Presented at: 9th International Symposium on Advanced Vehicle Control (AVEC2008), Kobe, Japan, 6. - 9.10.2008

Simulation-Based Automated Verification of Safety-Critical Chassis-Control Systems

Magnus Gäfvert[†] **Johan Hultén**[†] **Johan Andreasson**[†]
Andreas Junghanns[‡] **Jakob Mauss**[‡] **Mugur Tatar**[‡]

[†]Modelon AB
Ideon Science Park, SE-223 70 Lund, SWEDEN
Phone: +46 46 2862200
Fax: +46 46 2862201
E-mail: {magnus.gafvert, johan.andreasson, johan.hulten}@modelon.se

[‡]QTronic GmbH
Alt-Moabit 91d, D-10559 Berlin, Germany
Phone: +49 30 3512 1066
Fax: +49 30 3036 4941
Email: {andreas.junghanns,jakob.mauss, mugur.tatar}@qtronic.de

This paper presents and exemplifies a novel methodology to perform simulation-based automated testing for verification of complex chassis-control systems. The methods are derived from computer-game principles and regard the system under test as an opponent that is defeated when the specification is violated. The methodology is demonstrated on an auto-coding implementation of a brake-blending function for a heavy vehicle, in combination with a simulation model implemented in Modelica. It is shown that a vast number of scenarios can be analyzed with moderate manual effort, and that results corresponding to high-coverage FMEA / FTA analysis can be produced to verify system safety and robust performance.

Vehicle Control, Vehicle Dynamics, Modeling and Simulation Technology

1. INTRODUCTION

New chassis control functions add to the safety and comfort of road vehicles and provide the customers with great value. However, the complexity of the heterogeneous and distributed total control system that is the result of many control functions that operates in parallel and share sensors and actuators is continuously increasing. Methods are needed to cope with this complexity to guarantee safe and reliable performance at a reasonable cost.

Traditional control-design methods focus on robust performance and stability in terms of model uncertainty and disturbances in closed-loop systems. For the complex aggregate of vehicle control functions, the traditional methods can only be applicable on subsets of the total system, and must be combined with other methods in order to analyze global robust and safe performance with respect to the system design and possible faults. The complete system is of such complexity that formal proof-based verification of the design is practically impossible. Instead, the system must be subjected to verification testing with a sufficient coverage to yield confidence in safety and availability. Likewise, the analysis of system performance under possible fault scenarios must be analyzed by testing rather than proof.

The verification testing involves the analysis of a vast number of scenarios selected in a systematic way. This work is commonly done manually to a large extent in design revisions, hardware-in-the-loop (HIL) rigs, and prototype testing. The manual involvement is time-consuming and limits the number of cases that can be investigated and tested, and the verification process is also very costly. There are great improvements in efficiency and coverage to be gained from further automating these procedures. The human intervention is also a source of error that may be reduced by automation. Another issue is that verification testing usually is applied at later phases in the design process, and problems that are identified in these tests may require large and expensive steps back. Verification testing based on models or software may be applied at earlier phases and problems may thus be identified earlier.

The combination of a tool that automatically produces relevant test cases by intelligent search in the configuration space and model-in-the-loop (MIL) or software-in-the-loop (SIL) simulation is demonstrated in this paper. In Section 2, the tool and method for automated testing is described. Section 3 describes a brake-blending function implemented for auto-coding that is subject for verification testing. Section 4 describes the vehicle and environment simulation model. In Section 5, the tool used here to setup and perform MIL/SIL simulations is described. Section 6 presents and summarizes the results, and conclusions are given in Section 7.

2. THE AUTOMATED TEST ENVIRONMENT

Complex vehicle systems, like the one described in this paper, are difficult to test and validate. Traditional methods based on hand-written test scripts do not scale anymore with increasing system complexity. TestWeaver is a tool that implements a novel method for the automated generation and evaluation of tests for complex dynamic systems.

The overall design objectives for TestWeaver were to:

1. dramatically increase the test coverage, with respect to system behavior, while
2. keeping the workload for the test engineer low.

