An Efficient Strategy for Testing ADAS on HiL Test Systems with Parallel Condition-based Assessments

Christian Steinhauser[1][a], Maciej Boncler[2][b], Jacob Langner[1][c], Steffen Strebel[2][d] and Eric Sax[1][e]
[1]*FZI Research Center for Information Technology, Karlsruhe, Germany*
[2]*Dr. Ing. h.c. F. Porsche AG, Stuttgart, Germany*

Keywords: Automotive Testing, Verification and Validation, Advanced Driver Assistance Systems, Autonomous Driving, Automated Assessment, Test Strategy, Test Process.

Abstract: On the way to full automation the number of Advanced Driver Assistance Systems (ADAS) and the system's Operational Design Domain (ODD) increase. This challenges today's prevalent requirement-based testing paradigm in the automotive industry, as for each requirement at least one test is derived. While virtual testing offers scalability for large-scale testing, hardware integration-testing has to be performed under real-time constraints. A significant part of the verification on the target hardware is performed on Hardware-in-the-Loop (HiL) test systems. With the limited number of available HiL systems and their execution being bound to real-time constraints, test time becomes a precious resource. In this work we demonstrate a novel test strategy, that unites today's requirement-based test process with new concepts for more efficient HiL testing. Maintaining traceability throughout the development process is the main goal. The tests are split into stimuli and evaluation, where only the stimuli are executed on the HiL. This enables parallel assessment of multiple functionalities in one test execution. The concept has been implemented in a productive HiL environment at a German car manufacturer and the evaluation shows benefits in test coverage, as well as reduced test runtime. Moreover, it enables scenario based testing of Highly Automated Driving.

1 INTRODUCTION

Autonomous driving and electrification are the most significant trends in the automotive industry (BMW Group, 2020) (Daimler AG, 2018) (Volkswagen AG, 2020). On the way to fully autonomous driving, Advanced Driver Assistance Systems (ADAS) represent an important milestone. By gradually integrating more and more ADAS, such as adaptive cruise control (ACC) and automated lane keep assist (ALKA) the vehicle is able to support the driver in longitudinal and lateral driving tasks. Ensuring safety and correct functionality of those systems is a challenging task. For example, up to 150 sensors (Porsche Engineering Magazin, 2018) are constantly monitoring the vehicle's internal state and the surrounding traffic.

Today's vehicles can consist of more than 100 Electronic Control Units (ECU) (Jakobson, 2019). Therefore, ADAS have to be tested in a systematic way. At the integration level the correct functionality of the ADAS are commonly verified in Hardware-in-the-Loop (HiL) test systems as well as test drives in prototype cars. Over the last decade, a requirement-based test strategy has evolved within the development process of ADAS. System requirements describe the system's behaviour within its Operational Design Domain (ODD) and have to be fulfilled by the implementation of the system. In order to examine the fulfilment of all requirements, a test specification is derived for each ADAS which is then implemented in systematic test cases. For deriving the test specification, the following principles are used: 'no test without a requirement' and 'at least one test for each requirement' (Sax, 2008). Defined tests are executed in a systematic scheme. For each test case there is a fixed set of preconditions, a fixed set of input stimuli and a corresponding expected output. After each test step the system's reaction to the stimuli is compared with the expected result. For traceability purposes all test cases within the test specification have to be evaluated in each iteration of the development process. Since HiL test systems are bound to real time due to the usage of real ECUs, time is a strictly limited resource when executing tests. Therefore, evalu-

[a] https://orcid.org/0000-0001-8973-2758
[b] https://orcid.org/0000-0003-4449-7875
[c] https://orcid.org/0000-0003-4210-1056
[d] https://orcid.org/0000-0002-4781-7937
[e] https://orcid.org/0000-0003-2567-2340

Steinhauser, C., Boncler, M., Langner, J., Strebel, S. and Sax, E.
An Efficient Strategy for Testing ADAS on HiL Test Systems with Parallel Condition-based Assessments.
DOI: 10.5220/0011081400003191
In *Proceedings of the 8th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2022)*, pages 391-399
ISBN: 978-989-758-573-9; ISSN: 2184-495X

ating one requirement per test case is a very inefficient way to perform the tests. In order to extract as much information as possible from one test run this work presents a novel testing strategy using parallel assessments. Section 2 presents the theoretical background and the related work. In section 3 we present the concept for integrating parallel assessments in the established testing process. Therefore, requirements for the strategy and implementation are derived. Section 4 compares the new test strategy with the established one. We provide a proof of concept and an evaluation of the tests performed. In section 5 potential future work is described.

