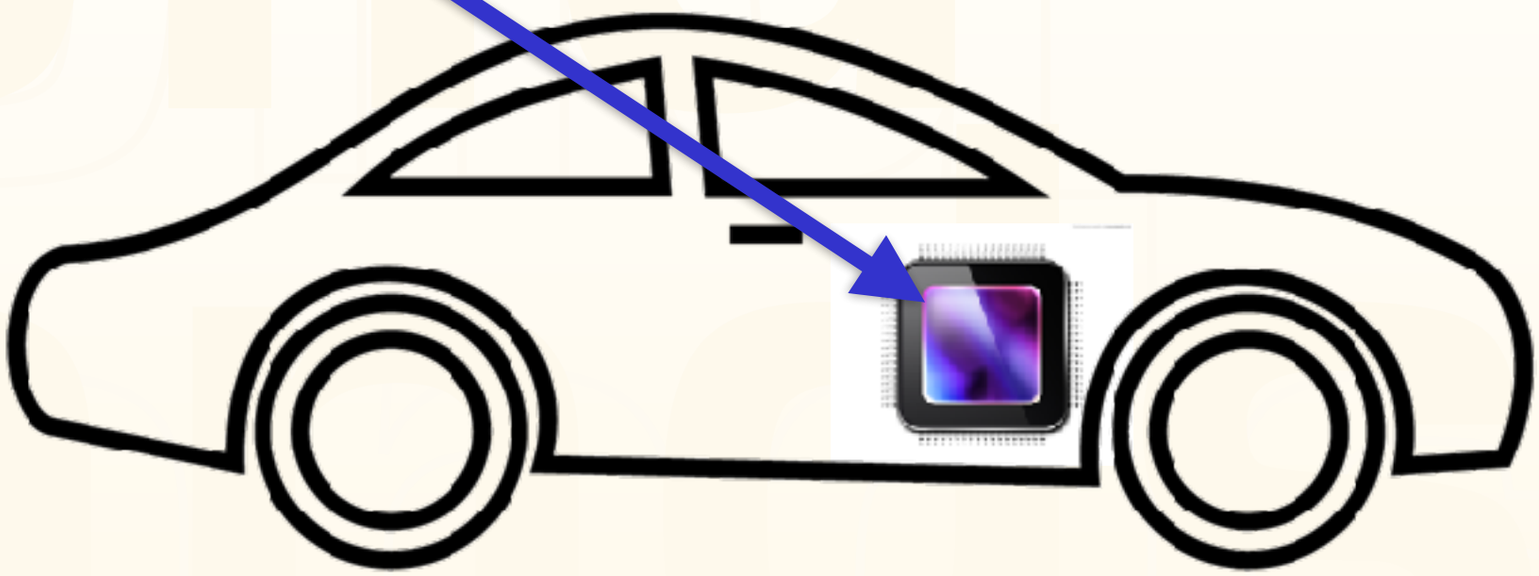
Using LLVM for Safety-Critical Applications

dr. Marcel Beemster
CTO Solid Sands

Would Drive that Car?

C-Source Code for Brakes

LLVM Compiler



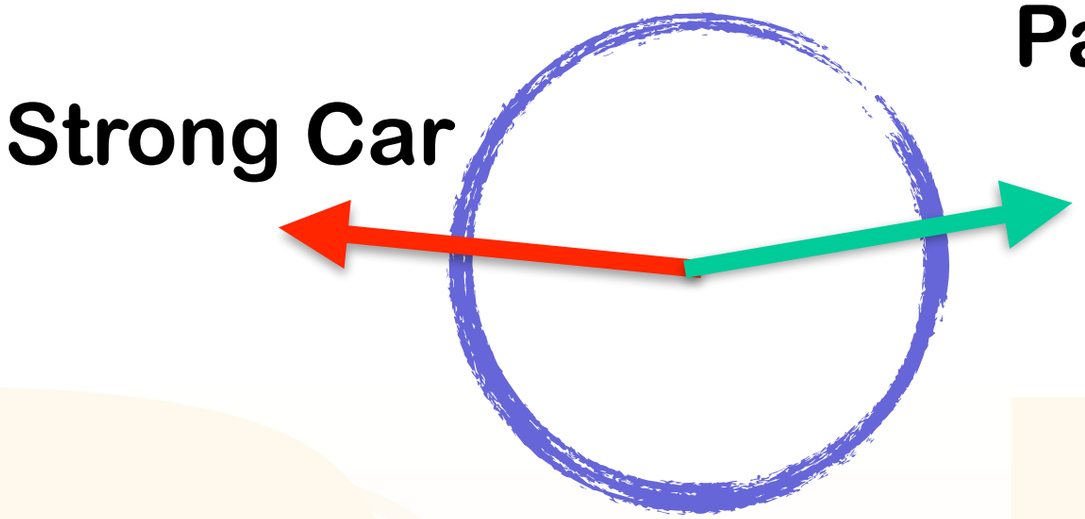
What is Safety?

