2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

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

Walter Tibboel, NXP Semiconductors, The Netherlands (walter.tibboel@nxp.com)

Jan Vink, NXP Semiconductors, The Netherlands (jan.vink@nxp.com)

*Abstract*— **This paper introduces a structured Verification and Validation (V&V) flow, which has a closed loop from an existing requirements management system to verification execution. Test specifications in the requirements management system need a proper synchronization mechanism to end up consistently in use cases, assertions and functional coverage located in the verification work area. We explain how this approach has been successfully applied in an automotive verification case. This paper continues on the methodology and tooling being introduced in a DVCon paper of last year [1], which described how traceability between requirements, test specifications and test results is ensured.**

***Keywords— Verification & Validation; requirements-driven verification; traceability; change management; coverage; test specification; Automotive verification***

## I. Introduction

In Verification & Validation one of the main challenges is to gain maximum confidence in reaching the product design and qualification in the automotive industry. Traceability between requirements and the actual verification execution are key. For this purpose, the essential element in the V&V flow is the definition of test items. In this paper we will explain more details how these test items are applied in functional verification, which is one of the V&V domains.

To guarantee that all functionality of the product is verified and/or validated, every requirement should have at least one test item. Such test item specifies criteria for one self-contained pass/fail check. To cover a full requirement multiple pass/fail checks might be needed, see Figure 1.

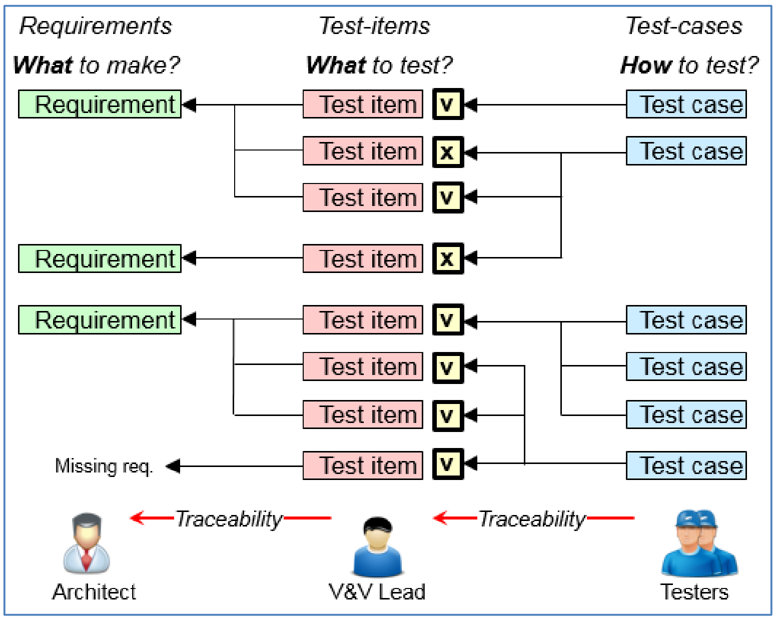


Figure 1 Verification hierarchy from requirements to test cases

* This work is partially sponsored by the European Commission in the CATRENE CA703 OpenES project.

The test item describes only what to test: Specific responses under certain conditions. How it is tested is described in the test case: For functional verification this is typically one simulation, defining all (sequences of) stimuli inputs for the DUT, needed for the test items being covered in that test case. The stimuli can be directed or guided random. The test items are often implemented in checkers containing the assertions which actually generate the pass/fail results. Part of the test item can be cover point specification, that should take care all specified conditions (states) are being tested. Let's call the traceability relations visible in Figure 1 the *Verification hierarchy*: the arrows indicate the hierarchy in the verification data.