2 THEORETICAL BACKGROUND

Testing is an important part of developing ADAS or automotive software in general. To keep the development costs low it is crucial to find errors as soon as possible (Sax, 2008). Therefore, multiple steps of software testing have been established in the automotive development process. Figure 1 depicts the X-in-the-Loop (XiL) methods within the V-Model for automotive development (Sax, 2008). In the early phase of the development process, the basic concept of specific functions is tested with Model-in-the-Loop (MiL). The goal of MIL is to find design errors as early as possible. In the next phase, the implemented code is tested with Software-in-the-Loop (SiL) to identify and correct errors and bugs within the code itself. For testing the proper function of the software integrated on target hardware, Hardware-in-the-Loop (HiL) test systems are used. There are multiple stages of HiL test systems. At the component level, one ECU is tested under real-time conditions. However, at the integration level there are multiple ECUs tested at the same time. For example, on an ADAS-cluster HiL all ECUs that are relevant for ADAS are integrated. The remaining bus communication that is necessary for the ECUs to function properly is simulated.

Alongside HiL testing there are also test drives in prototype vehicles for parameter application. Final validation of the software is conducted in the full vehicle. The underlying concept is known as requirement-based testing and will be explained in the following section.

2.1 Requirement-based Testing

The requirement-based test strategy is a systematic approach of defining the necessary tests, that need to be performed in order to verify the intended functionality of a system. Test cases are systematically de-

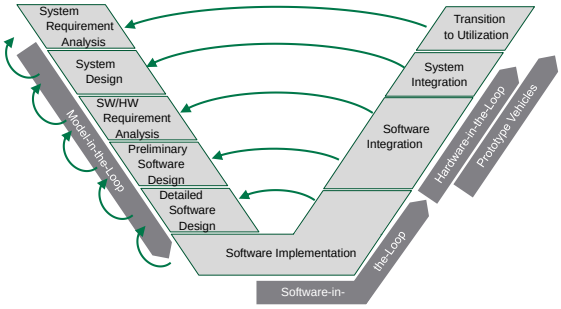

Figure 1: V-Model for automotive development.

rived from the system's requirements which define its behaviour. One test case consists of preconditions and various stimuli contained in tests steps with a corresponding expected result of the system. In each step the response of the System-under-Test (SuT) to the stimuli is compared with the expected result and a test verdict is derived. Typically, the results can either be 'pass' or 'fail'. Test criteria are derived by a multitude of methods like error guessing and equivalence classes as described in the ISO 26262 standard (ISO, 2018). In addition the criteria are supplemented by knowledge and experience. Test cases are collected and organized in a test catalogue, which is also known as the test specification. Within the test specification the distribution onto the test systems like SiL, HiL or full vehicle tests is defined.

2.2 Scenario-based Testing

Scenario-based testing is a promising approach for testing HAD-functions and suggested by several research projects (de Gelder and Paardekooper, 2017) (Eberle et al., 2019) (Menzel et al., 2018) (Wang and Winner, 2019). It was developed as an alternative to the statistical proof of a HAD function's safety, where billions of miles without an accident have to be driven (Wang and Winner, 2019) (Wachenfeld and Winner, 2016). According to Menzel et al. (Menzel et al., 2018) scenarios can be described in three abstraction levels. The first level is called 'functional scenarios'. Functional scenarios represent the most abstract level. They can be described in a linguistic formulation to make a scenario understandable for humans. The second abstraction level is 'logical scenarios', where parameter ranges of state values, that represent the scenario are added. The third level are 'concrete scenarios' which are a specific instantiation of a logical scenario where a specific value of the state values is selected within the parameter range. For testing, completion criteria for each scenario have to be defined. To evaluate the correct functionality of the SuT, parameter variations are suggested. The biggest

challenge for scenario-based testing is the identification or generation of critical scenarios, which are not part of the scenario catalogue. In contrast to the requirement-based approach statistical aggregation of the SuT's behaviour throughout all tested scenarios is suggested.

2.3 Related Work

In this section the main references on which our work is based on are presented. The testing approaches propose parallel testing, that is aimed to evaluate multiple assessments within a scenario. The basic concept of those approaches is adopted and expanded to ensure the usability in an established automotive testing process.

