Automatic Validation of Diagnostic Services

For the first time, a fully automated test case generator has been introduced in diagnostics validation at General Motors Europe (GME) Development. This article by GME and Vector Informatik describes the introduction of this automated testing of diagnostic implementations based on the example of the new Opel Insignia. Integrating the Vector Informatik tool into the existing tool environment resulted in improvements in the fields of economy, time and processes when compared to conventional, manual validation at the Opel Corsa.

1 Introduction

One consequence of strong competition in the global automotive market is that it is forcing a shortening of development cycles. Another is that the complexity of the electronic networking architecture is continually increasing. Key goals in replacing conventional systems by electronically controlled systems relate to cost reductions, a high level of safety and reliability as well as better manageability. Despite all of the benefits, it must not be forgotten that increased numbers of electronic components in vehicles can increase the probability of electronics-related faults. Since reliability is an important criterion for customers when purchasing a new vehicle, it is essential to introduce new methods that enable mastery of this complexity, accelerate the development process and guarantee proper operation of the installed electronic control units (ECUs). Particularly in the area of diagnostic functionality provided by the ECU, it is crucial that diagnostic services are correct. They transport information that helps mechanics in the service garage to quickly determine the cause of a fault and correct it. This information must make it possible for the mechanic to decide which component is the source of the problem and what needs to be replaced to restore full operational readiness. If this is not assured, the result may be erroneous replacement of properly operating units [1], which causes a rise in warranty costs and a decline in customer satisfaction.

The E/E architecture of the Opel Insignia consists of several Controller Area Network (CAN) [2] and Local Interconnect Network (LIN) [3] bus systems. All bus systems are accessed via a central diagnostic port (DLC), **Figure 1**. Communication is defined by a GM-specific protocol. This GM diagnostic specification is based on KWP2000 [4] and the CAN 2.0A standard. It contains all diagnostic services allowed for addressing an ECU's diagnostic system to obtain diagnostic information. These services are then output by the diagnostic tester to establish diagnostic communication. As soon as a request is sent, the addressed ECU(s) react with either a positive or negative response:

- Positive responses contain the diagnostic information requested by the diagnostic device. If there is a lot of diagnostic information, the response may include multiple message frames.
- Negative responses contain a clearly defined Negative Response Code, which gives information indicating the reason for the negative response. Negative Response Codes are given in accordance with the GM Diagnostic Specification.

The received responses must enable technicians to determine the cause for a

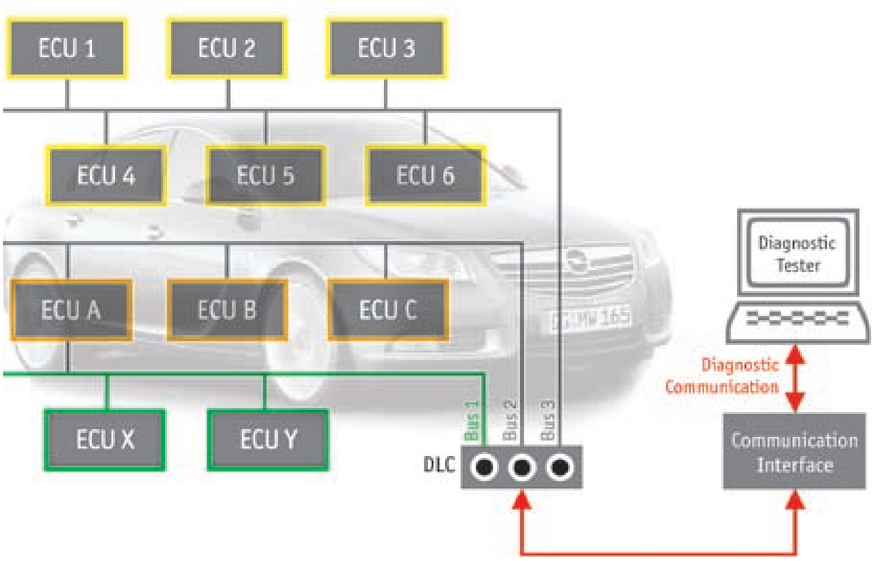
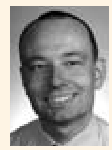


Figure 1: E/E architecture and diagnostic communication on the Opel Insignia

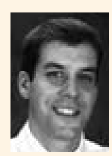
The Authors

Dr. Philipp Peti
is development engineer in the Global Systems Engineering group at General Motors Europe in Rüsselsheim (Germany).

