SDVDiag: A Modular Platform for the Diagnosis of Connected Vehicle Functions

Matthias Weiß, Falk Dettinger, and Michael Weyrich
Institute of Industrial Automation and Software (IAS)
University of Stuttgart
Pfaffenwaldring 47, 70550 Stuttgart, Germany
E-Mail: {matthias.weiss, falk.dettinger, michael.weyrich}@ias.uni-stuttgart.de

***Abstract*—Connected and software-defined vehicles promise to offer a broad range of services and advanced functions to customers, aiming to increase passenger comfort and support autonomous driving capabilities. Due to the high reliability and availability requirements of connected vehicles, it is crucial to resolve any occurring failures quickly. To achieve this however, a complex cloud/edge architecture with a mesh of dependencies must be navigated to diagnose the responsible root cause. As such, manual analyses become unfeasible since they would significantly delay the troubleshooting.**

To address this challenge, this paper presents SDVDiag, an extensible platform for the automated diagnosis of connected vehicle functions. The platform enables the creation of pipelines that cover all steps from initial data collection to the tracing of potential root causes. In addition, SDVDiag supports self-adaptive behavior by the ability to exchange modules at runtime. Dependencies between functions are detected and continuously updated, resulting in a dynamic graph view of the system. In addition, vital system metrics are monitored for anomalies. Whenever an incident is investigated, a snapshot of the graph is taken and augmented by relevant anomalies. Finally, the analysis is performed by traversing the graph and creating a ranking of the most likely causes.

To evaluate the platform, it is deployed inside an 5G test fleet environment for connected vehicle functions. The results show that injected faults can be detected reliably. As such, the platform offers the potential to gain new insights and reduce downtime by identifying problems and their causes at an early stage.

***Index Terms*—Connected Vehicle, Diagnosis, Root-cause Analysis, Causality Mining, Anomaly Detection, Dependency Graph**

I. Introduction

The automotive industry is undergoing a fundamental transformation, shifting from traditionally mechanical systems toward connected and software-defined vehicles (SDVs) [1] [2]. These vehicles increasingly rely on complex software stacks, continuous over-the-air updates, and cloud-based services to deliver advanced functions such as predictive maintenance, real-time traffic routing, and autonomous driving support. As these systems become more sophisticated, so do the expectations placed upon them—drivers and passengers alike demand seamless user experiences, minimal downtime, and robust reliability even under dynamic operating conditions.

To ensure these expectations are met, connected vehicle platforms must be closely monitored and continuously evaluated. This is typically achieved by establishing data loops that capture operational data across vehicle components, edge nodes, and backend cloud systems [3]. These loops form the basis of observability pipelines that provide valuable insights into system health, service availability, and anomalous behavior. However, the increasing complexity and distribution of these systems introduce significant challenges.

Diagnosing issues in such an environment is far from trivial. When a function degrades or fails, the root cause may lie deep within a chain of dependent services spread across in-vehicle software, edge processing units, and cloud-native components. Manual diagnosis in this context is not only time-consuming but also prone to human error, often requiring engineers to navigate layers of telemetry, logs, and metrics without a unified view of the system's internal state. This latency in troubleshooting can lead to prolonged downtimes and diminished user trust—both critical concerns for the success of SDVs [4].

Given these challenges, a pressing research question arises: *How can an automated and scalable diagnosis of connected vehicle functions across complex, distributed environments be enabled?* This paper addresses this question by introducing SDVDiag, a flexible and extensible platform designed to automate the end-to-end process of diagnosing faults in connected vehicle systems.

The remainder of this paper is structured as follows: Section II and Secton III provide an overview of related work with regards to distributed automotive systems and observability. Section IV introduces the architecture of the SDVDiag platform and outlines the automated diagnosis, including graph creation, incident analysis and self-adaptation. Section V presents the evaluation within a 5G test fleet environment and discusses the effectiveness of SDVDiag in identifying injected faults. Finally, Section VI concludes the paper and outlines directions for future research.

II. Background

A. Distributed Systems in Vehicular Environments

The increasing automation and benefits for autonomous navigation, such as cooperative map generation or driving, highlight the need for powerful and scalable backend systems. Typically, both cloud servers and edge clusters are used. While the cloud offers scalability and high computing power [5], its greater distance to vehicles results in higher latency, making it unsuitable for time-critical applications. In contrast, edge servers, mounted on Road Side Unit (RSU)s or cell towers, provide local access to computing resources for latency-critical applications but have limited processing capacity compared to the cloud.