Automotive System Testing by Independent Guarded Assertions

Gustafsson et al. (Gustafsson et al., 2015) presented a parallel testing approach on HiL test systems. It was derived from the declarative testing approach(Triou et al., 2009). The basic idea of independent guarded assertions is the separation of state-changing stimuli and verdict-generating assertions. The so called 'guard condition' continuously evaluates the state of the SuT to make sure that the assessment is performed only if the SuT is in an assessable state. The assessment itself is performed if the 'guard condition' is fulfilled. The assessment is similar to the expected result described in 2.1. This enables the parallel execution of multiple assessments in one single test run. Flemström et al. further enhanced Gustafsson's approach by proposing a process to transform requirements into guarded assertions(Flemström et al., 2018a) (Flemström et al., 2018b).

Automated Function Assessment in Driving Scenarios

King et al. (King et al., 2019) proposed a similar situation-based approach in the context of ADAS. Instead of purely relying on internal bus signals, external information from the simulation environment is taken into account. This enables an evaluation on the behavioural level of the vehicle. The authors further subdivide the activation of an assessment into several logical activation conditions. During a test run, the fulfilment of the activation conditions is constantly monitored by observers. The evaluation itself is described by test conditions, that are performed if the activation conditions are fulfilled. The goal of this approach is to enable automated quality assessment within digital test drives (Otten et al., 2018) (Wohlfahrt et al., 2016), where a priori knowledge of the traffic behaviour in the simulation is not available.

3 CONCEPT

Within the current test strategy a test case consists of three phases: precondition (PRE), action (ACT) and post conditions (POST). In addition there is the expected result (ER), which describes the expected output of the SuT for a given set of stimuli. A simple example is given to illustrate the current test procedure in the context of ADAS. At first, the SuT is initiated in a start-up phase. This includes loading the simulation environment, resetting all fault-codes within the ECU's storage, the start-up of the vehicle within the simulation, gear selection and finally acceleration up to a desired speed. To bring the SuT in an assessable state all conditions contained in PRE have to be fulfilled. After that, the test itself begins. The system's response to a set of stimuli is compared with the expected result, which are executed in the ACT-Phase. A test run ends with a tear down phase, which contains stopping the vehicle, resetting ECUs and the initiation of a test protocol. The signals that are relevant to the SuT are recorded and stored on a server, in case further manual analysis by the developer is needed. Thus, a test case is a mixture of test execution and assessment of the scripted test steps. This means that both the execution and the evaluation are performed on the HiL test system. Hence, only one specific function is tested within a test case. This means that there is a full test run for the evaluation of one single requirement. Considering the time used for the initiation and tear down of the HiL test system this is an inefficient way of using the available HiL resources. To address an increase in efficiency, the goals of the proposed test strategies are:

- Increasing the test coverage extracting more information from every test run performed on HiL test systems
- Reducing usage of HiL resources by splitting test run and evaluation
- Ensuring traceability by systematic mapping of a requirement to a specific assessment

The fulfilment of the first and the second goal will be discussed in section 4. The third goal is shown in sections 3.5.

3.1 Requirements for a Parallel Condition-based Test Strategy

The testing approaches discussed in section 2.3 are both based on separating the assessment of a SuT and the test run itself. This enables parallel execution of multiple assessments within a single test run. Gustafsson introduces an observer for assessing a wider range of possible states. In traditional approaches for example if the car is in reverse gear, it is assessed if the reverse light is on. The reverse light's behaviour in other gears is not tested.

In King's approach a similar observer is implemented. It does not only evaluate the SuT's internal signal but also external ground-truth data derived from the simulation environment. By doing this, evaluating the SuT's behaviour in a priori unknown scenarios is possible. However, from a test management perspective those approaches lack of a structured reporting of the test results. In today's requirement-based approach each test run has to have it's unique ID. This is due to the fact, that for each requirement at least one test is derived. To integrate a parallel testing approach similar to King's and Gustafsson's in the existing test process the following requirements have to be fulfilled:

- Uniqueness:
  Each assessment and each test run shall have a unique ID. Each test run or assessment shall only be defined once.
- Traceability:
  It shall be possible to match each result of an assessment with the corresponding test run. With that the relevant test report with measurement data and meta data is clearly traceable for further analysis of a test result.

Furthermore, unique test IDs are necessary for the integration in the existing test management tool chain, that is based on the IDs. The IDs are, among others, used for the synchronization among different tools or for an automatic test case generation in the test automatisation tool.

