From “Tinies” and “Test Benches” to “End-to-End” Tests:
How to Bring Test Automation to Life

MAGNA Telemotive GmbH

by Rebekka Haisch, Product Manager Software Solutions
and Erik Wiedemann, Test Automation Expert

Abstract

This whitepaper outlines a proposal for the three-step validation of automotive electronic control units: from testing the single ECU to testing the communication in conjunction with other ECUs, and finally testing the entire vehicle. It describes several tools and methods that are used in the process.

Keywords: Test automation, Telemotive Test Automation (TTA), Hardware-in-the-loop (HIL), Software-in-the-loop (SIL), DevOps, Agile Testing, Smoke Testing, Ethernet, Keyword Matching

Content

Introduction

Modern-day vehicles contain more than 100 electronic control units, or ECUs for short. They control the car's increasingly complex electronic processes. While processing motor control signals, the vehicle operates infotainment systems, provides Internet connectivity, and protects and assists the driver and passengers with Advanced Driver Assistance Systems (ADAS).

Since the first introduction of complex electronic functionalities in the late eighties, the car's ECUs communicate via so-called bus systems, such as CAN, LIN, FlexRay, MOST, and Ethernet. The increasing data processing in modern cars has led to a huge jump in the number of sent signals. For example, the infotainment system's main component, the so-called head unit (HU), sends and receives well in excess of 10,000 signals alone. These signals form more than 2,000 features the customer can use.

Not only is the design of such complex structures a challenge, ensuring the quality of the whole product is just as demanding. Due to the embedded nature of the vehicle's software, testing needs to focus on signals and data, their processing for functional features, and of course the integration into the whole system to ensure a perfect overall user experience.

So how can we handle this huge demand for testing? How can we obtain test results just in time if agile teams provide new software or hardware versions at very frequent intervals? How can we integrate testing in a modern continuous integration/continuous delivery (CI/CD) toolchain?

Our answer is test automation! We believe that if test automation is set up and used intelligently in a project, it drastically reduces the costs and complexity.

In this whitepaper, we demonstrate our three-stage concept for the (automated) validation of embedded systems in a vehicle:

1. Communication and application – apply standardized automated signal tests and test the individual features of an ECU
2. Integration and graphical user interface (GUI) – automate iterative integration tests
3. End-to-end – apply specialized feature tests and help the manual tester with intelligent solutions.

The test pyramid

The so-called test pyramid is the basis for our three-stage validation concept. Mike Cohn first introduced this concept in his book “Succeeding with Agile” [1] in 2009. Following his idea, there are many different representations of test pyramids, all based on the same idea: to test the system from the smallest and most basic functionality to the largest and most general. Depending on the project and its complexity, the number of test stages might increase to ensure the highest possible test coverage.

Figure 1 illustrates the test pyramid. The levels of the pyramid represent the individual test stages. The width of the levels corresponds to the number of tests for each stage. The higher up the level, the fewer tests should be necessary in this particular stage.