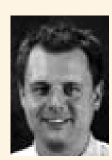


Dipl.-Ing.
Armin Timmerberg
has been working as development engineer in the Global Systems Engineering group at General Motors Europe in Rüsselsheim (Germany).

Dipl.-Ing.
Thomas Pfeffer
is group manager for Diagnostics and Test Automation in the Global Systems Engineering group at General Motors Europe in Rüsselsheim (Germany).



Dipl.-Inf.
Simon Müller
is product manager respondible for CANoe, DiVa on the Vehicle Diagnostics product line division at Vector Informatik GmbH in Stuttgart (Germany).



Dipl.-Ing. (BA)
Christoph Rätz
is global product line manager for the Vehicle Diagnostics product line division at Vector Informatik GmbH in Stuttgart (Germany).

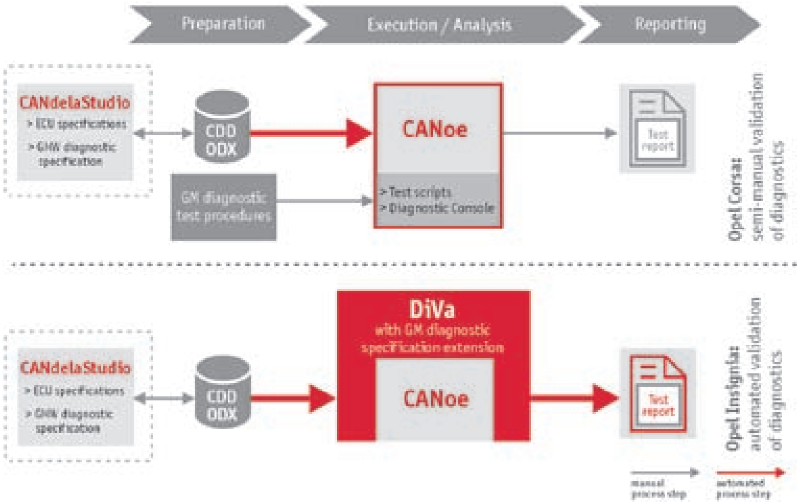


Figure 2: Comparison of diagnostic validation and tool environment on the Opel Corsa and Opel Insignia

fault, so that they can perform the right tasks to solve the problem.

Therefore, the success of a fault correction in the service garage depends considerably on the accuracy and precision of the data output by the diagnostic system. Proper implementation of diagnostic services is essential in performing quick and professional service or maintenance to the satisfaction of customers. Diagnostics also playing an important role in end-of-line testing: it is used to program ECUs and assure product quality. That is why comprehensive validation of diagnostic functionality is absolutely necessary.

2 Validation Process and Tool Environment at GME

In development of the Opel Insignia, GME introduced the "CANoe.DiVa" (Diagnostic Integration and Validation Assistant) tool from Vector Informatik for the first time. "DiVa" automates generation and execution of diagnostic tests. Figure 2 shows the tool environments for the Opel Corsa and Opel Insignia. In both cases, "CANoe" [5] is used as a test tool. While validation is largely performed manually in development of the Corsa, in development of the Insignia the vast majority of testing is covered by fully automated tests.