3.2 Condition-based Assessment

To extract more information out of a single test run, a parallel, condition based test approach similar to the approaches mentioned in 2.3 is used. In contrast to Gustafsson's and King's approach, we propose an offline assessment approach. Instead of implementing observers, that constantly monitor the signals that are relevant to the SuT, we record those signals. The recorded signals are not limited to internal bus communication, but also external signals provided by the simulation environment. Since the assessments are independent of the stimuli scripts, there has to be a mechanism that only triggers an evaluation if the SuT is in the corresponding state. This is ensured by a logical expression called 'activation condition'. The logical expression is derived from the specification of each SuT. It is composed by signals that are relevant for the specific function and their corresponding values from the ECU or external signals from the simulation environment. As depicted in figure 2 each requirement is evaluated with a corresponding assessment. The assessments are then assigned to a specific test run, which contains a concrete scenario. As shown in figure 2 multiple assessments can be assigned to a single test run and an assessment can be assigned to multiple test runs. This is important for more complex ADAS, which have to be evaluated in different situations or conditions. The evaluation of the expected result is performed by 'test conditions', which are triggered by the activation conditions. The naming of the conditions is adopted from King, due to the same basic principle of situation detection. Furthermore, this principle enables multiple evaluations of the same assessment in different test runs, thus increasing test coverage without additional test runs.

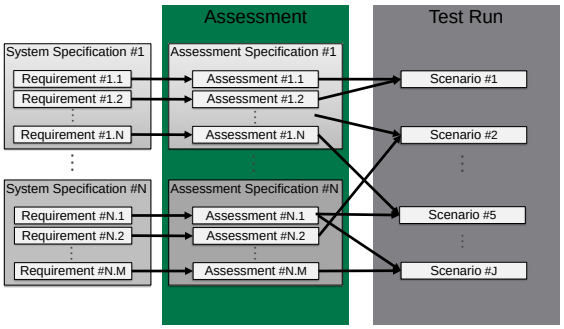

Figure 2: Derivation of Assessments.

3.3 Separation of Test Run and Assessment

In this section, we focus on the required traceability of the test result. In the current test process each test result has to be clearly matched to a specific test run and has to have its unique ID. With that, it is always possible to trace the relevant report with measurement data and analyse the test run, if an error is identified. To accomplish traceability within a parallel test approach, two classes are defined. We call them 'Test Run' and 'Assessment':

Test Run

The class 'Test Run' contains all information regarding the test Run on the HiL test system. This includes a complete scenario description, which contains all actions the SuT and other traffic participants have to perform during simulation. Furthermore, the complete description of the corresponding road topology within the driving simulation is included. Additional information regarding different entities like number of traffic participants, objects and pedestrians and their corresponding behaviour within the Test Run is defined as well. Meta information like signals that have to be recorded for this specific Test Run are also part of the Test Run.

Assessment

Within the class 'Assessment' the assessment of the SuT's behaviour is defined. This includes the formal description in which conditions the system's behaviour can be examined. Therefore, all relevant internal bus signals and external information from the simulation environment have to be listed including threshold values. With that, the activation condition for the assessment can be derived. Furthermore, the signals and corresponding signal ranges for the system's intended behaviour have to be specified within the Assessment class.

Each Assessment corresponds to a specific system requirement. To map the Assessments to the Test Runs in which they shall be evaluated the Test Run-Assessment specification layout is described in section 3.5.

Allocation of Test Run and Assessment

As it can be seen in figure 3, after the simulation is completed the traces are transmitted to an analysis server, which Runs the assessment scripts. This causes a reduction of HiL Run time, since the evaluation is performed on a separate system. This is due to the fact that the HiL test system is able to perform the next simulation while the assessment scripts are still evaluated on the analysis server. In addition, extensive video analysis for example for the evaluation of the Human Machine Interface (HMI) in the instrument cluster is not performed on the test system itself, thus reducing the Runtime needed for a test Run.

3.4 Test Strategy

The combination of the Condition-based Assessment (section 3.2) with the separation of Test Run and Assessment (section 3.3) is capable of optimizing the usage of the HiL test system in two ways. Depending on the goal of the optimization, either the number of test results per Assessment or the test capacity per development cycle can be increased.