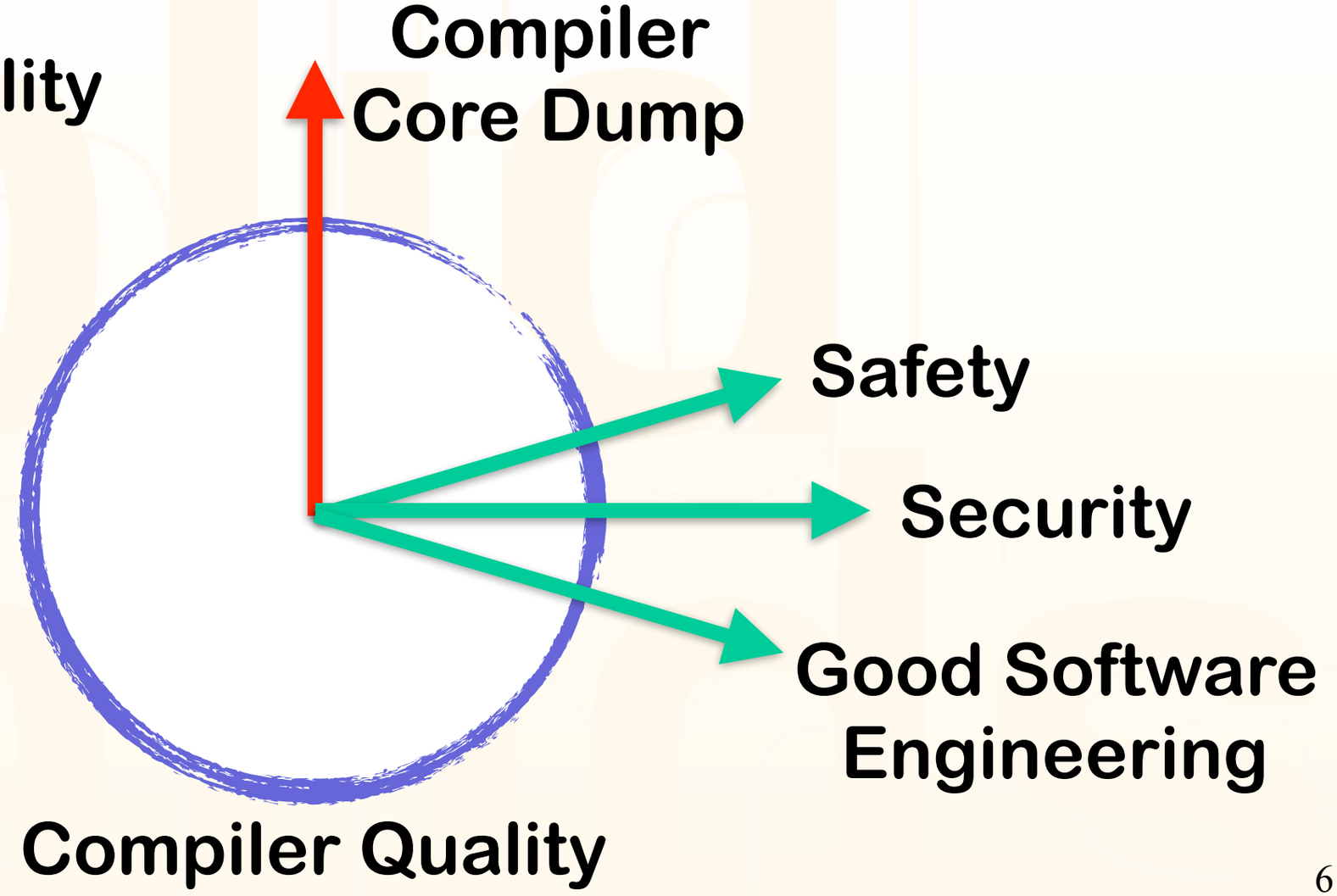
Can a Compiler be “Safe”?

How does Safety Relate to Quality?