Figure 3 shows a typical diagnostic validation process for an ECU performed by a test engineer at GME. Development of the ECU software is subdivided into several phases. At the beginning of an ECU development, the focus is more on implementation of ECU functionality than on diagnostic services. The latter are then elaborated and developed in subsequent software versions. As shown in Figure 3, with introduction of the Phase 1 (SWR 1) software version, only a small number of diagnostic services are implemented. The use of diagnostic software components at GME ("CANdesc") has made it possible to implement a portion of the diagnostic content early at the start of development, and as a result it is integrated in the ECU earlier (see Figure 3).

The number of diagnostic functions to be tested grows with each development cycle. Once all diagnostic services have been implemented, regression tests are performed (SWR 7). If no more faults are reported in diagnostic services at that development stage, the ECU is production mature in the execution of diagnostic services.

Since a test engineer normally tests a number of different ECUs simultaneously, without adequate tool support it is impossible for the engineer to perform the large number of tests necessary to cover all of the implemented diagnostic services of the individual software versions. As a result, only newly implemented diagnostic services are tested in-depth, and test engineers perform representative regression tests for previously integrated individual services based on their experience. By using a suitable automation tool, more tests may be performed in validation while simultaneously reducing effort.

3 Requirements for the Validation Tool

A tool for automated diagnostic validation must satisfy the following requirements:

- seamless integration in the existing tool chain
- transparency and reproducibility: the test engineer must be able to track the executed tests and repeat them
- Conformity to existing testing methods at General Motors: the tool must support existing test methods; in the diagnostic area, the GM Diagnostic Specification already defines mandatory test procedures for "GMLAN Diagnostic Services" of the ECUs
- expandability by the test engineer
- automatic generation of test cases: the specification must exist in a machine-readable format to enable this

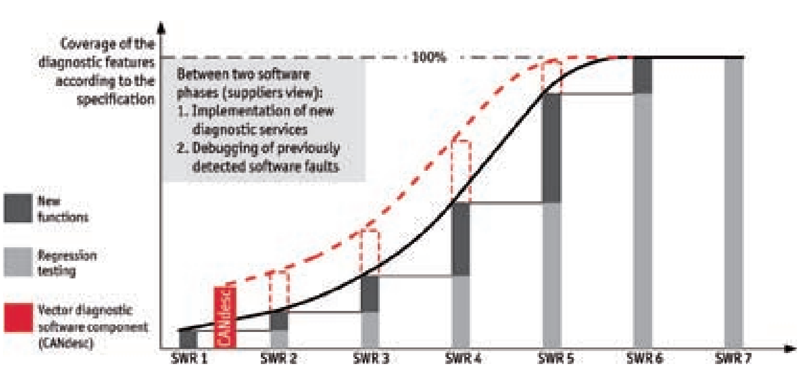


Figure 3: Scope of diagnostic functions in various phases of ECU development at GME

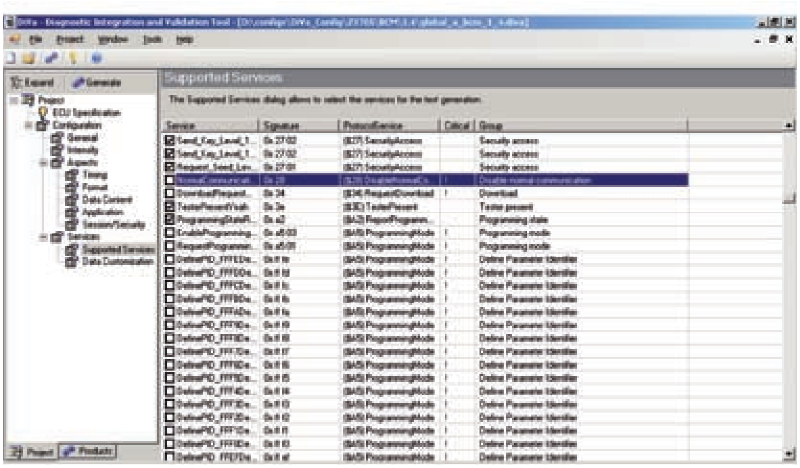


Figure 4: Setting test constraints in Diva (here: configuration of services)