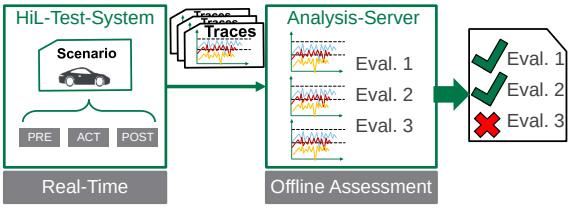

Figure 3: Seperation Test Run and Assessment.

Strategy for Increasing the Number of Assessments per Requirement

To increase the number of evaluated requirements, the corresponding assessment can be allocated to multiple Test Runs. If the assessment is evaluated on Test Runs, which contain a set of stimuli that differs from the initial Test Run, the test coverage is increased. In the previous requirement-based strategy there was a clear one-to-one relationship between a Test Run and its evaluation. However, by separating the assessment and Test Run it is possible to perform one assessment in multiple Test Runs, as shown in the bottom right in figure 2. This strategy allows an increase in test results while the number of Test Runs remains the same.

Strategy for Increasing Testing Capacity

In the requirement-based strategy explained in section 2.1 there needs to be one test per requirement. However, it is possible that multiple systems need the same sequence in the ACT part to evaluate their correct behaviour. This leads to redundant Test Runs in a development cycle. Since HiL test systems have limited capacity due to real-time constraints, it is favourable to eliminate redundant Test Runs while maintaining at least one assessment per requirement. Therefore, multiple assessments can be allocated to one specific Test Run as shown in the top right in figure 2. The strategy eliminates redundant Test Runs by identifying similar sequences of test steps. The identified sequences of test steps are merged into a single Test Run.

3.5 Test Specification Layout

For each driving function, for example LKA or ACC, a test specification is defined. The test specification for the separation of Test Run and Assessment is divided in two chapters: Chapter 1 contains all Test Runs, that are relevant to the specific SuT. Each Test

Run is represented as a module. Modules describe entities within a chapter. A module has its unique ID. The second chapter contains all Assessments. In contrast to the first chapter, each Assessment is a chapter by itself with its corresponding ID. The Assessment's definitions are denoted in the annotations of the chapter. For consistent and traceable mapping of each Assessment to a Test Run modules are introduced within the chapter. Each module has its own unique ID. The module's IDs are composed of the Test Run's ID and the Assessment's ID. Therefore, the Assessment is linked to the Test Run. The layout can be pictured as follows:

- Chapter Test Run
  - Module: Test Run ID 1
  - Module: Test Run ID 2
  - ...
  - Module: Test Run ID *N*
- Chapter Assessment
  - Assessment ID 1
    - Module: Assessment ID 1 Test Run ID 1
    - Module: Assessment ID 1 Test Run ID 2
    - ...
    - Module: Assessment ID 1 Test Run ID *M*
  - Assessment ID 2
    - Module: Assessment ID 2 Test Run ID 3
    - Module: Assessment ID 2 Test Run ID 8
  - Assessment ID *N*
    - Module: Assessment ID *N* Test Run ID 4
    - Module: Assessment ID *N* Test Run ID *M*

With the unique modules for each assignment the test automation tool is able to generate test scripts for each Assessment. With the assignment, triggering of an assessment-request in a specific Test Run can be accomplished automatically.

With this concept, each Test Run and Assessment can be maintained and updated independently. This increases the manageability by test engineers. Moreover, the Separation of Test Run and Assessment enables the integration of scenario completion criteria within the scenario-based test approach. Specific completion criteria can be added as new Assessments, enabling traceability, comparability and reusability throughout different scenarios.

4 APPLICATION AND EVALUATION

To provide a proof of concept for the proposed strategy from section 3 a test specification of a prototype

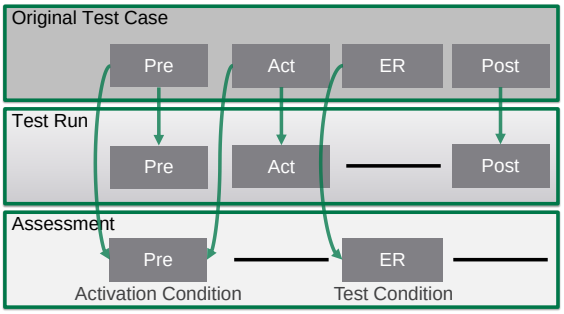

Figure 4: Decomposition in Test Run and Assessment.

