International Journal of Advances in
Soft Computing and Intelligent Systems (IJASCIS)
2025, Vol 04, Issue 01, 60-73
ISSN: 3048-4987

Original Paper

SOFTWARE MAINTENANCE IN AUTOMOTIVE AND AEROSPACE APPLICATIONS – AN OVERVIEW

[a]Vikas Vyas, [b]Zheyuan Xu

[a]vikas.vyas@mercedes-benz.com, [b]zheyuan.xu@mercedes-benz.com

ABSTRACT

It is of vital importance for safety-critical applications in automotive and aerospace systems to receive timely software maintenance. Systems deemed safety-critical are those wherein failure could result in loss of life, property damage, or environmental harm, and are extensively used in aircraft, vehicles, medical devices, nuclear reactors, and more. Consequently, standardized and regular software maintenance is imperative to guarantee these systems stay secure and sheltered against vulnerabilities. This paper gives an overview of the role of software maintenance in automotive and aerospace domains, emphasizing the necessity of preserving an appropriately maintained software environment. It additionally addresses particular difficulties faced in the automotive and aerospace industries, widespread practices and approaches, and developing trends in software maintenance. The paper starts with a comparative evaluation of the maintenance practices across these domains, stresses the commonalities and differences, and outlines potential research directions to further enhance the safety and reliability of software in these critical applications.

KEYWORDS

Software maintenance, safety-critical applications, automotive, aerospace.

1. INTRODUCTION

1.1 Background

Systems that are safety-critical are those for which failures could cause loss of life, property damage, or environmental harm due to their important functions relying heavily on software integrity for overall safety assurance. The challenges of integrating agile practices into such safety-critical system development are considerable.

While large-scale agile frameworks like SAFe and LeSS have streamlined processes in the automotive sector, carefully customizing their rapid cycles to mandate safety cases is imperative as vehicle software complexity escalates, with high-end product now containing over a hundred million lines of code requiring rigorous validation and verification [19]. However, with diligence, progressive iterative refinement aligned with risk-based priorities and independent assessment can enable both productivity and protection when developing innovations reliant on trustworthy software functionality for public security.

Incident reporting remains crucial to risk governance in safety- sensitive contexts. However, traditional platforms relying heavily on relational databases encounter difficulties including data extraction bias, precision and recall imperfections, abstraction challenges, and inter-examiner consistency issues [1]. To tackle these complications approaches such as computer-aided monitoring, free-form recovery and probabilistic reasoning, and conversational case-based rationalization have been recommended [1].

Software quality assurance in safety-critical domains rightly demands increased attention. It is particularly relevant in the context of the automotive industry, where software drives innovation and enables the delivery of new features [19]. The paper also reviews various safety-critical applications across automotive, aerospace, and other domains. The increasing prevalence of software-defined vehicles (SDV) is transforming the automotive industry from technology, products, services, and enterprise competition perspectives [21]. In the aerospace sector, the shift from time-based to condition-based maintenance, driven by advancements in sensor technology, necessitates sophisticated data analysis techniques, with deep learning emerging as a promising solution [27].

1.2 Objectives and Scope

The primary objective of this paper is to provide a comprehensive overview of software maintenance in safety-critical applications, with a particular focus on the automotive and aerospace industries. The paper highlights the importance of standardized software maintenance methodologies, identifies common challenges, and discusses current practices and emerging trends. The scope of this paper includes an examination of various types of maintenance, the specific challenges faced in the automotive and aerospace sectors, and a comparative analysis of maintenance practices across these domains.

1.3 Structure of the paper

In this paper, section 2 defines software maintenance and discusses its importance in safety-critical applications, out- lining different types of maintenance activities. Section 3 provides an in-depth look at the automotive industry's software systems, maintenance challenges, current practices, and tools, including the impact of software-defined vehicles [21]. Section 4 focuses on the aerospace industry, exploring the challenges of developing software for aerospace systems [24] and the application of artificial intelligence in aerospace maintenance [27], [28]. Section 5 showcases a comparative analysis of soft- ware maintenance practices across automotive and aerospace domains, highlighting common principles and domain-specific differences. Section 6 explores emerging trends and future directions in software maintenance, including the role of automation, AI, and predictive maintenance [14], [29]. Section 7 concludes the paper with final thoughts on the importance of software maintenance in safety-critical applications.