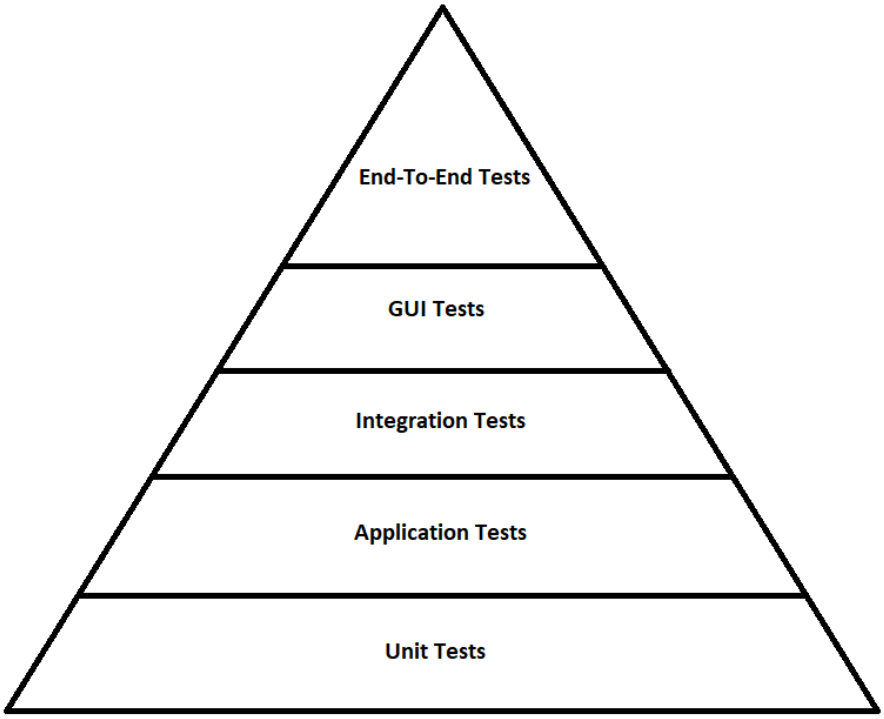

Figure 1: Five-level test pyramid

First, let us look at unit tests, which form the basis of any testing. Unit testing focuses on the smallest amount of code without looking at bigger structures. Developers mainly use unit tests to test their latest work directly and monitor whether any changes affect the code already written and its functionality. Unit tests are often specific to the programming language, thus they change depending on the language.

Second, application or component tests validate the fulfillment of the specific ECU's requirements. Those tests generally take two different approaches: white box tests, which look deep into the system's operations, and black box tests, which check the output for specific inputs. However, these tests do not concentrate at all on the user experience.

On the next level, integration tests validate whether the single ECU integrates into the complete system, and if the communication to other ECUs within the system works correctly.

Close to the top of the pyramid are GUI tests. Here, testing takes a step towards including user experience features and focuses on GUIs and their interaction with input parameters.

At the top of the pyramid are end-to-end tests. Here, testing focuses on the user experience of the feature. Manual testing becomes more important and may take a leading role. The most important point is that the interaction of all parts of the system is tested indirectly.

The border between integration tests, GUI tests, and end-to-end tests sometimes blurs, since some integration tests consider GUI features and end-to-end tests always also check the GUI.

All levels of the test pyramid can and should use automated tests. The lower level tests should be fully automated – especially if iterative development cycles come into play. However, creating automated tests is not without effort. When only looking at the starting phase of test automation, it generates more costs and consumes more resources than manual testing. The positive and cost-saving effect of highly automated tests is revealed in the longer term. Automated regression tests, otherwise performed manually, save time and the number of test cases, built up over time, ensure high code and product quality.

Telemotive Test Automation (TTA)

The automotive world, as well as other industries, comprises a great diversity of different technologies and technical solutions. Most OEMs and suppliers work with a proprietary, specific test solution, which requires a whole array of highly sophisticated, often divergent, hardware and software tools. Moreover, important and specific requirements often demand special tools.

In our experience, what is mostly missing is the special “glue” that holds this complex toolchain together. To solve this challenge we built a software framework, Telemotive Test Automation (TTA), that makes it possible to integrate every tool into an automated toolchain. For example, tools for testing web interfaces and back ends can be connected just as well as the software under test using REST APIs. TTA also enables flexible and quick creation of test cases.

Learning and adaptation based on Python 3 and open source are possible for every user. Pre-assembled units, sample code, and easy-to-apply test case implementation support a straightforward learning process for all new users as well as accelerated development cycles. A powerful but clearly structured GUI gives access to running test sequences, reports, and statistics with unambiguous visualization.