### *A. Change is a process, not an event*

Figure 2 shows a different aspect of the V&V data, here the arrows indicate *data updates* between specification documents, each having their specific owner. Because the test items are based on the requirements, the test item specification document will have a copy of some requirement attributes, in the figure indicated by the green part in the top of the Test item arrow in the figure. These requirement attributes in the test items specification will become inconsistent when the architect changes a requirement in the requirement specification. Using the check & update mechanism as described in [1], the V&V lead is triggered explicitly on the inconsistencies. After updating the test items, he can accept the corresponding requirement updates in his test item spec. The first priority of the V&V lead is maintain *all* data attributes in the test item specification consistent. Second priority is synchronize all requirement updates from the architect in a controlled way: one by one, taking care of the mentioned first priority. The key is *"divide and conquer"*: Just only updating the inconsistencies of one item that you can absorb completely before updating the next one.

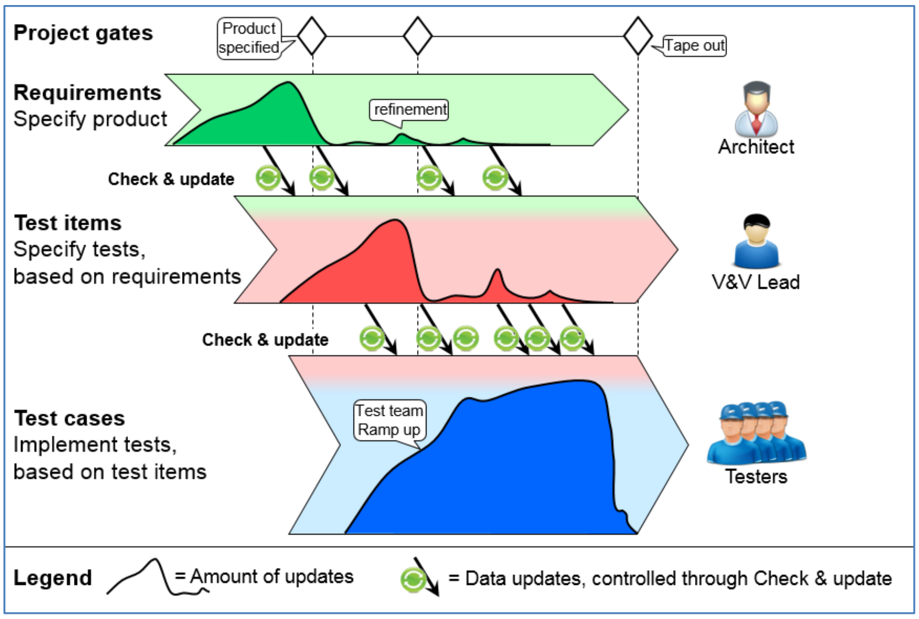


Figure 2 Dynamics in verification hierarchy

Similar, test item updates ripple through to the test implementation. A tester will only accept a test item changes when he directly can adapt the test implementation. The amount of inconsistencies is a proper indication of how much test implementation work still to be done.

Figure 2 shows a lot of dynamics (updates) that will happen during the project cycle. A frozen spec is often a relative term, refinements / change request will pop-up rather late in the project life cycle. E.g. new insights, customer feedback, cannot be ignored. Because of time pressure, the different owners do not execute in pure sequential order. An activity in the figure (one of the big arrows), will start before the activity, on which it depends, is finished completely. A maturity status (e.g. draft, proposal, approved) must be indicated per spec item. Also the total amount of test items can vary, because new insights during test implementation. In short, the project gates give important structure in the dependencies of the activities. Still we cannot avoid that the dependent activities are executed in parallel, for good reasons. So we need a proper mechanism to tackle the updates rippling through the activities.

## II. Related Work

In [1], the proposed check and update mechanism can support the traceability between specifications in Excel. In Automotive example of this paper DOORS [2] is used for this purpose. DOORS is a completely different environment compared to the Linux environment of the testers. The aim of the check & update mechanism is not to replace one of the environments but to "glue" them together. This kind of "glue" in certain areas like requirement management is called Application Lifecycle Management (ALM) [3].

OpsHub is one of the providers of ALM integration and migration solutions for application development organizations. OpsHub creates a unified ALM ecosystem by seamlessly combining individual ALM systems [4]. For data management in a functional verification flow, this kind of software is too generic and too complicated. But their applied technologies is useful in a solution which is more tailored to the V&V domains we are supporting.

