"Never tell me the odds!"
— Han Solo

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.
Anti-Patterns for Critical Systems:

- You haven’t characterized worst case failures
- You haven’t assigned SILs to system hazards
- Validation plan doesn’t match fleet exposure

Critical systems require low failure rates

- **SIL** = Safety Integrity Level
  - Higher level of integrity needed for higher risk
- **Safety critical:**
  Loss of life, injury, environmental damage
  - Special care must be taken to avoid deaths
- **Mission critical:**
  Brand tarnish, financial loss, company failure
  - Consider a safety critical approach
Worst case might not be obvious
- Aircraft – software can cause a crash
- Thermostats/HVAC – software can freezing plumbing
  - Can – rarely! – also kill small children due to overheating

Key thought experiment:
- What’s the worst that can happen if …
  … your system intentionally tried to cause harm?
- This identifies system hazards to mitigate

Failure consequence varies, typically:
- Multiple fatalities (e.g., plane crash)
- Single fatality (e.g., single-vehicle car crash)
- Severe injuries
- Minor injuries
- Can consider analogies for mission-critical goals
SIL represents:
- The risk presented by a system-level hazard
- The engineering rigor applied to mitigate the risk
- The permissible residual probability after mitigation

Example: DO-178 (aviation flight hours)
- DAL A (Catastrophic): $10^9$ hrs/failure = 114077 years
- DAL B (Hazardous): $10^7$ hrs/failure = 1141 years
- DAL C (Major): $10^5$ hrs/failure = 11 years
- DAL D (Minor): $10^3$ hrs/failure = 42 days

Example: IEC 61508 (industrial controls)
- SIL 4: $10^8$ hrs/dangerous failure = 11408 years
- SIL 3: $10^7$ hrs/dangerous failure = 1141 years
- SIL 2: $10^6$ hrs/dangerous failure = 114 years
- SIL 1: $10^5$ hrs/dangerous failure = 11 years

1984: Bophal Chemical Plant
Thousands of deaths
(not software related; pre-dates IEC 61508)

https://en.wikipedia.org/wiki/Bhopal_disaster
https://goo.gl/GGHWRn
https://goo.gl/l2RWUv

© 2020 Philip Koopman
Higher SIL Invokes More Engineering Rigor

**Example:**

IEC 61508

- **HR** = Highly Recommended
- **R** = Recommended
- **NR** = Not Recommended (don’t do this)

<table>
<thead>
<tr>
<th>Technique/Measure*</th>
<th>Ref</th>
<th>SIL1</th>
<th>SIL2</th>
<th>SIL3</th>
<th>SIL4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Fault detection and diagnosis</td>
<td>C.3.1</td>
<td>---</td>
<td>R</td>
<td>R</td>
<td>HR</td>
</tr>
<tr>
<td>2 Error detecting and correcting codes</td>
<td>C.3.2</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>HR</td>
</tr>
<tr>
<td>3a Failure assertion programming</td>
<td>C.3.3</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>HR</td>
</tr>
<tr>
<td>3b Safety bag techniques</td>
<td>C.3.4</td>
<td>---</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>3c Diverse programming</td>
<td>C.3.5</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>3d Recovery block</td>
<td>C.3.6</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>3e Backward recovery</td>
<td>C.3.7</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>3f Forward recovery</td>
<td>C.3.8</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>3g Re-try fault recovery mechanisms</td>
<td>C.3.9</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>HR</td>
</tr>
<tr>
<td>3h Memorising executed cases</td>
<td>C.3.10</td>
<td>---</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>4 Graceful degradation</td>
<td>C.3.11</td>
<td>R</td>
<td>R</td>
<td>HR</td>
<td>HR</td>
</tr>
<tr>
<td>5 Artificial intelligence - fault correction</td>
<td>C.3.12</td>
<td>---</td>
<td>NR</td>
<td>NR</td>
<td>NR</td>
</tr>
<tr>
<td>6 Dynamic reconfiguration</td>
<td>C.3.13</td>
<td>---</td>
<td>NR</td>
<td>NR</td>
<td>NR</td>
</tr>
<tr>
<td>7a Structured methods including for example, JSD, MASCOT, SADT and Yourdon.</td>
<td>C.2.1</td>
<td>HR</td>
<td>HR</td>
<td>HR</td>
<td>HR</td>
</tr>
<tr>
<td>7b Semi-formal methods</td>
<td>Table B.7</td>
<td>R</td>
<td>R</td>
<td>HR</td>
<td>HR</td>
</tr>
<tr>
<td>7c Formal methods including for example, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z</td>
<td>C.2.4</td>
<td>---</td>
<td>R</td>
<td>R</td>
<td>HR</td>
</tr>
<tr>
<td>8 Computer-aided specification tools</td>
<td>B.2.4</td>
<td>R</td>
<td>R</td>
<td>HR</td>
<td>HR</td>
</tr>
</tbody>
</table>
Bigger fleets have increased exposure

- 250 Million US vehicles @ 1 hour/day = $2.5 \times 10^8$ hrs/day exposure
- If “unlikely” failures happen every million hours...
  - that’s: $2.5 \times 10^8$ hrs / $10^6$ hrs per event
  - $\Rightarrow$ 250 events every day
- This is why $10^8$ to $10^{10}$ hrs is a typical goal

Hardware components fail at $\sim 10^5$-$10^6$ hrs

- Need two independently failing components to get to $10^9$ hours!
  - This motivates redundancy for life-critical applications (SIL 3 & SIL 4)

For mission-critical systems, consider:

- Fleet exposure = # units * operational hours/unit
- Number of acceptable failures
- Compute failure rate = failures / hours; pick an appropriate SIL
“Recently we detected a bug in the firmware of our scooter fleet that under rare circumstances could cause sudden excessive braking during use.”
Best Practices For Critical Systems

- Characterize worst case failure scenarios
  - Assign SIL based on relevant safety standard
  - Use engineering rigor for software SIL
  - Use redundancy for ultra-low failure rates
  - Consider fleet exposure, not just single unit

- Pitfalls:
  - Software redundancy is difficult, and diversity is usually impracticable
  - Designer’s intuition about “realistic” faults usually optimistic
    - At $10^{-9}$/hr, random chance is a close approximation of a malicious adversary
  - Going through the motions not enough for SIL-based process
ASKING AIRCRAFT DESIGNERS ABOUT AIRPLANE SAFETY:

Nothing is ever foolproof, but modern airliners are incredibly resilient. Flying is the safest way to travel.

WAIT, REALLY?

Don't trust voting software and don't listen to anyone who tells you it's safe.

Why?

I don't quite know how to put this, but our entire field is bad at what we do, and if you rely on us, everyone will die.

ASKING BUILDING ENGINEERS ABOUT ELEVATOR SAFETY:

Elevators are protected by multiple, tried-and-tested failsafe mechanisms. They're nearly incapable of falling.

They say they've fixed it with something called "blockchain."

AAAAA!!

Whatever they sold you, don't touch it. Bury it in the desert. Wear gloves.

ASKING SOFTWARE ENGINEERS ABOUT COMPUTERIZED VOTING:

That's terrifying.