DEMO
Safety vs Quality

clang++ compilation


Strong Car
Passenger Safety
Car Quality


Compiler Core Dump
Safety
Security
Good Software Engineering
Compiler Quality

Definition: Functional Safety

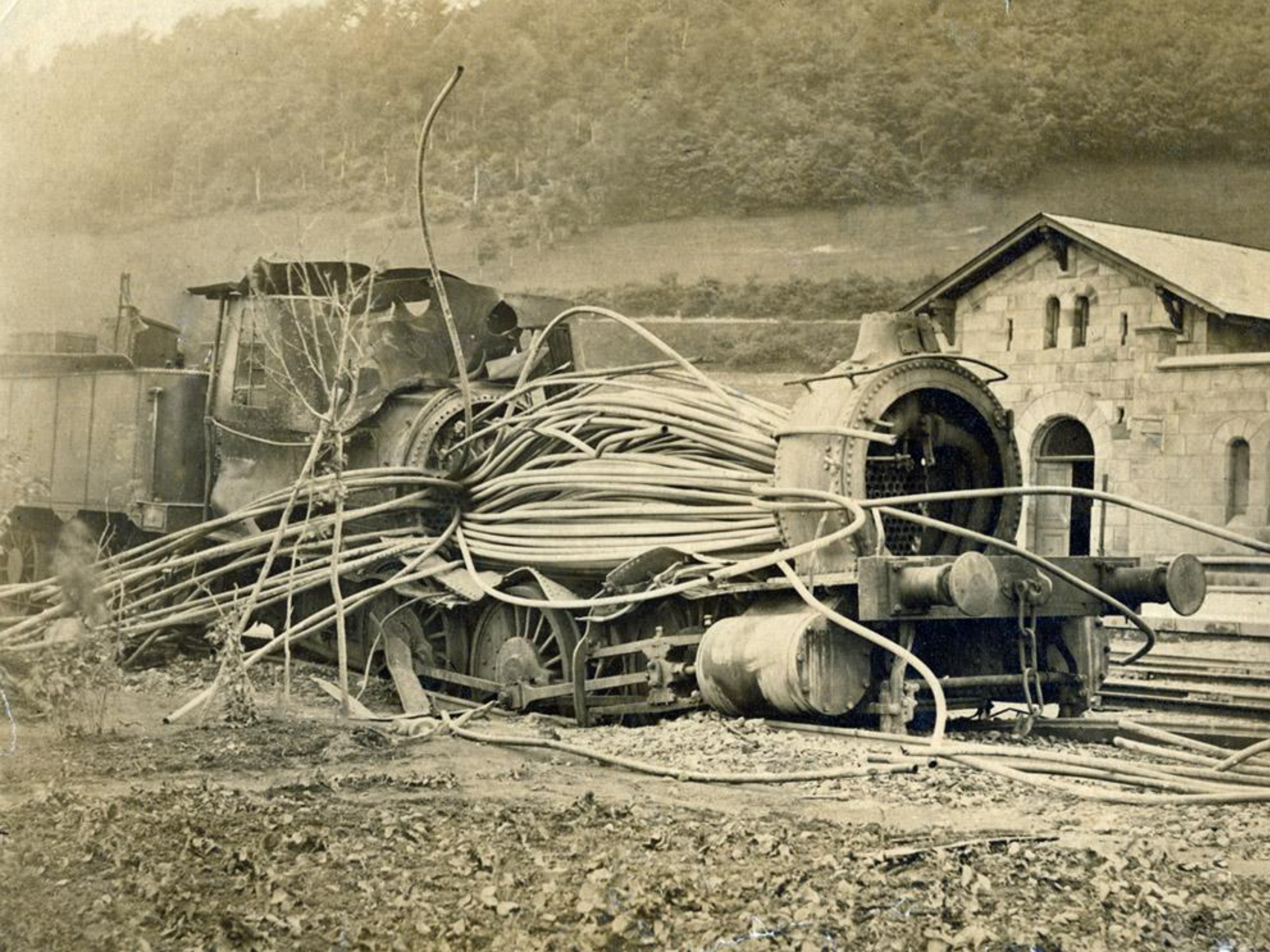
"Absence of unreasonable risk due to hazards caused by malfunctioning behaviour"

system. Use care
to the side of the oven when
The oven may still be VERY HOT.