4 From Specification to Test Execution and Report Evaluation

As shown in Figure 2, "DiVa" represents the link between "CANdelaStudio" (diagnostic specification) and the proven validation tool ("CANoe"). "DiVa" can be seamlessly integrated in the existing and established GME tool chain. Test cases for checking the individual services are automatically derived from the "CANdela" diagnostic specification (CDD file). The generated code is based on the "CANoe" programming language "CAPL" (Communication Access Programming Language) and can therefore be examined at any time. If problems occur, the test engineer can intervene in the automated test sequence and troubleshoot their causes (transparency). Furthermore, "CANoe"'s logging functions enable traceability and evaluation of the diagnostic data flow on the CAN communication level.

The following steps are necessary to conduct a test with "DiVa":

– select the ECU and its variant
– configure the test
– generate the test
– add the generated test module to the "CANoe" test environment
– execute the tests
– evaluate the test report.

The user can modify test constraints in "DiVa" at any time. Among other things, the intensity parameter is used to configure the test contents, such as full test, quick test or good case test. In addition, under supported services the user can exclude certain services from the test or modify data contents of the services under data customization, Figure 4.

In updating the diagnostic specification (the CDD file) "DiVa" enables synchronization to the new specification while preserving previously defined settings. From a technical perspective, "DiVa" generates CAPL code for the "CANoe" test module in order to test all diagnostic services supported by the ECU. To assure conformity to the GM diagnostic specification, the "DiVa" extension maps the test procedures of the GM standard. The test generation process produces a detailed description of the generated test cases, CAPL test codes for the "CANoe" test module and the associated "CANoe" test environment.

5 Test Execution and Report Evaluation

After the test has been generated, the user opens the generated test environment in "CANoe" and starts the test. The test duration depends on the complexity of the diagnostic specification and the user-defined test scope that is selected, and it may vary from just a few minutes to several hours, Table. At General Motors, the "CANoe" test environment serves as a joint platform for test automation and simplifies reuse of existing GM test programs. For example, end-of-line flash test procedures are also programmed in the "CANoe" programming language "CAPL". To simplify analysis by the test engineer, test reports are structured according to the GM diagnostic specification. Figure 5 shows a typical test report.

6 Test Coverage

Automating the tests extends test coverage and simultaneously shortens the time needed for test execution. The extent to which "DiVa" covers the test procedures described in the GM Diagnostic

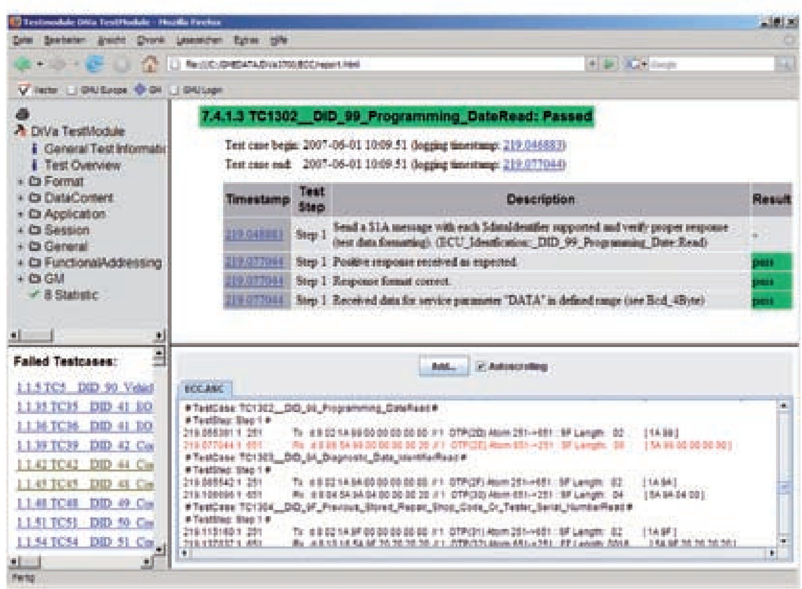


Figure 5: Automatically generated test report in "DiVA"

Specification is described below. The quality and number of generated test cases depend in large part on the completeness of the machine-readable diagnostic specification (CDD file). All generated tests are derived from it.

