2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)

Software Engineering for Automated Vehicles: Addressing the Needs of Cars That Run on Software and Data

(Extended Abstract)

Krzysztof Czarnecki
Waterloo Intelligent Systems Engineering (WISE) Lab, University of Waterloo, Canada
Email: kczarnec@gsd.uwaterloo.ca

***Abstract*—Automated vehicles are AI-based safety-critical robots that fulfill transportation needs while interacting with the general public in traffic. Software engineering for automated vehicles requires a DevOps-style process with special considerations for functions based on machine learning and incremental safety assurance at vehicle and fleet level. This technical briefing reviews current challenges, industry practices, and opportunities for future research in software engineering for automated vehicles.**

I. Introduction

Automated driving is a multidisciplinary challenge that creates new requirements for software engineering. The functionality of an automated driving system (ADS) is mainly expressed in software, and thus, software engineering is also a key discipline in automated driving. ADS development places special demands on software engineering, requiring data-driven development and agility, while assuring safety.

This briefing discusses these demands and reviews current practices and challenges in addressing them. The practices include requirements engineering for data-driven functionality, dependable software architectures, safety assurance for functions that rely on machine learning (ML), and DevOps with incremental safety assurance. Much of the foundation for these practices is still under development and creates rich opportunities for research.

II. RE for Automated Driving

Requirements engineering for automated driving (READ) is challenging [1]. Driving is a comparatively simple task for humans to perform, but hard to specify. Any general driving rule, such as determining a safe following gap, is subject to a myriad of context-dependent exceptions and adjustments. Further, roads are open environments, with the possibility of new, unseen situations. Thus, READ needs to be data-driven and continuous, allowing expert-assisted extraction of driving specifications from traffic data. Human experts are necessary to judge the encountered situations and select the most appropriate responses. READ also needs to generate requirements on what data and how much to collect.

AVs are not just replacements for human-operated vehicles. Automation fundamentally changes some aspects of driving. For example, the absence of a human driver eliminates driver-pedestrian communication, requiring other means to communicate intent. Rather than just copying human driving, AV behavior needs to be creatively designed, calling for a continuous, data-driven, experimental design cycle: new behaviors are rolled out and telemetry data is collected and analyzed to evolve the behaviors based on the insights from the field.

Automated driving will have a huge impact on the society, and the strategies that drive the decisions that AV fleets make need to be explicitly managed. The human values embedded in these strategies must be elicited and represented explicitly, and methods for doing so are still in their infancy [1], [2].

III. ADS Architecture

Automated driving systems are complex [3]. Perception functions interpret sensor data to build models of the static and dynamic environment, including models of future behavior. Using these, planning functions decide maneuvers and implement them as precise trajectories and actuator commands. Additional support functions are needed to ensure dependable operation in varying environmental conditions, system health, and mission types, such as passenger or cargo delivery or empty trips for pick-up.

Thus, an ADS requires a highly modular and reconfigurable architecture and the application of dependability patterns. An ADS typically relies on a publish-subscribe framework, such as Robot Operating System (ROS) or Adaptive AUTOSAR, which defines components with message-based interfaces and easy runtime reconfiguration. ADS architectures use multiple dependability patterns, including monitoring of system assumptions and system health [4], redundancy and diversity, and simplex architecture [5]. A promising direction is to formalize interfaces and apply automated assume-guarantee reasoning to show that the architecture adequately addresses hazardous failures of components and their communication [6].

IV. ML-Based Functions

Machine learning (ML) is indispensable for implementing perception functions, and it plays an increasingly important role in behavioral planning. Today's ADSs rely on supervised learning using deep neural networks for perception, which currently outperforms any other technology for computer

Extended abstract of a Technical Briefing presentation at ICSE'19

DOI 10.1109/ICSE-Companion.2019.00024

vision tasks [7]. Behavioral planning using manually crafted rules typically results in suboptimal driving policies, because of the huge number of exceptions to general driving rules that are required. Machine learning can exploit road user behavior data collected in traffic to produce sophisticated driving policies. A winning approach to behavior planning has not yet emerged; however, hybrid approaches combining reinforcement and imitation learning with manually crafted policies appear most promising [8], [9].

