presented at the: 8th International CTI Symposium Innovative Automotive Transmissions, Berlin, 1 - 2.12.2009.

Test-driven Development of DCT Control Software

Dr. Nikolaos Papakonstantinou, Dr. Sigrid Klinger
GIF - Gesellschaft für Industrieforschung mbH
{nikolaos.pap, sigrid.klinger}@gif.net

Dr. Mugur Tatar
QTronic GmbH
mugur.tatar@QTronic.de

Abstract

Transmission systems, such as DCTs, require systematic test and validation methods in order to guarantee correctness and quality despite shorter development times and increasing complexity. Therefore, during the development of a new dual clutch transmission at GIF, a process for extensive test and optimization of the control software has been adopted. For this, system correctness and quality criteria are evaluated for thousands of simulated test scenarios (driving maneuvers). Problems and system weak points are identified and corrected as early as possible and the process is repeated after each correction or system change. The tool TestWeaver [2] from QTronic was used for the systematic generation and evaluation of the test scenarios. This allows achieving high test coverage with minimal test specification effort. This extensive test process drives and accelerates the optimisation of the control software. Beside MIL tests also intensive SIL, HIL and tests with the real hardware components in the loop must be performed in order to guarantee the functionality of the automatically generated code, the proper functionality of the TCU including electronics and CAN communication, and the robustness of the software functions under real operating conditions [3].

1 Introduction

Transmission systems are under continuous improvement with respect to efficiency, robustness, costs and comfort. Many of these requirements have to be addressed also by the transmission control software, which is becoming more and more intelligent. Many driving conditions have to be detected fast and reliably. Specific, optimized actions and control strategies have to be performed in order to achieve the optimum balance between often conflicting goals: efficiency, agility, comfort. Many modules are interacting and cooperating in order to achieve the system requirements. The test and the validation of the control strategies become, however, also more difficult. Module tests performed in isolation, no matter how extensive, cannot assess the emerging system behaviour, while the test of the system in only a limited set of predefined test scenarios does not scale with the huge number of differing situations that need to be tested and validated.

The software of the new GIF DCT is developed using a model-based development process with Simulink and TargetLink. The complete control software model can be simulated in closed-loop with a realistic plant model developed also in Simulink - including transmission hydraulics, mechanics and vehicle model. For faster simulation the complete model is compiled with Real-Time Workshop from Mathworks. This way, the simulation runs about 20 times faster than real-time on a commercial PC. The simulation includes, beside the control model and the plant model, also several correctness and quality observers implemented in Simulink. TestWeaver then controls and evaluates the simulation runs. Thousands of scenarios are automatically generated and assessed in a reactive loop in which TestWeaver actively attempts to reach all possible system states with at least one scenario and to find scenarios with correctness or quality problems. For instance, it will try to reach all transmission shifts, simple (e.g. 4-5) and multiple (e.g. 6-4), with many differing speed and torque loads, with differing street slopes, with and without braking, with or without chan-

ging the acceleration pedal during the shift, with ideal sensor models or with active injection of sensor faults. All problems found can be recalled and analysed in detail in simulation by the development engineers. The corrections are then validated again in simulation. The complete test process is repeated after each significant software change.

Beside the software tests performed in the simulation environment, functional tests on special DCT test rigs are performed by GIF. On these test rigs the software functions are validated and further developed with the real hardware parts of the transmission in all driving and environmental conditions. In that way the robustness of the DCT software, which is desired for the series production, is guaranteed.

2 The new GIF DCT

GIF is currently developing a compact dual clutch transmission for a large Asian customer. A broad development process is employed to address all aspects of the total development responsibility, including:

- Definition of load profiles suitable for the DCT
- Test specification
- Design concept
- Detailed DCT design
- Functional software development
- Durability tests
- Vehicle application and calibration with different climate trips

A very significant and challenging task also is the management and coordination of the different suppliers. The integration of the mechanical and hydraulic components and the choice of the proper sensors and transmission control unit are of significant importance for the proper functionality of the DCT. The optimal DCT operation can only be guaranteed under the proper and safe operation of the complete mechanical-hydraulic and electronic sub-systems.

The new GIF DCT includes the following defining features:

- Gear set with dual usage of the 3/5 and 4/6 gear
- Short and compact synchronizers
- Wet dual clutch in radial form and integrated torsion damper
- Separate hydraulic module
- External transmission control unit

3 The software development process

The high complexity of the DCT, the large amount of interfaces and the strict demands of the safety standards, increase the requirements of the electronic control system. In the last years a significant progress has been made in improving the tools and processes used for developing software for electronic control systems. The neat integration of the new tools and processes in the complete development of the transmission is of great importance: only this way the shorter time to market can be achieved. Inside GIF the wide spread V-process for the development of electronic control systems is used, as illustrated in Figure 1.

Before the software development, the specifications of the functional and of the safety relevant software have to be available. The functional software specification derives from the analysis of the OEM's demands and from the product specification. The resulting functional software specifications are validated with a detailed benchmarking of a similar DCT already launched on the European market. The specification of the safety software is based on the FMEA, hazard and risk analysis of the DCT. Also the demanded safety standards are considered. In the end the specification of the safety software by means of fail-safe functions and failure reaction is defined.