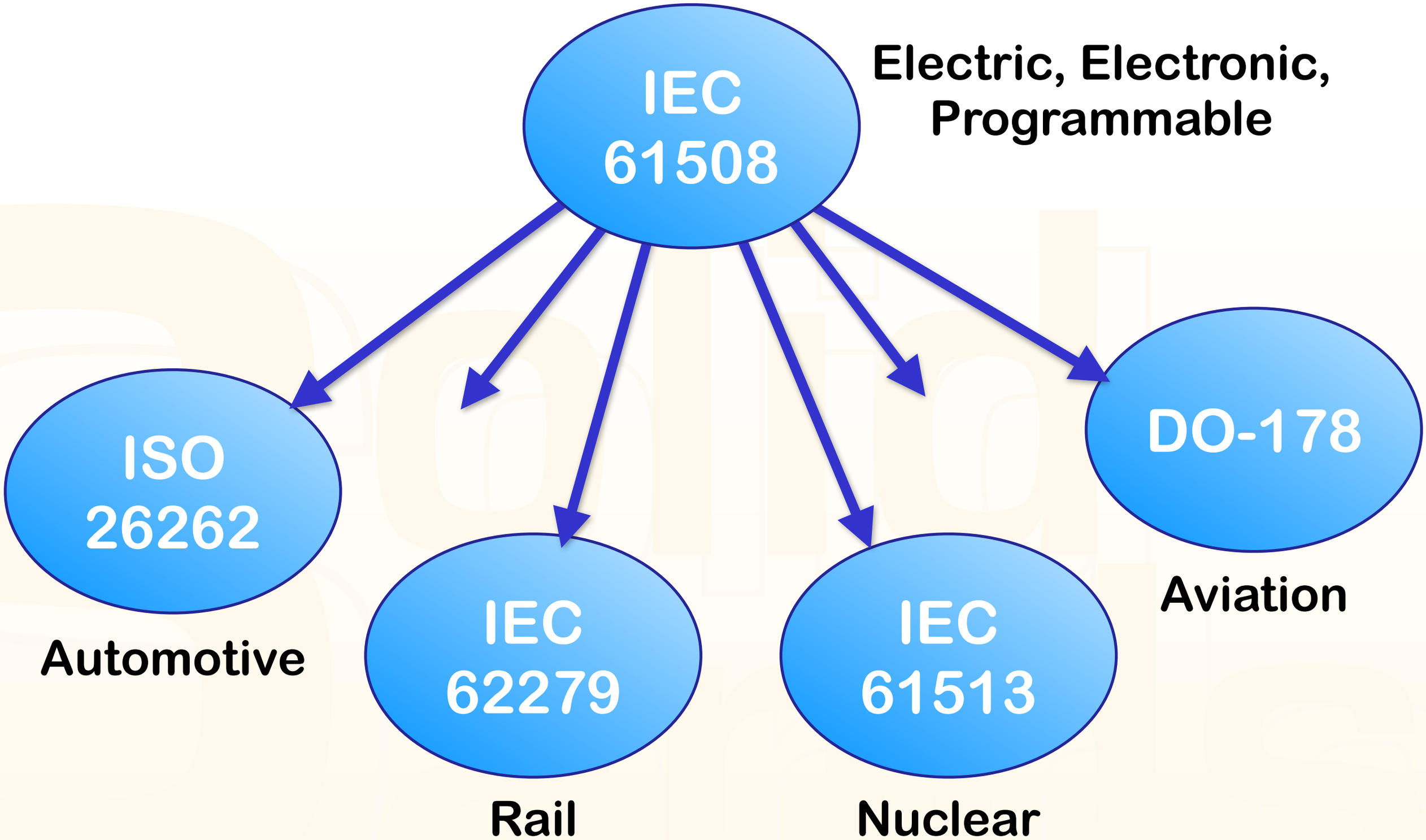
- **DO NOT use harsh etching, abrasive cleaners**
the oven door glass since they can scratch the
glass to shatter.
- **Be careful when removing and lifting the doo**
- **DO NOT lift the door by the handle.** The door i
- **DO NOT** use utensils for removing refuse (ash, f
- Refer to the installation manual for proper anti-
- Never remove the oven legs. The range will not
the legs are removed.

⚠ CAUTION

- **DO NOT line the oven walls, racks, bottom,**
aluminum foil or any other material. Doi
produce poor baking results and cause per
(aluminum foil will melt to the interior surfac
- Never attempt to dry a pet in the oven.



Taxonomy of Functional Safety Standards



1. Vocabulary

2. Management of functional safety

- **2-5** Overall safety management
- **2-6** Safety management during the concept phase and the product development
- **2-7** Safety management after the item´s release for production

3. Concept phase

- **3-5** Item definition
- **3-6** Initiation of the safety lifecycle
- **3-7** Hazard analysis and risk assessment
- **3-8** Functional safety concept

4. Product development at the system level

- **4-5** Initiation of product development at the system level
- **4-6** Specification of the technical safety requirements
- **4-7** System design
- **4-8** Item integration and testing
- **4-9** Safety validation
- **4-10** Functional safety assessment
- **4-11** Release for production

5. Product development at the hardware level

- **5-5** Initiation of product development at the hardware level
- **5-6** Specification of hardware safety requirements
- **5-7** Hardware design
- **5-8** Evaluation of the hardware architectural metrics
- **5-9** Evaluation of the safety goal violations due to random hardware failures
- **5-10** Hardware integration and testing

6. Product development at the software level

- **6-5** Initiation of product development at the software level
- **6-7** Software architectural design
- **6-8** Software unit design and implementation
- **6-9** Software unit testing
- **6-10** Software integration and testing
- **6-11** Verification of software safety requirements

