# Synchronization between Run-Time and Design-Time View of Context-Aware Automotive System Architectures

Philipp Obergfell*, Christoph Segler*, Eric Sax† and Alois Knoll‡
*BMW Group Research, New Technologies, Innovations, Parkring 19-23, 85748 Garching bei München
{philipp.obergfell, christoph.segler}@bmwgroup.com
†Karlsruhe Institute of Technology, Engesserstraße 5, 76131 Karlsruhe, eric.sax@kit.edu
‡Technical University of Munich, Boltzmannstr. 3, 85748 Garching bei München, knoll@in.tum.de

***Abstract*—In contrast to current automotive system architectures, future architectures will continuously gain knowledge at run-time with the help of machine learning techniques. For making this knowledge visible to the developer, synchronization mechanisms between run-time systems and design-time models have to be introduced. In this paper, we propose a framework for continuously updating design-time models with learned knowledge from the run-time system. The core of our framework is a system architecture with a gateway which retrieves data from different car functions. On this gateway, feature selection algorithms are implemented in order to select a data subset that describes the nominal behavior for the usage of car functions. Considering the resulting nominal model as baseline, the run-time system is able to detect violations of the nominal behavior. For assessing these violations from the perspective of a developer, we connect the run-time system with the design-time models by means of a semi-automatic feedback loop. For the evaluation, we test our approach on an exemplary scenario based on the function of the window lever by using real car data.**

## 1. Motivation

In order to learn the nominal behavior of an user, we first have to identify the context in which a function is used by the user. Context can be characterized as "any information that characterizes the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves" [1]. Some context information for the usage of a car function are caused by functional dependencies and are therefore always present. As an example, the usage context of a cruise control function is correlating to the status of the engine because the cruise control is functionally depending on a running engine. In contrast to this interaction, some context information are only caused by the behavior of the user, e.g. the context for using the seat heating includes the car's inside temperature.

In this paper, we apply machine learning techniques on automotive system architectures for deriving a model that describes the nominal behavior when using a car function. This model is considered as the nominal model. On this basis, it is possible to continuously compare the learned nominal model with the actual behavior of the user in order to detect anomalies that might give indications for potential threats. For making the anomalies visible to developers, we suggest to automatically update the design-time models and deploy possible mitigation techniques.

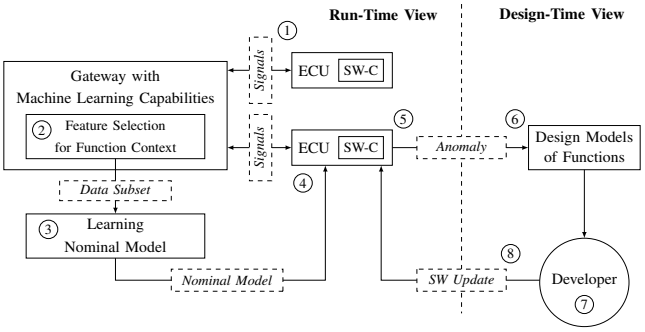


Figure 1: Proposed context aware framework for synchronization between run-time and design-time in automotive system architectures

## 2. Approach

Within our framework (shown in Figure 1), we propose an approach that consists out of five steps within the run-time view and three steps within the design-time view. The overall goal is to detect anomalies compared to the nominal use of functions and reflect those using design-time models within the development department. On this basis, the assessment of possible threats for users, their mitigation, and deployment on the run-time system are supported.

### 2.1. Run-Time View

Our approach is based on a conventional automotive system architecture. This architecture implements functions using software-components (SW-C) that are deployed on a topology of electronic-control-units (ECUs). Each function creates and consumes data which represent features, like temperature, speed, etc. (1). For identifying features that

978-1-5386-4446-1/18/$31.00 ©2018 IEEE

are relevant for the context while using a car function, we initially collect all car data over a gateway. For a single car function, reasoning about the context is challenging because just a subset of the data is relevant. In the field of machine learning this is known as the "curse of dimensionality" [2] and leads to the challenge of selecting only the relevant data subset. In order to reduce the dimensionality, we apply a feature selection algorithm within the gateway ②. Based on the derived relevant features, a model of the nominal behavior when using a car function is learned ③. This model is then send from the gateway to the ECU that controls the corresponding function ④. From now on, this ECU will retrieve context-relevant data from the gateway in order to compare their current value with the received nominal model. In the case of violating the nominal model, a notification is send to the OEM's backend infrastructure ⑤.