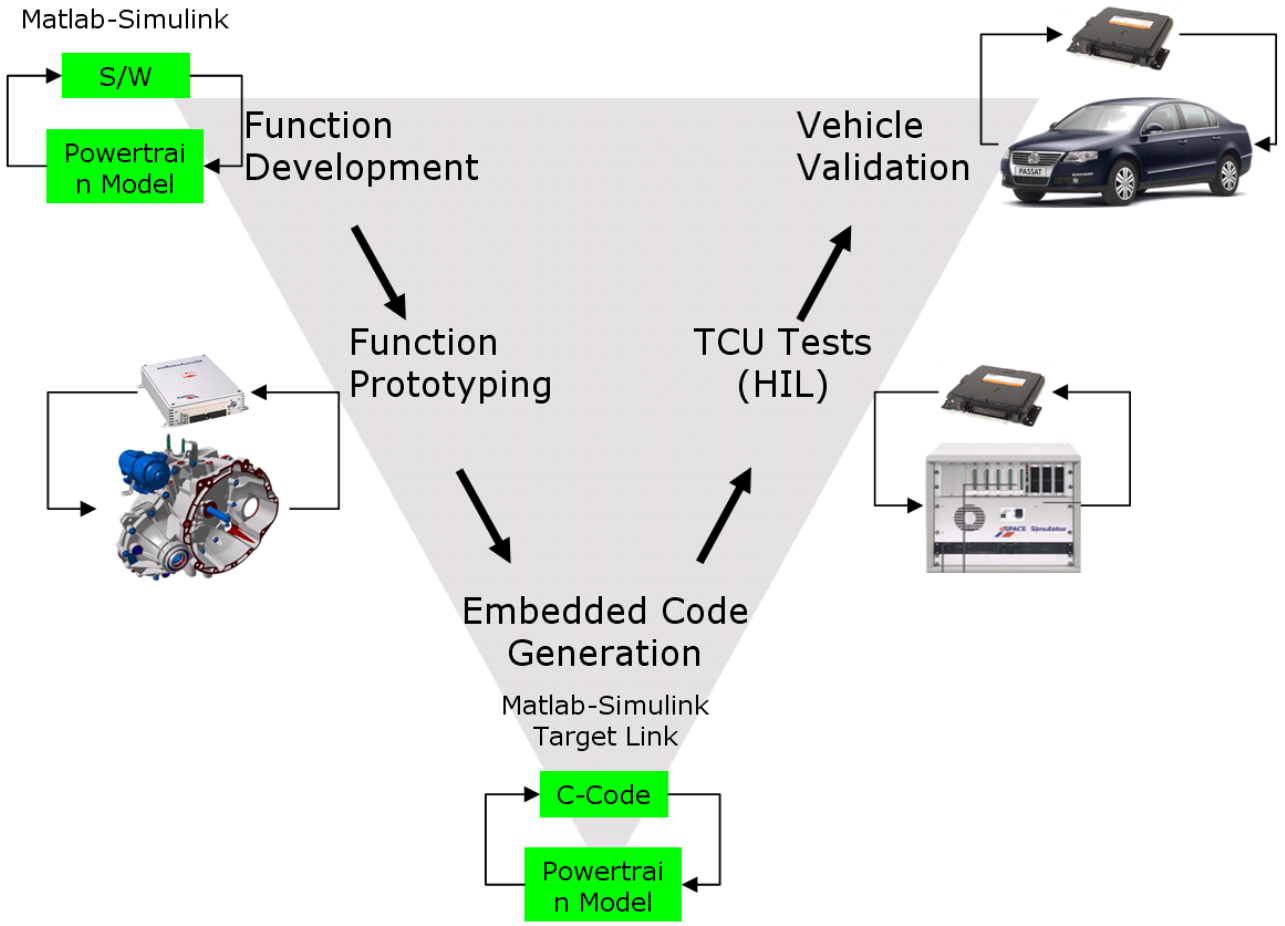


Figure 1: V-Process used for the DCT software development

The DCT control software, consisting of functional and safety software, is developed with model-based techniques using Matlab/Simulink as modelling and simulation tool. For this purpose also a simulation model of the physical subsystems of the powertrain is developed. This model consists of a simplified engine model, the DCT hydraulics and mechanics, and a reduced vehicle model. It has to be suitable for simulating the dynamic behaviour of the powertrain in all driving situations which are relevant for the transmission functionality. For this purpose a detailed hydraulic model, developed by the hydraulic supplier, is coupled with the mechanical model of the transmission.

The algorithms described in the functional and safety software specification are modelled with Simulink block diagrams and are tested in closed loop simulation with the powertrain model. With help of a user interface it is possible to interactively control the driver commands and other system inputs. Alternatively, this simulation is coupled to TestWeaver for automatic scenario generation and assessment. Section 5 and 6 will explain this process in more detail.

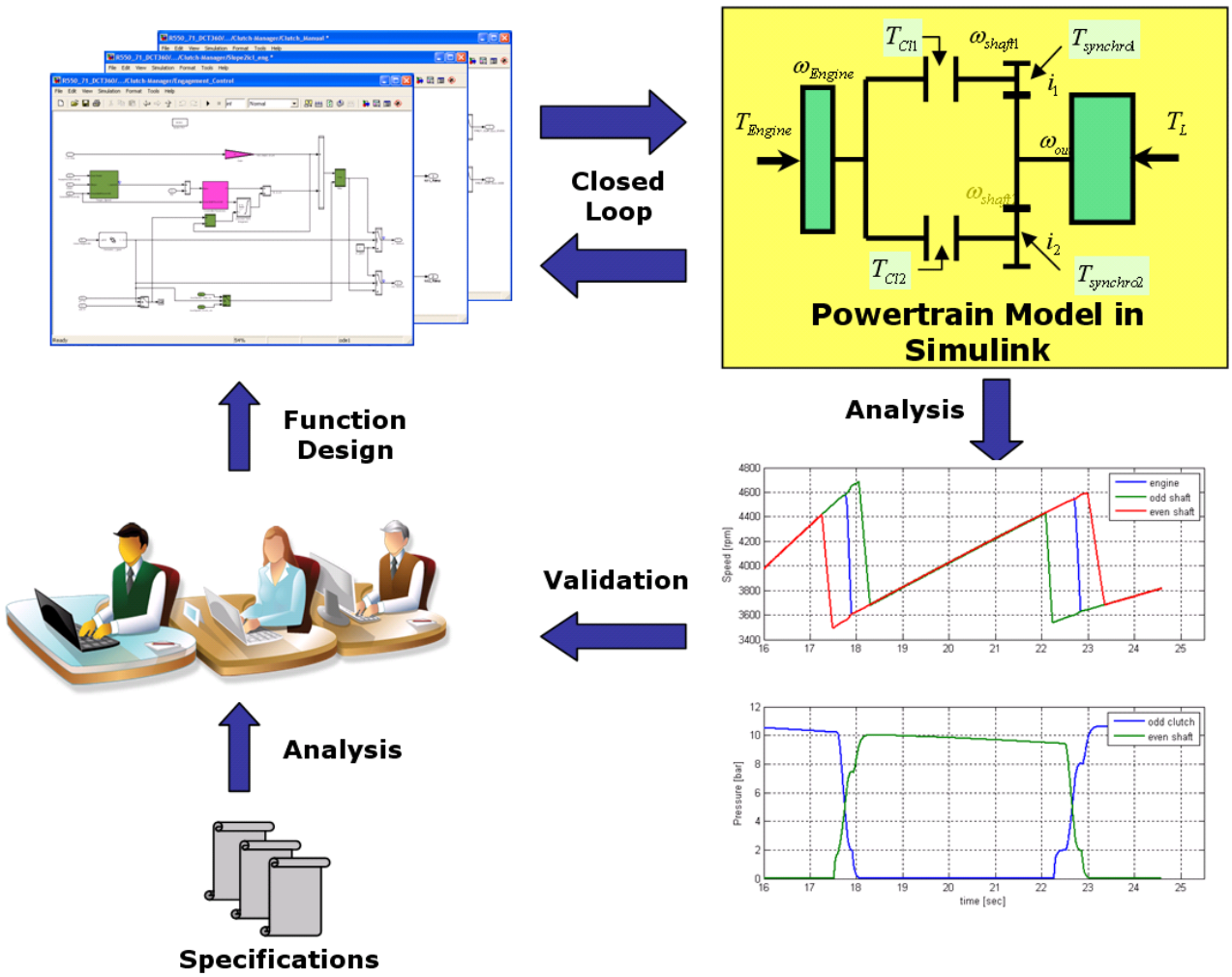


Figure 2: Model-based function development

The next phase in the development process is the function prototyping phase, where the developed algorithms are tested and validated with the real hardware components. For this purpose rapid prototyping tools from dSPACE are used. From the Simulink modules for the functional and safety software, C-code is automatically generated using Real Time Workshop / Mathworks and Real Time Interface / dSPACE. The prototype code can then run in a prototype controller from dSPACE. This tool chain is very flexible and minimises the time to prototype tests, since the engineers can concentrate purely on the function development.