Specific implementations are enabled through the alternative or combined use of cloud and edge. Here, the terms Mobile Cloud Computing (MCC) [6], Mobile Edge Computing (MEC) [7], and, in hybrid systems, Cloud Edge Computing (CEC) [8] are significant. MCC refers to systems that solely rely on the cloud data center for computation, while MEC corresponds to the use of edge servers. CEC, as a hybrid approach, attempts to combine the advantages and disadvantages of MCC and MEC, enabling both the scalability of the cloud and the processing of latency-critical applications on the edge.

Applications in this context are implemented as services that include all dependencies and can be deployed in a scalable manner on cloud and edge servers [9], then retrieved by the vehicle. During deployment, reliability and availability of the services are ensured, which is why they are often implemented as distributed applications within computing clusters, such as Kubernetes.

B. Vehicle Data Loops and Observability

In connected vehicle environments, vast amounts of data are continuously collected to facilitate collaborative use cases, enhance existing software models, and monitor system health to ensure reliability and availability. This systematic collection, processing, and utilization of vehicle-generated data are commonly described as a data loop. Typically, a data loop comprises multiple interconnected stages: data acquisition from vehicles, transmission to backend infrastructure, processing and analysis, model updates or decision-making based on analysis results, and finally, deployment of the updated models or corrective actions back into the vehicle fleet.

Given the distributed nature of connected vehicle systems, which often span both cloud and edge computing environments, specialized methods and tools are essential for effectively managing and analyzing the collected data. A crucial concept in this context is observability, defined as the ability to understand and diagnose the internal state of a system solely based on its external outputs, such as metrics, logs, and traces [10], [11]. Observability enables system engineers to reconstruct the state and behavior of complex systems, thus making it possible to pinpoint the root causes of observed failures quickly and accurately.

To achieve observability in distributed vehicular environments, various tools and practices have been established. These tools typically capture three primary types of data: metrics that quantify the system's state numerically (e.g., CPU load, memory usage, response times), logs that record discrete events and state changes, and traces that document the path of requests across different services [12], [13]. Together, these data types provide comprehensive visibility into the functioning of vehicle systems and their backend infrastructures.

For the analysis of such observability data, anomaly detection methods are widely employed. Anomaly detection identifies deviations from expected patterns, highlighting potential problems and enabling proactive maintenance. Techniques can broadly be categorized into statistical methods, machine learning-based approaches, and hybrid models. Statistical methods, such as Z-score analysis or seasonal decomposition of time series (e.g., STL), rely on predefined assumptions about data distributions [14]. Machine learning-based methods, including supervised approaches like isolation forests and unsupervised models like autoencoders, are particularly prevalent in productive environments due to their effectiveness in detecting complex anomalies without explicit definitions [15]. However, anomaly detection models employed in production are typically trained statically, necessitating periodic retraining to maintain accuracy as data characteristics evolve.

III. State of the Art

A. Vehicle Diagnostics

For traditional, non-connected cars, Unified Diagnostic Services (UDS) is a widely adopted standard in vehicle diagnostics, serving as a key communication protocol between diagnostic tools and a vehicle's Electronic Control Unit (ecu)s [16]. It provides standardized processes for fault detection, analysis, and resolution, helping to identify operational anomalies. In addition to UDS, several well-established diagnostic protocols contribute to anomaly detection in vehicles, including On-Board Diagnostics (OBD)-II [17] and Society of Automotive Engineers (SAE) J1939 [18].

As vehicle architectures increasingly integrate high performance computing (HPC) units and methods for hardware abstraction, service-oriented approaches are gaining prominence alongside traditional diagnostic protocols. Service-Oriented Vehicle Diagnostics (SOVD) is a key innovation designed to address the increasing complexity of software-defined and connected vehicle architectures [19]. By leveraging standardized APIs, SOVD enables real-time data retrieval, flexible system monitoring, and seamless integration into cloud-based diagnostic frameworks. Yet, to the best of our knowledge, to this date there exists no such diagnostic framework for distributed, connected vehicle functions, which incorporates both vehicle and cloud services.

B. Causal Inference in Distributed Systems

The increasing complexity and scale of modern distributed systems cause conventional analysis methods to become impractical. Thus automated methods have gained significant attention over the past years, typically involving a two-step process: anomaly detection to identify unexpected system behavior and subsequent causal inference to pinpoint underlying root causes [20]. For the latter, graph-based methods have become particularly prevalent due to their ability to intuitively represent relationships within distributed systems. Two principal types of graphs are commonly utilized: dependency graphs and causal models [21]. Dependency graphs explicitly represent relationships among system components, generally based on observed interactions such as communication patterns or resource usage. Causal models, on the other hand, explicitly capture causal relationships between components, quantifying how changes or anomalies in one component influence others.