### 2.2. Design-Time View

In the design-time view, we automatically generate design-time models that reflect the behavior of a car function when an anomaly was detected ⑥.

We generate design-time models in the form of state machines that reflect the occurrence of an anomaly by means of a transition between two states. In this respect, the initial state represents the nominal behavior when using a car function whereas the final state (anomalous state) represents the non-nominal behavior that was detected.

Based on the generated design-time models, the developer of the corresponding function is notified and in charge of assessing the anomaly. This assessment includes the expertise of safety as well as security engineers. The safety engineers have to classify if the anomalous state still illustrates a safe state [1]. The task for the security engineers is to derive whether possible security attacks lead to the transition into the anomalous state. Both assessments may consequence in the selection of mitigation approaches ⑦. Based on the mitigation approach, software updates are deployed ⑧.

## 3. Evaluation

To evaluate our approach, we learn the nominal behavior when using the driver's window lever. We assume that a subset of all features in the car is correlating to the status of the driver's window (based on the driver's behavior). We distinguish between the window states *fully open*, *partly open*, and *closed*. Using feature selection, we identify the relevant data subset and then describe the correlation of the single features to the state of the driver's window. On this basis, we learn a model for the nominal behavior when using the driver's window lever. For example, this model could assert that the window is only opened below a certain speed. In the case of non-nominal behavior (e.g. opening the window at high speed), we assume to detect the non-nominal situation when comparing it to the nominal model.

1. In this respect, we refer to the term of a safe state which is introduced in the ISO 26262 [3].

### 3.1. Evaluation Setup

For the evaluation, we collected data from a current generation BMW 5 Series which was driven by different drivers for eight months in order to learn a model for the nominal behavior when using the driver's window lever. The drivers have not been instructed how and when to use the window lever. The non-nominal situations were recorded with the same car within a test drive where we provoked anomalies multiple times by opening the window at high speed. We collected the data that contains the anomalies and tested the nominal models within the lab. For the detected anomalies, we imported the anomalous situation in a model-based tool that automatically creates design-time models in the form of state machines for the assessment by developers.

For step ① in our approach, we used a data logger for collecting all data which was send as signal via the car's internal CAN-Buses. The dataset includes ~2000 features and ~46000 instances with a sampling rate of ~50 seconds. After labeling, data preprocessing, and removing static features, ~1350 features and ~44000 instances remained. For selecting features which correlate to the status of the driver window, we used the supervised feature selection algorithm fisher score [4], [5] on every second instance from our dataset ②.

### 3.2. Evaluation Results

Our evaluation focuses on the top 25 features from the output of the feature selection algorithm (in future we want to investigate this threshold). From these 25 features, we discarded nine features since they are functionally dependent on the function of the window lever and do not contain any information about the context ("fire" features [6]). For the 16 remaining features, we created independent nominal models in which the correlation between the window status and each feature is described. Within this paper, the set of all independent nominal models is considered as the nominal model for the context ③.

*Description of the independent nominal models:* Ten of the independent nominal models describe a correlation to the speed of the car with the only difference that the value is provided by different sensors. One of these models is illustrated in Figure 2 and described later on. Besides the correlation to the speed, one model is describing the correlation to the current gear which is depending on the car's speed. Another model is describing the correlation to the status of an engine function which is only available a few minutes after starting the engine and one model describes the correlation to the quality of the location tracking. These two models are present because the test car is parked in an underground car park where the driver has to open the window and place his ID card on a card reader to leave the car park. The corresponding features correlate because in this situation the specific engine function is not yet available and the satellite reception for tracking is bad and due to the card reader the window has to be opened by the driver. Another model describes the correlation to the

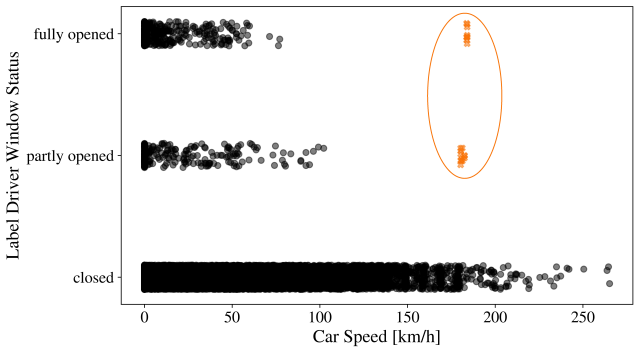


Figure 2: Window status in correlation to car's speed for nominal situations (black dots) / non-nominal situations (orange crosses within circle)

parking assistant which shows that the function was only used when the window was closed. The penultimate model describes the correlation to the road type. In brief, this model describes that the window is rarely opened while traveling on a highway. The last model describes the correlation to a debug message. This circumstance derives from the development level of the test vehicle.

