A Method and a Set of Tools for Automated Evaluation and Aggregation of Test Data

Tim Martini[1]✉, André Kempf[2], Michael Auerbach[1], André Casal Kulzer[3]

[1]Labor Fahrzeugantrieb,
Hochschule Esslingen,
Kanalstraße 33, 73728 Esslingen a.N.
{tim.martini, michael.auerbach}@hs-esslingen.de

[2]Entwicklung Betriebsstrategie,
Mercedes-Benz AG,
Mercedesstraße 120, 70372 Stuttgart
andre.a.kempf@mercedes-benz.com

[3]Lehrstuhl Fahrzeugantriebssysteme,
IFS Universität Stuttgart,
Pfaffenwaldring 12, 70569 Stuttgart
andre.kulzer@ifs.uni-stuttgart.de

Abstract: Faced with the desired shift-left concept in automotive development, it is essential to automate and digitise the development process. It is common practice in automotive software engineering to implement DevOps processes, such as continuous testing. In the context of the off-car development method, extended data from test drives and in-service operations are utilised as stimuli for continuous off-car robustness testing. The robustness testing complements regression testing by using the exploratory nature of road tests. This allows the software to be analysed in a real-world operational context at an early stage in the development process. The implementation of automated road testing in simulations and on test benches enables the execution of a significant number of measurements, the evaluation of which can prove to be a challenge. In addition to the assessment of functional tests, other characteristics, such as the quantity of the change of states and the availability of electrical energy, are also relevant for behaviour of a performance-hybrid vehicle in real-world conditions. This work presents a method and tools for automated evaluation and aggregation of data obtained from off-car testing. Object-oriented approaches and interactive dashboards are employed for data and code management, as well as for visualisation purposes. An efficient analysis is enabled, supporting data-driven development. The use of software-in-the-loop simulations and the implementation of analysis classes ensure a consistent evaluation throughout the product development process in different development environments.

1 Research Motivation and the Challenge of Off-Car Testing

The complex nature of control systems in modern vehicles, which involve a multitude of actors and interactions, requires a comprehensive and intensive testing process. As it is not feasible to test all operational conditions, the software system’s functionality can only be proven within a covered range [Wo18, My12]. Intensive testing of the system reduces the likelihood of errors occurring during in-service operation. Road testing, which is a highly resource-demanding process, is of pivotal importance. In addition to functional tests, the vehicle's behaviour is tested subjectively and exploratively under varying boundary conditions. However, due to shorter development cycles and limited availability of test vehicles, there is diminishing time available for road testing. Therefore, it is essential that the vehicle software reaches a high level of maturity before the road testing phase in order to ensure an efficient use of time. The

goal is to continuously increase the software's maturity throughout the product development process (PDP). [Ma22]

In order to guarantee this, the continuous integration, deployment, and testing processes, as defined by the DevOps methodology [HPJ20], will be employed. A new software or calibration dataset is tested against defined requirements. To extend the test coverage and subject the evolving system to exploratory analysis and error investigations at an early stage of its development, off-car robustness testing is implemented as part of the off-car development method (OCM) in the PDP [Ma22].

The OCM employs data derived from both test drives and in-service operation to emulate the conditions of road tests in simulations and on test benches. Published works on this topic employ Markov chains to aggregate speed measurement reports from test drives into compressed, equivalent advance profiles [FIG20]. By emulating road tests, the system behaviour can be evaluated in a non-road environment, subjected to comprehensive analysis, and adapted to real-world operational requirements. The off-car test (OCT) used in this paper is a 2000-kilometre road test comprising of 100 individual trips varying in length. The total distance of OCTs and, consequently, the number of individual trips can be scaled as required. A probabilistic approach to the configuration of the trips in question ensures that a different OCT is conducted if the software or calibration dataset is modified.

The accumulation of data through the fast-moving nature of continuous off-car testing presents a challenge for analysis. A time series based interpretation of results requires a significant amount of resources for this quantity of data. To capture insights and identify potential avenues for enhancing system behaviour, it is essential to employ efficient methods.