7. Production and operation

- **7-5** Production
- **7-6** Operation, service (maintenance and repair), and decommissioning

8. Supporting processes

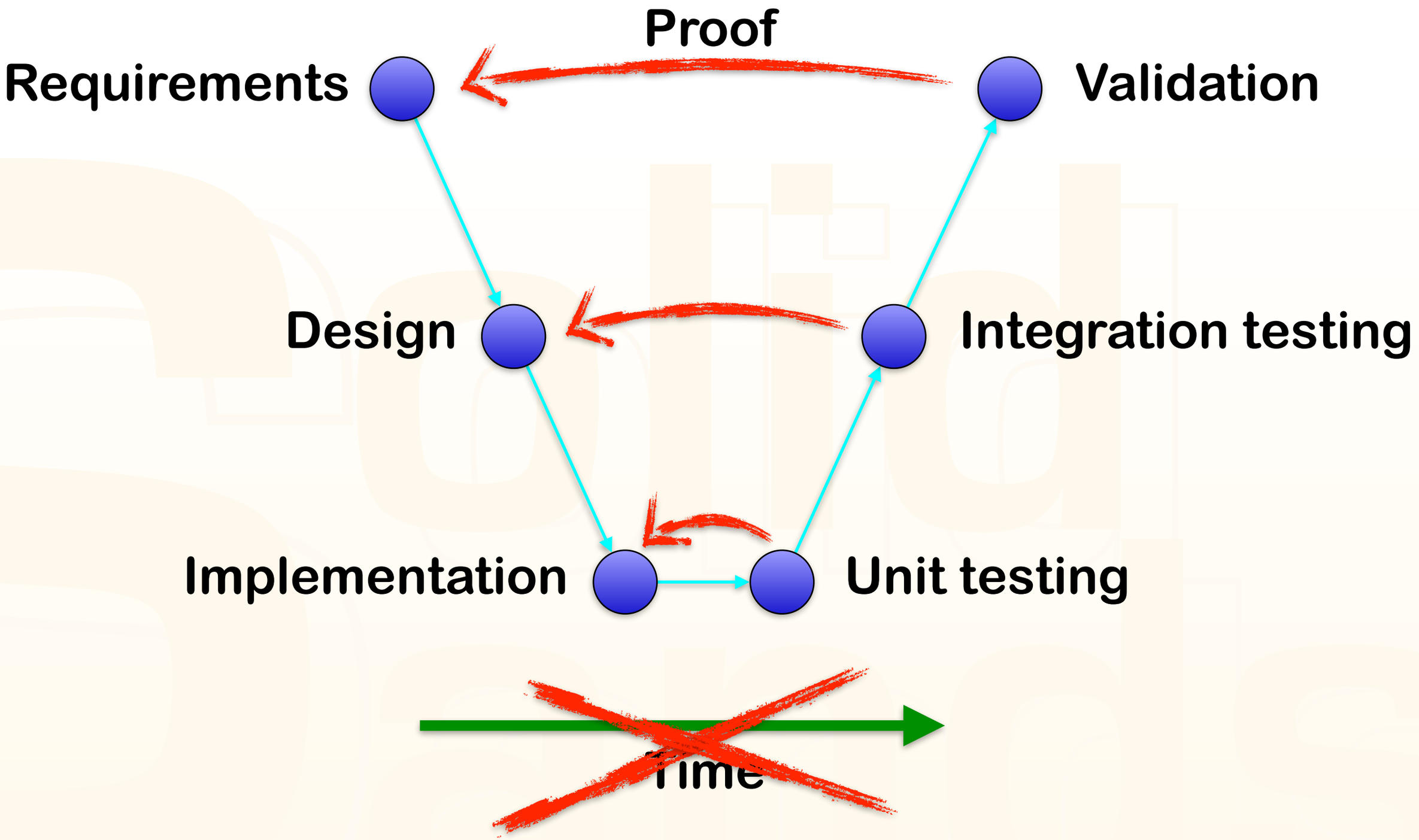
- **8-5** Interfaces within distributed developments
- **8-6** Specification and management of safety requirements
- **8-7** Configuration management
- **8-8** Change management
- **8-9** Verification
- **8-10** Documentation
- **8-11** Confidence in the use of software tools
- **8-12** Qualification of software components
- **8-13** Qualification of hardware components
- **8-14** Proven in use argument