A total of about 350 test sequences are defined in the GM Diagnostic Specification. The test sequences cover both good case and bad case tests. A large share (approximately 80 %) of the test procedures are covered by fully automated tests in "DiVa". An application-specific user input is required for 45 (15 %) of the test procedures defined in the GM Diagnostic Specification. In such cases, "DiVa" pauses test execution and asks the user to put the ECU in the required state. The remaining 5 % of test procedures are not supported by "DiVa" and must be tested either manually or by other means. This includes tests that would put the rest of the test procedure at risk (like generating EEPROM errors and detect them) or would cause long-term changes to the ECU (like an ECU without calibration data). Testing depth is further enhanced by including execution of additional non-GM-specific test cases.

Comparisons made at GME between validation for the Opel Corsa and for the Insignia conclude that "DiVa" shortens test execution time enormously by predominantly automated execution of all generated test cases, **Figure 6**. The table shows a summary of execution times and the number of generated test cases for ECUs in the Opel Insignia. Often, manual tests can only be performed sporadically due to time demands. Therefore, test results largely depend on the experience of the test engineer and the amount of time available. At GME, "DiVa" enables both complete testing of ECUs per diagnostic specifications and greater test coverage in all development stages.

Table: Execution times and number of generated test cases for ECUs in the Opel Insignia

Insignia ECU	Generated Test Cases	Execution Time (w/o User Interaction for DTC Checking)
Instrument Cluster	1700	13:25 min
Climate Control	3350	39:10 min
Airbag	4630	1:19:32 min
Mobile Phone	1120	11:05 min

7 Economic Aspects and Efficiency Increases

When a tool is introduced, its economic benefit is a primary consideration. The new Opel Corsa is very successful on the market, and there are no negative reports of diagnostics-related electronic problems. That is why the manually performed validation process on the Opel Corsa was selected as a reference project. In contrast, on the new Opel Insignia, "DiVa" was being used as the primary tool for validation of diagnostic services. It was used to automate a large share of validation tests for the first time. For comparison purposes, the study evaluated the time required for test execution and evaluation in the validation phase, based on representative ECUs. The values given are based on implementation level SWR 5, Figure 3. Most services have already been implemented at that point, and a large number of failed test cases had already been captured. Figure 6 shows validation effort in hours for manual testing on the Opel Corsa and automated testing on the Opel Insignia.

By using "DiVa", execution and evaluation times were shortened considerably on the Opel Insignia compared to the Corsa. In the studied case, 3- to 5-fold improvement was attained (see Figure 6). In particular, the time savings was enormous for ECUs with a large number of diagnostic services. If one considers later development phases such as SWR 6 or SWR 7, the time needed for evaluating test results is reduced even further. This can be traced back to the smaller number of failed test cases in the more mature implementation. This trend continues in each new phase up to the production launch. The production ready ECU must not exhibit any defects; consequently, the evaluation time is equal to the execution time. In this stage of Opel Insignia development, depending on the complexity of the ECU, efficiency might be increased by a factor of 20 to 40.