ADAS is transformed into the novel condition-based strategy. The prototype ADAS is already integrated in a HiL test system and is tested in the current development process. The initial test specification, HiL test system, SuT and implementation of the test cases are all real-life examples. For the proof of concept, a set of 15 test cases is chosen. It reflects the content of the complete test specification, but not the number of test cases of the ADAS. Therefore, the set contains basic tests, such as 'function availability', as well as tests for complex requirements, like deployment of intervention mechanisms. The aim is to ensure the transferability of the proof-of-concept's results to a real testing process in the automotive industry.

4.1 Proof of Concept

The original test specification is analysed and redesigned according to the separation of Test Run and Assessment described in section 3.3. Each original test case refers to a specific requirement, thus one test run and one Assessment is derived. The combined content corresponds directly to the original testcase. The decomposition of the original test specification is visualized in figure 4. It depicts that the PRE, ACT and POST parts for the specification of the test run can be directly inherited from the original test case. In the test run there is no function-relevant expected result because all assessments will be performed offline with the traces recorded from the test run. For future implementations, the correct execution of the test run or the proper storage of all relevant traces can be examined within test runs. Thus, their result should only be seen as verification of a correct test execution. For example, if there were any exceptions by running the test script itself or if there was an error while saving files. The most significant changes in the test specification for the separation of Test Run and Assessment are necessary for the definition of activation conditions within the Assessments. Figure 4 depicts that the activation conditions are derived from the PRE and ACT parts of the original test case. In

Table 1: Evaluation of test coverage and runtime.

	Original Test Case	Condition-Based Assessment
Run Time	47 min 09 s	46 min 44 s
Number of Evaluations	15	77

addition, the activation condition is supplemented by information from the requirements of the SuT as described in section 3.2. The fulfilment of completeness of activation conditions is an especially challenging part (Gustafsson et al., 2015). Therefore, additional information from analysing the SuT's intended behaviour is used to ensure proper activation. The aim is to find the relevant section within the recorded traces from the test run to perform the evaluation part of the assessment. The evaluation part of the Assessment itself can be directly inherited from the former Expected Result (ER). According to the layout presented in section 3.5, a total number of 225 modules are defined in the test specification.

Evaluation of the Run Time

To evaluate the difference in test run time, there are as few changes as possible done in the original test case implementation to extract a corresponding Test run. The ER part is removed from the Test run's implementation and moved to the Assessment. With the Test runs and Assessments implemented, two test suits are created. A test suite is a group of tests that should be successively, automatically conducted by the test automation tool. The first test suite contains the 15 original test cases. The second one contains the 15 Test runs and 225 Modules. We did not conduct any preselection of Assessments for Test runs and executed all defined modules implementing every Assessment for every Test run.

The test suites are executed one after another on a ADAS-cluster HiL test system and their execution time is tracked. Another chosen metric is the number of evaluations. It shows how often the activation condition of the Assessment is fulfilled. Those are then denoted as significant test results. Otherwise, the test result of a module remains 'open' and does not bring any further information about the fulfilment of the respective requirements.

The results in table 1 show, that the Separation of Test Run and Assessment is beneficial according to both of the chosen metrics.

However, the improvement in the test suite's execution time, of 1%, should be further evaluated with a bigger test suite. The authors expected a bigger improvement due to the parallel computing. The observed difference might be a result of the run time fluctuations in the test system. Another point worth to mention is that, although only 77 modules could be successfully evaluated, all 225 modules were proceeded during the run of the test suit. Elimination of these unproductive runs of the modules might lead to further time improvements.

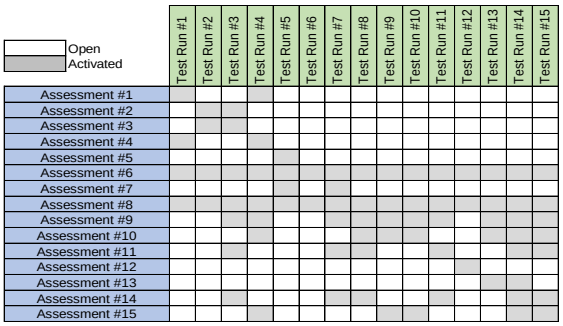

Figure 5: Activation matrix.