Machine learning has a profound impact on the development of automated driving software. Major new engineering activities include collection and curation of large datasets, design of neural network architectures for perception and planning tasks, and ensuring their effective training. As a result, engineering of automated driving software now involves data and training engineers in addition to software engineers.

A major challenge is safety assurance of systems that rely on machine learning. Traditional safety assurance of complex automotive software according to ISO 26262 relies on the presence of a complete specification for the safety-critical systems, and the ability to inspect the program logic. Both aspects are lacking for ML-based functions, however [10]. A subset of verification and validation methods intended for programmed software, such as testing and monitoring, are also applicable for assuring ML-based functions, but new methods are needed, including dataset-requirements engineering, measuring similarity between the training data and inputs at runtime, uncertainty management, architectural patterns to detect and mitigate failures of ML-based components, and methods to enable human interpretability and machine analyzability of learned logic (see [10] for a detailed discussion).

V. DevOps and Incremental Assurance for ADS

The data-driven and continuous nature of ADS development calls for a DevOps approach. The key idea behind DevOps is to deliver small increments of functionality frequently and with short delays, and to evaluate them using telemetry data collected in the field [11]. Over the air (OTA) updates enable DevOps in the automotive context [12]. While DevOps is a proven paradigm in cloud computing, it is challenging for safety-critical systems.

An ADS is safety-critical, and deploying ADS updates requires an adequate safety assurance process. A key practice is to develop and maintain a safety case, which is a rigorous argument supported by evidence that all relevant hazards have been identified and eliminated or sufficiently controlled or mitigated. DevOps requires incremental safety cases, which argue about the safety of each system update, and, for each piece of evidence, the adequacy of reusing or updating it. Incremental safety assurance is an active area of research, and the application of model-based engineering appears to be a promising approach to support the evolution of safety cases and reduce the risk of oversights during updates [13].

Standards such as ISO 26262 (Part 6) and ISO/PAS 21448 prescribe the verification and validation techniques to be used in engineering automotive software, and testing methods play a central role among them. A standard test pipeline takes functionality through simulation, closed course, and field tests. Computer simulation allows testing dangerous situations, but ensuring adequate realism is challenging. Closed course also allows testing dangerous situations, for example, using foam car targets on robotic platforms, but the environment complexity is usually limited and the tests are expensive. Mixed reality testing can inject virtual objects into vehicle's perception in closed course tests. Field testing offers the ultimate realism, but it lacks repeatability and cannot target dangerous situations.

Major challenges include test design, deciding what tests to perform at each stage, and deciding how much testing is enough. A standard approach to system testing is scenario based, with scenarios derived from the operational design domain (ODD). Combinatorial test methods are required because of the many varying ODD parameters, such as the number and behavior of road users, road geometry, and environmental conditions. Each test stage should focus on its strengths, and later stages should not merely replicate the tests from earlier stages, but rather focus on the assumptions and limitations of the previous stage [14]. For example, if certain types of sensor noise could not be faithfully produced in simulation, the closed-course test should target creating the specific noise conditions to test their effects on the ADS decisions. A major challenge is to decide the amount of testing needed to demonstrate safety. Using field testing alone is impractical for a proven-in-field argument, as billions of kilometers would be required to demonstrate a comparable or better failure rate than that of the best human drivers [15]. Billions of kilometers are still needed to validate system functionality due to the openness of the road environment, but verification testing in response to system updates does not need to replicate all these kilometers in the field [14].

Promising approaches to address the testing challenges are accelerated testing, fleet testing including a shadow mode, and incremental testing and arguments. Accelerated testing using rare event sampling methods actively seeks out rare and critical events, avoiding testing empty kilometers [16], [17]. These methods target simulation and closed course testing, but field tests can also be planned to maximize disengagement rates and thus learning. Field tests can be scaled with fleet testing, which opens up opportunities to test different configurations in parallel and to learn from the entire fleet. While field tests are typically performed with safety drivers on board, shadow mode testing, which is widely used in DevOps, can be safely deployed on vehicles used by customers. Field testing methods must be coupled with adequate runtime monitoring to collect relevant field data and to assure the safety of testing. These different testing methods are still in their early development stages and offer rich opportunities for research.