In the same development environment also the embedded code generation is integrated. Using the same Simulink modules and with Target Link from dSPACE a 100% automated embedded code for the target TCU is generated. Based on Target Link, Software-In-the-Loop tests (SIL) are possible. In the SIL tests the generated embedded code is tested in closed loop simulation using the same powertrain model. The differences between the functional Simulink models and the generated code are analyzed in order to evaluate the efficiency of the generated code. The number of performed test cases, selected from a generated data base, depends on the target test coverage.

The generated embedded code of the functional software is integrated with the software from the TCU supplier. This code contains the operating system, the BIOS and the fault monitoring functions of the electronic system. This consists of the microcontroller, the sensors and the actuators of

the hydraulic module. The complete embedded code is then flashed in the production TCU. The software integration and the correct operation of the functional code on the target processor is checked and validated in a HIL system. In the HIL test the driving situations can be simulated using again the same powertrain model as in MIL and SIL. Furthermore, some critical situations, such as those caused by electrical failures, are simulated and the behaviour of the TCU is validated. The tested TCU is then released for vehicle use and for the final calibration procedure – for more details see [3].

4 Tests during functional developments

The software development process described above consists of different steps where the test and validation of the current software must be done. In the following sections the test procedures will be explained with special focus on the tests done during the functional development phase. Due to the complexity and the large number of modules (160), it is very important to start tests in the early development phase. Beside the module tests performed for each function in isolation, tests have to be done with the complete functional software in closed loop simulation with a realistic powertrain model. Only in that way the interdependencies between the functions, the interfaces and proper data exchange can be tested. For this purpose the complete software is handled as a “black box” with certain defined inputs and outputs. The inputs of the control software can be separated in three categories. In the first category are the driver dependent inputs, such as acceleration pedal, brake pedal and shift lever position. In the second category are the transmission and vehicle dependent inputs, such as sensor information for speeds, actuator positions, etc. The third category includes the environment dependent inputs, such as temperature and road slope. The software “black box” can be tested in closed-loop simulation with freely varying inputs from the driver and the environment and with constrained transmission and vehicle inputs computed by the powertrain model.

The amount of test cases relevant for the system is huge. It is probably impossible, or unfeasible, for a test engineer to define a complete set of test scenarios by hand. Beside the definition of the relevant test stimulus, also the test evaluation and the report generation are difficult and time intensive tasks. For this reason the tool TestWeaver from QTronic was integrated for automatic test generation, evaluation and reporting. TestWeaver is integrated in the model-based software development tool chain, since it is 100% compatible with Simulink. The following two sections describe the usage of TestWeaver during the development of the GIF DCT.

4.1 Scenario generation and assessment with TestWeaver

The specification of the interface between TestWeaver and the simulation is done in Simulink with the help of a block-set for TestWeaver. Each system input that is controlled by TestWeaver is set using a Chooser block, while the signals which are reported to TestWeaver are connected to Reporter blocks. During simulation TestWeaver autonomously sets the chooser-signals and reads the reporter-values. Thousands of differing test scenarios are generated and the system response is analysed and stored in a database. TestWeaver learns from the observed system behaviour and continuously adapts the scenario generation such as to (i) increase the coverage of the system states observable through the interface, and to (ii) detect problems in the system. For this purpose, in the Chooser and the Reporter blocks the value domain of the continuous signals is mapped to a discrete set of intervals that are relevant for the state coverage analysis. TestWeaver records and analyses the trajectories of the system state in this discrete state space, and intelligently tries to reach as many differing discrete states as possible during the scenario generation. Several software faults that provoke a simulation crash, such as divisions-by-zero, access violations, infinite loops and others, are automatically registered by TestWeaver. Furthermore, some of the reporter intervals can be marked with an abnormal or critical severity – this way TestWeaver will be informed about additional undesired or abnormal system states.

The system states reached during a test are reported using HTML tables. Via further links, descriptions are provided about how each state can be reached. Based on this information, at the end of the test, the scenarios with problems can be replayed in simulation by the development engineers for detailed analysis and debugging. More details about the tool can be found in [2].