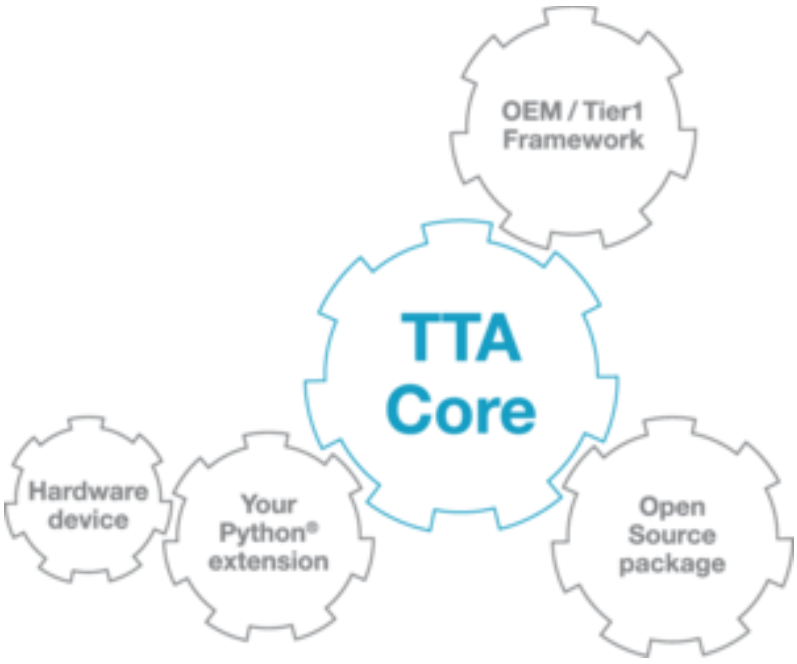

Figure 2: Structure of the TTA tool

Testing communication and application - The ‘Tiny’ test set-up

While a great number of unit tests provide good code coverage, application tests look at the actual functionality of the software. In an embedded, complex system with multiple ECUs, the individual ECU to be developed is a good starting point for the testing cycle.

The test set-ups reflect the three-stage validation concept.