This work proposes a solution to the challenge of off-car testing, namely a method and tools for automated data evaluation and aggregation. In order to automate and standardise the evaluation of data across the PDP, an object-oriented approach to data and code management has been integrated into the tool chain (see section 2). The tools have been designed with the specific purpose of aggregating vehicle‘s internal measurement signals, calibration data, and calculation instructions for the usage of evaluation and visualisation. In accordance with user predefined calculation instructions, the results of the OCTs are presented in the form of interactive dashboards, accompanied by the calculation of key figures, as described in section 3. The paper concludes with a comparison of two calibration datasets. The examples are demonstrated through software-in-the-loop (SiL) simulations [SZ16, Sy24] and a control system for the change from electric to hybrid drive of a plug-in performance hybrid vehicle.

2 Consistent Evaluation and Aggregation: Data and Code Management

The reusability of calculation procedures and the aggregation with data are of fundamental importance for efficient and error-free analysis of the evolving vehicle software. In order to implement these principles, object-oriented programming (OOP)

in Python [St18] is employed. One of the core concepts of OOP is the definition of classes, which serve as blueprints for instances. The classes designed for the OCM describe methods for the evaluation and visualisation of measurements. Measurement-specific data are transferred to an instance as attributes, which may be single or multi-measurement data. The following pseudo-code outlines the structure and the functionalities of an example analysis class utilised in this paper. [St18]

Pseudo-Code: Analysis Class for Automated Evaluation and Visualisation of Electric Drive Characteristics
Input: Measurements in diverse formats from a multitude of development environments
Return: Measurement-specific instance with the class functions as implemented methods

```
1: Class Analysis:
2:         Function Initialise Instance():
3:                 Assign measurement(s) to instance
4:         Function Read Time Series():
5:                 Process diverse data formats
6:                 Save defined signals in a standardised data format
7:         Function Evaluate Electric Drive Characteristics():
8:                 Calculate the number of starts of the internal combustion engine per trip
9:                 Calculate the minimum value of the state of charge of the traction battery per trip
10:        Function Visualise Electric Drive Characteristics():
11:                Plot the quantity of starts of the internal combustion engine as a histogram
12:                Plot the minimum values of the state of charge of the traction battery as a histogram
13:        Function Read Calibration():
14:                Save calibration parameters of selected functions of the vehicle software
15:        …
```

When an OCT is conducted as a SiL simulation, an analysis instance is generated for each individual measurement. The analysis class may be calculated in simulations or, alternatively, posterior instances may be assigned to measurements. In addition to the storage of time series data from the control device, signals and key figures (e.g., lines 8 and 9 of the analysis class) are calculated to analyse the system. Furthermore, calibration parameters of selected functions are read and assigned to an instance. A single result file is created, comprising measurement-specific instances. These instances contain the time series data of the control devices, the calculated signals and key figures, the employed calibration data, and the methods utilised for the evaluation and visualisation of the measurement data and calibration parameters. The file is stored in Pickle format [Py24] for archiving.

In addition to its utility in SiL environments, OOP has the potential to enhance efficiency and reduce error probability in a multitude of development environments. The analysis class may be employed in a variety of additional development applications. The processing of diverse data formats (see pseudo-code, line 5) and the capacity for posterior assignment allows measurements to be generates from any environment as instances. The evaluation of simulations, test benches and vehicle measurements remains consistent. In the OCM framework, the above analysis class is also employed to analyse measurements documented within a big data infrastructure [IS23]. Alongside testing, the signals and key figures of OCTs can be used to determine target functions and facilitate automated calibration (see [Ma22]). Figure 1 provides an overview of the various development applications of the analysis class in the PDP.