Evaluation of Test Coverage

The increase of the test coverage is on the other hand undisputable. There is an increase of 433% in the amount of the significant test results observed. The distribution of the additional evaluation can be seen in figure 5 which we call the 'activation matrix'. The columns of the activation matrix correspond to the Test runs and the Assessments are shown in the rows. Modules are depicted as single cells of the matrix. If the cell $x_{i,j}$ is marked as 'activated', the activation condition of the Assessment of the row i has been fulfilled during the test run of the Test run in the column j and the result for the module is successfully determined. The results that would have been obtained with the use of an original test strategy are to be seen on the main diagonal of the activation matrix. As it to be seen in the activation matrix, the additional results are unevenly distributed over the Assessments and Test runs. Test runs that were initially aimed to prove complex requirements usually have a high number of activated Assessments. On the other hand, the Assessments for the complex requirements are only rarely activated outside of their corresponding Test run. An example for such test cases are #4, #7 and #13. The Assessments for basic requirements, like 'function activation' or 'function passivation' tests, are activated most frequently, see Assessments #6 and #8. Those two are actually activated in all of the examined Test runs. Worth to mention is that in Test run #6 only those two Assessments are activated. It can be assumed, that this test case brings very little added value to the test process and might be neglected to optimize the whole process.

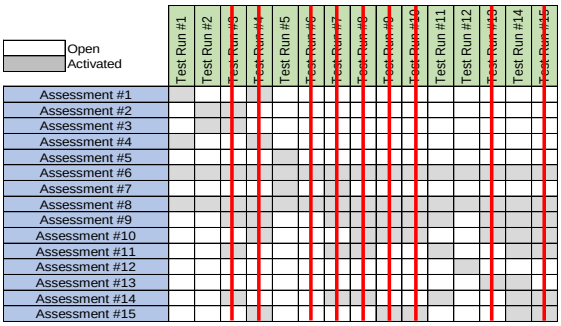

Figure 6: Optimization of required Test Runs.

Optimizing the Number of Test Runs

The proposed strategy of separating the Test Run and Assessment can also be used to optimize the number of Test Runs which are performed on the HiL test system. For optimizing the number of Test Runs each Assessment should be evaluated at least one time to ensure test coverage of 100% for each requirement. Figure 5 shows that with this approach, each assessment is evaluated multiple times. This is due to the fact that the Test Runs performed are similar to each other, thus leading to activations even if they were not intended in the original test case. Figure 6 shows that in this case 9 of 15 Test Runs can be eliminated, while still evaluating each assessment. This leads to a reduction of required test capacity of 60%. The exact achievable improvements in overall test capacity depend on the initial definition of test specification as well as the whole testing strategy. Different results can be achieved when data from a single Test run is used by Assessments defined for a single ADAS or multiple functions in the vehicle. Furthermore, the possibility of combining tests depends on available HiL test systems and their design.

Comparison with the Traditional Test Strategy

Compared to the traditional requirement-based test strategy, the major drawback of the proposed strategy is an increased effort for designing the activation condition of each assessment. However, if the requirements of the SuT are available to the test engineer, it is possible to derive the activation condition. The benefit of the proposed test strategy is a more flexible and efficient overall test process. It can either be used to increase the test coverage or to increase test capacity while at least maintaining full coverage of each requirement. From the test management view, the novel strategy establishes a clear one-to-one relationship from a requirement to an assessment. This leads to a significant increase on traceability, since each assessment with its corresponding test run can be identified clearly.

4.2 Application on Productive Test Specifications

To further investigate the benefit of the proposed strategy, it is used to optimize test specifications that are used on a productive HiL system for evaluating prototype ADAS functions. In addition, we checked if the concept can be applied to the body domain, where for example door locking mechanisms or crash reactions are tested. Finally, we applied the strategy to a HiL system, where functions of the drivetrain domain are tested. The initial test specifications are developed with the traditional requirement-based strategy. Since the test coverage provided by the initial test specification is already sufficient, the optimization of the test capacity is performed. Table 2 shows the potential for reducing the required test capacity for each specification. On the first column, the initial required number of tests is presented. The second column shows the required tests runs, if the novel strategy is applied. The third row shows the reduction of test runs in percent. The Authors note, that the number of test cases presented in table 2 only represent a fraction of the overall tests efforts conducted by the manufacturer.

The results show that the potential of reducing the required test capacity is highly dependent on the initial definitions of the test cases. The overall improvement is at $21,5\%$ in required test runs. However, the potential of reducing the required test capacity differs between the specifications. For example, compared with the specification of ADAS there is a higher potential than within the drivetrain specification (see row two and three in table 2). To further investigate the overall potential of reduction, more test cases for every specification have to be considered. The achievable result also depends on the original test specification that is to be adopted. The optimum might have already been achieved by aggregation of requirements in the single test case. In that case, the proposed method does not bring significant improvement in the overall run time. However, it significantly improves the traceability within the test process, as each requirement might be evaluated in an independent module, instead of test result aggregation in a classic test case.

