math

Security

Safety is about system failing without an attacker model

Security is about system failing because of adversarial actions

Strategies for security

Do not connect devices to networks unless you need to

Use strong cryptography

Principle of least privilege

- Each component only has access to as much of the system as it needs

Assume user wants to do the bare minimum (**default passwords are dangerous**)

The top 10 most common passwords list:

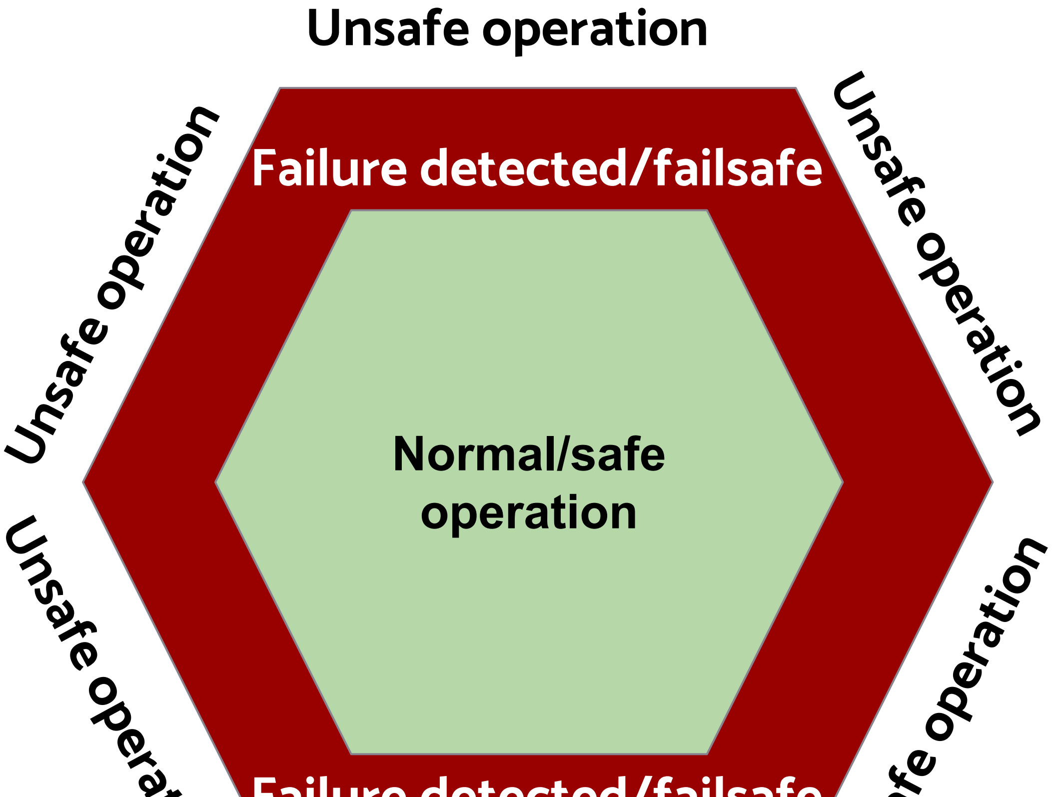
1. 123456
2. 123456789
3. qwerty
4. password
5. 12345
6. qwerty123
7. 1q2w3e
8. 12345678
9. 111111
10. 1234567890

https://cybernews.com/best-password-managers/most-common-passwords/

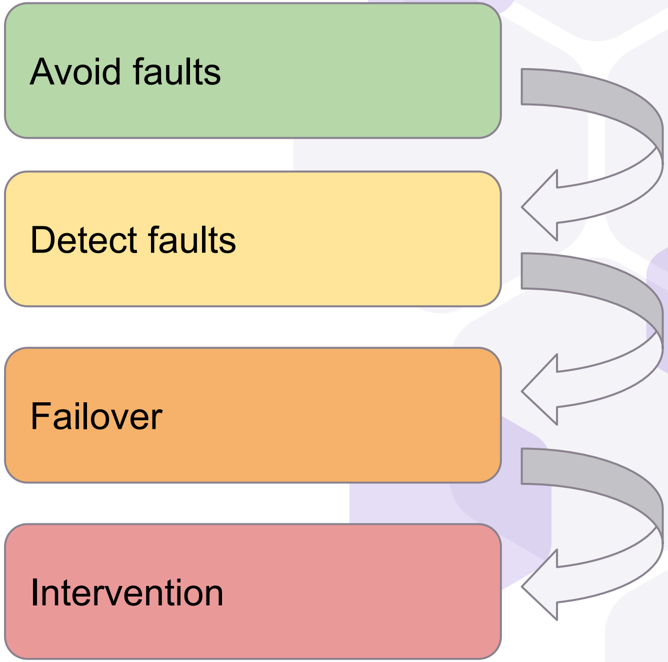
It’s not just software….

Image source

Summary



Unsafe operation
Unsafe operation
Unsafe operation
Failure detected/failsafe
Normal/safe operation
Unsafe operation
Failure detected/failsafe



Avoid faults
Detect faults
Failover
Intervention