To generate causal models, causal discovery methods have been developed, aiming to infer causal relationships directly from observational data. One example that will also be used in this paper is the Amortized Causal Discovery (ACD) framework proposed by Loewe et al. [22], which leverages deep learning to efficiently infer causal structures from time series data, particularly suited for dynamic and noisy distributed environments. Despite substantial advancements, the application of current causal inference approaches to connected vehicle environments requires careful scrutiny and extensive adjustments. Notably, these systems present high dynamicity due to frequent software updates and environmental changes, significant heterogeneity across various vehicle functions, and complex integrations spanning vehicle-edge-cloud infrastructures. Addressing these gaps represents a pivotal research direction to enhance reliability and maintainability within connected vehicle environments.

IV. Concept of SDVDiag

A. Overview

To deal with the intricate challenges of connected vehicle environments, this paper introduces the concepts of distributed tracing and causality mining to the automotive domain and proposes additional approaches to improve reliability in complex distributed systems. In particular, the following requirements to the concept have been isolated:

1) **Real-time Data Collection**: The platform must continuously collect data from vehicles and backend services in real time to reflect the current system state.
2) **Distributed Tracing Capability**: SDVDiag should provide distributed tracing to effectively track requests through various services and infrastructure layers.
3) **Dynamic Dependency Mapping**: The platform must dynamically detect and maintain an updated map of dependencies between different vehicle functions and services.
4) **Anomaly Detection**: Robust anomaly detection techniques must identify deviations in metrics to proactively recognize system issues.
5) **Causality Analysis and Root-cause Identification**: SDVDiag must perform causality mining to trace relationships between anomalies, identifying potential root causes.
6) **Modularity, Extensibility, Scalability**: The platform architecture should support modularity, allowing runtime component exchange, extension, and scaling.
7) **Self-adaptation and Continuous Model Updating**: The platform must continuously update causality mining and anomaly detection models to ensure consistent performance even under concept drift.
8) **Intuitive Visualization and Reporting**: Diagnostic outcomes, including anomalies and causality graphs, should be presented through clear and intuitive visualizations.

To address these requirements, SDVDiag, which is facilitated by a service-based architecture, is designed. Fig. 1 presents the resulting application and its key subsystems. In general, the architecture can roughly be divided into:

- A **data aggregation and storage** layer (grey), which includes methods and technologies for accessing the information required for the analysis from all involved data sources,
- the process of **graph creation** (blue), which is responsible for generating a comprehensive system overview based on the given data, which can subsequently be used when investigating incidents,
- a **learning environment** (green), in which the involved models are trained continuously based on most recent data to ensure a robust analysis process, and
- the actual **incident analysis** (yellow), where anomalies become causally linked and the most probable root causes are determined.

More details on these components are provided in the following.

B. Data Aggregation and Storage

Foundational Layer that holds all data that is relevant for the continual diagnosis. Data must be collected from all sources (i.e., vehicles, edge nodes and the cloud), stored and preprocessed within acceptable time constraints. To accomplish this, the platform provides support for most state of the art monitoring and observability protocols as they are used in productive environments. In the implemented examples, OpenTelemetry, Apache Kafka and the Delta Lake framework have all been used for data collection and storage in real-time. For the correct operation of the analysis platform, two specific data types must be collected: Traces, by which the data and communication flow between components can be measured, and metrics, which provide information about the system status and performance, such as the resource consumption on available compute nodes. The integration of additional data types like logs or application-specific data is supported as well via the provided platform interfaces.

C. Graph Creation

This layer is responsible for generating and maintaining a comprehensive overview of the system using a graph-based data model. In this model, nodes represent system components along with their characteristics, while edges symbolize the relationships between these components. SDVDiag uses two distinct types of graphs (also ref. to Section III-B), whose elements are explained in the following:

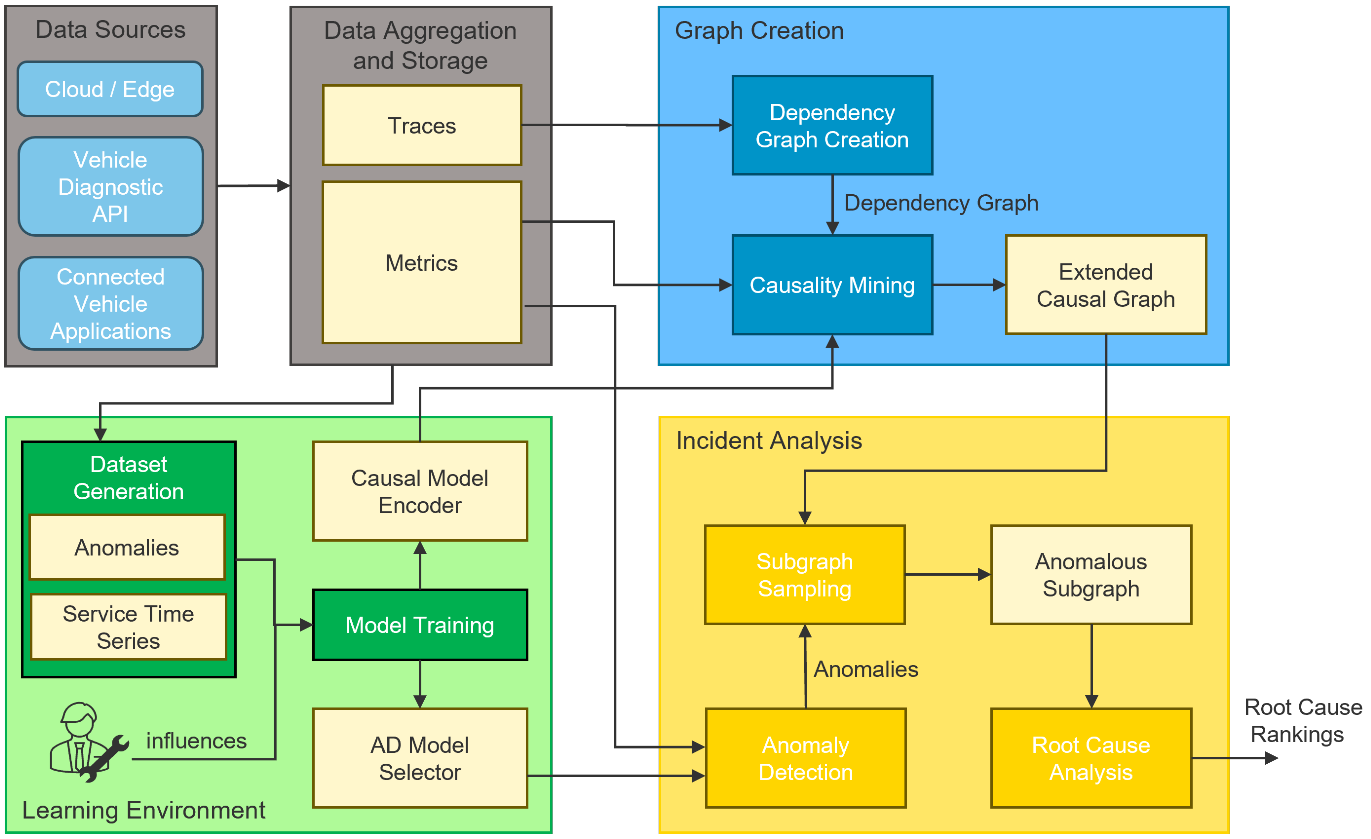


Fig. 1. Conceptual Overview of SDVDiag. The platform can roughly be divided in subsystems for the generation of graphs, the analysis of these graphs for detected incidents and a learning environment, in which the used models evolve continuously.

1) Dependency Graphs: In distributed systems, runtime information forms the foundation of all analysis processes due to the dynamic behavior of the environment. For example, dependencies between services and instances must be derived from live communication flows, as they are not determined statically at time of development. As such, SDVDiag integrates tools for distributed tracing, by which dependency graphs are constructed, and saves the result in a graph database for further analysis. Additionally, SDVDiag supports to extend the dependency graph to enable a dedicated analysis of specific applications or types of incidents. For the purpose of this paper, timestamps of the most recent recorded communication between two services are added as an example in order to provide context on failure propagation to the analysis. Other possible information to integrate includes current performance metrics and additional knowledge provided by engineers, all of which require compatible causality mining techniques.

2) Causal Graphs: As mentioned in Section III-B, causal relationships are essential for automated analyses since they quantify the strength of causality between components. To derive these relationships, SDVDiag supports integrating various causal discovery models, which can be exchanged dynamically at runtime due to the modular architecture of the platform. In the context of this paper, the Amortized Causal Discovery (ACD) framework (see Section III-B) has been incorporated into the platform to identify causal relationships between service instances. During operation, ACD identifies causal relations among monitored performance metrics and constructs an initial causal graph structure, which currently includes metric nodes interconnected by weighted edges representing causal strength. Subsequently, SDVDiag combines this causal information with the existing dependency graph, resulting in an extended causal graph, as illustrated in Figure 2. In this combined graph, metric nodes are assigned to their corresponding service instances. Furthermore, to enhance analytical stability, causal edges between metrics are pruned if the associated services or instances are not related according to the dependency graph. The resulting graph serves as the basis for subsequent incident analyses and is continuously updated to reflect the latest system state.

