SiL-First Approach in a Test Ecosystem as Success Factor for Automotive Software

In-house software development is playing a crucial role in the automotive industry, as innovative software features are one of the key differentiators in the market. Before being used in a vehicle, they undergo countless tests in various environments using the latest methods. Mercedes-AMG and tracetronic show how the SiL-first approach is used for a control unit in the e-Drive to identify and correct errors in the code as early as possible.

As the complexity of software in modern vehicles increases, manufacturers are focused on establishing a consistent, automated, and transparent testing process. At the same time, methods need to be used that enable early testing activities in the development. In this context, Software-in-the-Loop (SiL) testing is increasingly being used in virtually scalable environments.

Testing is being moved to SiL environments primarily for economic reasons. In the context of agile software development, this approach is the most cost-effective and fastest way to run a large number of tests continuously in parallel and ensure the shortest possible feedback time. This is because the necessary virtual infrastructure can be set up with low-cost resources and expanded as needed. In addition, traditional environments such as Hardware-in-the-Loop (HiL) test benches and vehicle testing remain important to validate vehicle functions under real-world conditions. The key to success is highly automated and continuous testing across diverse environments. But how do you ensure maximum usability while keeping track of thousands of results?

WRITTEN BY

Dipl.-Ing. Jan Georges
is Senior Engineer at tracetronic in Dresden (Germany).

Moritz Sauren, M. Sc.
is e-Drive Software Developer at Mercedes-AMG GmbH in Affalterbach (Germany).

Dipl.-Ing. Tobias Fochtmann
is Senior Software Developer at tracetronic in Munich (Germany).

Dipl.-Ing. (FH) Carmen Schaak
is Manager Digital Methods e-Drive at Mercedes-AMG GmbH in Affalterbach (Germany).

The increasing focus on in-house development of software in the e-Drive has led to a transformation at Mercedes-AMG. With the introduction of the Scaled Agile Framework (SAFe), cross-functional teams have been used to develop software features, requiring new roles and skills, and placing an emphasis on high quality testing. Moreover, the development of a powerful system for software testing was given special priority. As a result, an end-to-end test ecosystem for the e-Drive has been developed and continuously expanded in close cooperation with tracetronic since 2022. The goal is to standardize testing activities across several diverse environments and detect failures as early as possible.

DEFINITION OF A TEST ECOSYSTEM

Multiple test types need to be combined to test millions of lines of code, from classic unit tests to those on drive components and in the vehicle. Often there are entire departments, which are dedicated to software testing. The main challenge is that the code embedded in ECUs must respond to

events in the vehicle in near real time. This requires the development of complex HiL- or SiL systems that use physical models to induce critical states of the developed code and verify responses using appropriate measurement technology. Test automation tools such as ecu.test are used to manage the growing number of tests based on the constant changes and extensions to software functionality.

Individual test bench automation has been in practice for years. The challenge lies in the next level of automation [1]. This means that it is not just a matter of running tests quickly and cost-effectively on a test bench, but rather that the entire path, from development to individual test systems to the consolidation of all results, must be considered in an ecosystem for the release of new driving functions. This is the only way to optimize the entire process.

The first challenge is to define the goals for improved automation [2]. The Mercedes-AMG test ecosystem focuses primarily on:

- Speed: new software releases must be integrated and tested in a brief period of time.
- Suitable test environment: the most suitable environment that provides cost-effective and reliable results according to the test objectives should be used.
- Test case quality: the implemented tests need to fit into defined quality standards to be reproducible and easy to comprehend.

A test ecosystem is a complex IT system with hundreds of participants who work together to develop software, deliver test cases and models, review results and manage releases. This requires many tools to be linked together. One important requirement is stability, because tool updates or configuration changes in participating systems can break the entire system and must therefore be tested accordingly. Another one are data standards, which means, that creating and performing tests requires a wide range of data, from artifacts and parameters to results. Standards should enable the smooth exchange of data between tools. The third requirement is the alignment of the working method, because the entire process must be considered, rather than optimizing individual teams. Cross-functional teams of IT and test system experts form the basis for developing and operating the ecosystem.