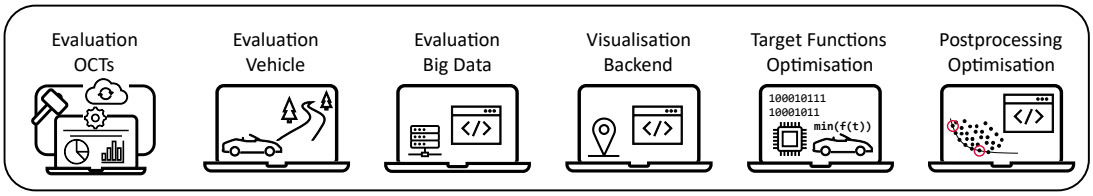

Figure 1: Overview of various development applications and environments of the analysis class in the PDP

Following the successful execution of an OCT, the only prerequisites for analysis are the Pickle file and a Python installation. All components for visualisation in interactive dashboards and the calculation of key figures are contained in the result file. This enables efficient and error-free analysis of the software under development.

3 Visualisation and Key Figures for Data-Driven Decision Making

Analysing large amounts of complex and fast-moving data, such the output from an OCT during vehicle software development, is a challenge. Visual preparation is a key to manageability and the integration of dashboards enables interaction with the data. This supports the search for errors and their causes while allowing for data-based software adjustments and calibration. [De24]

Using a set of analysis class methods, it is feasible to create a dashboard that is customised to a specific function by integrating the user predefined components, such as the histograms described in pseudo-code lines 11 and 12. A summarised interactive visual representation of data enables efficient comprehension of system behaviour and eases the investigation of anomalies [De24]. Figure 2 schematically illustrates the structure of a dashboard designed for off-car robustness testing of the control system that is switching between electric and hybrid drive. The dashboards are built using the Python library Dash [Pl24a]. The data is based on 100 individual trips with a total distance of 2,000 kilometres, which represents the intended real-world operation of the system under development.

The dashboard shows a list of measurements performed with information on each individual trip. The table list of measurements offers the possibility of filtering, for example, by the dynamics of a trip in the form of the relative positive acceleration index (RPA, [Tu15]). Furthermore, the dashboard updates the displayed graphs and calculations according to the filtered trip selection. In addition to histograms and key figures, the diagrams also display the associated calibration maps and corresponding operating points.

Time series data can be analysed by employing a predefined method of the instance using the Python library ASAMMDF [Hr23]. Measurements can be exported as MDF files [AS24] and analysed with common tools. The export can be initiated from the dashboard via a download button or through command line. Alternatively, interactive time series plots for selected signals can be generated. Using the Python library Plotly [Pl24b], the analysis class can deliver plots as an HTML file [Wo24].