D. Learning Environment

To ensure the adaptability of the analysis platform, the involved models must be continuously updated to account for concept drift (e.g., introduced by software updates) or dynamic changes in system resources. To facilitate this, SDVDiag includes a dedicated learning environment that supports system engineers in generating suitable datasets and

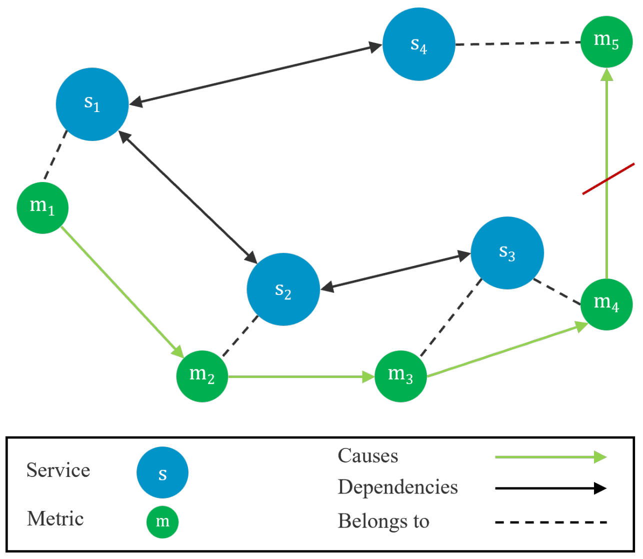


Fig. 2. Extended causality graph of SDVDiag. Information about service dependencies is combined with causal relationships, resulting in a comprehensive system overview. Causalities are pruned when there is no direct dependency between two nodes.

controlling the training process of the models. Specifically, two primary models require ongoing updates:

1) Causal Model Encoder: As mentioned in Section IV-C, the causality mining process requires a model that learns the causal behavior of the system. During operation, this model computes the causal weights between system components, determining the strength of causal relationships between pairs of components. For the purpose of this paper, a model for Amortized Causal Discovery has been trained within the learning environment. The model learns system dynamics based on time series data extracted from operational data collections. A significant limitation of many causal models, including ACD, is that even minor system changes can rapidly degrade the quality of results. To address this issue, retraining of the model is initiated whenever such a change occurs—typically after software updates or deployment modifications. Nevertheless, given the frequent changes typical in complex distributed systems, this limitation remains challenging for conventional causal discovery methods. Consequently, SDVDiag integrates an additional feedback loop, enabling more effective interventions by system engineers and incremental model adjustments that facilitate faster learning during live operation. The detailed concept behind this approach is beyond the scope of this paper and will be elaborated upon in future work.

2) Anomaly Detection Model Selector: To ensure robust anomaly detection, SDVDiag implements automated anomaly detector selection based on the characteristics of individual time series. During operation, features are extracted from each metric, upon which a suitable anomaly detection model is selected, trained, and deployed. This selection is facilitated through a continuous training loop, wherein an agent learns to select the optimal model from a predefined pool based on the extracted features of each time series. For the scope of this paper, both supervised and unsupervised anomaly detection models have been integrated into the available solution space. Initially, the agent undergoes training using historical, labeled datasets to ensure optimal performance. The learning environment supports system engineers in creating these labeled datasets by automatically recording historical data and suggesting labels. Once the initial training phase is complete, the agent can be continuously fine-tuned, which is performed on live data during operation and relies on a feedback mechanism wherein system engineers indicate whether the selected anomaly detection model's assessment for a given time series was correct. The detailed design and evaluation of this loop will be covered in future work.

E. Incident Analysis

Given the availability of adaptive models (Section IV-D) and the extended causal graph (Section IV-C), SDVDiag can proceed to analyze specific system incidents. Incident analysis can either be initiated manually by system engineers or triggered automatically through anomaly detection.

Regardless of the triggering method, SDVDiag subsequently creates a snapshot of the most recent system state for detailed investigation. This snapshot includes the current extended causal graph along with all anomalies detected within a configurable timeframe. Interfaces are provided for additional modifications to this snapshot. For instance, this paper demonstrates a module that prunes graph paths involving services without any detected anomalies, as such services are unlikely to contribute to the investigated incident. Further ML-based sampling techniques have also been developed and can be selected within the platform.