2. SOFTWARE MAINTENANCE IN SAFETY-CRITICAL APPLICATIONS

2.1 Definition and Importance

Software maintenance, which includes the modification of either software source code or binary after delivery [7], is crucial for correcting bugs, improving performance, and ensuring consistent user experience [8]. In safety-critical application fields like automotive and aerospace, where software is increasingly coupled with crucial functionalities, the reliability, and performance of this software are critical to overall system safety [1], [5].

Recent research mentions several aspects of software maintenance in safety-critical environments:

- Standards on functional safety and fail-safe design approaches are non-negotiable for ensuring system integrity [2], [5].
- Developing comprehensive, clear, and verifiable requirements specifications is a major challenge, demanding sustained stakeholder alignment throughout the system's lifespan [3], [24].
- While software implementation realizes complex functionalities and mitigates possible risks, it can bring in new issues if not rigorously maintained and updated [1], [4].
- The growing dependence on modern software systems in these sectors requires effective software maintenance, which is crucial given the potential for substantial financial consequences or even loss of life in case of system failures [5], [23].

2.2 Types of Maintenance

2.2.1 Corrective Maintenance

Corrective maintenance, as a type of reactive maintenance approach to correct identified software problems, is indispensable for enforcing the reliability and integrity of safety-critical software systems in the automotive and aerospace industries [7]. Studies show that software maintenance constitutes a

significant portion, often 60-90%, of the total cost of ownership for a software product [8], while a substantial portion of maintenance efforts, approximately 75%, is dedicated to adaptive and perfective maintenance. As a result, developing robust maintenance models that minimize rework expenses and ensure consistent customer experience is paramount, especially in domains where software failures carry severe consequences [7].

2.2.2 Adaptive Maintenance

Adaptive maintenance, on the other hand, emphasizes the compatibility and functionality of 1 software applications within changing environments. This includes adapting to hardware changes, operating system changes, software dependency modifications, and changes regulation and quality assurance requirements [9]. This form of maintenance is concerned about modifying software after the initial deployment phase to align with these requirements, ensuring its continued reliability and performance [9]. Automation in adaptive maintenance tasks increases efficiency in managing the evolution of software systems, often taking advantage of high-level specifications to generate specialized code for this purpose [9].

2.2.3 Perfective maintenance

Perfective maintenance 1 aims to refine software applications by enhancing performance and ease of updates. As mentioned in [11], developers focus on improving efficiency, reliability, and flexibility for potential changes in the source code. Visual tools can help highlight areas for improving code quality, lowering subsequent expenses.

2.2.4 Preventative Maintenance

Preventive maintenance, as one of the proactive maintenance approaches, involves modifying target software to detect and address potential fault and failures before they escalate into more severe issues [13]. This approach aims to mitigate software failure risks, minimize downtime, and ensures the reliability and performance of critical systems [13]. A well-structured preventive maintenance program is essential in safety-critical domains, where software failures can lead to severe consequences, resolving issues before they escalate or impact other modules.

2.2.5 Predictive maintenance

Predictive maintenance 2 aims to identify anomalies and forecast failures before they happen through leveraging data analytics and artificial intelligence. Machine learning algorithms and statistical models are applied to runtime system data in an effort to predict and prevent issues that could lead to downtimes or system failures. Research indicates this approach can help boost reliability in sectors like automotive manufacturing, where anomaly detection, inference methods, and optimized maintenance strategies [14], [15] take advantage of learned patterns from empirical performance data [29].

3. AUTOMOTIVE INDUSTRY

3.1 Overview of Automotive Software Systems

Modern-day automotive systems rely heavily on vehicle firmware for the realization of various functions, ranging from basic bring-up and operation of the steering and fuel system to advanced driver-assistance functionalities. The increasing complexity of these systems coupled with the introduction of the software-defined vehicles (SDVs) concept present unique challenges for software maintenance in the automotive industry [21].

3.1.1 Types of software used in automotive systems

Auto- motive software covers a wide range of applications, which include:

- Embedded Firmware: firmware deployed in embedded hardware serves as the backbone of various vehicle functions, and controls critical aspects like engine ignition, braking, and infotainment systems [20].
- Model-based Software Development: To manage the growing complexity and shortened development cycles, the automotive industry increasingly relies on automotive V-model software development. This approach utilizes modular design approaches to enhance efficiency and enforce safety standard compliance [17].