Important topic in the ALM area is to aim for standard interfaces, implemented in API's (Application Programming Interface). The relevant standardization community in this area is the OSLC (Open Services for Lifecycle Collaboration) [5]. An open community building practical specifications for integrating software. Their aim is "Working for easier integrations: We want software that easily integrates with other software, which will help you build your ideal development and operations environment, connect disjointed workflows, minimize frustration, and save time and money."

Cadence vManager [6] is a regression environment for functional verification, with primary focus on digital simulations. The vManager tool supports launching regressions, creation of test plan (vPlan) and collecting the corresponding results. The vPlan has similarities with the functional verification part of test item specification, we propose in this paper. The results can be a huge amount of data. The vManager architecture is being changed into Client Server architecture (vManager C/S). One of the aims is to improve (external) accessibility of specification and result data, e.g. via the new vManager API. This major architecture change takes a few years. The check and update mechanism of this paper will become one of the clients accessing the server.

Standardization of V&V terminology and methodologies for requirements and test item management is lacking. ISTQB (International Software Testing Qualifications Board) [7] offers a good foundation, however these definitions cover only a limited part of the V&V domain. TMMi (Test Maturity Model integration) [8] is mainly focused on the specification of processes. The proposed methodology in this paper can serve as a starting point for further standardization.

## III. Verification flow

In this chapter two main topics are discussed. As mentioned before, applying traceability in verification flow, means in practice of NXP that we have to bridge a gap. Next to this we will discuss in more detail how test items and cases are being applied in the automotive design case: A Mixed Signal In Vehicle Network product. More details of the design will discussed in the next chapter.

### *A. Bridging the gap in the Verification flow*

In the NXP automotive industry, the requirements and test items have their source in the DOORS requirements management system [2]. Before, projects were often hampered by the gap between requirements management (the "paper world") and the actual Verification execution (the "simulation world"), since their common data remains dynamic during the project. The new applied approach is bridging the gap by introducing a so called "*Work area Excel*" file, located in the verification environment. The obtained three levels are connected with the check & update mechanism, as depicted in Figure 3.

The check & update mechanism as described in [1], clearly and explicitly triggers the inconsistent attributes values and offers means to resolve these in a structural and reliable manner. At specific moments in time, the test engineers will synchronize the data by reviewing the inconsistent attributes. Especially later in the verification process the test item updates will be only done one-by-one, such that related attribute updates and test implementations can directly be adapted too.

Besides the requirements and verification specific data, the DOORS database also contains data of the other V&V domains, like parametric verification and validation, as depicted in the figure. All the corresponding test results are also imported. Furthermore, the DOORS database contains data of multiple automotive projects. The project planning and tracking data is exported to a separate SQL server, which has html clients to give (project) management a consistent dashboard over all V&V domains, over all enclosed projects. The *Work area Excel* is only one view, relevant for a specific group of testers. The Check & update mechanism is used to update the *Work area Excel* using the .csv files being exported/imported daily from DOORS, as an intermediate format (not shown in the figure).

