KPIT

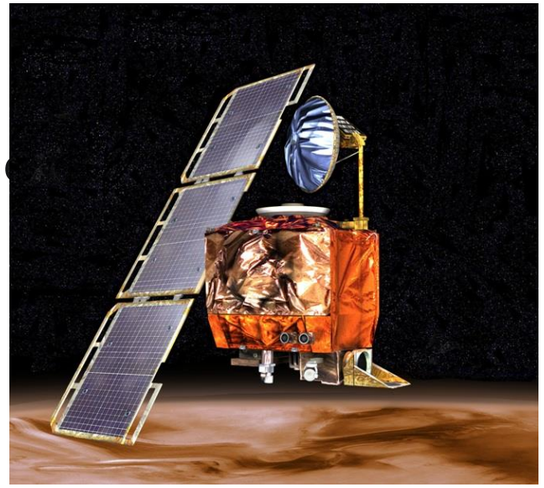
Session-VII
Verification/Validation and Code Quality

Session outline

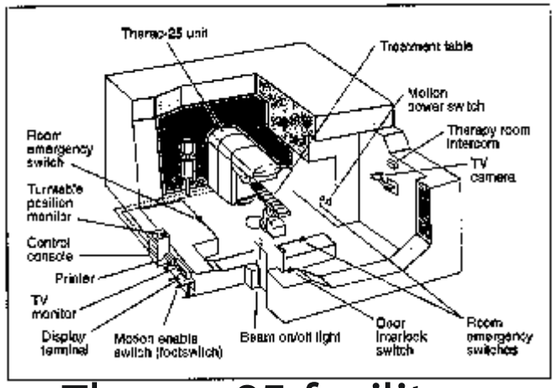
- Various system failures in the past
- Understanding verification and validation
- Role of testing in Verification-Validation
- Automotive perspective on Verification/Validation and code quality
- MISRA-C role in ensuring code quality

Learning from past Software disasters

- **1999:** NASA's Mars Climate Orbiter burned up in the Martian atmosphere b
  failed to convert units from English to metric.
- **Potential Error:**
- Problem was in the software controlling the orbiter's thrusters.
- Calculations happened in Pound instead of Newton.



NASA's Mars Climate Orbiter



Therac-25 facility

1985 and1987: (Failure of the Therac-25)

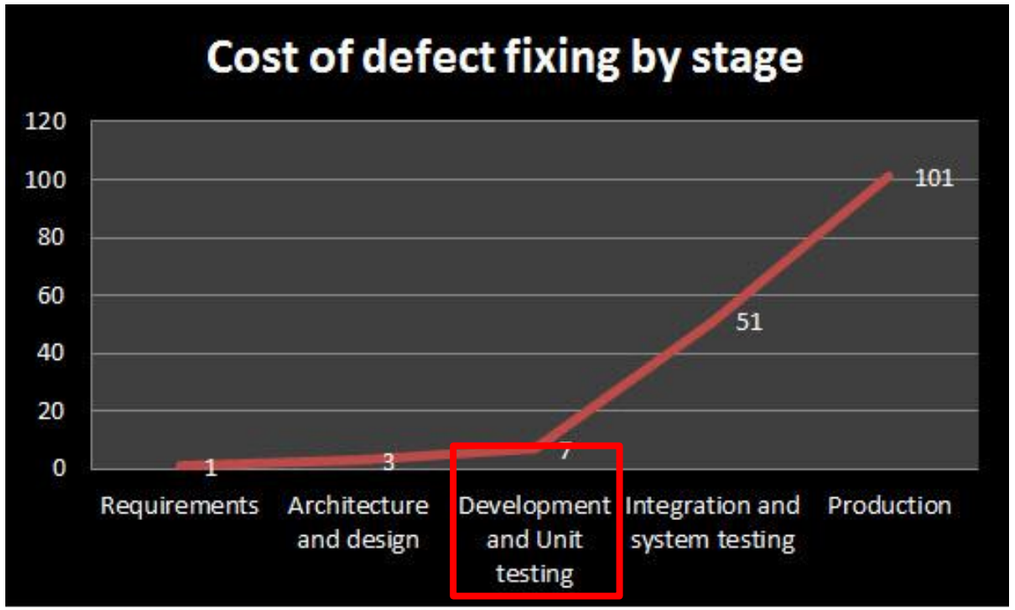
- Not well-designed error and warning messages lead to administered high dosage of radiation to patient.
- No proper hardware is installed to catch safety glitches.