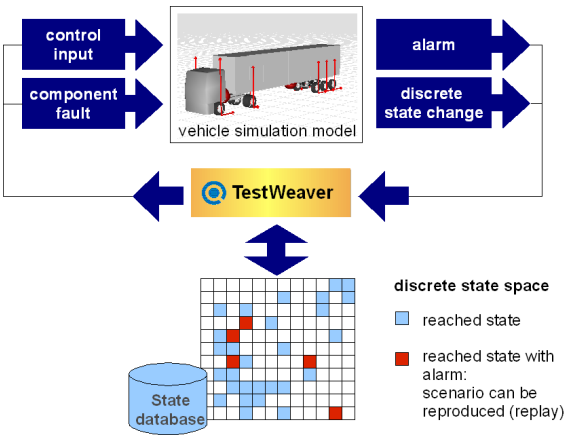

Figure 1. The test environment.

TestWeaver can autonomously generate and evaluate thousands of qualitatively differing simulation runs, see Figure 1. No handwritten test scripts are needed for that. The interaction with the simulated system under test is done via a typically small set of instrumentation components that are placed in the simulation model. These instruments allow TestWeaver to remotely control the system's input signals and parameters and to monitor selected state variables. Instrumentation libraries exist for Simulink, Modelica, C and for Silver co-simulations. The instrumentation is system-dependent, but the search algorithms implemented by TestWeaver are generic. Part of the instrumentation can monitor system safety and quality conditions and can report alarms to TestWeaver when certain safety requirements or specification are violated, or are nearly so.

TestWeaver systematically generates thousands of differing simulation scenarios. For this purpose TestWeaver analyses the results of the past simulations in order to intelligently (a) search for violations of specifications and to (b) maximize the test coverage. Test coverage is defined as follows: the domain of each variable controlled or monitored by a TestWeaver instrument is partionend into a small set of intervals by that instrument. This way, the instruments of a model define an n-dimensional discrete (i.e. finite) state space. The coverage goal of TestWeaver is to reach every reachable discrete state in that space at least once, see Figure 2.

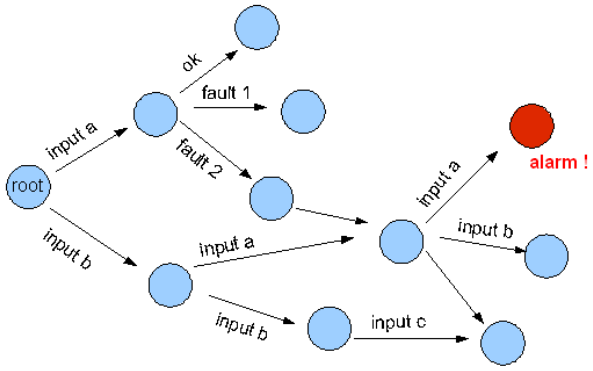

Figure 2 Scenarios generated by TestWeaver.

All recorded simulation runs can be "replayed" for detailed analysis and debugging.

To provide intuitive feedback to the user about critical scenarios found and what portion of the search space was covered, TestWeaver contains a sophisticated, customizable reporter component. This allows the user to focus on a few potentially interesting scenarios, instead of shifting through thousands or even tens of thousands of mostly uninteresting simulation runs.

3. VEHICLE FUNCTION – BRAKE BLENDING

The large mass of heavy-vehicle combinations such as tractors with semi-trailers implies that large braking power is required for retardation. This leads to a risk for very high temperatures in the service brakes as well as unnecessary disk and pad wear. Normal engine braking is unable to provide sufficient retardation torque to support the service brakes. Therefore, heavy vehicles are often equipped with auxiliary brakes such as retarders and special engine brakes (*e.g.* exhaust brakes).

These retarders can be controlled manually or automatically with brake blending. Brake blending is a feature where the service brakes and auxiliary brakes are seamlessly blended in such a way that the service brakes are relieved while the actual retardation corresponds to the driver brake pedal requests. The function shall not be noticeable for the driver, *i.e.* when the driver presses the deceleration pedal, the deceleration response of the vehicle shall be identical whether brake blending is active or not.