TECHNICAL STRUCTURE OF THE TEST ECOSYSTEM

The primary goal of the ecosystem within the given boundary conditions is to standardize tools and workflows across the board and to achieve the highest possible level of automation for functional testing. This requires the development of a cloud-based business logic with the final integration of five specific tools, including test.guide and ecu.test, into the existing infrastructure. The business logic will consider these different requirements and ensures that the right tests are run at the right time, with the right data and configurations. All testing activities of the e-Drive should be centrally accessible to all stakeholders. In such an architecture, a lot of information needs to be processed and exchanged. It is therefore particularly important to rely on robust APIs for all the tools and systems involved and on the availability of interpretable data formats. In addition, it must be ensured that the respective artifacts such as test scripts, configuration files or mapping files can be developed and made available to the corresponding tools asynchronously from the automated execution, **FIGURE 1**.

A major challenge in building such an ecosystem is the integration of environments with different levels of automation. At the heart of this is the business logic that automates specific workflows and standardizes data exchange to enable seamless integration within your own IT infrastructure. It also acts as an interface to external systems.

For the e-Drive, both fully automated processes in virtual environments and less automatable processes on the test

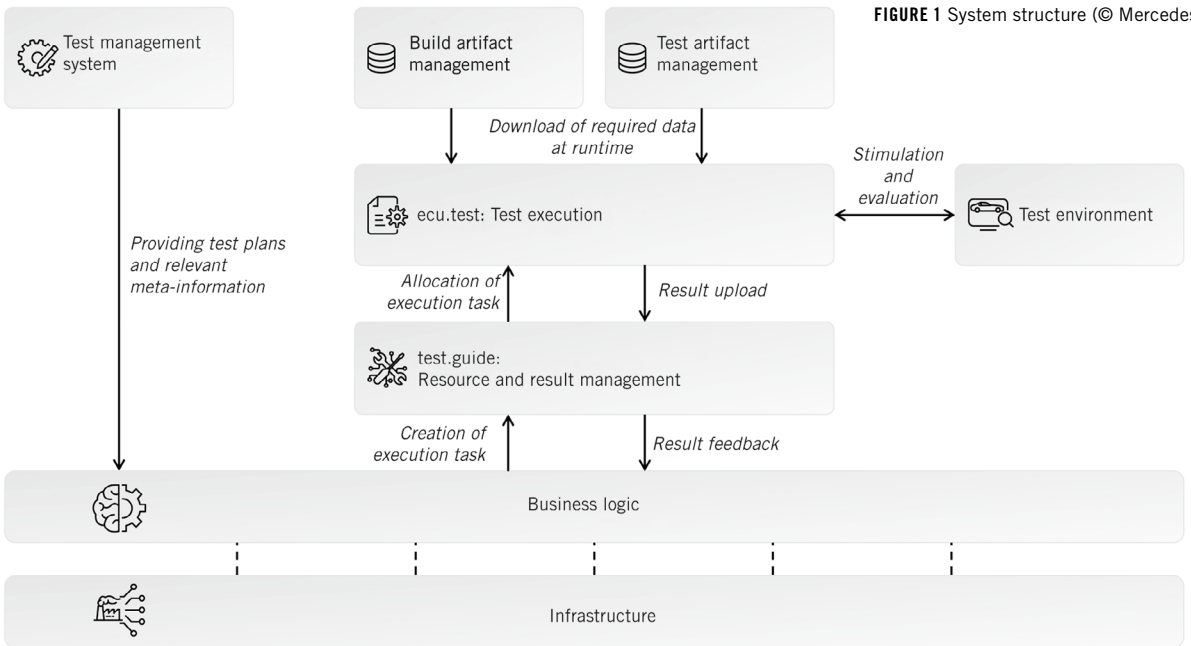

FIGURE 1 System structure (© Mercedes-AMG GmbH)