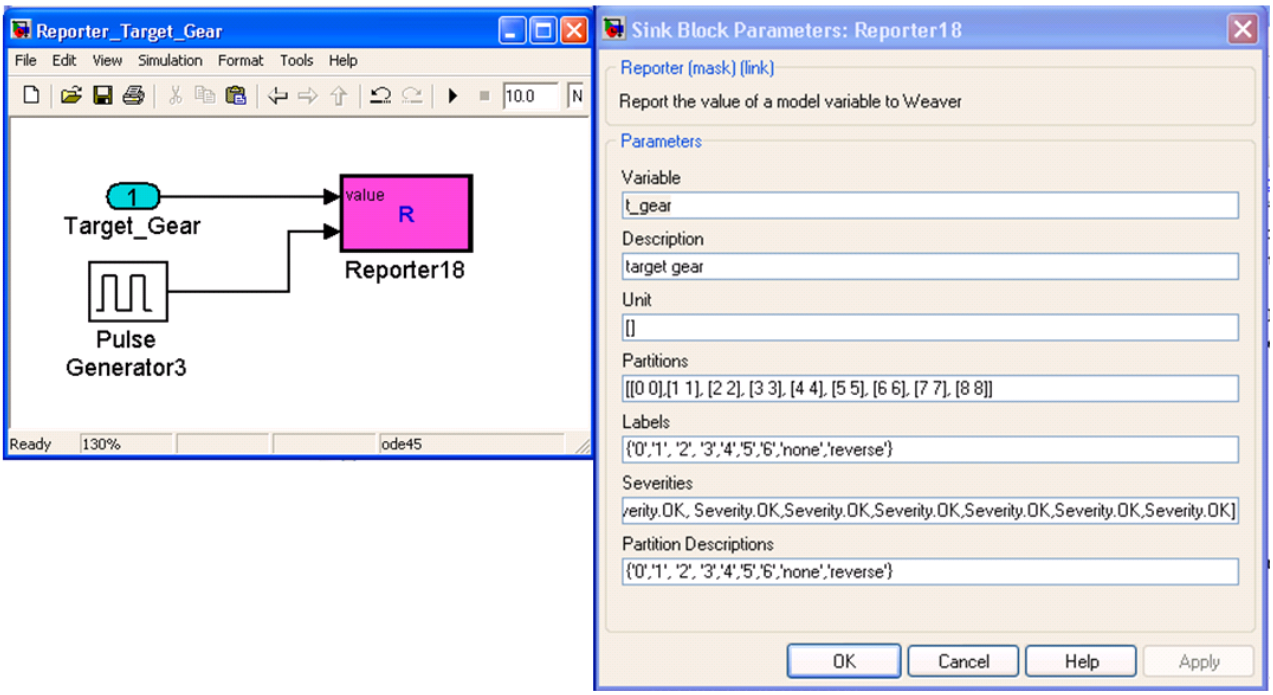


Figure 3: The definition of a reporter for TestWeaver

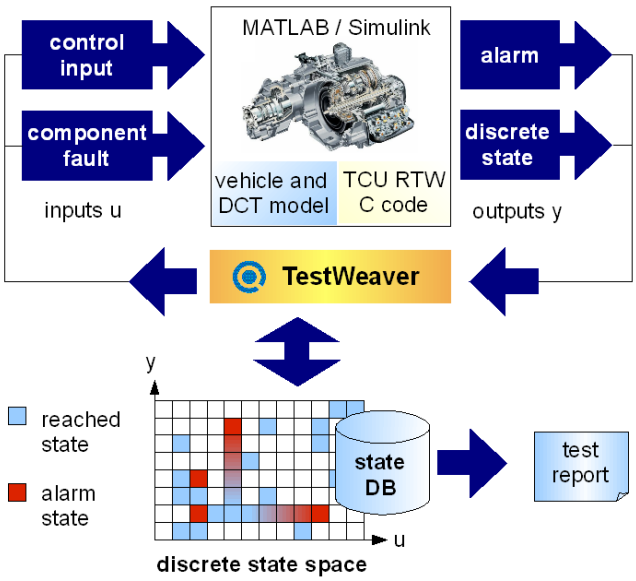


Figure 4: Automatic scenario generation and assessment

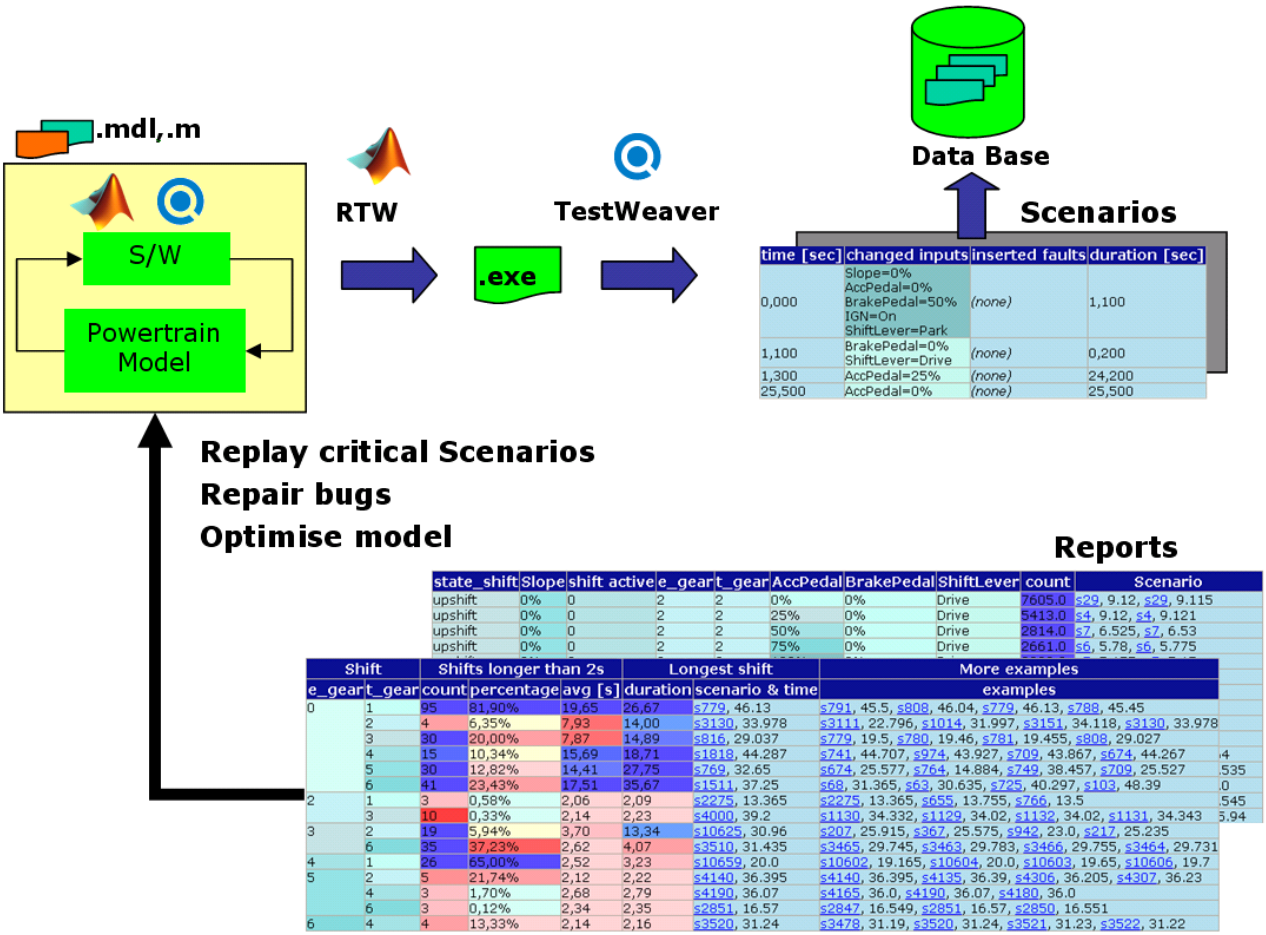
4.2 Testing the DCT functional software

The simulation started by TestWeaver includes: the DCT control software, the powertrain model, the driver's input block, where the choosers of TestWeaver are inserted, and a scope block where different state variables and the reporter blocks of TestWeaver are integrated. Additional functions can be added, beside the functional software and the powertrain model, and can be used as quality observers and for a better scenario assessment during the tests.