The cost of the new solution is low, since all that is needed are licenses for

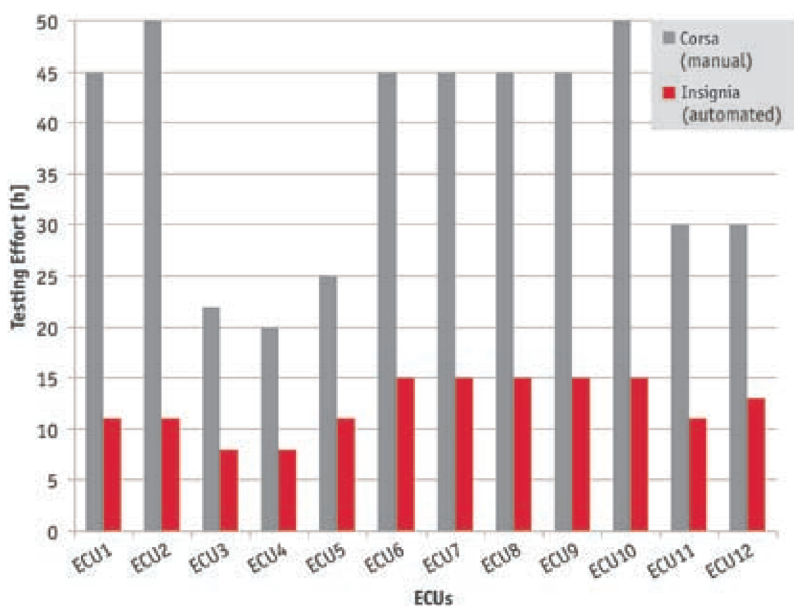


Figure 6: Test effort per ECU on the Opel Corsa with manual validation, compared to automated validation of diagnostic services on the Opel Insignia (execution and evaluation time)

"DiVa". A user at GME who is familiar with "CANoe" can perform "DiVa" tests – without prior training. Additional hardware is not required for test execution, since "DiVa" utilizes the available CAN infrastructure via "CANoe".

8 Limitations on Automatic Test Case Generation and Test Execution

Even if automated tools are better than manual test strategies in terms of test scope and time effort, automatic test generation does run into limitations:

- quality of the specification: since the specification represents the basis for generating test cases, completeness and accuracy of the specifications are essential – in other words, a test is only as good as its specification; furthermore, there must be conformity to the requirements of the General Motors diagnostic infrastructure (GGSE-I) [6]
- reproducibility: due to the non-deterministic properties of CAN communication in a vehicle, certain error situations are very difficult to reproduce in testing
- secondary fault: in case of error, the automated test tool – in contrast to a test engineer – cannot distinguish between an initial fault and a secondary fault
- user interaction: in application-specific tests it may be necessary to put the ECU in a state where additional hardware is necessary; these cases cannot be handled fully automatically in the approach described.

9 Summary

Without the use of test automation tools, it is hardly possible to achieve the desired coverage in validation of the diagnostic functionality of modern vehicles any longer. "CANoe.DiVa" has been adapted to GM requirements to support all established test processes, and it fits seamlessly in General Motors Europe's existing tool chain. It is used as an automated test tool for validation of diagnostic services on the new Opel Insignia.

With "DiVa", GME is not only shortening test duration, but is simultaneously increasing intensity of testing by its ability to perform regression tests more frequently. Furthermore, the scope of test coverage is extended by executing additional non-GM-specific test cases. In direct comparison to manual validation on prior successful projects, both technical and economical efficiency have been increased significantly. Depending on the development phase and quality of implementation, efficiency increases by a factor of 4 to 20 are realistic. At the same time, it is possible to satisfy the high expectations of customers in terms of quality.

References

[1] Thomas, D.; Ayers, K.; Pecht, M.: The "trouble not identified" phenomenon in automotive electronics. In: Microelectronics reliability, Vol. 42, S. 641-651, 2002

[2] LIN Consortium: LIN Specification Package Revision 2.1, OV. 2006

[3] Robert Bosch GmbH: CAN-Spezifikation 2.0, 1991

[4] International Organization for Standardization: Keyword Protocol 2000, ISO 14230, 1999

[5] Krauss, S.: Testing with "CANoe", Application Note AN-IND-1-002. Vector Informatik, 2005

[6] General Motors. GGSE ECU Diagnostic Infrastructure Requirements, Version 1.07, 2007