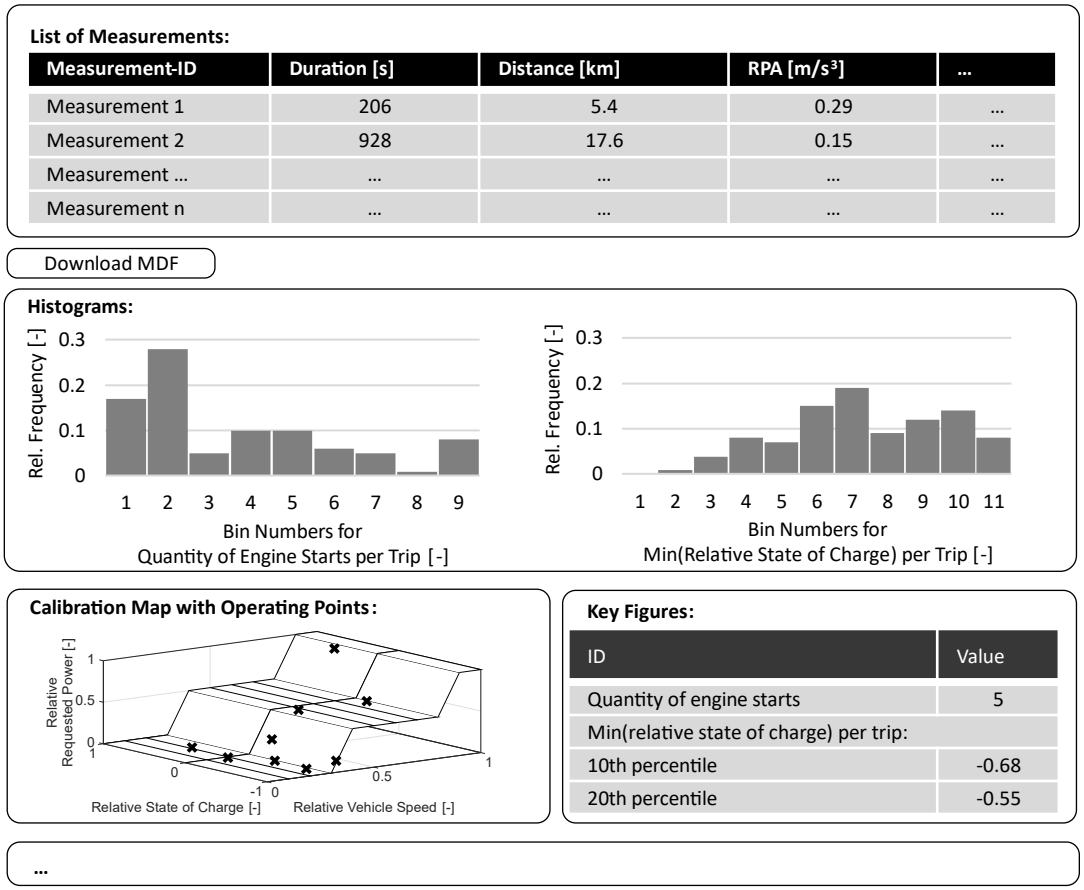

Figure 2: Schematic dashboard for analysing the control system for the change from electric to hybrid drive

The previously mentioned open-source Python libraries along with the Pickle module provide a platform for the analysis of a multitude of simulations and test bench measurements. Upon generation of new software or datasets, an analysis of the implemented modifications can be conducted based on the intended real-world operation within a few hours. To illustrate the power of the aforementioned method and tools, the results of an example variation of calibration data is presented in the following section.

Demonstrating Method and Tools: Results of an Example Off-Car-Test

The calibration of the control system in charge of changing drive states from electric to hybrid has a significant impact on the subjective perception of vehicle behaviour in performance hybrids. The degree of freedom of operation permitted by the drive train allows for a multitude of variants to be implemented. The calibration determines the utilisation of electrical energy stored in the traction battery and the extent of the change of states. The following section presents a comparison of two example calibration datasets and examines the resulting system behaviour as in-service operation. The example describes the results of an OCT for the charge-sustaining operation [UNR154] of a performance hybrid vehicle. The modified calibration map is shown in terms of the requested power as function of the vehicle speed and the state of charge of the traction battery.

Both variants concentrate on binary operation, divided into two speed ranges. In order to prevent the internal combustion engine from operating at an inefficient partial load, the low speed range is usually run in an all-electric mode. At higher speeds, a hybrid operation is employed to facilitate recovery of electrical energy through load point shifting. Moreover, the avoidance of partial load operation is extended by boundary conditions, namely performance potential and comfort. The differentiation between the variants is based on the targeted separation speed for the binary mode ranges. Calibration dataset A is designed with performance in mind, with the objective of maintaining sufficient electric energy for accelerations in the traction battery. This is achieved by starting the internal combustion engine at speeds of 30 kilometres per hour. Calibration dataset B is designed to provide a more comfortable system behaviour with less frequent changes of states than dataset A. This requires a system to reduce the number of internal combustion engine start and stop events in urban areas. The targeted engine start-up speed is therefore defined as 50 kilometres per hour.

In order to quantify the comfort of the system, the number of starts of the internal combustion engine is analysed. The key figure for the performance potential is described by the 10th percentile ($x_{0.1}$) of the lowest occurring state of charge of the traction battery per trip, thereby representing the minimum electrical energy that can be made available in 90 percent of cases. To calculate these values, OCTs of each variant are simulated in SiL environments. Based on the method discussed above and the tools for automated evaluation and aggregation of data, a pickle file is available after a few hours of simulation time.

Once the pickle file and a Python installation have been prepared, the results can be analysed. Figure 3 illustrates components of the analysis class used for the creation of dashboards. The left-hand side of the plot displays the comfort quantification, represented in a histogram showing the specific number of start-ups of the internal combustion engine per trip. The histogram on the right-hand side of the plot represents the lowest charge states of the traction battery per trip, which quantifies the performance potential. The state of charge is normalised to an example charge-sustaining set point for the purpose of comparing variants in this paper. The class centres for categorising the number of starts are arranged in an arithmetic sequence with a difference of two, ranging from zero to 20 starts. The relative state of charge exhibits a class step size of 0.1 within an interval of minus one to zero.