9. ASIL-oriented and safety-oriented analyses

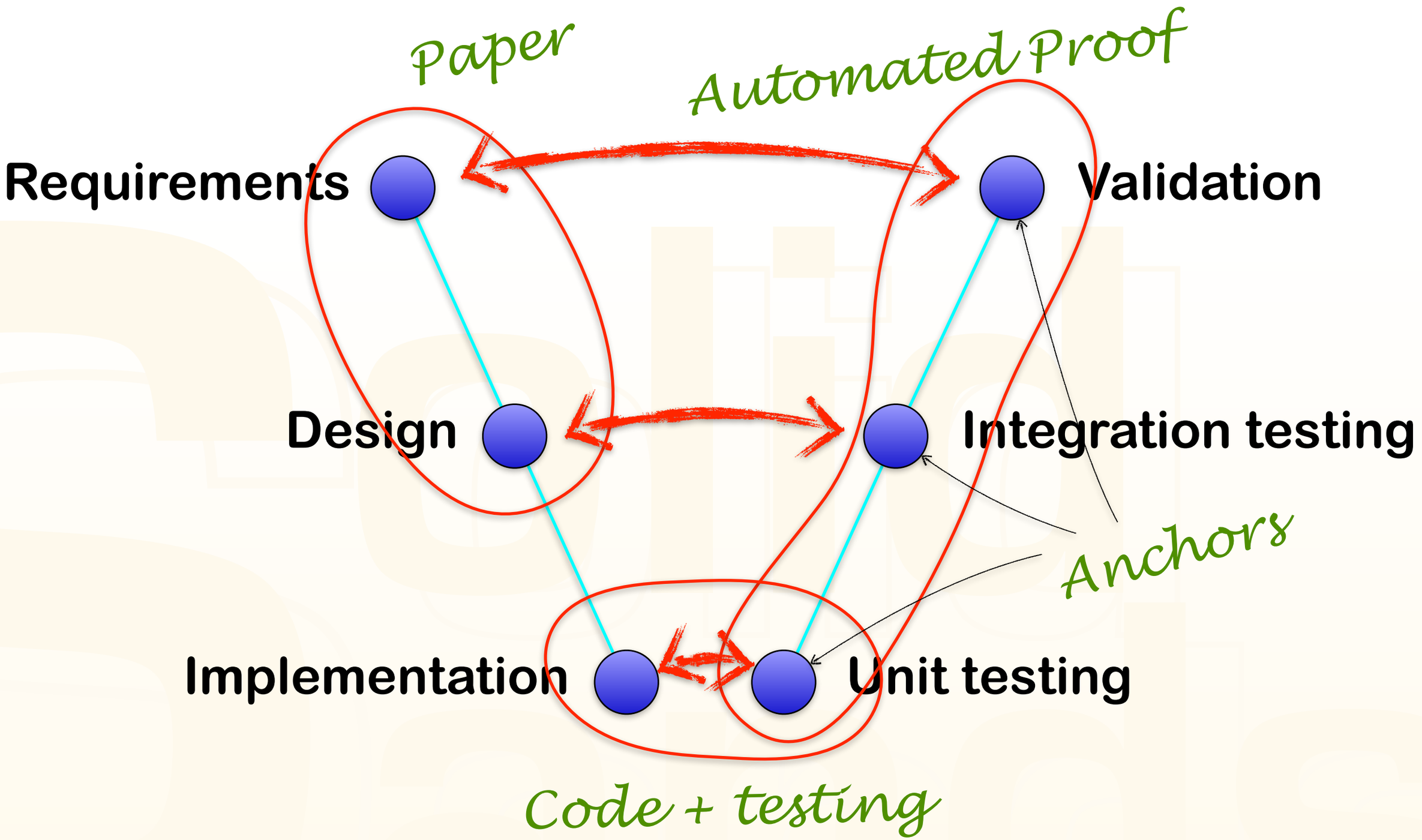
- **9-5** Requirements decomposition with respect to ASIL tailoring
- **9-6** Criteria for coexistence of elements
- **9-7** Analysis of dependent failures
- **9-8** Safety analyses

