Verification and Validation of Embedded Software Systems at DAF Trucks

Raymond Tinsel
Model Based Design Specialist, DAF Trucks NV

Co Melissant
CTO, MonkeyProof Solutions BV

ABSTRACT

At the MathWorks Automotive Conference 2010, DAF Trucks and MonkeyProof Solutions presented the challenge DAF Trucks was facing with the increasing importance and complexity of software in vehicle development, the chosen strategy (migration to Model-Based Design), the transition and migration needs, a tool suite, and some early findings on the chosen approach and environment.

In this session, we will present our progress, tangible results, and findings two years down the road. Perspectives that will be covered are design quality and project management in vehicle development; the design environment and tool suite for design data management, traceability, and testing; and ISO 26262 reflections on the current workflow. We share results yielded by the adoption of Model-Based Design and discuss how project management is supported by a well-defined workflow and supporting infrastructure and design environment. We also share an evaluation of the workflow and design environment in light of ISO 26262.

INTRODUCTION

DAF Trucks N.V. is a wholly owned subsidiary of the North-American corporation PACCAR Inc. The company was founded in 1928 and has its core activities focused on the development, production, marketing and sale of medium and heavy-duty commercial vehicles according to a ‘Build to Order’ principle. This means that all vehicles are built to satisfy each customer’s individual wishes, resulting in tens of thousands of different vehicle configurations. DAF’s engines not only find their way to DAF’s own trucks, but also to trucks produced by the American vehicle divisions under PACCAR.

As presented at the MathWorks Automotive Conference two years ago, DAF Trucks faced the ever existing challenge of reducing cost and time-to-market while increasing quality for the exponentially growing scale and complexity of its embedded systems. To take on this challenge DAF Trucks decided to adopt a lean but full scale Simulink centric Model-Based Design approach to introduce the principles of V-cycle front-loading, requirement traceability, executable specifications and elimination of waste, with the objective to remove incompleteness, inconsistencies, miscommunication and other inefficiencies from its development processes. While in 2010 a good start had been made with the most essential processes and tools, two years down the line it was time to look back and make up the score.

To get an independent evaluation of the Model-Based Design achievements DAF trucks involved the Mathworks and other external experts in an ISO26262 compliance analysis focusing on chapter 6 of the standard “Product development: software level”. With some constructive feedback in the areas of integration testing, documentation and supplier involvement, the overall conclusion was very positive: “A mature Model-Based Design process and tool chain is in place for application level software. This result has been achieved with a minimum of DAF Trucks specific customization”.

While working on the standard the main topic became very clear: Traceability. With this topic of traceability in mind we will now present our achievements in the next chapter.

TRACEABILITY

When introducing a new way of working it all starts with the goals one wants to achieve. The next step is to define a process with deliverables to reach that goal. And the final trick is to introduce an intuitive tool suite that supports people in following that workflow/processes. Critical for the DAF Trucks process and toolsuite was to maximize design freedom while ensuring efficiency, quality and traceability from customer requirement to code. A challenge for which DAF Trucks teamed up with MonkeyProof Solutions.
In this chapter we will discuss the traceability relevant aspects following the steps in the high-level workflow as shown in Figure 1. For each process step we will show how one or more tools from the toolsuite as shown in Figure 2 support the engineer with providing the required traceability.

A short introduction to the process steps in the high-level workflow:

- **REQUIREMENT**: The process starts by importing the customer requirements in the Requirement Manager. Requirements are then distributed over hardware and software and further refined to the black-box requirements of a software application module.
- **REQUIREMENT REVIEW**: The black-box module requirements are reviewed to ensure there is a common understanding about what the module must do before the actual design work starts.
- **DESIGN & TEST**: Based on the requirements a module and its testcases are designed in parallel.
- **DESIGN REVIEW**: Both the module and testcases are reviewed to ensure they are correct and in line with the customer requirements.
- **RELEASE**: When all application modules are finished a release is composed and sent to the supplier.
- **SUPPLIER & ACCEPTANCE**: The supplier verifies the DAF Trucks release, codes and compiles the application and provides DAF Trucks with the results of their verification activities.

But it all starts with requirements …

**REQUIREMENT**

The Requirements Manager provides traceability from the highest level requirement down to sub Simulink, Stateflow and MATLAB model implementation and vice-versa. Simulink/Stateflow objects are automatically annotated to provide visual information which requirement is implemented, where in MATLAB code comments are added using code cell comment syntax. So in short, full bi-directional traceability including visual information is in place. Where the first versions of the requirements manager were plain MATLAB GUI's, now the strength of MATLAB is combined with the strength of Java. This has impact not only on usability, but also on ease of traceability and navigation, as now the high-level architecture of all functions and components is directly available from a tree view (see Figure 3). Being able to trace the requirements to the design and vice versa not only ensures completeness, but also enables impact analysis and fast adoption of changes to either requirements or design. Figure 3 shows the dependencies of an individual requirement to facilitate impact analysis of a change.
Safety requirements

An ASIL attribute can be added to a requirement (see Figure 4) to allow for decomposing safety requirements into redundant safety requirements. This enables ASIL tailoring at the next level of detail as described in ISO/FDIS 26262-9 – chapter 5.

Integrity of links

Although the desired requirements workflow is top-down, in practice the requirements process is iterative or even bottom-up in case of for example incomplete customer requirements. The tool supports both approaches and provides visualization and checks of the relations between high-level and low-level requirements. An iterative process generally also introduces broken or invalid links between requirement and design, due to moved, copied or changed designs, obsoleted requirements etc. These changes can for example result in low-level decisions that no longer originate from high-level requirements. As traceability needs to be trustworthy, automated checks are available with options to correct the invalid or broken links (using Model Advisor technology, see QUALITY CONTROL below).