The brake blending system consists of a reference generator for the brake forces, a coordinator for all the brake systems, a retarder torque controller and a brake torque controller. See further the control scheme in Figure 3. The function is implemented in Simulink/Targetlink for automatic generation of production code.

The brake blending function is in an early development phase and this is the first safety and robustness assessment. Not all needed countermeasures are implemented yet. The function is, however, auto-coded and integrated in an ECU. It is tested in prototypes as well as in simulations.

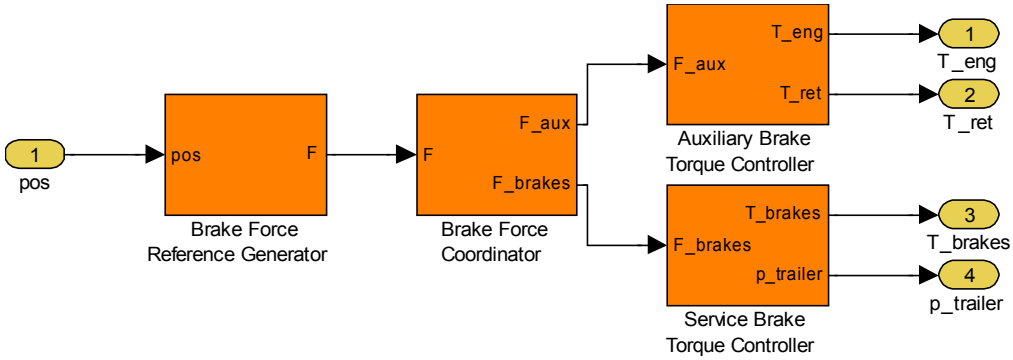

Figure 3. Schematic control diagram for Brake Blending. Only the main control flow is shown.

The brake pedal position, pos, is interpreted as a brake force, F, on the whole combination by the Brake Force Reference Generator. The Brake Force Coordinator distributes (the actual blending) the force between the different types of actuators, F_aux to the auxiliary brakes and F_brakes to the service brakes. The Auxiliary Brake Torque Controller controls the engine torque, T_eng, and the retarder torque, T_ret while the Service Brake Torque Controller controls the service brake torques, T_brakes and the trailer pressure, p_trailer.

The engine retarder is only possible co control in a number of fixed steps and the achieved torque has great engine speed dependency. The dynamics vary heavily with engine speed, and the resulting braking torque on the wheels also depends on the engaged transmission gear. The driveline retarder can be controlled continuously, and its dynamics vary with the prop shaft speed. To have a smooth control of the retardation, the brake blending function must include compensations for these aspects.

However, as the brake blending function prioritizes auxiliary brakes and the auxiliary brakes are braking only the drive axle, there is a risk for excessive slip on that axle, and thus reduced vehicle stability as a consequence. As a means for avoiding that, a slip control for the rear axle is needed. If the rear axle has too high slip, brake force is moved to the front axle. Therefore, brake blending is a relatively complex function.

Now, to make sure that the brake blending function is safe and robust, two studies are made in TestWeaver:

- A fault injection and
- A parameter variation robustness study

The analysis is made on a system configured for a vehicle with a compression and exhaust retarder as a part of the engine and a hydrodynamic driveline retarder mounted between the gear box and the final gear. In this study the analysis is limited to a single tractor, without trailer

Hazards

Hazard identification and classification is an important and difficult task. As a lot of proprietary knowledge about a function is documented in this classification, only a small portion of the hazard list can be published here. To give an example of what is possible with the tool chain presented here, we just give one example of one type of hazards, namely retardation deviations.