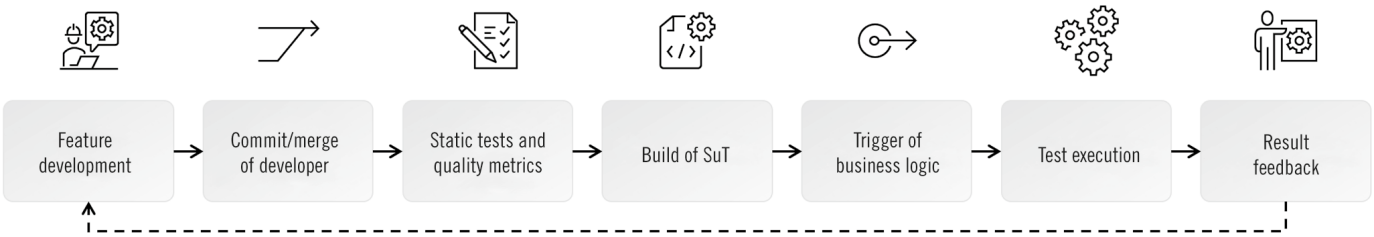

FIGURE 2 Feature development process flow from code change to test result (© Mercedes-AMG GmbH)

bench or in the vehicle must be integrated into the system. This means that automated execution can be triggered by events from an upstream CI pipeline as well as by manually configured jobs. As soon as such a trigger event occurs, the necessary data from all involved tools and systems is processed, an execution task is created and assigned to the appropriate test environment, and the planned tests are executed. Test results are stored centrally, independent of the environment, enabling stakeholders to easily get a comprehensive overview. All relevant metadata and measurements are also stored. The results are then made available for further processing to trigger automated follow-up processes, such as automated software deployment.

The entire platform is based on the approach of centralization by integrating distributed systems and tools into a separate business logic. Managing all the tools involved is particularly complex and challenging due to the many existing dependencies. In addition, individual updates to the IT infrastructure or downtimes can affect the operability of the entire system, which can quickly lead to a slowdown or even blockage in the development process of the ECU software.

REAL-LIFE USE CASE IN SIL

The test ecosystem is used for several e-Drive ECUs in different environments in real operation. This also includes a domain controller, which is being developed in-house at Mercedes-AMG. It relies heavily on the SiL test environment and