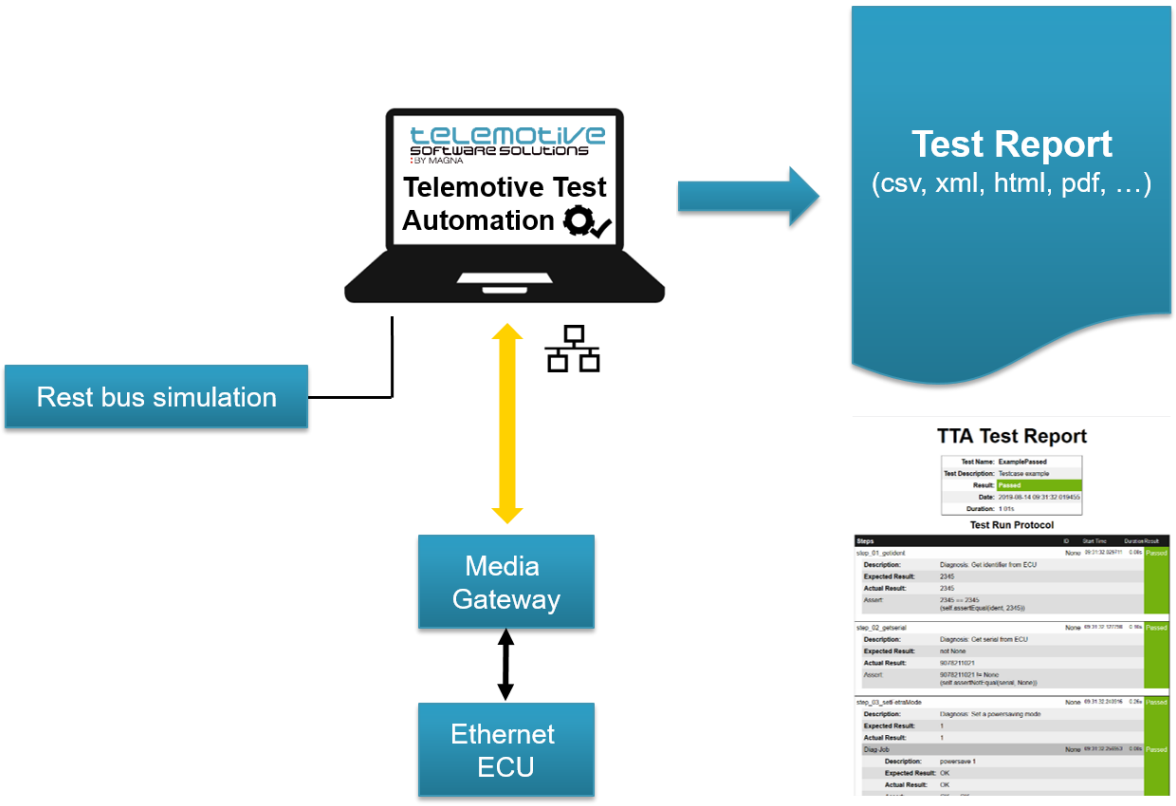

Figure 2: ‘Tiny’ test set-up structure

We start with a simple test set-up for signal level testing, which we call ‘Tiny’. The ‘Tiny’ is literally a very small test bench, as depicted in Figure 2. A media gateway connects the ECU under test with the test computer. This gateway translates and transports bus signals from the ECU to the computer. The test computer is equipped with TTA, which controls all the test activities from sending signals, to which the ECU reacts, to recording the ECU’s feedback and evaluating it. Additionally, TTA controls the so-called residual bus simulation (RBS) that provides a rudimentary vehicle environment and simulates the

remaining vehicle onboard network. As an example, it keeps the ECU alive or fulfills network management tasks.

Sometimes special test hardware is necessary for specific test cases. For example, an oscilloscope can examine physical signals and their quality, or signal generators can create signals to trigger a certain behavior. Unit developers can adapt TTA to gadgets like these, making them usable in automated test cases.

The test focus in the 'Tiny' set-up is on evaluating the communication protocol's implementation as well as testing the processing of the single ECU's application data.

This set-up provides a basis for testing the Ethernet bus communication, which is becoming increasingly important in modern vehicles. The Ethernet test cases for ECU validation are specified by the OPEN Alliance (One-Pair Ethernet), an open industrial alliance of automotive industry and technology providers, which is committed to the large-scale introduction of Ethernet-based networks as the standard in automotive networking applications. It has formed various technical committees (from TC1 to TC14) for the validation of Ethernet communication in cars. TC8 shares the requirements for testing automotive Ethernet ECUs and defines specifications that apply for all ECUs in an automotive Ethernet network, based on these shared requirements. TC8 currently defines more than 850 test cases; in addition, each OEM requires specific test cases.

The Telemotive Ethernet Testhouse (TET) was established especially to implement these tests. It offers semi-automated tests for the ISO/OSI layers 1 and 2 and fully automated testing for the ISO/OSI layers 3 to 7.

Testing standards and application specifications can lead to a large number of test cases. As mentioned above, 10,000 signals alone need to be tested, not mentioning the applications based on them. The possibility to automate not only the testing, but also the test case creation, helps a lot in handling these tests. To set up automated test case creation, test developers might use a method, which we call **keyword matching**. Here, a bot analyzes the test case specifications for reoccurring instructions, so-called keywords. The bot connects those instructions to programmed methods and puts them into an automatically created test case skeleton. As a result, the process of test case creation is greatly sped up. However, the keyword matching method relies on consistent and exact test case specifications. If the same description of a functionality uses several terms, the keyword matching will not work correctly, thus a manual analysis and adaption is required. In addition, an exact 'black or white' description of the test instructions and test result is mandatory. Automated test case creation and the automated test itself cannot interpret instructions, they can only validate true or false.

To sum up, exact specifications lead to exact test cases and unambiguous test results. It is therefore of great importance that the automation expert takes part in the creation of the specification as early as possible, to direct the project towards these matters. If these topics are not considered, the test automation will become ineffective and inefficient.

Testing integration and GUI - The extended ‘test rack’

The next validation step takes place in an extended test set-up, the ‘test bench’ or ‘test rack’.