Function Hazard Identification and Classification – Brake Blending

ID	Description	Expression	Controllability	Severity	Risk	SIL
BB_H_004	The retardation is slightly lower than intended for a long time.	delta_r < -0.5 m/s^2 time > 1 s				
BB_H_005	The retardation is lower than intended for a long time.	delta_r < -1.0 m/s^2 time > 1 s				
BB_H_006	The retardation is much lower than intended for a long time.	delta_r < -2.0 m/s^2 time > 1 s				
BB_H_007	The retardation is slightly higher than intended for a long time.	delta_r > 0.5 m/s^2 time > 1 s				
BB_H_008	The retardation is higher than intended for a long time.	delta_r > 1.0 m/s^2 time > 1 s				
BB_H_009	The retardation is much higher than intended for a long time.	delta_r > 2.0 m/s^2 time > 1 s				

Signals and Parameters

Inputs and parameter values are partitioned into intervals that are classified as nominal, tolerance deviations, or faults. The faults injected in this study are described as offset or gain variations on signals in the function interface. See examples in the tables below.

Interface Signals

Name	Unit	Scaling
Vehicle to Control Function		
Engine Retarder Actual Torque	Nm	[0, 0.5, 1, 1.5, 2]
Driveline Retarder Actual Torque	Nm	[0, 0.5, 1, 1.5, 2]
Vehicle Wheel Angular Velocities (x4)	rad/s	[0, 0.5, 1, 1.5, 2]
Engine Speed	rad/s	[0.5, 1, 1.5]
Rear Axle Wheight (air spring pressure sensors)	kg	[0, 0.5, 1, 1.5, 2]
Vehicle Longitudinal Acceleration	m/s2	[0, 0.5, 1, 1.5, 2]
Control Function to Vehicle		
Brake Clamp Forces (x4)	N	[0, 0.5, 1, 1.5, 2]
Engine Retarder Requested Torque	Nm	[0, 0.5, 1, 1.5, 2]
Driveline Retarder Requested Torque	Nm	[0, 0.5, 1, 1.5, 2]
Engine Retarder Torque Mode (valid request)	-	[0 1]
Driveline Retarder Torque Mode (valid request)	-	[0 1]

Internal Vehicle Signals

Name	Unit	Scaling
Engine Retarder Actuated Torque	Nm	[0, 0.5, 1, 1.5, 2]
Driveline Retarder Actuated Torque	Nm	[0, 0.5, 1, 1.5, 2]

Parameters

Name	Unit	Scaling
Front Tire Longitudinal and Cornering Stiffness	N	[0.5, 1, 1.5, 2]
Rear Tire Longitudinal and Cornering Stiffness	N	[0.5, 1, 1.5, 2]
Wheel Base	m	[0.5, 1, 1.5]
Front Axle Load	kg	[0.5, 1, 1.5]

Test Scenario

The test consists of a driver that brakes and steers the vehicle. Gear shifting is made automatically. The Driver can give inputs according to the table below.

Drivers

Name	Unit	Values
Brake Pedal Position (0 to 1)	-	[0, 0.15, 0.3, 0.45]
Steering Wheel Angle	rad	[0, 0.5]

4. VEHICLE MODEL

To test and verify the brake-blending function it is necessary to simulate its operating environment including the tractor towing-vehicle, a driver, and the road. This environment must produce realistic responses to the control actions computed by the function. For this purpose a model is built using the Modelica-based tools Dymola and Vehicle Dynamics Library , and . Modelica is an open-specification high-level object-oriented and component-based language designed for multi-domain modeling for simulation of complex systems. Modelica is designed for acausal and equation based modeling, which makes the models highly readable and reusable. An outstanding feature of Modelica-based solutions in the context of verification testing is the exposed and modifiable model source code in combination with the high-level model descriptions. It is possible to view and modify or extend the implementation of component and system models. In this way instruments can be added to the vehicle model, such as sensor instruments for criteria evaluation, and instruments for input excitation and parameter variations to describe operating conditions, tolerance variations, and fault injection. Figure 5 shows the tractor model and the main powertrain, brakes, and chassis subsystems, and the instruments used by TestWeaver. The powertrain model includes a gearbox, engine retarder, driveline retarder, and driveline with final gear and differential. The brake subsystem includes components for service disk-brakes with ABS. The chassis contains front and rear axles, with steering, cabin, frame, and wheels. The vehicle runs on an infinite plane with configurable inclination and surface friction. The model contains about seven thousand equations and 63 continuous states. It is translated into C-code using the Dymola inline trapezoid integration feature for generation of efficient real-time simulation code. It is then compiled and linked into a DLL to be used with Silver and TestWeaver. The final model simulates in about realtime with 5ms time steps on a standard pc. Figure 4 shows the tractor in a heavy braking maneuver.