10. Guideline on ISO 26262

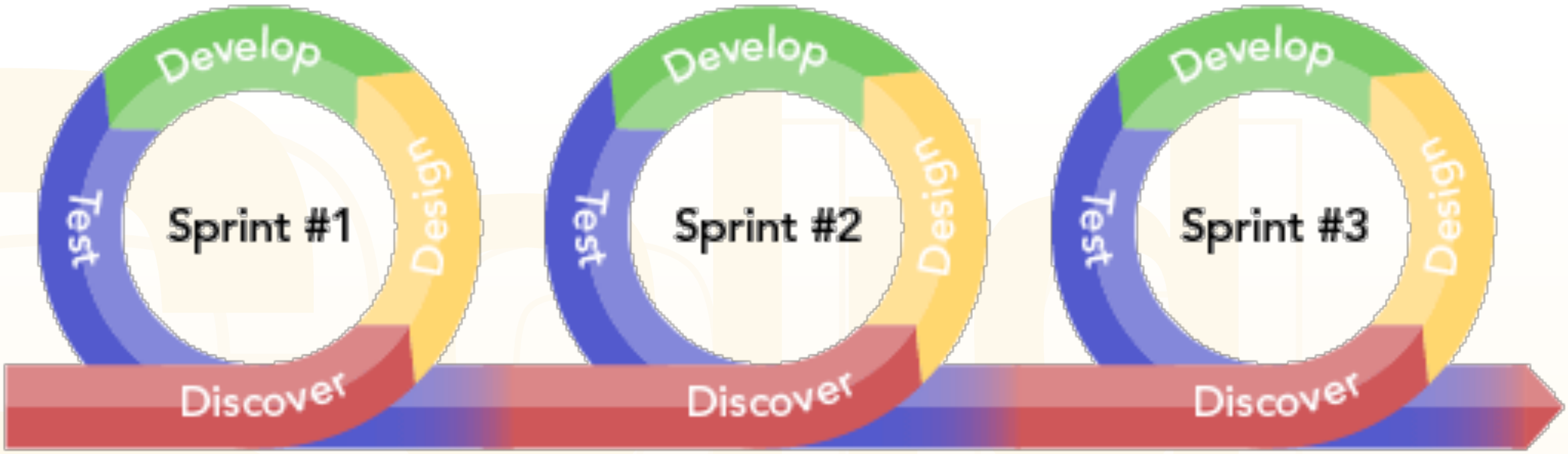
V-Model


Proof
Requirements
Validation
Design
Integration testing
Implementation
Unit testing
Time

Closing the Loop



Iterative Development Fits Well with V-Model



ISO 26262

By requiring documentation

ISO 26262 ensures that components are used within the bounds of their specification

How safe is a piece of wire?

“Use-Case”

How safe is a piece of wire?

Who is responsible?