DESIGN & TEST

Once the requirements have been agreed upon the engineers can start designing the module and creating the testcases. From a traceability point of view three topics are of interest: Simulink model, data dictionaries and test cases.

Simulink model

All models are based on one centralized Simulink library referred to as the MBDBlockset. The MBDBlockset is based on a subset of standard Simulink and Stateflow blocks and has been agreed upon by the supplier for code generation purposes. Due to the Mathworks library mechanism each block is traceable to the module and changes to the MBDBlockset can be managed from one central location.

Requirements are linked to the module using the Mathworks V&amp;V toolbox capabilities, with the bi-directional traceability as mentioned before.
Data dictionaries
To trace and control the use of units a mutually agreed list has been created with the supplier. This list has also been implemented in a custom package inheriting the Mathworks Simulink.Signal and Simulink.Parameter objects. A screenshot of this integrated list of units is shown on the right.

As each unit has a unique identifier and a unit must be chosen from this list (of course checked with a Model Advisor check), also here the traceability and change impact analysis is easily automated.

In addition to the traceability aspects such a fixed list of units is of course also a great basis for data validation of the object’s user defined properties.

Test cases
Module testcases are linked to the same requirements as used for the module design and similar checks are performed to guarantee consistency. Each testcase gets a unique ID which makes the testcase traceable throughout the development process from DAF to supplier and back. For added traceability of specific issues, also issues can be linked to one or more testcases. A screenshot of the UNITT tool in Figure 6 shows both types of links.

In addition to the software modules also each MBD Blockset block is accompanied by testcases for regression testing and verification by the supplier.

QUALITY CONTROL
The goal of traceability is lasting quality so Quality Control (QC) is a key task during the complete workflow (see Figure 1 High-level overview workflow, review elements). Although reviews are very important, many of the review tasks are time-consuming and boring (so error-prone). To improve quality as well as to reduce time, many of the review tasks are automated using MathWorks’ Model Advisor technology, extended with an Object Oriented model to accomplish full traceability between guidelines, checks, tasks, tests, documentation and implementation. The relations between guidelines, tasks (steps in the workflow) and automated checks are maintained within the checks repository. Report generation provides a full traceability chart from this (see Figure 7).
The QC process contains several artifacts (guidelines from various sources, checks, and result reports) that in general evolve over time. Furthermore, a QC session is very much related to the version & status of the object under QC (requirements, design test, data, etc). To solve these traceability issues, several steps are made.

First of all, all artifacts are stored under version control using TortoiseSVN and Subversion (SVN). SVN is integrated with the Project Selector (see Figure 8) – a tool to select a project, the phase, and component. When performing a QC-review on the design, a baseline/tag is created (see Figure 9, note that all reviews are stored, including those not used for a release).

Secondly and important from process perspective is that each review (or release) baseline is only made if a predefined (sub-)set of all guidelines is completely verified. As Model Advisor allows for grouping checks in tasks, several tasks are available throughout the development process: requirements review, entry check, requirements traceability charts, etc.
review full check, design review entry check, etc. The entry tasks for a review are created to improve efficiency of the formal review and full tasks ensure the quality. Once a model has passed review (both manual tasks as well as the full review task in Model Advisor) a review tag can be tagged as release.

Last, but not least, access to the artifacts is controlled using Role-Based Access Control (RBAC). Several roles are defined, like "System Architect", "Design Engineer" and "Test Engineer" to control the read/write access to file and database entries. With this, apart from shear control, also full traceability to all changes and whom made these changes is in place.

REVIEW

Each DAF Trucks release is identifiable by a release tag, which in case of the example in Figure 9 would be DEMOS_phase_01_R01. By using this tag in the SVN repository and for snapshots of data that is not under Version Control, like the Requirements Manager database, the concept of Configuration Management has been realized and DAF Trucks knows exactly what goes out the door.

Once all modules have been tagged for release, making the release itself is a matter of a click of the mouse. The toolsuite will compose the deliverable according to agreements with the supplier, after which the package is automatically verified in a clean Matlab environment to ensure that the supplier gets a 100% working package.

SUPPLIER

When the supplier has verified the consistency of the DAF Trucks release a fully automated toolsuite using the Mathworks Embedded Coder converts the models into code. The DAF Trucks delivered test cases combined with the testcases of the supplier are executed on both model and C-code and the results of these tests are sent back to DAF Trucks. With this feedback loop we have got requirement and testcase traceability up to the C-code.

CONCLUSION

Looking back on the last two years we believe it is fair to conclude that our Model-Based Design adventure has been a great success:

- Traceability of all relevant information is achieved from customer requirement to application level C-code,
- Overall lead time has been reduced by approximately 30% and cost reductions have been achieved to as much as 50%,
- On the supplier side the labor intensive, manual and error prone process has been replaced by a fully automated one,
- And since the introduction of the Model-Based Design approach DAF Trucks has gone from a technology push to a market pull situation where both engineers and managers want to see more.

The challenges DAF trucks and MonkeyProof Solutions will be working on next are the requirements imposed by the upcoming ISO26262 standard, the introduction of next generation reports to reduce the amount of paperwork, and to branch-out and align the Model-Based Design up the left side of the V-cycle.

More good news is sure to follow soon!

REFERENCES

1. “Verification and Validation of Embedded software Systems at DAF Trucks”, Co Melissant / Raymond Tinsel, Mathworks Automotive Conference 2010
2. ISO 26262:2011