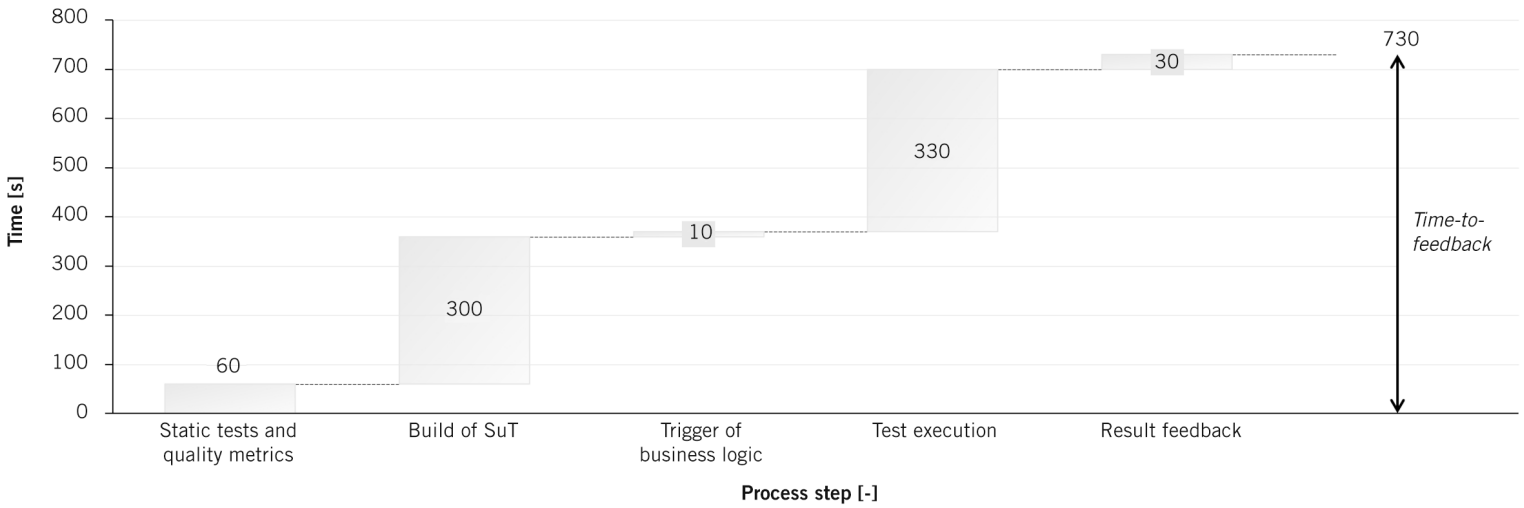

FIGURE 3 Composition of time-to-feedback (© Mercedes-AMG GmbH)

is very close to the ideal state of platform usage, both technically and methodologically. **FIGURE 2** shows the feature development process from code change to the test result. The integrated SiL approach allows the system to quickly provide feedback for the entire ECU.

The implementation or adaptation of any module of the ECU code in a repository represents the beginning of the feedback loop. The code change is first committed in the selected tool for software version management, or the sum of the individual changes from a development branch is merged into a "stable branch". Such events are the triggers for the fully automated process flow – a serial sequence of multiple stages. The corresponding time-to-feedback composition is shown in **FIGURE 3**.

The first stage involves performing static tests and checks against defined quality criteria according to modeling or coding guidelines. This takes approximately 60 sec. In the next stage, the SiL is built as a SuT. Here, the sources of the repository are first used for the respective commit and a virtual ECU (vECU) is built in a special build environment. The vECU is then integrated into the SuT with the relevant plant model and configuration data from the simulation. The generated data is then stored in the build artifact management. The entire process takes about 300 sec.

Once the SiL is successfully built, the business logic of the ecosystem is triggered and an execution task is created in test.guide, which takes about 10 sec. The execution task is then assigned to the appropriate test station, and the defined functional tests are executed with ecu.test in approximately 330 sec according to the plan. The results are bundled and uploaded to test.guide. Finally, the business logic checks to see if the initial commit or merge request from **FIGURE 2** was successful, and then the next steps can begin. This final process step takes another 30 sec. This means that in around 730 sec development teams can receive feedback on code changes at the overall ECU level and initiate subsequent steps such as automated deployment or the execution of additional tests in other environments.

SUMMARY AND OUTLOOK

The test ecosystem for the e-Drive, created in collaboration between Mercedes-AMG and tracetronic, shows the state of the art in modern automotive software development and how this platform can be used to provide fully automated feedback to the development teams for every single code change. A software developer gets easy access to complex software environments. Software functionality can be tested in much less time, which is a significant advantage when developing new products.

In addition, parallelizing test execution in the cloud or distributing individual plans across multiple virtual test benches offers tremendous scalability, creating further potential to reduce time-to-feedback. In the future, the optimum execution time will have to be found by balancing the resource consumption through parallelization and the runtime of execution tasks according to their number and duration.

REFERENCES

[1] Eldh, S. et al.: Test Automation Improvement Model - TAIM 2.0. In: 2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), Porto, Portugal, 2020, pp. 334-337. Online: https://ieeexplore.ieee.org/document/9155990, access: February 12, 2025

[2] Eldh, S.; Andersson, K.; Ermedahl, A.; Wiklund, K.: Towards a Test Automation Improvement Model (TAIM). In: 2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops, Cleveland (OH), USA, 2014, pp. 337-342. Online: https://ieeexplore.ieee.org/document/6825682, access: February 12, 2025

THANKS

The authors would like to thank the members of the development teams involved at tracetronic GmbH and Mercedes-AMG GmbH for their daily commitment and excellent cooperation. Special thanks go to Alexander Beck and Alexander Schneider, developers at Mercedes-AMG, for their support in describing the real use case.