1991: Patriot missile defense system failed to track scud missile

- The system was operating continuously for over 100 hours" that leads to timing error.
- Software was Unable to track location of scud missile

Challenges for software developers

- **Main challenges any software development team faces:**
- Time
- Cost and
- Quality



Ref - Software Engineering Economics by Barry Boehm

Economic Impact ***(Source: Planning Report of NIST US Dept. of Commerce)***

Inadequate software testing costs the US alone between $22 and $59 billion annually.

Automotive Sector accounts for $1.2 billion.

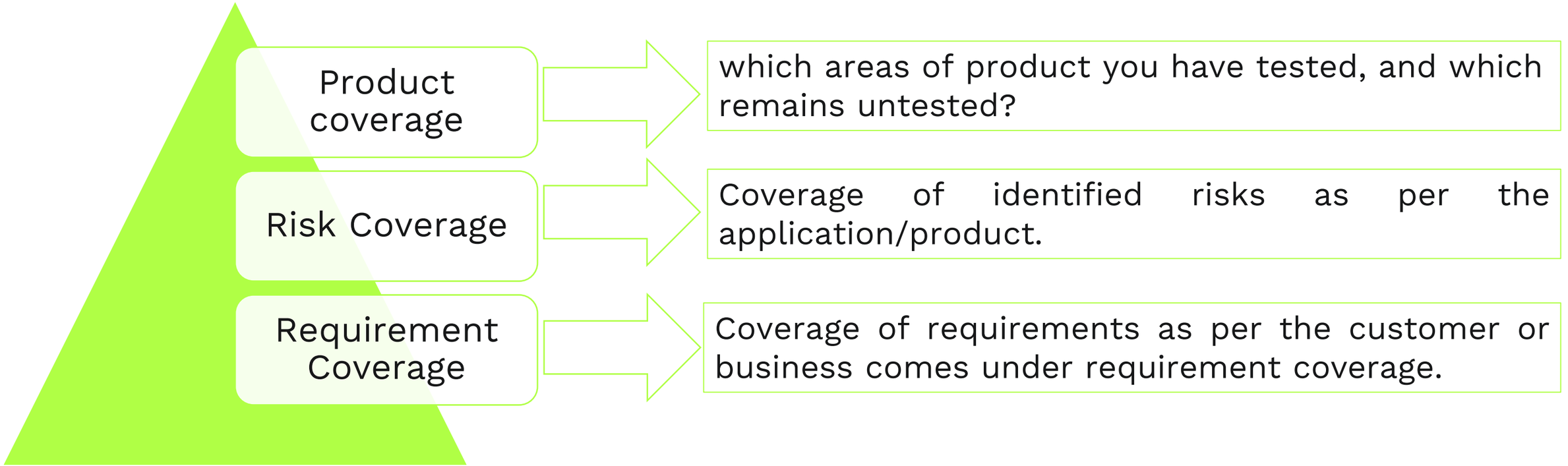
Goal of Software Developer

- Improve quality
- Reduce cost
- Preserve customer satisfaction

"Early Lifecycle Validation" in software development is must

What need to be validated and Tested

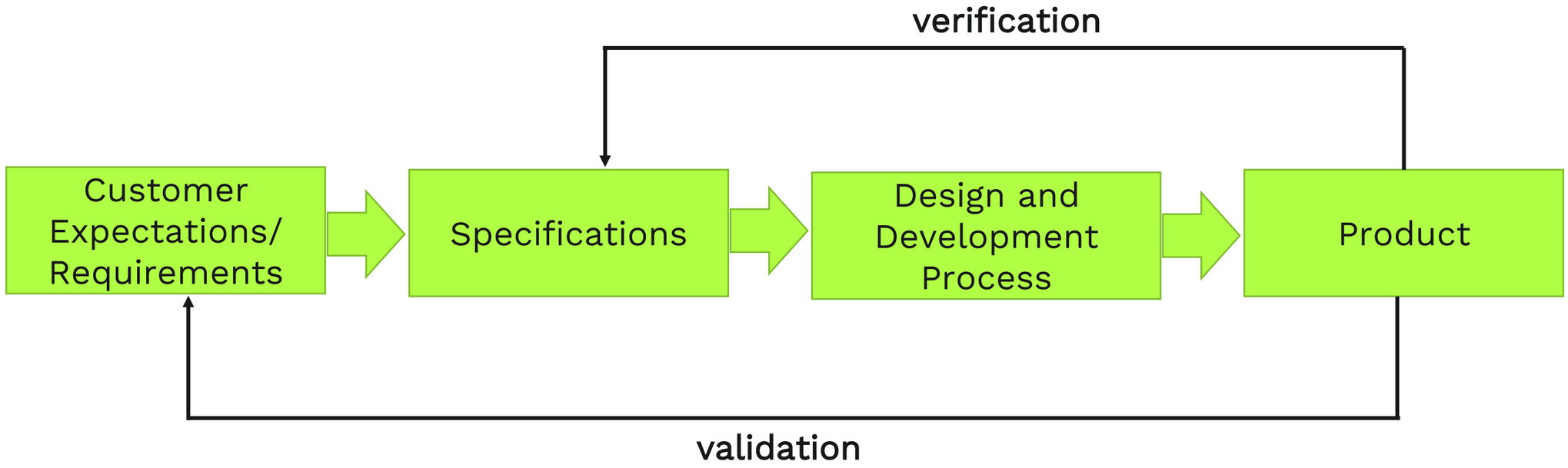
Test coverage can have different meaning in different context.