VI. Conclusion and Future Directions

Software engineering for automated vehicles requires a DevOps-style process to allow innovating and optimizing automated driving policies in complex, open road environments.

The special needs and emerging practices in engineering automated driving software create an exciting space for research, which includes data-driven, continuous, and values-based requirements engineering; adaptive and dependable software architectures; and incremental safety assurance, especially in the presence of ML-based functions.

References

[1] K. Czarnecki, "Requirements engineering in the age of societal-scale cyber-physical systems: The case of automated driving," in *2018 IEEE 26th International Requirements Engineering Conference (RE)*, Aug 2018, pp. 3–4.

[2] S. M. Thornton, F. E. Lewis, V. Zhang, M. J. Kochenderfer, and J. C. Gerdes, "Value sensitive design for autonomous vehicle motion planning," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1157–1162.

[3] R. Matthaei and M. Maurer, "Functional system architecture for an autonomous on-road motor vehicle," in *Automotive Systems Engineering II*. Springer, 2018, pp. 93–120.

[4] I. Colwell, B. Phan, S. Saleem, R. Salay, and K. Czarnecki, "An automated vehicle safety concept based on runtime restriction of the operational design domain," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, June 2018, pp. 1910–1917.

[5] A. Kohn, R. Schneider, A. Vilela, A. Roger, and U. Dannebaum, "Architectural concepts for fail-operational automotive systems," SAE Technical Paper, Tech. Rep., 2016.

[6] D. Cofer, A. Gacek, J. Backes, M. W. Whalen, L. Pike, A. Foltzer, M. Podhradsky, G. Klein, I. Kuz, J. Andronick, G. Heiser, and D. Stuart, "A formal approach to constructing secure air vehicle software," *Computer*, vol. 51, no. 11, pp. 14–23, Nov. 2018. [Online]. Available: doi.ieeecomputersociety.org/10.1109/MC.2018.2876051

[7] J. Janai, F. Güney, A. Behl, and A. Geiger, "Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art," *arXiv preprint arXiv:1704.05519*, 2017.

[8] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," 2018.

[9] C. Li and K. Czarnecki, "Urban driving with multi-objective deep reinforcement learning," 2018.

[10] R. Salay and K. Czarnecki, "Using machine learning safely in automotive software: An assessment and adaption of software process requirements in iso 26262," 2018.

[11] L. Bass, I. Weber, and L. Zhu, *DevOps: A software architect's perspective*. Addison-Wesley Professional, 2015.

[12] T. Chowdhury, E. Lesiuta, K. Rikley, C.-W. Lin, E. Kang, B. Kim, S. Shiraishi, M. Lawford, and A. Wassyng, "Safe and secure automotive over-the-air updates," in *Computer Safety, Reliability, and Security*, B. Gallina, A. Skavhaug, and F. Bitsch, Eds. Springer, 2018, pp. 172–187.

[13] Z. Diskin, T. Maibaum, A. Wassyng, S. Wynn-Williams, and M. Lawford, "Assurance via model transformations and their hierarchical refinement," in *MODELS 2018*. ACM, 2018, pp. 426 – 436.

[14] P. Koopman and M. Wagner, "Toward a framework for highly automated vehicle safety validation," *SAE Technical Paper, 2018-01-1071*, 2018.

[15] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016.

[16] D. Zhao, H. Lam, H. Peng, S. Bao, D. J. LeBlanc, K. Nobukawa, and C. S. Pan, "Accelerated evaluation of automated vehicles safety in lane-change scenarios based on importance sampling techniques," *IEEE transactions on intelligent transportation systems*, vol. 18, no. 3, pp. 595–607, 2017.

[17] R. B. Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing vision-based control systems using learnable evolutionary algorithms," in *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*. IEEE, 2018, pp. 1016–1026.