Comparing calibration dataset A to B, there is a shift towards fewer starts per trip and an increase in trips where the internal combustion engine is not started at all. The proportion of time spent in all-electric driving mode is 2.25 times higher with calibration dataset B compared to dataset A (see the left plot, bin zero). Overall, the number of starts reduces by 22 percent. An increase in the proportion of electric driving results in a greater demand for electric energy from the traction battery, which has two consequences in real-world operations. Firstly, the 10th percentile drops to -0.68 with calibration dataset B compared to the calibration dataset A value of -0.55 for the minimum relative state of charge. Secondly, in dataset B, the value of -0.55 corresponds to the 20th percentile ($x_{0.2}$). Therefore, with dataset B, the number of trips with the same minimum available electric energy is 10 percent lower than with dataset A.

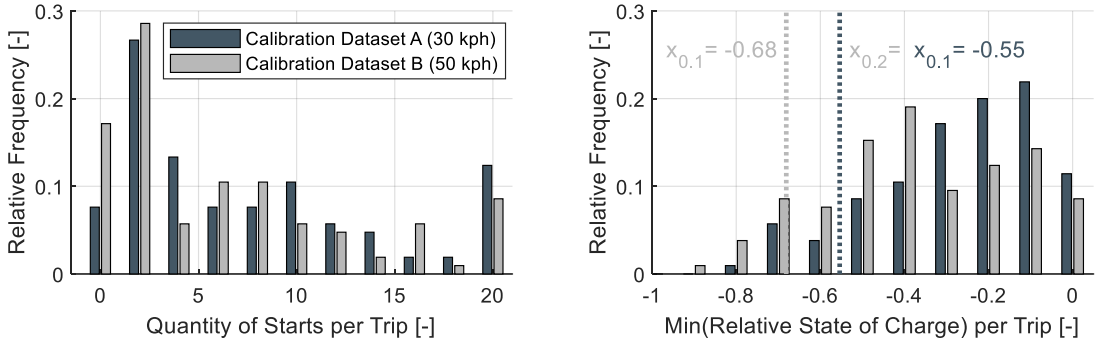

Figure 3: Quantification of the characteristics of the dataset variants in real-world operation – comfort (left) and performance potential (right)

This initial assessment serves to comprehend the potential advantages and disadvantages of the proposed modifications at a glance. A plan of action for subsequent steps can be devised based on the quantified characteristics. In this example, calibration dataset B results in a reduction in performance potential in comparison with dataset A. However, it also demonstrates the potential to reduce the quantity of change of states. To investigate a potential trade-off between comfort and performance, it is recommended to employ the interactive dashboards for a more comprehensive analysis. Based on the insights gained, modifications can be made to the calibration data as appropriate.

The off-car testing equivalent to in-service operations enables the quantification of potential advantages and disadvantages of modifications prior to costly road tests. The aggregation of time series data, key figures, and evaluation methods, along with the aggregation of multiple instances within a Pickle file, facilitates an efficient analysis of data produced. The insights thus obtained can then be employed to inform decision-making processes. In the event that a tested variant fails to meet the required specifications, the calibration data can be adjusted in subsequent iterations without the need for road tests. Following a successful off-car demonstration, the calibration data can be deployed and further refined on development vehicles.

4 Conclusion

The method and tools presented in this paper facilitate the efficient analysis and data-driven decision-making of new software versions and calibration datasets. In order to ensure an automated and consistent evaluation, an object-oriented approach is utilised. An analysis class has been designed to provide blueprints for the evaluation and visualisation of measurements taken in a variety of development environments. Following the successful completion of an off-car test, the specific instances of individual trips, along with time series data, key figures, calibration parameters, and predefined methods are aggregated in a result file. The data is then analysed using interactive dashboards.