After preparing the subgraph, the actual root-cause analysis is performed. SDVDiag provides modules supporting both first-order and second-order random walks, as well as conventional Fault Tree Analysis (FTA). In this paper, the first-order random walk is used as the default method due to its efficiency in smaller distributed environments. However, the authors suggest to rely on the other options in larger systems with long causal chains. For the first-order walk, system engineers can choose whether the analysis should identify the root cause of a single anomaly or all detected anomalies collectively. When analyzing a single anomaly, the random walk algorithm begins at the node representing the anomaly's metric and traverses the graph along causal relationships. Neighboring nodes are selected randomly, weighted by the strength of the causal relationships—meaning nodes with stronger causal connections are visited more frequently. For analyzing multiple anomalies, the process repeats with different start nodes corresponding to each anomaly, until all anomalies have been processed. Finally, the visit counts for each node are summed and ranked in descending order, providing the system engineer with an ordered list of the most probable root causes for further investigation.

V. Evaluation

To evaluate the analysis capabilities of the SDVDiag platform, a smart charging application was developed and deployed within the 5G vehicle test track at the University of Stuttgart. Fig. 3 provides an overview of the entire system. The hardware setup includes multiple Unmanned Ground Vehicles (UGVs) equipped with various sensors for environmental perception, such as LiDAR and depth cameras, enabling autonomous navigation. Designated parking spots with integrated charging stations are available throughout the track. For efficient data transmission, each UGV is connected to a local, scalable computing cluster via 5G. The cluster, implemented using Kubernetes, consists of one control node and three worker nodes.

Multiple instances of the charging station service are deployed on the cluster, providing real-time access to charging infrastructure data [23]. These services process publicly available information to offer key details such as the location, operator, and capacity of charging stations. A REST API enables vehicles to retrieve nearby stations, check availability, and perform spatial queries using geometric distance calculations. The vehicle service allows seamless access to this information by responding to vehicle requests for nearby charging options.

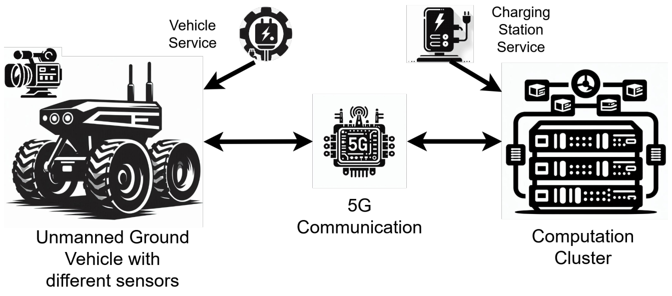
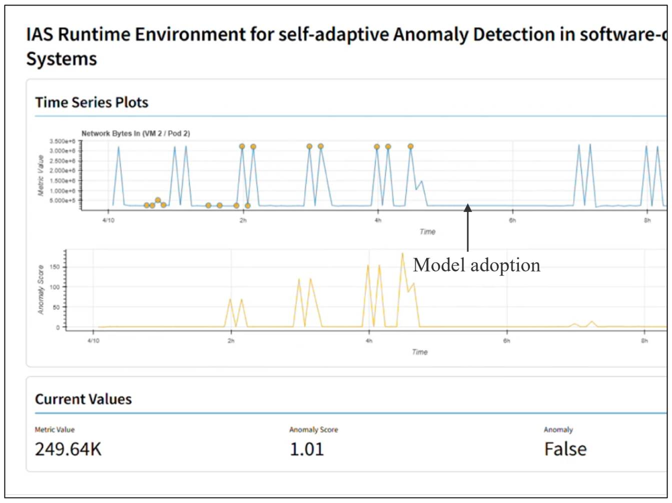


Fig. 3. System overview consisting of Unmanned Ground Vehicle with defined sensor set, connected via 5G with a computation cluster providing a charger station service

To enable analysis in this environment, SDVDiag was integrated to monitor both the Kubernetes cluster and the deployed vehicle services. Traces are collected to construct a dependency graph, while causal discovery is applied to the collected metrics to infer causal relationships. The Causal Model Encoder and the AD Model Selector were trained on historical data and subsequently deployed into the live system. Fig. 4 displays the SDVDiag user interface during anomaly detection. The highlighted metric is the transmitted data volume of a specific worker node, which shows characteristic spikes whenever vehicle requests are processed. Anomalies (marked as yellow dots) appear at the beginning of the recording but are identifiable as false positives. As more data becomes available, the AD Model Selector improves its model selection and training, resulting in a noticeable reduction of false positives over time.