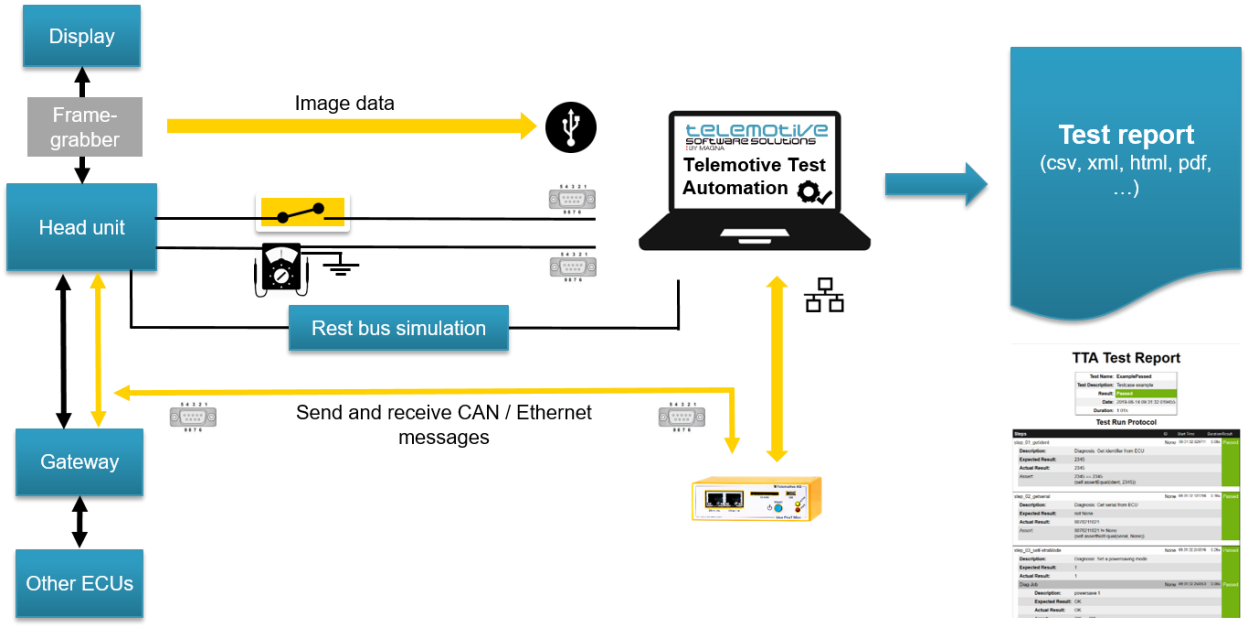

Figure 3: ‘Test rack’ set-up

With all protocols and ECU-specific applications tested, the integration tests focus on the interaction of several ECUs and the whole system’s requirements. Therefore, the test rack includes several connected ECUs that contribute to a user function under test.

Figure 3 shows an example suitable for infotainment test purposes, one of the main test fields in modern vehicles and certainly one of the main application fields for test automation. Today, the car (or the GUI) displays status information and notifies the user about a change of functionality. Hence, the interaction with almost all ECUs has to be tested.

In the test rack shown in Figure 3, the infotainment’s main ECU, the head unit, connects to a display and via a gateway to other ECUs. A so-called frame grabber records images from the head unit and transmits the image data to a test computer for further analysis. A data logger saves the log files of the whole test bench for subsequent analysis. Further

measurement instruments, such as signal generators, oscillators, or a multimeter, help testing certain purposes such as verifying the start-up behavior of the ECU.

Again, a RBS simulates signals from missing ECUs in the test rack, such as engine or steering ECUs. A simulation might also trigger simple signals like the signal coming from a closed door. Due to the integrated components, the tests performed in such a ‘test rack’ set-up tend to become very complex. Thus, finding a bug can be difficult and often requires analyzing several ECUs individually. Running a ‘Tiny’ set-up for every subcomponent in parallel to the test bench helps to isolate the bug.