In Figure 2, the car's speed and its correlation to the window status is visualized. Each black dot represents one instance of the recorded nominal behavior [2]. As seen in Figure 2, the maximum speed with a fully or partly opened window is 102.7 km/h in the nominal situations. Therefore, the nominal model asserts that opening the window occurs between the absolute speed of 0 and 102.7 km/h. In this case, opening the window above a speed of 102.7 km/h is considered as anomalous behavior of the user. Using these thresholds on the non-nominal trace ④, we can detect an anomaly from the test drive ⑤. In Figure 2, each orange cross within the circle represents one instance of the window status in the non-nominal situation and the corresponding speed that violates the nominal model. As seen in Figure 2, the provoked situation will lead to a detected anomaly. Based on the illustrated data, we triggered an import function that is implemented within the model-based engineering tool PREEvision [7]. This import function read the detected anomaly and updated the design-time models for the window lever in order to reflect the anomalous situation ⑥, ⑦. These models can be used to asses whether possible threats regarding safety or security are linked with the detected anomaly and to initiate mitigation approaches in the form of software updates ⑧.

## 4. Conclusion

In this paper, we present a framework for continuously updating an existing design-time model with newly gained knowledge from the run-time system. The core of our framework is illustrated by a system architecture that is equipped with a gateway for retrieving all data created by different car functions. On this gateway, feature selection algorithms are implemented in order to identify relevant correlations between different car functions and car data. Using this information, the system is able to learn the context in which certain car functions are used. We considered this as nominal behavior of the user. Vice-versa, the system is capable to detect non-nominal behavior during run-time. In order to react to detected non-nominal behavior, we developed a semi-automatic feedback loop between the run-time system and the design-time models. We evaluated our approach on real car data and showed the capability on detecting anomalies that are caused by non-nominal behavior of the user.

Within our research project [8], the intended, final evaluation set up for our test car is not yet present. Here, the learning of nominal models for car functions and the detection of anomalies would happen at run-time. Currently, we collect the data in the form of traces. The feature selection algorithm itself is running in the lab on test hardware and selecting the data subset for the nominal models.

In future work we want to implement our approach to operate at run-time in the car and explore more complex functions and scenarios. Furthermore, we want to investigate the threshold for the number of selected features considering the tradeoff between performance and accuracy. We also want to refine the learning algorithm for the creation of the nominal behavior model and evaluate different approaches.

2. For a better illustration of the data, each observation is randomly distributed over the y-axis within each class and plotted with an opacity of 50%.

## References

[1] A. K. Dey, "Providing architectural support for building context-aware applications," Ph.D. dissertation, College of Computing, Georgia Institute of Technology, 2000.

[2] R. Bellman, *Dynamic Programming*, 1st ed., ser. Dover Books on Computer Science. Princeton, NJ, USA: Princeton University Press and Dover Publications, 1957.

[3] ISO, "Road vehicles–Functional safety (ISO 26262)," 2011.

[4] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection," *ACM Computing Surveys*, vol. 50, no. 6, pp. 1–45, 2018. [Online]. Available: http://arxiv.org/pdf/1601.07996v4

[5] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*, 2nd ed., ser. A Wiley-Interscience publication. New York NY u.a.: Wiley, 2001.

[6] Z. Zhao, J. Wang, S. Sharma, N. Agarwal, H. Liu, and Y. Chang, "An integrative approach to identifying biologically relevant genes," in *Proceedings of the 2010 SIAM International Conference on Data Mining*, S. Parthasarathy, B. Liu, B. Goethals, J. Pei, and C. Kamath, Eds. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2010, pp. 838–849.

[7] "Preevision," https://vector.com/vi_preevision_de.html, accessed: 2017-02-10.

[8] S. Kugele, V. Cebotari, M. Gleirscher, M. Hashemi, C. Segler, S. Shafaei, H.-J. Vögel, F. Bauer, A. Knoll, D. Marmsoler, and H.-U. Michel, "Research challenges for a future-proof e/e architecture - a project statement," in *Informatik 2017*, ser. GI-Edition Lecture Notes in Informatics Proceedings, M. Eibl and M. Gaedke, Eds. Bonn: Gesellschaft für Informatik, 2017.