Regarding the evaluation of the incident analysis, a scenario was simulated in which a single charging station service


IAS Runtime Environment for self-adaptive Anomaly Detection in software-d
Systems

Time Series Plots

Current Values

Metric Value	Anomaly Score	Anomaly
249.64K	1.01	False

Fig. 4. UI of SDVDiag for Anomaly Detection. Yellow dots symbolize detected anomalies. Detection is improved over time by the learning loop.

experiences increased CPU usage. This anomaly produces several observable effects in the system, including:

- A decrease in CPU usage for other charging station services on the same worker node due to limited available resources.
- An increase in CPU usage for charging station services on other nodes, as requests are rerouted to available services.
- Increased CPU usage for vehicle services interacting with the affected charging station, as they eventually assume the service is unreachable and attempt to compute the closest alternative locally.

Fig. 5 presents the resulting anomalous subgraph (service nodes are hidden for clarity). The graph reveals a dense network of causal relationships among the affected metrics, which is expected given the small-scale demo system and the load balancing mechanisms that operate across all worker nodes. Despite this complexity, SDVDiag successfully isolates

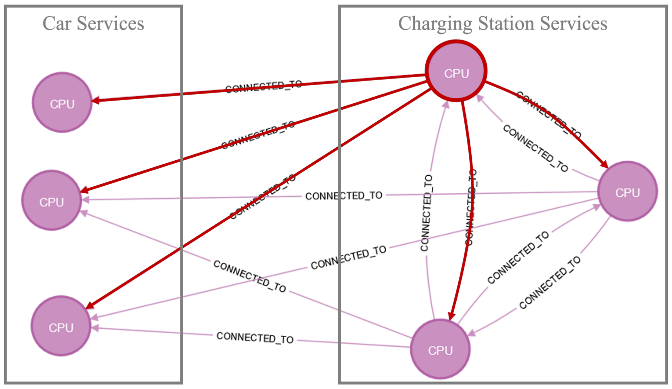


Fig. 5. Anomalous causality graph for the charging station test scenario. The increased CPU usage of a single charging station service causes CPU fluctuations across the system.

the root cause of the incident, which can be attributed to the accurate causal discovery model and graph pruning procedure.

Overall, the evaluation demonstrates that SDVDiag is capable of reliably detecting anomalies and identifying their root causes in a dynamic, distributed vehicle environment. The platform's modular architecture, adaptive learning capabilities, and graph-based analysis techniques contribute to a robust and effective diagnosis process.

VI. Conclusion

To enable the diagnosis of the challenging environment posed by software-defined vehicles, this paper presents SDVDiag as a modular and scalable platform that improves the reliability of connected vehicle systems by combining distributed tracing and causality mining. The platform consists of subsystems for the creation of dependency and causal graphs, a learning environment for continuous model refinement, and incident analysis. Its service-based design enables real-time diagnostics, anomaly detection, and root-cause analysis. Evaluation within a 5G test environment confirms its effectiveness in monitoring applications, providing a robust anomaly detection, and ensuring scalable diagnostics for intelligent mobility. The main findings can be summarized as follows:

- Diagnosing failures in distributed vehicle systems is challenging due to complex interdependencies and limited system transparency.
- SDVDiag employs distributed tracing and causality mining to map dependencies and identify root causes with precision.
- The platform provides effective mechanisms for continuous model adaptation on live operational data, maintaining robust performance despite frequent system changes.

While SDVDiag demonstrates satisfactory performance in diagnosing incidents observable through system metrics, many real-world failures require additional expert knowledge for accurate identification. Therefore, future work will focus on integrating advanced causal discovery techniques to reduce training overhead and enable the seamless incorporation of domain expertise into the causality mining process. Additionally, the human-assisted feedback loop for fine-tuning anomaly detection and causal inference will be further developed and comprehensively addressed in future work.

References

[1] D. Baumann, M. Sommer, F. Dettinger, T. Rösch, M. Weyrich, and E. Sax, "Connected vehicle: Ontology, taxonomy and use cases," in *2024 IEEE International Systems Conference (SysCon)*, 2024, pp. 1–6.

[2] M. Weiß, M. Müller, F. Dettinger, N. Jazdi, and M. Weyrich, "Continuous analysis and optimization of vehicle software updates using the intelligent digital twin," in *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2023, pp. 1–7.

[3] M. Weiß, S. Thich, M. Artelt, and M. Weyrich, "A survey about self-adaptive anomaly-detection in software-defined systems," in *IECON 2024 - 50th Annual Conference of the IEEE Industrial Electronics Society*, 2024, pp. 1–4.