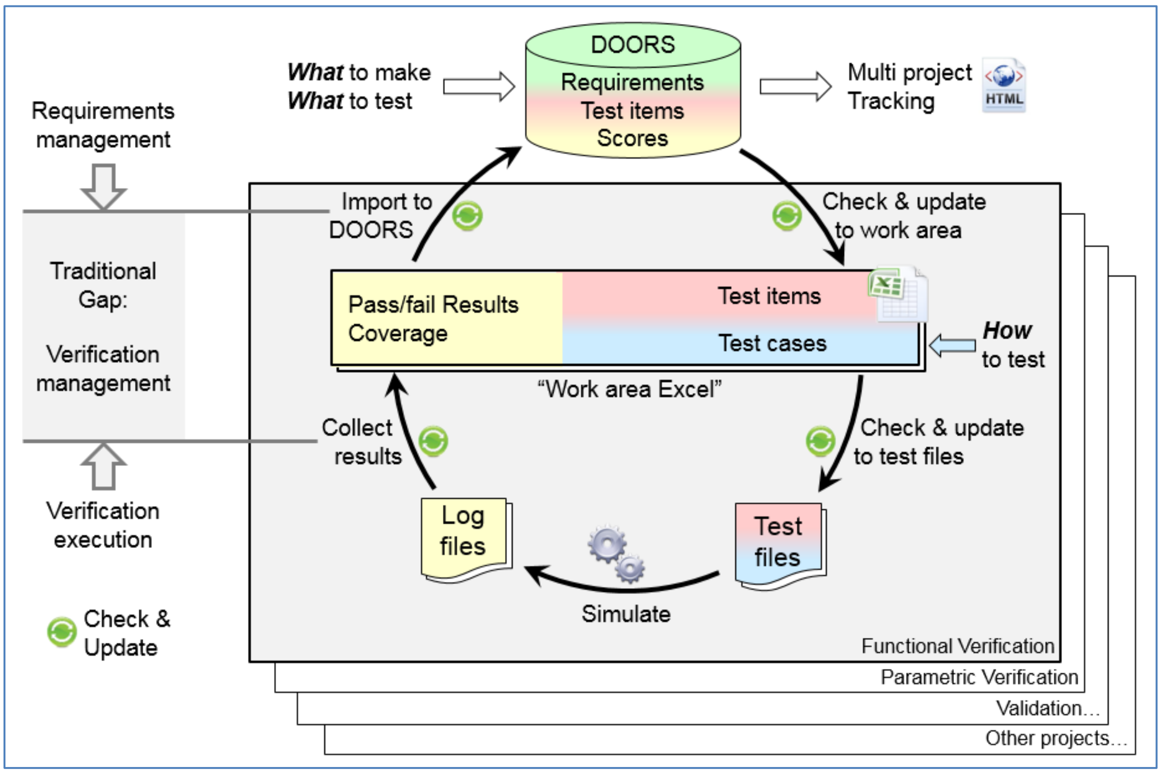


Figure 3 Functional verification data flow

The *Work area Excel* contains verification data (test items and some requirement attributes) of this specific project only. If needed, this content can be separated in multiple Excel files. The check and update mechanism often include user specific selection filters, to restrict the data processing only to data belonging to the owner.

## B. *Test items and cases in functional verification*

During implementation of the test-environment and test-cases, the test-items in the work-area excel must be linked to the implementation. This is done by adding comment sections to the implementation code which is - by means of the check and update mechanism - synchronized with the test-items from the work-area excel. In case a test-item is updated by the V&V lead, the tester will get a trigger if the comment section in the code does not match with the description in current version of the work-are excel. The tester will update the implementation code and when this is in line again with the test-item description an update of the comment section will be done.

A second aspect of linking the test-item in the work-area excel to the implementation code is generation of pass/fail information. This must be done in such a way that it is possible to automatically get pass/fail information for every test-item after running a regression. This automation is achieved by inserting message generation in the test-cases which will provide a pass/fail for the test-items covered. The pass/fail information will be filtered from the log-files. The status for every test-item will be analyzed in this way. In case all results show a pass, a pass will be annotated to the test-item in the work-area excel. In case there is a single fail, a fail will be annotated.

Next to collecting the pass/fail information from messages in the log-files, a pass/fail can also be collected from functional cover point or assertion results. Assertions are typically used for pin-level checks. Functional cover points are particularly interesting in case the test item must pass under multiple conditions. Multiple bins can be defined to show the check is performed under all required conditions. This is comparable to PVT (Process Voltage Temperature) corners for parametric simulation.

Using a combination of directed tests, assertions and cover points, which are explicitly linked to the test items, is generally called Metric Driven Verification (MDV). One of the MDV promoters in the EDA arena is Cadence vManager, as discussed in the related work section. Parts of the result parsing, filtering and summarizing can be done at different places. The challenge of the check & update mechanism described in this paper is to agree on and make use of standardized interfaces to synchronize data, and leaving data processing to the dedicated tooling of the specific V&V domain (functional verification in this case).