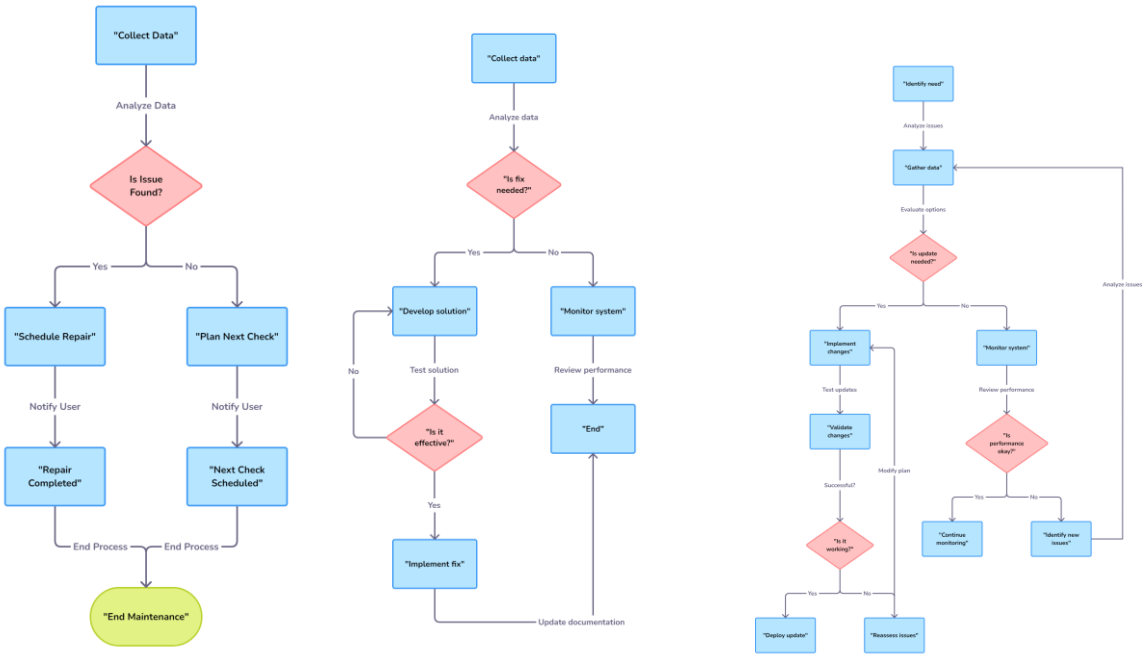

Fig.1. Typical workflow of preventative(left), corrective(middle) and adaptive(right) maintenance in automotive software. Along with perfective maintenance, they form the 4 essential phases in modern software maintenance.

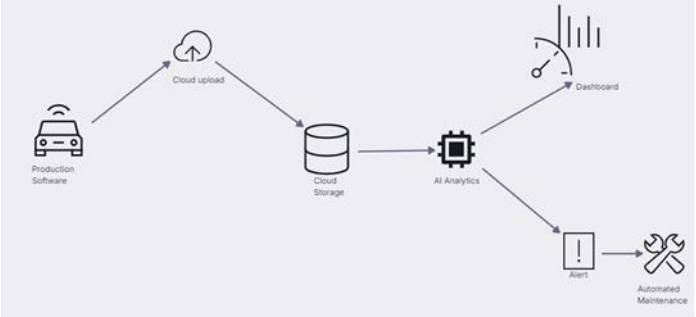

Fig.2. Predictive maintenance workflow with AI Analytics algorithms.

- Vehicular Internet-of-Things (IoT) and Telecommunication: Connected vehicles leverage IoT and telecommunication systems to exchange data with backend servers, ap- plications, and various vehicular components, to achieve objectives like smart mobility and advanced vehicle-based services [17].

3.2 Characteristics and requirements

Automotive software systems possess unique characteristics and adhere to strict safety, performance, and reliability requirements.

3.2.1 Standardization and Modularity

The AUTomotive Open System ARchitecture (AUTOSAR) standard is key to the development of automotive software and system architectures, addressing modularity, variant and extension handling, and standardized execution management practices [18].

3.2.2 Safety and Security

Given the safety-critical nature of automotive systems, compliance with industry standards like MISRA and AUTOSAR, along with robust practices addressing cyber security, is crucial for meeting safety and security requirements [19].

3.3 Maintenance Challenges in Automotive

Maintenance of automotive software brings in several challenges due to the convoluted and safety-critical nature of the systems:

The exponential increase in software complexity of vehicular software brings considerable challenges for traditional software engineering practices. This complexity partially arises from the coupling of various systems and the increasing demand for more intricate and advanced functionality [19], [20].