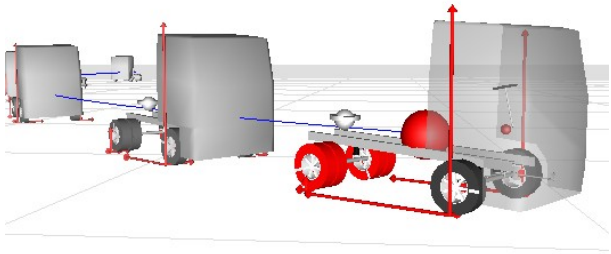

Figure 4. Animation screenshot of a heavy braking maneuver.

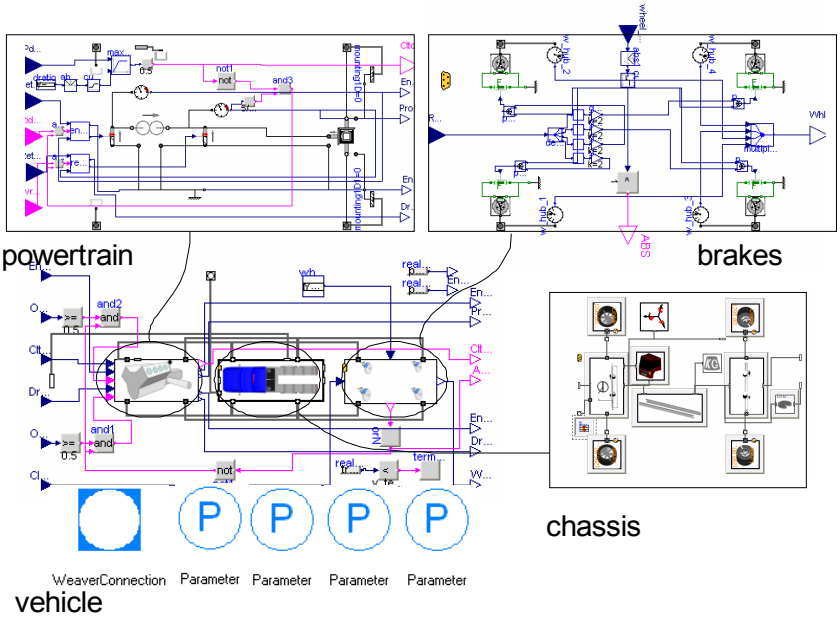

Figure 5. Vehicle model with powertrain, brakes, and chassis subsystems and TestWeaver instruments.

5. VIRTUAL VEHICLE INTEGRATION

For a precise and flexible simulation of system-level behavior we chose a virtual integration platform based on Software-in-the-Loop (SIL). The vehicle software functions and the simulation of the vehicle dynamics can be integrated by co-simulation on standard PCs. The simulation is fast due to the use of compiled modules (DLLs) for the *(i)* vehicle functions, exported with RealTime Workshop from Simulink, and for the *(ii)* vehicle dynamics simulation, exported from Modelica / Dymola. As integration tool Silver was used here. In addition to the basic integration functionality already mentioned, Silver offers the possibility to parameterize the virtual system using the same standardized interfaces and data exchange formats as those used in cars, e.g. ASAP2 (A2L), XCP/CCP, DCM (ETAS), MDF (Vector). For calibration tools, such as Inka (ETAS) or Canape (Vector) the Silver simulation behaves like an ECU. Further tools are provided for controlling, visualizing, recording and debugging simulation runs in Silver, as well as a direct interface to the automatic test generation and evaluation tool used here: TestWeaver. Figure 6 depicts the virtual integration architecture used here.