The signals controlled by TestWeaver in this application include the ignition, the acceleration pedal, the brake pedal, the shift lever position and the road slope. For assessing the coverage of the reached system states and activated control functions, several signals from the software and from the powertrain model are reported, such as car and engine speed, engine torque, internal gearbox speeds, actual and target gear, status of the synchronization according to the TCU controller and according to the powertrain model, clutch swapping status, shift time and timers running when the software is in critical loops.

Beside software problems that cause a crash of the simulation process, such as divisions-by-zero and access violations, etc., the following system problems can be detected automatically: engine over-speed, odd/even shaft over-speed, fault codes set from the safety software, too long shift times, too long clutch slip times, too high clutch temperatures, too long durations of certain control phases, and others.

Because TestWeaver records all discrete events, all transition sequences and all transition times, certain correctness and quality indicators are reported by analysing the scenario database recorded during the test. Examples of such problems are bad state estimation algorithms (estimation not matching state in the model for long time), unintended or spurious control sequences (such as repeated changes of the target gear, illegal control sequences), too long shift durations, etc. Certain quality indicators can be reported globally per experiment or for differing operating conditions, such as: average and maximal shift durations, average and maximal clutch temperatures. The quality indicators can be used to compare the effectiveness of software optimizations, for instance by comparing the test results obtained by different versions of the control software. The workflow for the iterative software test, analysis and optimization is illustrated in the following figure.



time [sec]	changed inputs	inserted faults	duration [sec]
0,000	Slope=0% AccPedal=0% BrakePedal=50% IGN=On ShiftLever=Park	(none)	1,100
1,100	BrakePedal=0% ShiftLever=Drive	(none)	0,200
1,300	AccPedal=25%	(none)	24,200
25,500	AccPedal=0%	(none)	25,500

state_shift	Slope	shift active	e_gear	t_gear	AccPedal	BrakePedal	ShiftLever	count	Scenario
upshift	0%	0	2	2	0%	0%	Drive	7605.0	s29, 9.12, s29, 9.115
upshift	0%	0	2	2	25%	0%	Drive	5413.0	s4, 9.12, s4, 9.121
upshift	0%	0	2	2	50%	0%	Drive	2814.0	s7, 6.525, s7, 6.53
upshift	0%	0	2	2	75%	0%	Drive	2661.0	s6, 5.78, s6, 5.775

Shift		Shifts longer than 2s			Longest shift		More examples
e_gear	t_gear	count	percentage	avg [s]	duration	scenario & time	examples
0	1	95	81,90%	19,65	26,67	s779, 46.13	s791, 45.5, s808, 46.04, s779, 46.13, s788, 45.45
	2	4	6,35%	7,93	14,00	s3130, 33.978	s3111, 22.796, s1014, 31.997, s3151, 34.118, s3130, 33.978
	3	30	20,00%	7,87	14,89	s816, 29.037	s779, 19.5, s780, 19.46, s781, 19.455, s808, 29.027
	4	15	10,34%	15,69	18,71	s1818, 44.287	s741, 44.707, s974, 43.927, s709, 43.867, s674, 44.267
	5	30	12,82%	14,41	27,75	s769, 32.65	s674, 25.577, s764, 14.884, s749, 38.457, s709, 25.527
	6	41	23,43%	17,51	35,67	s1511, 37.25	s68, 31.365, s63, 30.635, s725, 40.297, s103, 48.39
2	1	3	0,58%	2,06	2,09	s2275, 13.365	s2275, 13.365, s655, 13.755, s766, 13.5
	3	10	0,33%	2,14	2,23	s4000, 39.2	s1130, 34.332, s1129, 34.02, s1132, 34.02, s1131, 34.343
3	2	19	5,94%	3,70	13,34	s10625, 30.96	s207, 25.915, s367, 25.575, s942, 23.0, s217, 25.235
	6	35	37,23%	2,62	4,07	s3510, 31.435	s3465, 29.745, s3463, 29.783, s3466, 29.755, s3464, 29.731
4	1	26	65,00%	2,52	3,23	s10659, 20.0	s10602, 19.165, s10604, 20.0, s10603, 19.65, s10606, 19.7
5	2	5	21,74%	2,12	2,22	s4140, 36.395	s4140, 36.395, s4135, 36.39, s4306, 36.205, s4307, 36.23
	4	3	1,70%	2,68	2,79	s4190, 36.07	s4165, 36.0, s4190, 36.07, s4180, 36.0
	6	3	0,12%	2,34	2,35	s2851, 16.57	s2847, 16.549, s2851, 16.57, s2850, 16.551
6	4	4	13,33%	2,14	2,16	s3520, 31.24	s3478, 31.19, s3520, 31.24, s3521, 31.23, s3522, 31.22

Figure 5: Workflow with TestWeaver

The Simulink model that contains the DCT control software, the powertrain model, the driver model and the scopes are compiled with Real-Time Workshop for faster simulation. During scenario generation with TestWeaver the compiled model is used. This simulation runs about 20 times faster than real time[1] on a normal PC. This way in 16 hours running time more than 14.000 scenarios are generated. The test results are summarized afterwards with respect to specific aspects, e.g. conducted gear shifts, shifts with long duration etc. For this purpose the test reports can be configured and stored by the user. Following the test results can be analyzed in detail by model experts by replaying specific scenarios in Simulink. Bugs are repaired and optimization of the functions can be performed. After repairs or optimizations the tests can be repeated. TestWeaver allows also exporting scenarios in order to create a test data base which is used for regression tests or for code coverage, as can be seen in the following section.

5 Tests during function prototyping

The phase of function prototyping is very important in order to validate the developed functional software with the prototype hardware. Testing takes place in test environments similar to test rigs. In the following figure the different test facilities and phases during function prototyping are presented.