Verification and Validation

Validation ensures **“specifications captures the customer need”**

Verification ensures **“product meets specifications”**



Verification and Validation (as per IEEE)

Validation : The process of evaluating software at the end of software development to ensure compliance with intended usage

Verification : The process of determining whether the products of a given phase of the software development process fulfill the requirements and specifications established during the previous phase

Automotive Safety Integrity Levels

- Functional Safety (based upon ISO 26262) need to be ensured while designing Automotive Systems. The large number of goals are identify based upon risk assessment and hazard analysis.

Risk Assessment & hazard analysis parameters:

i. **Exposure,** i.e. how often the vehicle is in a situation in which the people involved, e.g. driver, passengers or other road users, may be put at risk.

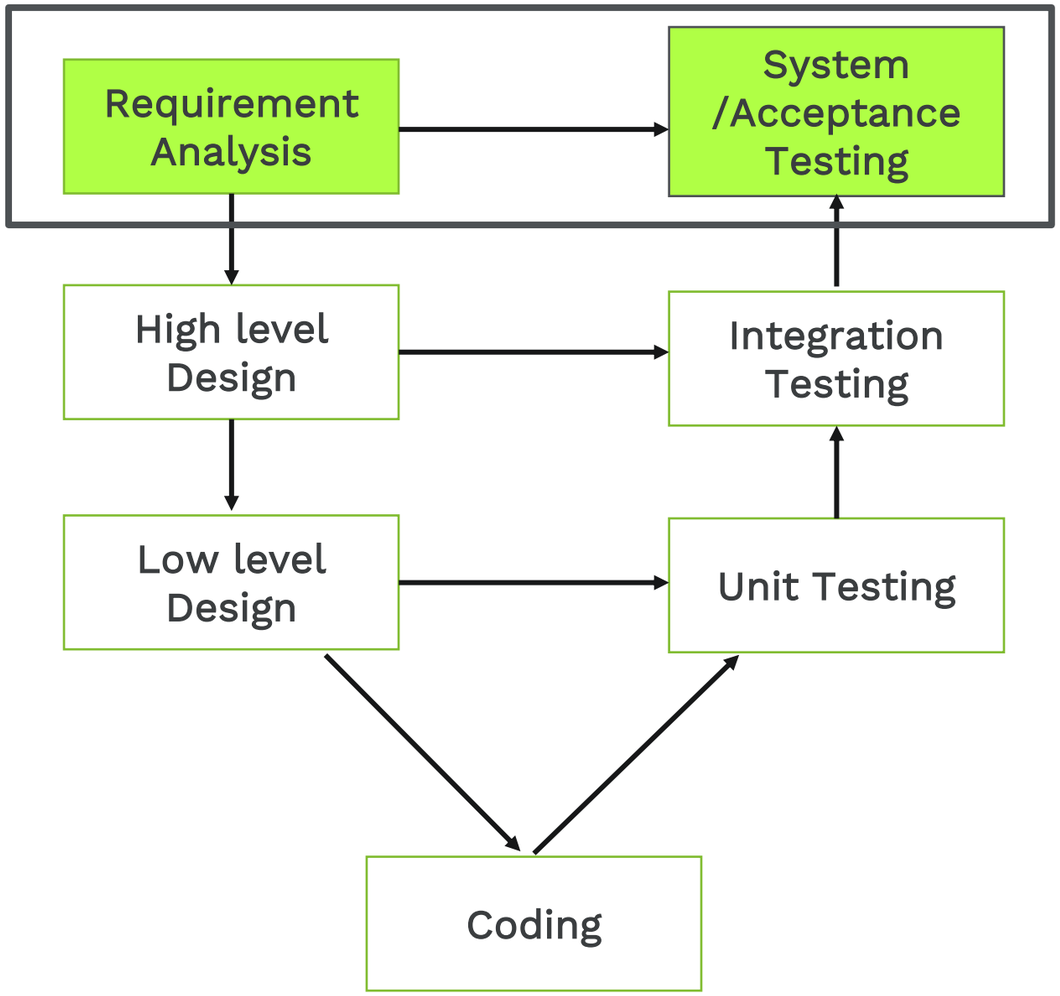
ii. **Controllability,** i.e. how well the individuals involved can handle an infringement of the safety goal.