## IV. Industrial example - In Vehicle Network product

Our application is a high-speed Mixed Signal - In Vehicle Network - product that provides an interface between a Controller Area Network (CAN) protocol controller and the physical two-wire CAN bus.

### *A. Introduction of the products and verification challenges*

The transceiver is designed for high-speed CAN applications in the automotive industry, providing differential transmit and receive capability to a CAN protocol controller. An SPI interface is used to configure and control the CAN transceiver with a micro controller. The transceiver provides different states related to power-management and safety features. Challenges for top-level functional verification is to guarantee correct cooperation between the analog components (such as under voltage detectors, temperature sensor, CAN wake-up, etc.) and the digital, under all conditions and register configurations. Requirements-driven verification provides traceability of what is expected from the product and what is really implemented/verified. It is an essential tool to achieve sufficient confidence the product can operate reliably in customer applications and meets compliancy with the increasing number of standards that control development of hardware for domains such as automotive.

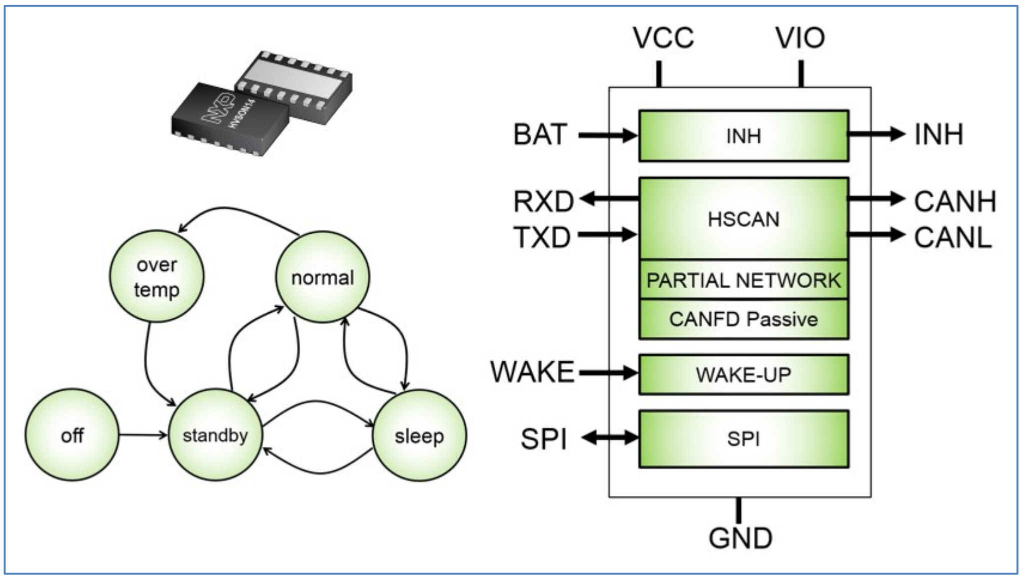


Figure 4 CAN transceiver system controller state diagram

### *B. The verification environment*

To be able to do thorough top-level verification, a mixed signal test environment is used. It contains Verilog AMS and system Verilog components. For the digital in the DUT either RTL or netlist+sdf is used. For the analog in the DUT either Verilog AMS models or a mixture of Verilog models and transistor level schematics are used. In the test-bench the Verilog AMS components interact with the DUT interfaces and do an A-to-D or D-to-A translation. The actual checkers and stimuli generators are written in system-Verilog.