5 CONCLUSION AND OUTLOOK

The presented strategy of separating the Test Run and Assessment has been proven to optimize the usage of available HiL test system resources. With this novel strategy for HiL testing, either a significant increase in test coverage or a reduction in required test run time

Table 2: Results of the evaluation.

	Original Test Cases	Optimized Test Runs	Reduction of Test Runs
Specification ADAS	205	182	11.2%
Specification Drivetrain	46	19	41.3%
Specification Body Domain	511	397	22.3%
Overall	762	598	21.5%

can be achieved. Due to the separation of Test Run and Assessment and parallel computing, minor reduction of runtime of a test suite is possible. The proposed strategy has been implemented and evaluated on a productive HiL system at a German car manufacturer. The authors see potential for further run time improvements. The strategy optimizes the test capacity by finding redundant test runs which are a result of the original requirement-based development and test process. The authors see the possibility to find an optimized set of test runs by analysing the test steps that are needed to test each specific requirement. This could lead to an optimal test run of a specific set of requirements. Therefore, optimization methods for the automated reduction of test run time are to be further investigated. In addition, test management methods like finding the compromise between the necessary run time at the HiL test system and achieved test coverage are subjects of further work of the authors.

REFERENCES

BMW Group (2020). Investor presentation.

Daimler AG (2018). Corporate presentation.

de Gelder, E. and Paardekooper, J.-P. (2017). Assessment of automated driving systems using real-life scenarios. In *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE.

Eberle, U., Hallerbach, S., Mannale, R., Kramer, B., Neurohr, C., Steimle, M., and Amersbach, C. (2019). Pegasus final event and symposium - the pegasus method for safeguarding automated driving: What else is needed?

Flemström, D., Enoiu, E., Azal, W., Sundmark, D., Gustafsson, T., and Kobetski, A. (2018a). From natural language requirements to passive test cases using guarded assertions. In *2018 IEEE International Conference on Software Quality, Reliability and Security (QRS)*. IEEE.

Flemström, D., Gustafsson, T., and Kobetski, A. (2018b). A case study of interactive development of passive tests. In *Proceedings of the 5th International Workshop on Requirements Engineering and Testing - RET '18*. ACM Press.

Gustafsson, T., Skoglund, M., Kobetski, A., and Sundmark, D. (2015). Automotive system testing by independent guarded assertions. In *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE.

ISO (2018). 26262 Road vehicles – Functional safety.

Jakobson, O. (2019). Vcc complete hil rigs meeting our next generation corebased service oriented architecture, dspace world conference, 2019.

King, C., Ries, L., Kober, C., Wohlfahrt, C., and Sax, E. (2019). Automated function assessment in driving scenarios. In *2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)*. IEEE.

Menzel, T., Bagschik, G., and Maurer, M. (2018). Scenarios for development, test and validation of automated vehicles.

Otten, S., Bach, J., Wohlfahrt, C., King, C., Lier, J., Schmid, H., Schmerler, S., and Sax, E. (2018). Automated assessment and evaluation of digital test drives. In Zachäus, C., Müller, B., and Meyer, G., editors, *Advanced Microsystems for Automotive Applications 2017*, pages 189–199, Cham. Springer International Publishing.

Porsche Engineering Magazin (2018). "testing", 01/2018.

Sax, E. (2008). *Automatisiertes Testen Eingebetteter Systeme in der Automobilindustrie*. Hanser Verlag, München.

Triou, E., Abbas, Z., and Kothapalle, S. (2009). Declarative testing: A paradigm for testing software applications. In *2009 Sixth International Conference on Information Technology: New Generations*. IEEE.

Volkswagen AG (2020). Unicredit kepler cheuvreuxgerman corporate conference. Presentation.

Wachenfeld, W. and Winner, H. (2016). The new role of road testing for the safety validation of automated vehicles. In *Automated Driving*, pages 419–435. Springer International Publishing.

Wang, C. and Winner, H. (2019). Overcoming challenges of validation automated driving and identification of critical scenarios. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE.

Wohlfahrt, C., Schmerler, S., Schmid, H., and Lier, J. (2016). Von systematischer absicherung zur digitalen erprobungsfahrt. 6. Fachkonferenz AUTOTEST, Stuttgart.