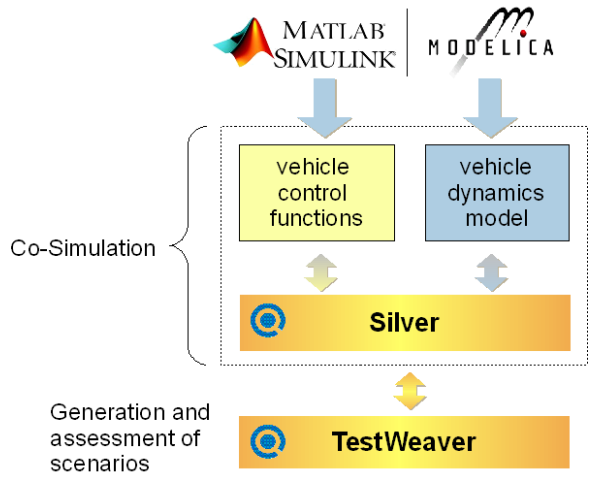

Figure 6. The virtual vehicle integration architecture.

Instrumentation

Based on the results of the hazard analysis described in Section 3, the instrumentation of the virtual vehicle simulation was done. The instrumentation included 43 instruments: one alarm for hazard monitoring, *i.e.* the retardation deviation, 2 inputs for the brake pedal and for the steering angle, 8 parameters in the vehicle dynamics model for robustness studies and 33 faults for the events surrounding the brake-blending function, as described before.

6. RESULTS

Using the instrumentation and the vehicle simulation TestWeaver generates many differing simulation runs and can build, with the help of the reporting functionality, an abstract view of the qualitative behavior of the vehicle. It is then possible to find out which faults result in safety critical alarms, and under which conditions in terms of parameter settings and input sequences. Moreover, it also possible to analyze different aspects of coverage, in terms of input and parameter space coverage, fault coverage or system state coverage – of course, only with respect to the discretized system space abstraction provided by the instruments.

When a fault is detected, the corresponding scenario can be replayed using the Silver environment to view details in the system response, see Figure 7.

With the control algorithm, simulation environment and game-inspired testing algorithms, the potential hazard causes have been investigated and some previously unforeseen system weaknesses have been revealed. One major result is that critical signals and code portions are effectively identified. Thus, this primary testing result points out the countermeasures needed. Countermeasures are then implemented in the form of plausibility checks for the critical input signals and safety monitors for critical code portions. With the countermeasures implemented, succeeding test runs are made in order to verify that the countermeasures are effective.

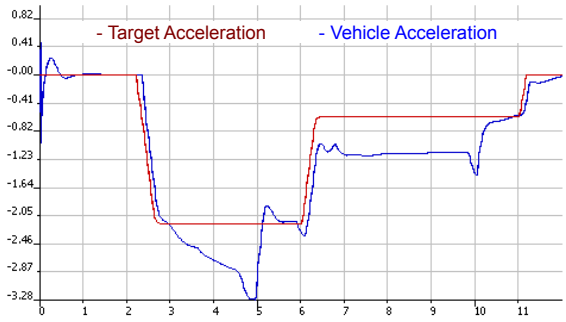

Figure 7. Example of result from fault injection. The engine speed signal is for this fault injection half of the true engine speed.

The presented tool chain can be used to support safety analysis such as FMEA / FTA as well as robustness and availability studies.

Safety Analysis

For the safety analysis, the fault injections are made on the interface signals only. The vehicle function is assumed to be monitored for code and numerical errors. In order to be able to study a reasonably big search space at the interface signals are discretized to certain gain values, as shown in section 3.

A hazard analysis has been used here to derive the

monitored safety hazards and the possible faults that could be related to them. The tool chain helped to analyze, in detail, the cause-effect relationship, an otherwise extremely difficult enterprise, if possible at all, considering the complexity and the amount of software functions involved. Of course, the simulation results need to be evaluated and complemented by the analysis of experienced engineers. As a second step, the tool chain presented here can be used to verify that the countermeasures designed to improve system safety are successful.