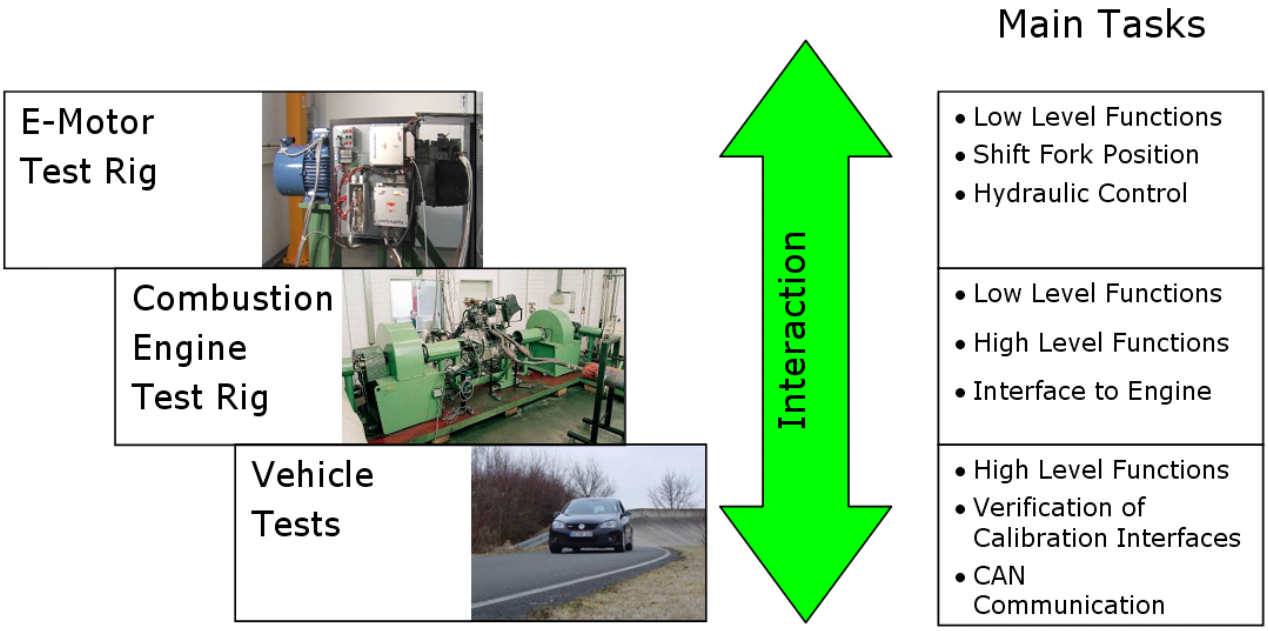


Figure 6: Test phases during function prototyping

The validation and the robustness of the Low Level Functions of the DCT are key components for the proper function of the transmission under all conditions (e.g. temperature, driving conditions, tolerances). The Low Level Functions are the functions responsible for positioning the shift forks, the torque transfer and the control of the hydraulic module (main pressure, lubrication, oil filling). Since these functions are depending on the behaviour of the hardware system very strongly, the development of these functions is very difficult, and it would not be sufficient if they were based solely on the simulated powertrain model. For this reason, after the model based development of the Low Level Functions, tests and further development are performed on a special electro-motor test rig. On this test rig the low level functions are tested and validated in all conditions in order to ensure the proper DCT operation. In this phase also the adaptation of the safety functions to the real hardware and sensor signals can be done. The test procedures and the test cases on this test rig are mostly related to differing operation conditions, such as temperature, speed and torque.

1 As opposed, the interpreted Simulink model runs about 15 times slower than real time.

Subsequently, a test rig with a combustion engine is used. Here the Low Level Functions are optimised and adapted to the operation with a combustion engine. On this test rig the same load conditions as in the vehicle operation can be achieved. This way it is possible to verify and validate also the High Level Functions designed with the powertrain model. The High Level Functions, consisting of the driving strategies, the transmission and the clutch coordinator, can be adapted to the tolerances of the mechanical system and the quality of the sensor signals - which cannot be simulated realistically with the powertrain model. A very critical issue for DCTs, which is also tested on the combustion engine test rig, is the communication with the engine and the validation of interfaces between the TCU (prototype) and the engine ECU. The software functions can be validated in that way with the real accuracy-tolerances of the torque information provided by the ECU.

The final tests during function prototyping take place in a vehicle equipped with the DCT and the control-application system from dSPACE. In this phase the High Level Functions are validated in the real vehicle and the functionality of the calibration interfaces is verified. The communication with other control units over CAN and the complete interface of the prototype TCU can also be tested.

6 Embedded code generation and validation

When leaving the stage of working with prototype control units and entering the stage of serial control units, embedded C-code has to be generated. In this project TargetLink is used to generate C-code automatically. TargetLink is an add-on tool for Matlab/Simulink. Simulink models can be converted into TargetLink models automatically. The TargetLink model contains specifications concerning the variable class, data type, the least significant bit, etc. With these settings the software specialists are able to generate C-code automatically. The function developers can use the same TargetLink model for further development as well. They use a TargetLink stand-alone software-version, that means only Simulink functionality is activated. The code specific options are not activated, so that the model can be used the same way as in the Simulink environment. The advantage of this workflow is, that even in early development stages, model configuration referring to code generation can be done - function developers as well as code specialists can work on the model in parallel.

With TargetLink also SIL simulations can be conducted. For this purpose the C-code is embedded into a simulation frame which is the interface between the code and the Simulink environment. The model first runs in Model-In-the-Loop simulation with floating point arithmetic. Afterwards the model runs in Software-In-the-Loop simulation, with data types specified in the C-code, i.e. integer arithmetic. Both simulation results can be compared in order to ensure that the C-code gives the same results as the Simulink model.

As suitable inputs a set of realistic driving scenarios is integrated into the simulation program, so that all possible shifts are conducted. With TargetLink the code coverage can be evaluated. If the code coverage is not sufficient the input test scenarios have to be expanded. Therefore the test data base with scenarios computed by TestWeaver is used. The tests are closed if sufficient test coverage according to the software integrity level is reached. In the following figure the process for the code quality test is illustrated.

