Closing the loop from requirements management to verification execution for automotive applications

Walter Tibboel, Jan Vink
NXP Semiconductors
Outline

• Motivation
• Verification flow
  – Verification hierarchy and traceability
  – Change is process; not an event
  – Flow & tooling
  – Future work
• Practical example
  – Automotive design example
  – Test environment
  – Progress tracking
• Conclusions
• Acknowledgement
Motivation

• Verification & Validation challenges
  – Full insight about verification coverage
  – Traceability between requirements, test items and test results
  – Change management: maintaining the overall consistency
  – Bridging the gap between requirements management system and verification environment

• This work focuses on
  – Methodology to make specification integral part of V&V flow
  – Tooling to ensure overall consistency:
    updates become explicit and well-controlled
Test items terminology

**Functional**
Pass/fail check on functional behavior

<table>
<thead>
<tr>
<th>Test item Description</th>
<th>Test item Setup</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check wake-up</td>
<td>From sleep mode event X is fired</td>
<td>DUT is active mode</td>
</tr>
</tbody>
</table>

**Parametric**
Check performance parameter under multiple PVT conditions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Conditions</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_{IL}$</td>
<td>Input Leakage low VDD3V0</td>
<td>VDD = 3.0:3.5; T=25°C</td>
<td>4</td>
<td>-</td>
<td>5</td>
<td>uA</td>
</tr>
</tbody>
</table>
Verification hierarchy

Requirements

What to make?

- Requirement
  - Test item
  - Test item
  - Test item

Test-items

What to test?

- Test item
  - Test item
  - Test item
  - Test item

Test-cases

How to test?

- Test case
- Test case
- Test case
- Test case

Traceability

- Architect
- V&V Lead
- Testers
Change is a process; not an event

Project gates

Requirements
Specify product

Test items
Specify tests, based on requirements

Test cases
Implement tests, based on test items

Product specified
change requests & refinements

Amount of updates

Data updates controlled by Check & update

Check & update

V&V Lead

Architect

Product specified

Test team Ramp up

Check & update

Testers

Specify tests, based on requirements

Specify product

Implement tests, based on test items

Check & update

© Accellera Systems Initiative
Functional verification flow

- **What to make**
  - Requirements management

- **What to test**
  - Traditional Gap: Verification management
  - Verification execution

- **Pass/fail Results**
  - Coverage

- **Test items**

- **Test cases**

- **DOORS**
  - Requirements
  - Test items
  - Scores

- **Multi project Tracking**

- **Import to DOORS**
  - Log files

- **Collect results**
  - Test files

- **Check & update to work area**

- **Check & update to test files**

- **Simulate**
  - Functional Verification

- **Parametric Verification, Validation, other projects**

- **Pass/fail Results**
  - Coverage

- **Test items**

- **Test cases**

- **“Work area Excel”**

- **How to test**

- **Check & Update**
Check & update tooling

<table>
<thead>
<tr>
<th>Object Identifier</th>
<th>RS Object Identifier</th>
<th>RS Object Identifier</th>
<th>Req Text [RQ]</th>
<th>Object Type</th>
<th>Object Text</th>
<th>Work-area</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS-23</td>
<td>FFS-108</td>
<td></td>
<td></td>
<td>Test item</td>
<td>It shall be possible to disable the WAKE functionality by setting WAKE-&gt;WAKE-0. The WAKE functionality is always enabled in NORMAL mode (nominal) independent from the WAKE/WAKE settings.</td>
<td>Test Item</td>
</tr>
<tr>
<td>TS-16</td>
<td>FFS-108</td>
<td></td>
<td></td>
<td>Test item</td>
<td>It shall be possible to disable the WAKE functionality by setting WAKE-&gt;WAKE-0. The WAKE functionality is always enabled in NORMAL mode (nominal) independent from the WAKE/WAKE settings.</td>
<td>Test Item</td>
</tr>
<tr>
<td>TS-11</td>
<td>FFS-108</td>
<td></td>
<td></td>
<td>Test item</td>
<td>It shall be possible to disable the WAKE functionality by setting WAKE-&gt;WAKE-0. The WAKE functionality is always enabled in NORMAL mode (nominal) independent from the WAKE/WAKE settings.</td>
<td>Test Item</td>
</tr>
<tr>
<td>TS-37</td>
<td>FFS-56</td>
<td></td>
<td></td>
<td></td>
<td>If the CAN state is OFFLINE, OFFLINE_MISC or LISTEN_ONLY, the RXD pin shall be HIGH except when a wake-up event is detected it shall be LOW.</td>
<td>Test item</td>
</tr>
</tbody>
</table>

**WARNING: INCONSISTENT Doors Test item ID: ts-37**

Attribute: Doors Test item Object Text -> Work 1

- In states STANDBY, SLEEP, NORMAL and OVERTEMP.
- In states STANDBY, SLEEP and NORMAL.
Future work: client/server

Requirements management

Server

DOORS
Requirements

API

Verification execution

Server

Cadence vManager C/S

vAPI

Launch

Test results

Regressions

HTTP - REST

Functional Verification

Check & Update client

Check & Update client

Standardization on the interfaces by means of API

API: Application Programming Interface

REST: Representational State Transfer
Practical example: the product
Practical example: test environment

Mix of directed tests at a specific point in time and automatic checkers always running in the back ground
Practical example: the V&V link

Pass/fail is automatically annotated to test-items using:

A. ‘Test-item messages’ in the log files from directed test-cases. One test-case can generate multiple messages.

B. Result from functional cover-points and assertions which are part of automatic checkers.

NOTE: test-case pass/fail information is NOT used directly.
Practical example: progress tracking

Progress is tracked using the status of all test-items.
Conclusions

• Methodology and tooling are presented that make specification an integral part of V&V flow.

• Achieved practical V&V flow bridging the gap between the DOORS requirements and the verification execution.

• During project execution, the check and update mechanism highly improved the overall consistency.
  – Triggering clearly on the exact changes.
  – Easy and well controlled updating.

• Future work will focus on standard interfaces (API) in client/server architectures.
Acknowledgment

• This work is partially sponsored by the European Commission in the CATRENE CA703 OpenES project.
• Special thanks for the automotive project team for smoothly picking up this new way of working in their tightly scheduled deliveries.
  – Steef Grimbergen
  – Leon Hesen
  – Marcel Oosterhuis
  – Veeresh Gaddepanavar
  – Ute Essen - Willemsen
  – Dawn Johnston
Questions?