Figure 8 shows a piece of a coverage report generated by TestWeaver, for analysis of the single fault events.

inserted faults	BrkPdlPosCmd_Sh	speed
DrvRetrdAct_Tq_gain=omission	no_brake	(10..20)
	little_brake	(0..10)
	some_brake	(0..10)
	much_brake	(0..10)
		(10..20)
EngRetrdAct_Tq_gain=omission	no_brake	(0..10)
		(10..20)
	little_brake	(0..10)
		(10..20)
	some_brake	(0..10)
		(10..20)
	much_brake	(0..10)
		(10..20)
...	...	...
driveLineRetarderAxleTorqueShareMode=omission	no_brake	(0..10)
		(10..20)
	little_brake	(0..10)
		(10..20)
	some_brake	(0..10)
		(10..20)
	much_brake	(0..10)
		(10..20)

Figure 8. Subset of inserted faults (3 out of 33).

Most of the entire cross product of inserted faults, brake pedal positions and speeds are evaluated - only a few speed combinations are missing. The speed is not directly controlled, but a consequence of the brake pedal position.

Robustness Study

For the robustness study, the effect of combined variations of vehicle parameters, road conditions, and driver input sequence are investigated. The search space is then of course enormous and it is implausible that completeness can be achieved. Nevertheless, the major result is that often critical signals and parameters can be effectively identified. Thus, also this kind of study can reveal existing risks or robustness weaknesses, that need to be handled.

The robustness analysis covered 8324 states in 1585 scenarios, and reported 6 alarms, see Figure 9. For each alarm it is possible to see what scenarios and input sequences that leads to the undesired state. The report gives all necessary information to judge the criticality of the condition. This is a great help to find and understand possible weaknesses and their remedies.

speed	acceleration_error	faults	max duration	scen
(0..10)	high	*(none)*	0.74	s900
	low	*(none)*	3.16	s956
(10..20)	high	*(none)*	0.61	s870
	low	*(none)*	2.84	s866
(20..30)	high	*(none)*	0.01	s793
	low	*(none)*	0.42	s1021

Figure 9. Example of alarm report for the robustness study.

7. CONCLUSIONS

For safety critical systems it is important to prove that the still existing risks are tolerable. Understanding and assessing software is difficult. The presented method can help increase the confidence in the system safety and helps the engineers to better understand the complex behavior of systems that include complex software. Our method complements the existing safety analysis and function validation methods with detailed and comprehensive analysis based on simulation.

The presented testing method reveals the safety critical signals and system portions in a very effective way and at an early stage. Required countermeasures in the form of plausibility checks and safety monitors are identified. The verification of the implemented countermeasures can be performed with succeeding tests.

8. ACKNOWLEDGEMENTS

Haldex Commercial Vehicle Systems division develops, manufactures and markets brake systems for heavy trucks, trailers and buses. The product offering includes all main components and subsystems included in a complete brake system.

The Brake Blending function is developed for Haldex. Haldex is gratefully acknowledged for letting us publish this study.

REFERENCES

[1] Philipson, N., Andreasson, J., Gäfvert, M., Woodruff, A.: Heavy Vehicle Modeling with the Vehicle Dynamics Library, Proc. 6th International Modelica Conference, Bielefeld, Germany, 2008.

[2] Anreasson J., Gäfvert, M.: The Vehicle Dynamics Library – Overview and Applications, Proc. 5th International Modelica Conference, Vienna, Austria, 2006.

[3] https://www.dynasim.se

[4] https://www.modelon.se

[5] https://www.modelica.org

[6] http://www.qtronic.de/en/silver.html

[7] Junghanns, A., Mauss, J., Tatar M.: TestWeaver: A Tool for Simulation-based Test of Mechatronic Designs, Proceedings of the 6th International Modelica Conference, March 3rd - 4th 2008, Bielefeld, Germany