ISO 26262, Part 8, Section 11

ISO 26262, Part 6

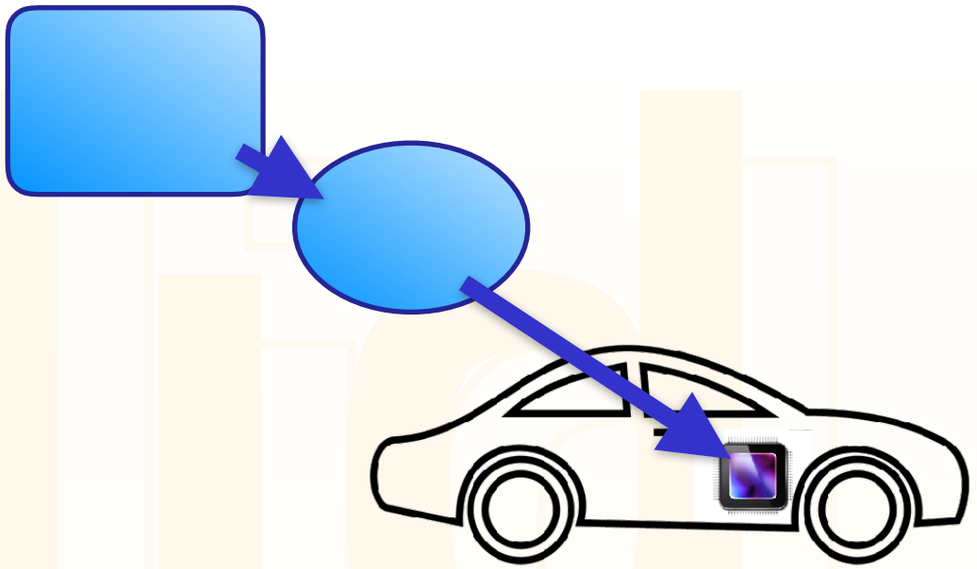
Compiler

Standard Library

Application

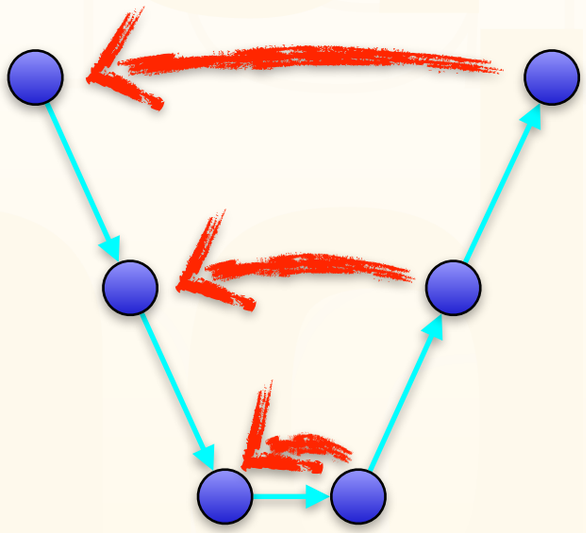
Compiler Qualification

- **Define and document use-case**

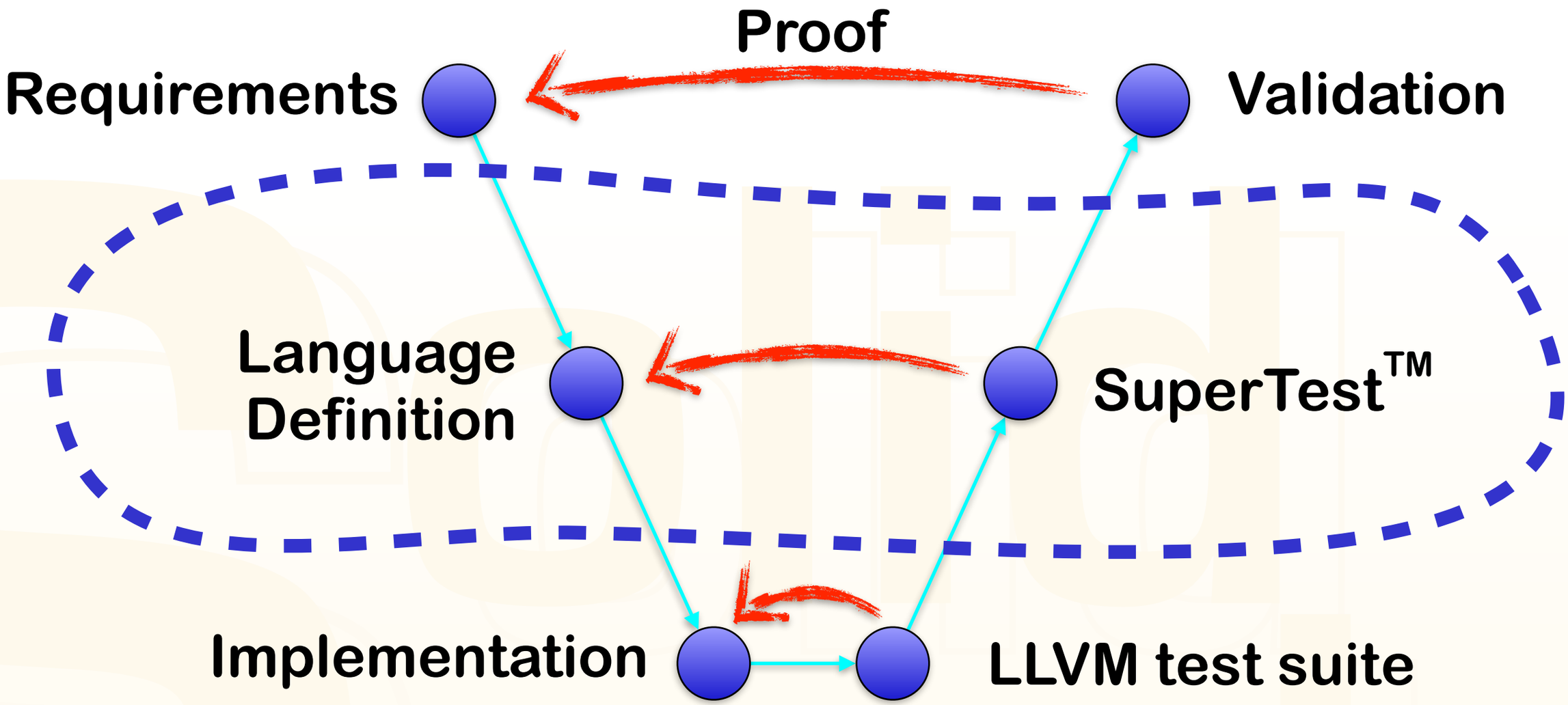


- **Assessment of risk**

- **If needed: validate by verification against the language standard**



Verification Against Language Specification



Software Development: Application & Library

- **Definition of Use-Case; Risk Assessment**
- **Proofs according to V-Model**
- **Using suitable programming guidelines, e.g., MISRA-C**
- **Unit-testing demonstrating 100% MC/DC coverage**

Take Home Messages

- **Quality (Software Engineering) and Safety are not the same, but go together well**

- **The concept of “Use-Case” plays a central role in defining Functional Safety, also for compilers**

- **Compiler qualification is not rocket science**

- **Not all test-suites are created equal***

*** That is why you also need SuperTest!**

Thank You!

Marcel Beemster
marcel@solidsands.nl
www.solidsands.nl