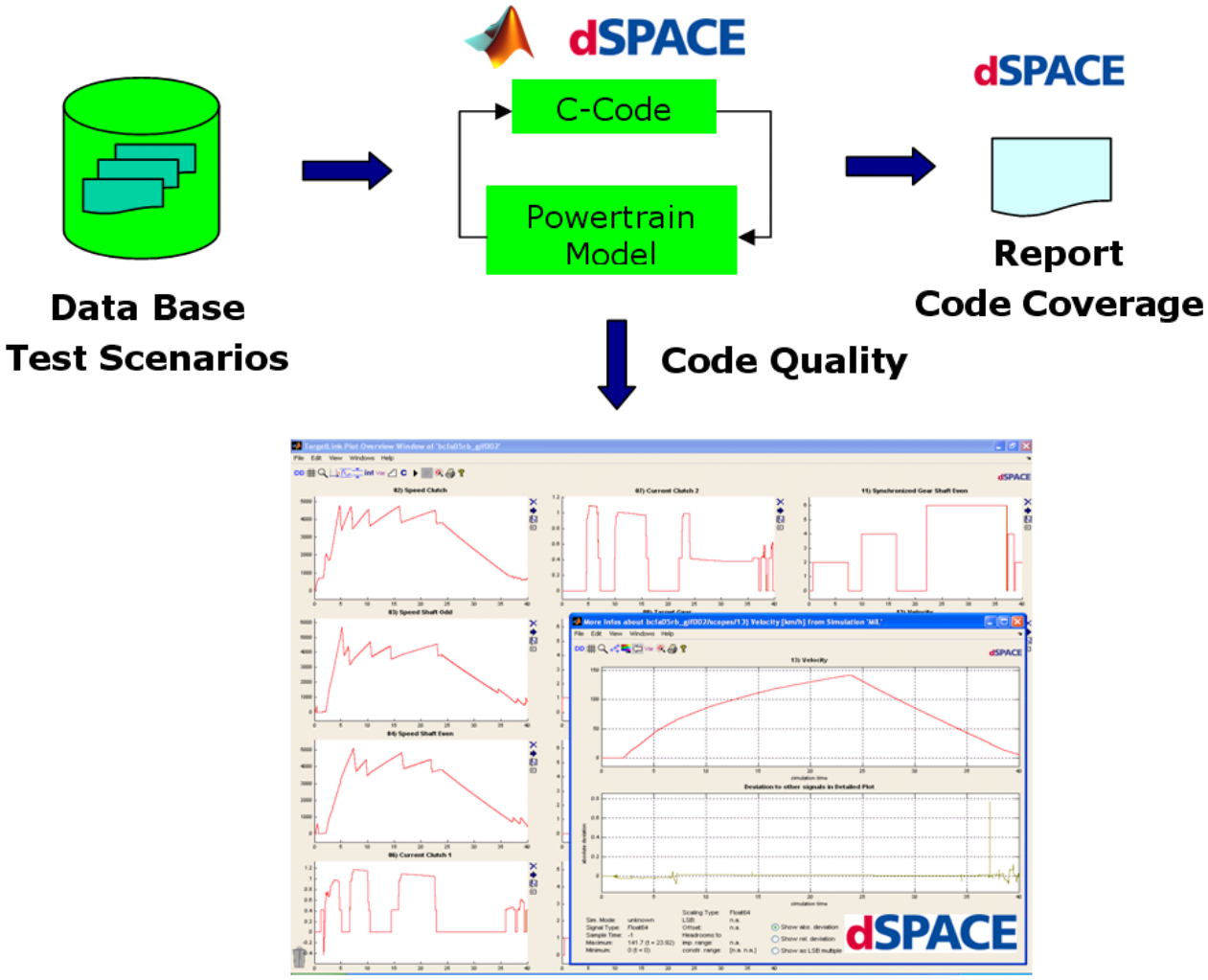


Figure 7: Embedded code quality control and coverage

7 HIL testing

The embedded code generated with Target Link contains the application software of the DCT. This code, in order to run in the production TCU, has to be integrated with the basic software of the TCU supplier. The basic software contains the operating system, the boot manager, the I/O management and the diagnosis functions. Using a build process, the final files which will be flashed in the TCU are generated (.hex and .a2l). For testing the correct software integration, the TCU is integrated in a HIL system supplied by dSPACE. The first tests performed in the HIL system are integration tests. In the integration tests the communication between the application and the basic software via the sharing interfaces is verified. Also the communication of the basic software with the virtual sensors, the real actuators and the CAN is tested.

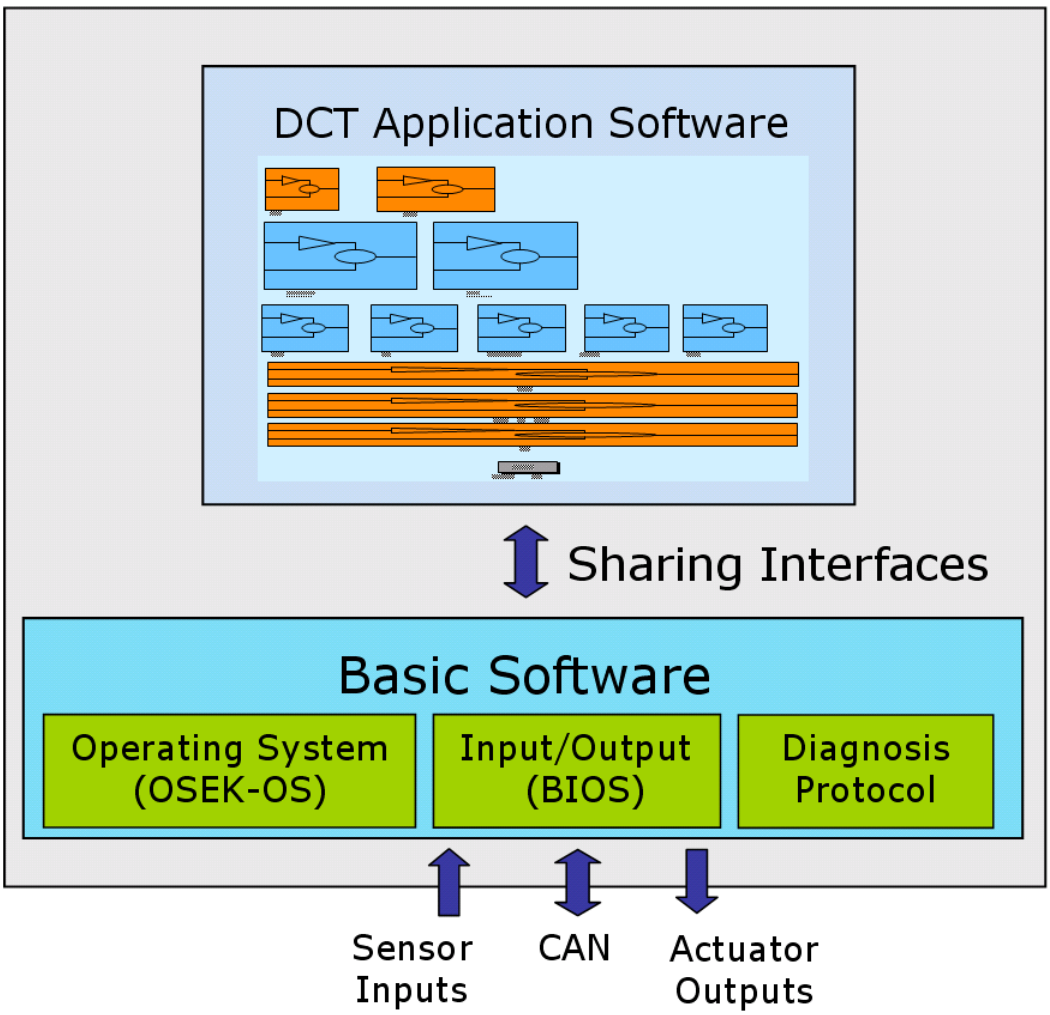


Figure 8: Software integration and flashing

Beside the integration tests, also functional tests are performed in HIL in order to validate the proper operation of the integrated software. The test scenarios used for performing the functional tests are chosen from the created TestWeaver test data base.

