Next Generation Design and Verification Today

Requirements-driven Verification Methodology (for Standards Compliance)

Mike Bartley, TVS
Agenda

- **Motivation**
  - Why Requirements Driven Verification?

- **Introduction to Safety**
  - The Safety Standards
  - What do we need to do? And deliver?

- **Supporting Requirements Driven Verification with Advanced Verification Techniques**

- **Tool Support**

- **Advantages of Requirements Driven Verification**
An Overview of Verification Approaches

- Metric Driven Verification
- Coverage Driven Verification
- Constrained random verification
- Directed testing
- Feature Driven Verification
- Assertion-based verification
- Formal property based verification
Why Requirements Driven Verification?

- **Metric Driven Verification**
  - Allows us to define targets
  - And monitor progress
  - The metrics can become the end rather than the means to the end

- **Coverage Driven Verification**
  - Most common metric driven verification approach
  - Code Coverage
  - Functional coverage
    - Might be related to features
  - How often have you chased a coverage goal with limited ROI?

- **Feature Driven Verification**
  - Features *MIGHT* be related to spec
    - Is that relationship captured?
  - Are features related to requirements?
  - Shouldn’t everything we do be related to a requirement?
Shift-Left “Sequential” Development Flow

Product Req → Requirements Verif Spec → Acceptance Verif

System Spec → System Verif Spec

Integration Verif Spec → Integration Verif

Unit Spec → Unit Verif Spec → Unit Verif → Unit Build → Static Analysis

Should we consider iterative flows?
Safety Standards

- **IEC61508**: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems
- **DO254/DO178**: Hardware/Software considerations in airborne systems and equipment certification
- **EN50128**: Software for railway control and protection systems
- **IEC60880**: Software aspects for computer-based systems performing category A functions
- **IEC62304**: Medical device software -- Software life cycle processes
- **ISO26262**: Road vehicles – Functional safety
Introduction to Safety

- The life cycle processes are identified

- Objectives and outputs for each process are described
  - Objectives are mandatory
  - But vary by Integrity Level
  - For higher Integrity Levels, some Objectives require Independence
Key Elements

- Plans & Standards
- Requirements
- Design Specifications
- Reviews and Analyses
- Testing (against specifications)
  - At different levels of hierarchy
  - Test Coverage Criteria
  - Requirements Traceability
  - Independence
Key Deliverables

- Hardware Verification Plan
- Validation and Verification Standards
- **Hardware Traceability Data**
- Hardware Review and Analysis Procedures
- Hardware Review and Analysis Results
- Hardware Test Procedures
- Hardware Test Results
- Hardware Acceptance Test Criteria
- Problem Reports
- Hardware Configuration Management Records
- Hardware Process Assurance Records
Requirements Engineering Definitions

Requirement:
1. A condition or capability needed by a user to solve a problem or achieve an objective
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification or other formally imposed documents
3. A documented representation of a condition or capability as in (1) or (2)

[IEEE Std.610.12-1990]

Stakeholder*:
- A stakeholder of a system is a person or an organization that has an (direct or indirect) influence on the requirements of the system

Requirements Engineering:
- Requirements engineering is a systematic and disciplined approach to the specification and management of requirements with the following goals:
  1. Knowing the relevant requirements, achieving a consensus among the Stakeholders about these requirements, documenting them according to given standards, and managing them systematically
  2. Understanding and documenting the stakeholders’ desires and needs, then specifying and managing requirements to minimize the risk of delivering a system that does not meet the stakeholders’ desires and needs

* All Definitions taken from IREB
Requirements Engineering

Requirements

Intent to Implement

Proof of implementation
Variants, Reuse & Communication
Issues

Conflicts

Comprehension
Data Integrity
Functional Hazard

Function
- What function ensures requirement is achieved

Functional Failures
- No Function
  - **HAZARD**: Doesn't do what its designed to
- Incorrect Function
  - **HAZARD**: Incorrectly does an incorrect function

Situational Analysis
- Usage situation - when is it likely to happen
- People at risk – who can be hurt by a failure
## Hazard Level Analysis