The testing tools become more diverse at the integration test stage. To see the correct sending, receiving, and processing of data, analyzing log files is essential. Therefore, the data loggers need to be part of test automation. Moreover, any kind of optical analysis plays an important role in validating the display on the screen, such as optical character recognition (OCR), image recognition and comparison, icon search, icon matching, or video sequence analysis. Furthermore, checking online data or analyzing server communication becomes essential for testing any connectivity services. Here, live trace data analysis plays an important role in seeing the correct triggering of functionalities. For services using phone connectivity, automating these devices from different manufacturers is a fundamental task in test automation.

Seeing all those different tasks with all their different, specialized tools, the need for a framework like TTA that enables the integration of all those tools and that combines them into test cases, becomes clear. TTA has already integrated many of these tools: various frame grabber devices, various data loggers, the Android Debug Bridge (ADB), different image, video, and OCR analysis procedures, and therefore forms the basis for the implementation of automated integration testing.

Testing end-to-end – The test bench and vehicle

End-to-end validation, the last level of the test pyramid, takes place at the test bench and in the vehicle. In this context, manual testing is still important, as it is not possible to test the user experience with a machine. Human testers can assess the feel of a button or the sound of a key tone far better than any computer.

Figure 4: The principle of semi-automated testing

Assuming a ‘hard’ technical requirement, test automation still helps to evaluate end-to-end tests at an extended test bench, especially in media or connectivity applications. The above section described tools and techniques for this purpose, although the test focus here changes to the overall system. For any other purposes, however, intelligent automation can support human testers, and make their work easier, more effective, and focused. In semi-automated testing, for example, automation can take over repetitive tasks and non-essential aspects, allowing the human tester to focus on the main tasks and objectives of the test.

For example, test automation can set up the car and make sure it fulfills all preconditions for the test. Test automation might also evaluate the data sent by the car to a phone or the backend while the tester focuses on operating the car. As an example, a comparatively new feature is the use of extended reality (XR) glasses, where the tester sees the test specifications in the lenses. Those XR glasses might also help evaluating the looked-at reaction of the car by recording and matching it with the test specifications.

MAGNA Telemotive has already developed such an XR solution platform, called Telemotive Xtended Reality (TXR), which MAGNA production plants utilize for their digitally enhanced initial parts release. End-to-end testing in a vehicle might as well use such technologies.

Conclusion

The three-stage validation concept and the testing solution this paper describes are not exclusive to the automotive sector. Every industrial segment, in which digitized machines are developed, can apply the test concept outlined here.

To keep up with the market, we continuously enhance TTA using new project experiences and customer input. We provide customized TTA solutions and offer engineering know-how and consulting for individual requirements.

The biggest challenge is yet to come. Autonomous driving requires an as-yet unknown dimension of test and validation to achieve the highest goal: the protection of lives. We believe that test automation plays an essential role in reaching those goals. It is better to address this role as early as possible.

For further information, please contact us at

TMO.sales@magna.com

About the authors:

Rebekka Haisch is a graduate industrial engineer (M.Sc.) and Product Manager of the Software Solutions division at MAGNA Telemotive. She is responsible for the strategic orientation of the division and the further development of Telemotive Test Automation, and other products based on market requirements. Her goals also include digitizing companies, for example, through automated rather than manual testing. She holds the Professional Scrum Master certification and has experience in both Agile Development and Agile Management.

Erik Wiedemann is team lead and software developer for test automation at MAGNA Telemotive. His work focuses on Python-based test automation of automotive embedded systems within agile teams. Aside from ensuring high code quality, he also maximizes the impact of test automation on projects by guaranteeing it the attention it deserves in the project structure. Therefore, he looks for methods and ideas all around the different agile frameworks. Consequently, Erik supports colleagues in various project in questions of automation-related test specifications and test concepts. He graduated in Experimental Quantum Physics with focus on Quantum Computing and Quantum Optics.

References

1. Mark Cohn. Succeeding with Agile: Software Development using Scrum - Addison Wesley, 2009

DRIVING **EXCELLENCE.**
INSPIRING **INNOVATION.**