[4] M. Weiß, F. Dettinger, N. Jazdi, and M. Weyrich, "Devops als enabler der kontinuierlichen funktionsverbesserung und automatisierten update-analyse in software-definierten systemen," in *Automation 2023*, 2023.

[5] F. Dettinger, M. Weiß, and M. Weyrich, "Future use cases for vehicular communication based on connected functions," in *2024 IEEE 100th Vehicular Technology Conference (VTC2024-Fall)*, 2024, pp. 1–5.

[6] K. Akherfi, M. Gerndt, and H. Harroud, "Mobile cloud computing for computation offloading: Issues and challenges," *Applied computing and informatics*, vol. 14, no. 1, pp. 1–16, 2018.

[7] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE communications surveys & tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[8] B. Wang, C. Wang, W. Huang, Y. Song, and X. Qin, "A survey and taxonomy on task offloading for edge-cloud computing," *IEEE Access*, vol. 8, pp. 186 080–186 101, 2020.

[9] E. Peltonen, M. Bennis, M. Capobianco, M. Debbah, A. Ding, F. Gil-Castiñeira, M. Jurmu, T. Karvonen, M. Kelanti, A. Kliks, T. Leppänen, L. Lovén, T. Mikkonen, A. Rao, S. Samarakoon, K. Seppänen, P. Sroka, S. Tarkoma, and T. Yang, "6g white paper on edge intelligence," 2020. [Online]. Available: https://arxiv.org/abs/2004.14850

[10] C. Sridharan, "Distributed systems observability," in *Velocity Conference*. O'Reilly Media, 2018.

[11] S. Niedermaier, F. Koetter, A. Freymann, and S. Wagner, "On observability and monitoring of distributed systems: An industry interview study," in *Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*. SciTePress, 2019, pp. 99–108.

[12] B. H. Sigelman, L. A. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, and C. Shanbhag, "Dapper, a large-scale distributed systems tracing infrastructure," *Google Research Technical Report*, 2010.

[13] B. Li, X. Peng, Q. Xiang, H. Wang, T. Xie, J. Sun, and X. Liu, "Enjoy your observability: an industrial survey of microservice tracing and analysis," *Empirical Software Engineering*, vol. 27, no. 1, pp. 1–42, 2022.

[14] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[15] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller, "A unifying review of deep and shallow anomaly detection," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 756–795, 2021.

[16] International Organization for Standardization, "Road vehicles — unified diagnostic services (uds) — part 1: Application layer," International Organization for Standardization, Geneva, Switzerland, Tech. Rep. ISO 14229-1:2020, 2020.

[17] G. Team. (2023) What is obdii? history of on-board diagnostics (obd). Accessed: 2025-05-05. [Online]. Available: https://www.geotab.com/blog/obd-ii/

[18] SAE, "Serial control and communications heavy-duty vehicle network," SAE International, Warrendale, PA, USA, Tech. Rep. J1939_202306, June 2023, revised June 2023, originally issued April 2000. [Online]. Available: https://www.sae.org/standards/content/j1939_202306/

[19] ASAM, "Service-oriented vehicle diagnostics sovd)," Association for Standardisation of Automation and Measuring Systems (ASAM), Ingolstadt, Germany, Tech. Rep. Version 1.0.0, Jun. 2022, accessed: 2025-05-04. [Online]. Available: https://www.asam.net/standards/detail/sovd/

[20] B. Li, X. Peng, Q. Xiang, H. Wang, T. Xie, J. Sun, and X. Liu, "Enjoy your observability: an industrial survey of microservice tracing and analysis," *Empirical Software Engineering*, vol. 27, no. 1, p. 25, 2021. [Online]. Available: https://doi.org/10.1007/s10664-021-10063-9

[21] T. Wang and G. Qi, "A comprehensive survey on root cause analysis in (micro) services: Methodologies, challenges, and trends," 2024. [Online]. Available: https://arxiv.org/abs/2408.00803

[22] S. Löwe, D. Madras, R. Zemel, and M. Welling, "Amortized causal discovery: Learning to infer causal graphs from time-series data," in *Conference on Causal Learning and Reasoning*. PMLR, 2022, pp. 509–525.

[23] M. Weiß, J. Stümpfle, F. Dettinger, N. Jazdi, and M. Weyrich, "Simulating cloud environments of connected vehicles for anomaly detection," in *SAE Technical Paper Series*, ser. STUT. SAE International, Jul. 2024. [Online]. Available: http://dx.doi.org/10.4271/2024-01-2996