- On the other hand, automotive software development is conducted under strict constraints, including cost, quality requirements, tight deadlines, etc. These constraints facilitate efficient maintenance strategies and tools [20].
- The trend towards software-defined vehicles (SDVs) concepts introduces new challenges.

Conventional research and development models need to evolve to accommodate the trend and new standards of SDVs [21].

3.4 Current Practices and Approaches

There are several common-practice approaches and method- ologies employed to address the challenges posed by the maintenance of automotive software:

- Lifecycle Plan: It is important to plan the management of the complete lifecycle of automotive software, taking into account factors like hardware constraints, over-the- air (OTA) updates, long-term maintenance, and eventual phase-out of the platform. Addressing these factors ensures the sustainability and reliability of automotive software throughout its entire lifecycle [19].
- Shift-Left Testing and Verification: Early and continuous testing throughout the software development lifecycle is essential for automotive applications. Model-based shift- left testing, static analysis, and dynamic verification techniques are crucial for identifying and resolving issues early on, contributing to improved software quality and reduced development time [17].

Various tools and frameworks support the development and maintenance of automotive software:

- Advanced Development Tools: The limitations of traditional automotive software frameworks necessitate the adoption of more advanced tools and development approaches. Tools like Eclipse Zenoh and frameworks that support building scalable architectures are becoming increasingly important in managing the complexity of modern automotive software [17].
- Cybersecurity Frameworks: With the rise of connected vehicles, robust cybersecurity measures are paramount. Open and flexible frameworks specifically designed for automotive applications, such as PENNE, are crucial for enhancing cybersecurity training and evaluating the effectiveness of in-vehicle network security concepts [17].

By adopting these practices, tools, and frameworks, the automotive industry can strive to overcome the challenges of maintaining increasingly complex software systems while ensuring the safety, reliability, and security of modern vehicles.

4. AEROSPACE INDUSTRY

4.1 Overview of Aerospace Software Systems

4.1.1 Types of software used in aerospace systems

Aerospace systems rely heavily on a diverse range of software applications to ensure safe and efficient operation. These applications encompass various functionalities, including:

- Flight control systems: Responsible for managing aircraft stability, navigation, and autopilot functions [22]– [25].
- Engine control systems: Governing engine performance, fuel efficiency, and monitoring engine

health [27].

- Avionics systems: Encompassing communication, navigation, surveillance, and cockpit display systems [26].
- Mission control software: Used for spacecraft command, control, and data handling in space exploration missions [24], [25].
- Unmanned Aerial Vehicle (UAV) control systems: Enabling autonomous flight, navigation, and mission execution for UAVs [28].

4.1.2 Key Characteristics and Requirements

Software in aerospace systems must meet stringent requirements due to the industry's emphasis on safety, reliability, and mission- criticality. These requirements include:

- High Reliability and Safety: Aerospace software must adhere to rigorous safety standards to minimize the risk of failures that could result in loss of life, environmental damage, or mission failure [22], [23], [25].
- Deterministic Behavior: The software must operate predictably and consistently, especially in real-time systems where timing constraints are critical [26].
- Certification and Compliance: Adherence to industry standards (e.g., DO-178B, ARP4754A) is mandatory to ensure software quality and safety [25], [26].
- Robustness and Fault Tolerance: The software must be able to withstand and recover from unexpected events, such as hardware failures or environmental disturbances [23].
- Security: Protecting aerospace systems from cyberattacks is crucial, demanding robust security measures within the software [28].

4.2 Maintenance Challenges in Aerospace

4.2.1 Specific challenges in maintaining aerospace software

Maintaining aerospace software presents unique challenges due to the industry's specific constraints and operational environment. Some key challenges include:

- System Complexity: Modern aircraft and spacecraft com- prise intricate, interconnected systems, making it difficult to understand the impact of software changes and potential ripple effects [23], [25].
- Safety-Critical Nature: Even minor software errors can have catastrophic consequences, requiring rigorous testing and verification processes [23].
- Long System Lifecycles: Aerospace systems often operate for decades, necessitating ongoing maintenance and updates to address evolving requirements and technology advancements [23].

Access and Downtime Constraints: Accessing software components for maintenance can be challenging, especially in flight-critical systems. Downtime must be minimized to maintain operational efficiency [27]. Due to safety and security concerns, software in aerospace domains is usually accessible to the outside via a wired, ad-hoc maintenance window.