### Lane Keeping assistant example

#### Identify hazards

<table>
<thead>
<tr>
<th>Hazard</th>
<th>Doesn’t stay in lane</th>
</tr>
</thead>
<tbody>
<tr>
<td>Situation</td>
<td>Unintended lane change</td>
</tr>
<tr>
<td>UID</td>
<td>123</td>
</tr>
<tr>
<td>Severity</td>
<td>S3</td>
</tr>
<tr>
<td>Rationale</td>
<td>Unintended change due to speed at which the system is active or required may be life threatening to multiple parties</td>
</tr>
<tr>
<td>Exposure</td>
<td>E4</td>
</tr>
<tr>
<td>Rationale</td>
<td>Possibility of occurrence over any frequency or duration of travel in car</td>
</tr>
<tr>
<td>Control</td>
<td>C3</td>
</tr>
<tr>
<td>Rationale</td>
<td>May be required override for danger situation - short time scale to consider appropriate other actions and system not reacting to request</td>
</tr>
<tr>
<td>ASIL</td>
<td>ASIL D</td>
</tr>
</tbody>
</table>
Safety Requirements

Safety goal

The Drivers and other road users shall not be exposed to unreasonable risk due to unintended lane change

Safe State

The Vehicle shall remain in the lane in which they intended

Functional goal

Avoid Undemanded Steering

Functional Safety Requirement

System shall detect excessive motor torque
Requirement Quality Gateway

• Requirements are expensive
  - ROI
  - Quality Criteria :
    - Unambiguous
    - Testable (verifiable)
    - Clear (concise, terse, simple, precise)
    - Correct
    - Understandable
    - Feasible (realistic, possible)
    - Independent
    - Atomic
    - Necessary
    - Implementation-free (abstract)

• How do we check for quality
  - Boilerplates
  - Manual inspection (review)
  - model rule checker (if model based)
Considerations

- Requirements stages
- Data management
- Where to store/communicate
- Change management
- Visualisation
- Process/Flow
- Communication
- How to prove
Requirements Driven Verification And Test

Integration testing
Parameteric testing

Manual review

Assertion testing

Directed testing

Structural coverage

Formal testing

Functional coverage

SOC Simulations
IP2 Simulations
IP1 Simulations

Where?
Pass?
Metadata?
Variant Management

Requirements Database

Variant x
xml

Variant y
xml

Variant z
xml

Variant a
xml

Import of feature level requirements

Partial import of just top-level requirements

Complete import include all mapping

Refine & map

Becomes

Copy of Variant x
asureS\n
Variant x
asureS\n
Variant y
asureS\n
Variant y
asureS\n
Supporting Advanced Verification

- Constrained random verification with automated checks based on models or scoreboards, etc.
- Coverage driven verification based on functional coverage models and code coverage metrics.
- Assertion-based verification.
- Formal property based verification.
Supporting Advanced Verification

- Feature Level Requirements (Top Level Test Plan)
  - Req1
    - Req1.1
    - Req1.2
    - Req1.2.1
    - Goal 1.2.2

- Refined Requirements (Sub-Features)
  - Req1.1
    - Req1.1.1
    - Req1.1.2

- Refined Requirements (Sub-Features and Goals)
  - Req1.1.1
    - Goal 1.1.1.1
  - Req1.1.2
    - Goal 1.1.1.2

- Measureable Goals
  - Goal 1.1.1.1
  - Goal 1.1.2.1
  - Goal 1.2.1.1

- Metrics (Coverage or Tests)
  - Coverage 1.1.1.1
  - Coverage 1.1.1.2.1
  - Coverage 1.1.2.1.1
  - Test 1.2.1.1.1
  - Test 1.2.2.1
Tracking

Metrics can be:
- From HW verification
- From Silicon validation
- From SW testing
Track Progress on Requirements Signoff
Supporting Hierarchical Verification

- A requirement might be signed off at multiple levels of hierarchy during the hardware development
  - Block
  - Subsystem
  - SoC
  - System
    - Including Software
  - Post Silicon
Tool Support Requirements

- Requirements -> test plan
- Data Integrity, hierarchy, data translation
- Change management – instant update
- Live database
- Tailored Documented proof
- Allows reviews of implementation document against test plan
- Mapping
- Test management
- Compliance / Audit Management
asureSIGN Dataflow
asureSIGN™ Solution Built on UCIS

Requirements Engineering Flow

- Identify
- Map
- Translate
- Analyse
- Compare

asureVIEW™

 UCIS
 UCDB
 XML
 LOG

34
Thank you!