Requirements-driven Verification Methodology (for Standards Compliance)

Mike Bartley, Serrie-justine Chapman (TVS)
DVCon 2015
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
- Directed Testing
- Coverage Driven Verification
- Constrained random verification
- 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

- **Coverage Driven Verification**
  - Most common metric driven verification approach
  - Code Coverage
  - Functional coverage
    - Might be related to features

- **Feature Driven Verification**
  - Features **MIGHT** be related to spec
    - Is that relationship captured?
  - Are features related to requirements?

The metrics can become the end rather than the means to the end.

How often have do you chase a coverage goal with limited ROI?

**Shouldn’t everything we do be related to a requirement?**
Sequential Development Flow

Product Reqs → Requirements Verif Spec → Requirements Verif

System Spec

System Spec → System Verif Spec → System Verif

Integration Spec

Integration Spec → Integration Verif Spec → Integration Verif

Unit Spec

Unit Spec → Unit Verif Spec → Unit Verif

Unit Build

Unit Build → Static Analysis
Shift-Left “Sequential” Development Flow

Product Reqs → Requirements Verif Spec → Acceptance Verif

System Spec → System Reqs

System Spec → Integration Verif Spec → System Verif

Unit Spec → Unit Verif Spec

Unit Build → Unit Verif

Integration Verif Spec → Integration Verif

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

Requirements Database

Product Requirement Document

Internal Target Specification

Safety Concept

Test Plan

logs
checkers
coverage
test

accellera
SYSTEMS INITIATIVE
Requirement
Atomic Sub requirement
Atomic sub requirement

Top level test plan
Grouped sub plan
Single sub plan

Atomic test plan
Atomic test plan
Atomic test plan

Top level Safety
Atomic Safety
Grouped Safety

Atomic Safety feature
Atomic Safety feature
Atomic Safety feature
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>
</tbody>
</table>

| ASIL         | ASIL D                       |
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)

Shift left
CONSIDERATIONS

- Requirements stages
- Data management
- Where to store/communicate
- Change management
- Visualisation
- Process/Flow
- Communication
- How to prove
REQUIREMENTS DRIVEN VERIFICATION AND TEST

- Test scripts
- Integration testing
- Parameteric testing
- Manual review
- Structural coverage
- Directed testing
- Assertions
- Functional coverage
- Formal testing
- SOC Simulations
- IP2 Simulations
- IP1 Simulations
- Where?
- Pass?
- Metadata?

Software tools:
- Do
- Word
- PDF
- ReqPro
- Jama
- Cradle
- Test
Variant Management

Requirements Database

Variant x
xml
Import of feature level requirements
Refine & map
Variant x
asureS✓/gn

Variant y
xml
Partial import of just top-level requirements

Variant z
xml

Variant a
xml

Copy of Variant x
asureS✓/gn
Complete import include all mapping

Becomes
Variant y
asureS✓/gn
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
  - Refined Requirements (Sub-Features)
  - Refined Requirements (Sub-Features and Goals)
    - Req1.1.1
    - Req1.1.2
    - Req1.2.1
    - Goal 1.2.2
  - Measureable Goals
    - Goal 1.1.1.1
    - Goal 1.1.1.2
    - 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™ SOLUTION built on UCIS

Requirements Engineering Flow

asureVIEW™

Identify → Map → Translate → Analyse → Compare

UCIS
UCDB
XML
LOG
Using the Accellera UCIS Standard

Using UCIS to Combine Verification Data from Multiple Tools
Motivation for UCIS

- Verification is hard
  - <insert standard slide: 70+%, increasing complexity, yadda, yadda, yadda>

- Variety of verification techniques and methods
  - Directed and constrained-random simulation
  - Formal verification
  - Testbench methodologies

- Design and verification engineers need coverage metrics:
  - What has been checked, what remains to be checked?
  - How many engineers do we need?
  - How much time do we need?
  - Where best to direct verification resources?
  - What is the best tool or method to efficiently cover problem areas?
  - What coverage overlaps?
  - What coverage is missing?
Unified Cases and Data Flow

- **Generate**
  - Single verification run, single/multiple coverage types
  - Multiple verification runs

- **Access**
  - Using UCIS Application Programming Interface (API)
  - Using Interchange Format (XML Interchange Format)

- **Analyze**
  - Report unhit coverage points
  - Track progress of coverage over time

- **Merge**
  - Across runs, components, tools
Open the Coverage DB.

```c
ucisT db = ucis_Open(string_pointer_to_db_name);
```
Traverse the Coverage DB, using Callback mechanism.
Traversed of Coverage DB, scans the Data based on reasons.