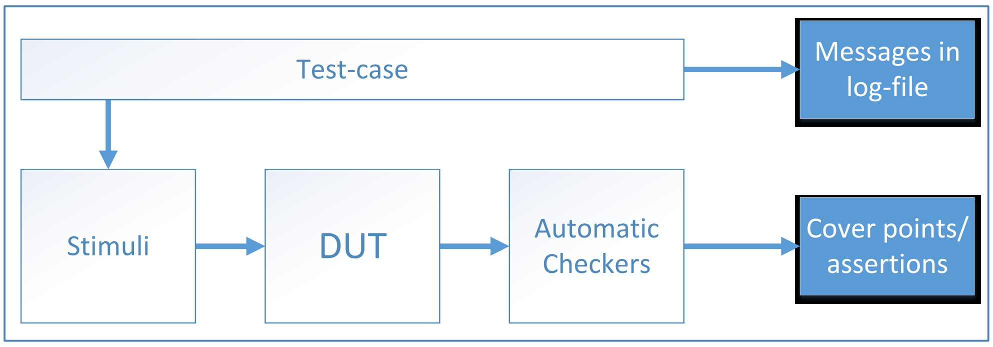


Figure 5 Test bench setup

The target is to automatically obtain pass/fail information for the test-items from a regression run. Like already mentioned in section III.B, two types of information are used for that, here it is shown how this is implemented (see Figure 5):

A. Messages in the log-files providing a pass or fail for a test-item. Generating these messages is implemented in the test-case. The expected values and checks are controlled at test-case level.

B. Cover-points and assertions providing pass/fail information from automatic checkers running in the background. These are independent from the test-case and checking all the time for any test-case executed.

The results from a regression are by means of the VnV link linked to the test-items, see Figure 6 for how this VnV link relates to the earlier provided Verification Hierarchy (Figure 1) and data flow (Figure 3).