In order to utilise the exploratory nature of road testing at an early stage of the product development process, an off-car imitation of road testing is employed as part of the OCM. The relocation of test drives to simulations and test benches allows a multitude

of measurements to be conducted in an efficient manner. However, the fast moving and rapid accumulation of data presents a challenge when off-car testing is conducted on a regular basis. To address this issue, the method and the aforementioned tools presented in this work are integrated into the OCM. Consequently, a few hours after a modification is made, key figures for the system behaviour in an in-service-equivalent operation can be provided and the system can be analysed. An example variant comparison demonstrates the power of the employed tools for efficiently analysing system behaviour. To quantify performance potential and comfort, key figures are calculated based on the state of charge of the traction battery and the frequency of internal combustion engine starts.

Further development of the OCM will entail extending the application of the analysis class to automated calibration. Additionally, key figures of off-car testing will be employed in the determination of target functions.

The presented work on automated data evaluation and aggregation plays a pivotal role in the success of automated, data-driven development of automotive software.

References

[AS24] ASAM e. V.: MDF (Measurement Data Format). https://www.asam.net/standards/detail/mdf/, accessed 16 Feb 2024.

[De24] Demirbaga, Ü. et al.: Big Data Analytics. Theory, Techniques, Platforms, and Applications. Springer Nature Switzerland; Imprint Springer, Cham, 2024.

[FIG20] Förster, D.; Inderka, R. B.; Gauterin, F.: Data-Driven Identification of Characteristic Real-Driving Cycles Based on k-Means Clustering and Mixed-Integer Optimization. IEEE Transactions on Vehicular Technology 3/69, pp. 2398–2410, 2020.

[HPJ20] Halstenberg, J.; Pfitzinger, B.; Jestädt, T.: DevOps. Ein Überblick. Springer Fachmedien Wiesbaden, Wiesbaden, 2020.

[Hr23] Hrisca, D.: ASAMMDF Documentation. https://asammdf.readthedocs.io/en/latest/, accessed 16 Feb 2024.

[IS23] Inmon, B.; Srivastava, R.: Rise of the Data Lakehouse. Building the Data Lakehouse. Technics Publications, Sedona, 2023.

[Ma22] Martini, T. et al.: Methodik und Tools zur Betriebsstrategieentwicklung für Performance-Hybride. 9. AutoTest Technical Conference, Stuttgart, 2022.

[My12] Myers, G. J.: The Art of Software Testing. J. Wiley & Sons, Hoboken, 2012.

[Pl24a] Plotly Technologies Inc.: Dash Documentation & User Guide. https://dash.plotly.com, accessed 13 Aug 2024.

[Pl24b] Plotly Technologies Inc.: Collaborative Data Science. https://plotly.com/python/, accessed 13 Aug 2024.

[Py24] Python Software Foundation: Pickle – Python Object Serialization. https://docs.python.org/3/library/pickle.html, accessed 13 Aug 2024.

[St18] Steyer, R.: Programmierung in Python. Ein kompakter Einstieg für die Praxis. Springer Vieweg, Wiesbaden, Heidelberg, 2018.

[Sy24] Synopsys, Inc.: Silver – Virtual ECU | Synopsys Verification. https://www.synopsys.com/verification/virtual-prototyping/silver, accessed 2 Aug 2024.

[SZ16] Schäuffele, J.; Zurawka, T.: Automotive Software Engineering. Springer Fachmedien Wiesbaden, Wiesbaden, 2016.

[Tu15] Tutuianu, M. et al.: Development of the World-Wide Harmonized Light Duty Test Cycle (WLTC) and a Possible Pathway for its Introduction in the European Legislation. Transportation Research Part D: Transport and Environment 40, pp. 61–75, 2015.

[UNR154] UNR154:18.08.2021, E/ECE/TRANS/505/Rev.3/Add.153/Rev.1.

[Wo18] Wolf, F.: Fahrzeuginformatik. Springer Fachmedien Wiesbaden, Wiesbaden, 2018.

[Wo24] World Wide Web Consortium: HTML Standard. https://html.spec.whatwg.org/multipage/, accessed 16 Feb 2024.