**UCIS reasons for Callback**
- **INITDB**: Start of DB, apply initial settings
- **DU**: Design unit scope
- **TEST**: Test data history object
- **SCOPE**: Scope object
- **CVBIN**: Cover item
- **ENDScope**: Scope end including design units
- **ENDDB**: End of DB

**Defined Reasons in Call back area**
- asureSign
- UCIS Library
- UCIS file (.ucd + .ucm)

**MySQL DB**
Cover Items captured on the basis of Design and Scope

UCIS CALL BACK

UCIS file (.ucd + .ucm)

MYSQL DB

asureSign

UCIS CALL BACK

UCIS Library

TOGGLE  BRANCH  EXPR
COND  INSTANCE  PROCESS
COVERGROUP  COVERINSTANCE  COVERPOINT
CROSS  COVER
INTERFACE  FSM
DU_ARCH  DU_PACKAGE
DU_INTERFACE
PROGRAM  PACKAGE  TASK
BLOCK  FUNCTION  FORKJOIN
GENERATE  GENERIC  CLASS
DU_INTERFACE  FSM_STATES
COVBLOCK  CVGBINSCOPE
Cover Items captured based on Classification Criteria.

Classification Criteria:
- Design Unit
- HDL Scope
- Cover Scope
  - Functional
  - Structural
  - Assertion
Cover Items captured based on Classification Criteria.

Based on Kind tool captures:
- Individual Cover Items
- Aggregated Cover Items

Classification Criteria:
- Design Unit
  - assuresign only uses instance and module coverage
- HDL Scope
- Cover Scope
  - Functional
  - Structural
  - Assertion
Based on Kind tool captures:
- Individual Cover Items
- Aggregated Cover Items

Information
- Coverage Kind
- Coverage Name
- Simulation Path
- File name
- Design Type
- Line Number
- Hits

Information Captured for each Cover Items using Library

UCIS file (.ucd + .ucm)

UCIS Library

Get String Property
Get Handle Property
Get Scope Type
Get Internal Int Property
Get Scope Source Info
Get Int Property
Get Cover Data
Source Iterate
Source Scan
Free Iterator
Captured information is passed to DB in form of Records.
asureSign uses Captured Data from all sources, and relates it to Requirements via Features and Goals.
Advantages of Requirements Driven Verif

- Requirements Management
- Verification Management
- Project Management
- Impact Analysis
- Product Line Engineering
- Variant management
- Improved Product Sign-Off
Conclusions #1

Requirements Driven Verification

- Compliance to various hardware (and software) 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

- And ….
  - Identify test holes and test orphans
  - Track the status of the whole verification effort (planning, writing, execution)
  - Build historical perspective for more accurate predictions
  - Better reporting of requirements status
  - Risk-based testing
  - Prioritisation and Risk Analysis
  - Filtering Requirements based on Customers and releases
  - Impact and conflict analysis
Conclusions #2

- Advanced verification techniques can be deployed in Requirements Driven Verification
  - Requirements engineering tools to capture the verification plan & mapping
  - Verification management tools to automate collection of results

- More info
  - CRYSTAL http://www.crystal-artemis.eu/
Any questions?

I'll need to know your requirements before I start to design the software.

First of all, what are you trying to accomplish?

I'm trying to make you design my software.

I mean what are you trying to accomplish with the software?

I won't know what I can accomplish until you tell me what the software can do.

Try to get this concept through your thick skull: the software can do whatever I design it to do!

Can you design it to tell you my requirements?
Discussion and feedback