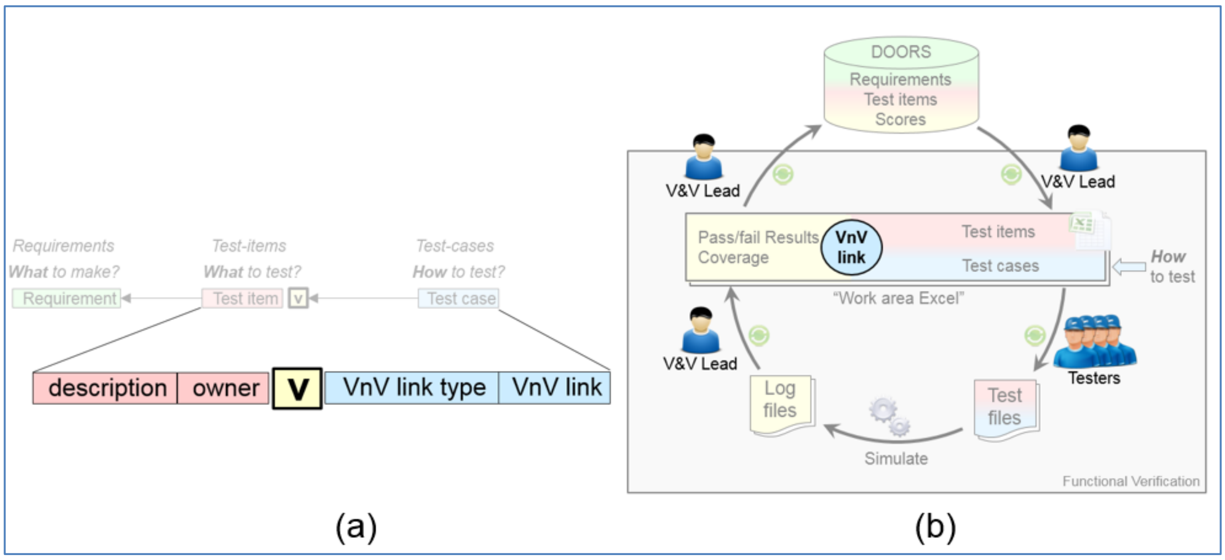


Figure 6 Attributes of the test-item and usage of VnV link

## *C. The way of working within the V&V team*

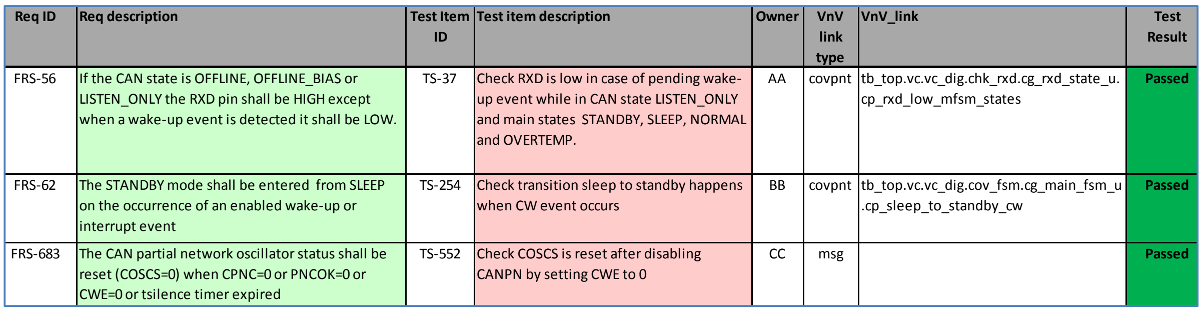
The way of working during preparation and execution of the project is as follows:

1. During the preparation phase of the project, the V&V lead creates the test-environment and defines the test-plan. The test-plan contains the test-items which define what needs to be tested. The test-items link to requirements (see Figure 1 and Figure 6). The target is to cover all functional requirements. This is formally checked by a requirements coverage report generated from DOORS. The test-items and linked requirements are present in the work-area excel (see Figure 3 and Figure 6).

   During the execution phase of the project, requirements are still changing or added. This is typically related to details being worked out, only sometimes to changes in the product behavior. Still it's very important to be able to propagate these changes from requirements to test-items to implementation of test environment. The V&V lead takes care that the synchronization with work-area excel is done and divides the work between the testers (see Figure 2).

2. During the execution phase of the project, the team will be ramped up and extended with many testers. The testers will define the optimal mapping of test-items to test-cases to optimize implementation effort and simulation run-time. The testers will update the work-area excel with the VnV link and implement related test-cases, message generation, cover points and assertions (see Figure 7). The testers will also take care that the implementation matches test-item description. Automatic synchronization between comments in implementation code and work-area excel is not done yet in this application example, this is seen as a next step. This was not a big issue, most of the time the changes are additions of test-items, not changes in existing test-items.

3. The V&V lead will regularly run regressions using all implemented tests. A script is used to automatically generate pass/fail results for all test-items. This is imported into DOORS and regularly test-item progress reports are generated.



| Req ID | Req description | Test Item ID | Test item description | Owner | VnV link type | VnV_link | Test Result |
|---|---|---|---|---|---|---|---|
| FRS-56 | If the CAN state is OFFLINE, OFFLINE_BIAS or LISTEN_ONLY the RXD pin shall be HIGH except when a wake-up event is detected it shall be LOW. | TS-37 | Check RXD is low in case of pending wake-up event while in CAN state LISTEN_ONLY and main states STANDBY, SLEEP, NORMAL and OVERTEMP. | AA | covpnt | tb_top.vc.vc_dig.chk_rxd.cg_rxd_state_u.cp_rxd_low_mfsm_states | Passed |
| FRS-62 | The STANDBY mode shall be entered from SLEEP on the occurrence of an enabled wake-up or interrupt event | TS-254 | Check transition sleep to standby happens when CW event occurs | BB | covpnt | tb_top.vc.vc_dig.cov_fsm.cg_main_fsm_u.cp_sleep_to_standby_cw | Passed |
| FRS-683 | The CAN partial network oscillator status shall be reset (COSCS=0) when CPNC=0 or PNCOK=0 or CWE=0 or tsilence timer expired | TS-552 | Check COSCS is reset after disabling CANPN by setting CWE to 0 | CC | msg | | Passed |

Figure 7 Snapshot of the Work area Excel

Below a progress graph is shown. Next to good traceability of requirements coverage, this approach also enables to track progress. The dynamics in test-items is also clearly visible, the total amount of test-items significantly grew during the execution of the project.