iii. **Severity,** seriousness of the consequences from a breach of the safety goal.

Airbag System		***ASIL***
Hazard	Unintentional inflation of airbag	**D**
Safety Goal	Prevent airbag inflation unintentionally	

Each Safety goal is classified in term of ASIL (Automotive Safety Integrity Levels) standard ASIL A, ASIL B, ASIL C and ASIL D

Relevance of testing in verification/validation



- **Unit testing** is done to verify that the lowest independent entities in any software are working fine.
- In **integration testing** the individual tested units are grouped as one and the interface between them is tested. Integration testing identifies the problems that occur when the individual units are combined
- **Acceptance Testing** is testing done by a customer (rather than a product developer) to ensure that the product they are buying meets their requirements.
- For example, a car manufacturer would perform acceptance testing on a airbag system, before they deploy it.

Note:

- From a technical perspective **acceptance testing** is like **system testing,** the difference is mainly in who is testing rather than what they are testing.

Automotive perspective on Testing as per ISO-26262

ISO-26262 in automotive domain poses stringent requirements for development of safety critical applications and on the testing processes for automotive software.

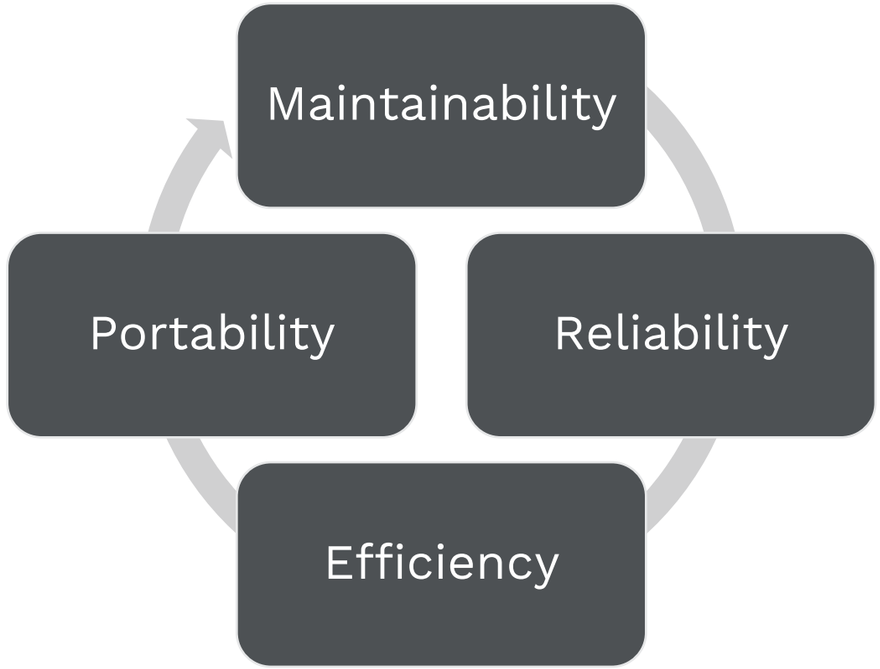
ISO 26262-6 recommend some set of software test coverage metrics for unit testing of automotive software.

	Metric	ASIL			
		A	B	C	D
1a	Statement coverage	++	++	+	+
1b	Branch coverage	+	++	++	++
1c	MC/DC	+	+	+	++

software test coverage metrics

Embedded developer’s perspective on code quality

- Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments
- Degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled or replaced in a specified environment
- Degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components
- Degree to which a code can be used in other programs
- Degree of Modifiability and Testability



MISRA C standard to ensure software code quality

- a set of "Guidelines for the Use of the C Language in Vehicle-Based Soft-ware," or "MISRA C,“ are developed with following objectives:

- Writing a safe program in C
- Remove base language issues and use subset as per ISO standard C
- Make C readable and portable to other platforms

- **Guidelines are set of advisory rules while writing C programs**