A further significant usage of the HIL system is to test the behaviour of the TCU under critical conditions, such as the ones caused by electrical failures. Using a Failure Insertion Unit (FIU) it is possible to provoke electrical failures on the TCU pins. All possible failure types, such as open circuit, short to ground and short to battery are initiated and the appropriate failure recognition and failure reaction is verified. For assessing the failure recognition the trouble codes generated by the TCU can be read. For assessing the fault reaction the behaviour of the powertrain model running in the simulator is analysed. The test cases for the electrical failures result from the electrical failure specification of the system. The electrical failure specification table (.xls) contains the corresponding failure code and failure reaction for each electrical failure. The control of the test flow is done using Automation Desk from dSPACE. In Automation Desk test scripts can be developed using graphical interfaces and Python scripts. Via Automation Desk the whole test flow can be controlled and test reports can be generated automatically. The signal flow used for testing the electrical failures of the TCU is illustrated in the following figure.

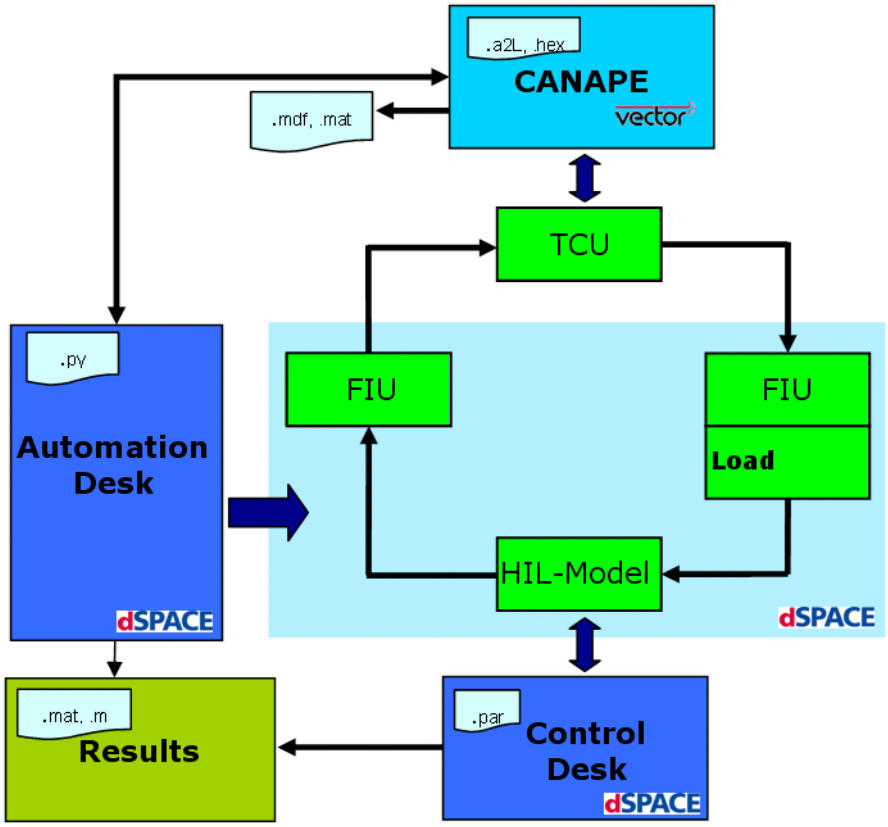


Figure 9: Signal flow during HIL testing

Since the communication of the different modules of the HIL environment is quite complicated, as can be seen in the figure above, some intelligent communication techniques are used in order to improve the HIL performance. For this reason the whole simulation is controlled by the Automation Desk. Automation Desk performs tests in loops. Measured data from the TCU and the powertrain model running on HIL are requested when a test is finished. Data from the TCU calibration tool CANAPE (Vector Informatik) are collected and a script running in Matlab is automatically started (not shown in the picture). This script contains evaluation criteria for the executed test and can decide if the test is passed or not. This is based on existing, predefined criteria from the failure specification list. Automation Desk is also able to erase the failure memory of the TCU (via CANAPE) in order to continue with the next test loop. This way the test execution and the test evaluation are done automatically. The user only needs to read the test report and to forward the possible failures to the function developers.

8 Summary and outlook

The complexity of transmission systems is steadily increasing due to growing market expectations regarding efficiency, agility and comfort. The corresponding development times are constantly shortened, while simultaneously keeping high quality standards. The growing complexity and limited resources impose an increasing pressure on both OEMs and suppliers to further improve the development process. In particular, also the test and validation have to become more reliable and more cost-effective.

For the new DCT development project GIF has adopted a comprehensive software test and validation method. Thousands of driving maneuvers are autonomously generated, executed and evaluated using simulation. Due to the high degree of automation, the test effort spent by the development engineers is significantly reduced, while, at the same time, the test coverage is significantly increased. Future improvement directions regard the application of the scenario generation and

evaluation for tests directly on SIL and HIL platforms. QTronic has recently extended its product range to cover these requirements.

Furthermore, beside the tests done in the simulation environment, very important complementary tests are done during the function prototyping phase. Tests with real hardware components must be done in order to ensure the robustness of the software functions under all environment and transmission states. During the development of the GIF DCT a large number of tests on test rigs has been performed in order to validate the proper and robust functioning of the transmission. The intensively tested software after the prototyping phase represents a good starting basis for the embedded code generation for the production TCU. The C-code generated from the Simulink block diagrams is subsequently tested in SIL and sufficient code coverage is guaranteed using test cases from the generated test data base. In this phase some further development is needed, since the execution time of the tests using the generated C-code in closed loop with the powertrain model is relatively long. HIL tests are done at the end of the process with the production TCU and the integrated software. These tests are very important in order to guarantee the proper integration of the different software parts and the function of the communication interfaces. During HIL tests, the functionality of the complete TCU, including the reaction to electrical failures, is tested. This consists of testing the failure recognition and the failure handling. Since the HIL is a very complex set-up with many different systems and different interfaces, a better system communication and standardization could improve the test performance, especially with respect to automation and automated evaluation of the test results.

References

[1] Rebeschieß, Liebezeit, Bazarsuren, Gühmann: Automatisierter Closed-Loop-Testprozess für Steuergerätefunktionen. ATZ Elektronik, 1/2007 (in German).

[2] Junghanns, Mauss, Tatar: TestWeaver - Simulation-based Test of Mechatronic Designs. Proceedings of International Modelica Conference, Bielefeld, 2008.

[3] Nissen, Papakonstantinou, Steinwascher, Kaufhold, Schneider, Schlipf: Entwicklung eines kompakten Doppelkupplungsgetriebes, 18. Aachener Kolloquium Fahrzeug- und Motorentechnik 2009 (in German).