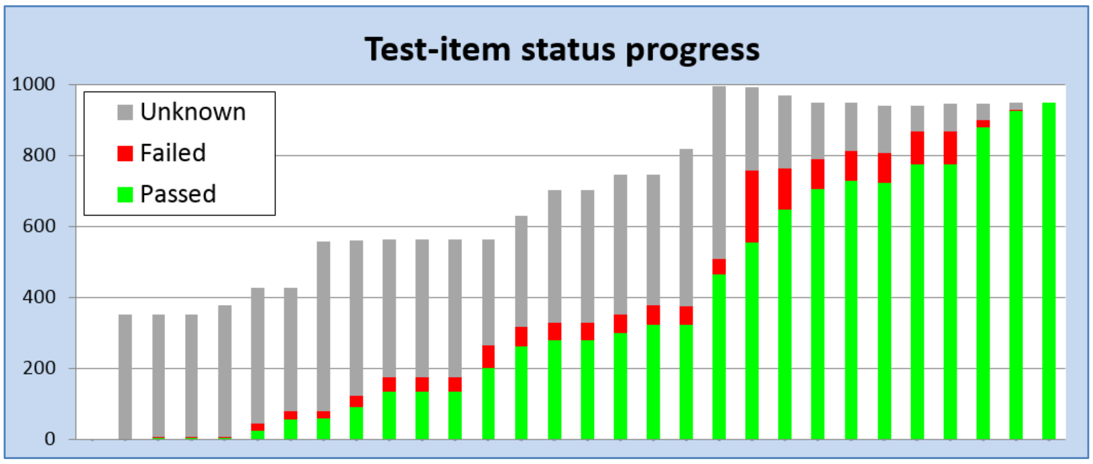


Figure 8 Test item progress monitoring

## V. Conclusions

In this paper we presented a practical V&V flow in which we bridged the gap between the requirement specifications in DOORs and the actual verification execution. The introduction of the Work area excel offers a detailed overview of all test items and their status, which increased the quality of the verification.

By means of structural and reliable check & update mechanism, the requirements specification in DOORS, the Work area in Excel containing test cases and test results, as well as the test case implementations are continuously kept synchronized and thus consistent.

The V&V flow has been applied on an automotive project, an NXP Mixed Signal In Vehicle Network product. It demonstrated that during project execution, where requirements and verification execution updates happened continuously, the process to manage all these changes was reliable and practical. Especially the triggering and updating of the inconsistencies is very much appreciated; it highly increased efficiency through the tool automation.

Future work will study the use of Application Lifecycle Management techniques and standardized (tool) interfaces to integrate our V&V methodology, flow and tooling with other commercially available solutions.

## VI. ACKNOWLEDGMENT

Special thanks for the automotive project team for smoothly picking up this new way of working in their tightly scheduled deliveries. Often biggest challenge is not the technical improvement, but the change in the way of working in the actual design and verification teams. Next to that we like to thank the Project Support Office team for their help in the requirement management area.

## REFERENCES

[1] Walter Tibboel, Martin Barnasconi, "Advancing traceability and consistency in Verification and Validation", DVCon Europe 2014

[2] IBM® Rational® DOORS® is a requirements management application for requirements communication, collaboration and verification http://www-03.ibm.com/software/products/en/ratidoor

[3] Application Lifecycle Management (ALM), http://en.wikipedia.org/wiki/Application_lifecycle_management.

[4] OpsHub, one of the providers of ALM integration and migration solutions for application development organizations. http://opshub.com/main/index.php/products/oim.

[5] OSLC (Open Services for Lifecycle Collaboration) see http://open-services.net.

[6] Cadence® Incisive® vManager™ is the solution of Cadence for verification planning and management http://www.cadence.com/products/fv/vmanager/pages/default.aspx

[7] ISTQB (International Software Testing Qualifications Board), http://www.istqb.org/downloads/glossary.html

[8] TMMi (Test Maturity Model integration), http://www.tmmi.org/pdf/TMMi.Framework.pdf