- Data Management and Analysis: Modern avionics system generates a great amount of data and log records for each of its operations. Effectively strategies for management, analysis, and utilization of this data for maintenance purposes are crucial [27].
- Certification and Qualification: Software updates often require re-certification, which can be time-consuming and expensive [26].

4.3 Current Practices and Approaches

4.3.1 Common Practice Approach and Mythologies

A range of common practice maintenance approaches are employed by the aerospace industry to address existent challenges related to software maintenance:

- Model-Based Development: Using models to represent software systems throughout the development lifecycle, enabling early verification and validation [17], [19].
- Formal Methods: Utilizing mathematical techniques for software specification, design, and verification to ensure correctness and reliability [24], [25].
- Condition-Based Maintenance (CBM): Employing sensor data and predictive analytics to anticipate maintenance needs and optimize maintenance schedules [27], [31].
- Deep Learning (DL): Leveraging DL algorithms for tasks such as anomaly detection, fault diagnosis, and remaining useful life (RUL) estimation [27].
- Digital Twins: Creating virtual representations of physical assets to simulate behavior, analyze data, and predict maintenance needs [14], [30].

4.3.2 Tools and frameworks

Several tools and frameworks support aerospace software maintenance:

- Integrated Development Environments (IDEs): Specialized IDEs for aerospace software development and de- bugging, often integrated with certification and verification tools.
- Testing and Simulation Tools: Tools for conducting rigorous software testing, including unit testing, integration testing, and system-level simulation.
- Configuration Management Tools: Managing software versions, configurations, and release processes to ensure traceability and control.
- Data Analytics Platforms: Platforms for collecting, storing, processing, and visualizing aircraft data to support maintenance decision-making.

5. COMPARATIVE ANALYSIS

5.1 Similarities Across Domains

While the automotive and aerospace industries are drastically different fields and have distinct software maintenance requirements, they share similarities in the approach to per- form software maintenance for safety-critical applications:

- **Safety and Reliability Focus:** Both fields prioritize the safety and reliability of their systems over other requirements. Software failures in either industry can cause catastrophic consequences, mandating the need for rigorous development, integration testing, and maintenance practices.
- Standards and Certification Process: Both industries strive to meet strict safety standards and regulations. Automotive software must comply with various standards such as ISO-26262, while aerospace software complies with the guidelines like DO-178C. Compliance with these standards is essential for ensuring system safety and completion of necessary certifications.
- Growing Software Complexity: Both industries are experiencing an increase in software complexity, which requires sophisticated maintenance strategies.
- Model-Based Development Strategies: Both industries leverage model-based development techniques to manage and decouple entangled software modules and improve the efficiency of software development and maintenance processes.
- Data Analytics and Management: The usage of sensors and data logging in both automotive and aerospace sys- tems generates a great amount of data. Both industries are exploring the potential of data analytics tools and data- driven methods to automate maintenance practices and predict potential failures.

5.2 Domain Specific Differences

Nevertheless, several differences exist between the automotive and aerospace industries related to software maintenance, including:

- **System Lifecycle:** Aerospace systems usually have much longer lifecycles than automotive systems. Aircraft often stay operational for decades and require long-term maintenance and support strategies for software systems. On the contrary, automotive software lifecycles are shorter due to intense competition in the field and rapid iteration of software and hardware. This difference affects maintenance strategy, with aerospace systems requiring greater emphasis on long-term sustainability and retirement management, while automotive systems require affordability and ease of maintenance.
- **Software Development and Certification Cost:** In general, the aerospace industry incurs

higher development and certification costs compared to the automotive industry. The strict safety requirements and complexity of aerospace systems, coupled with the difficulty of acquiring operational data, mandate extensive testing, verification, validation, and formal certification processes, which all contribute to the cost.

- **Production Volumes and Iteration Cycle:** The automotive industry has significantly higher production quantity compared to the aerospace industry, which affects how software updates are deployed and managed. Automotive manufacturers see wide application of over-the-air (OTA) updates for efficient software distribution and installation across large fleets of vehicles. On the other hand, aerospace software updates involve more complex procedures due to the complexity of the systems and rigorous safety regulations.
- Operational Conditions: Aerospace systems usually operate in harsher and more diverse environmental conditions than automotive systems. Common factors like adverse weather conditions, cruise speed, fatigue of air- frame, and extreme temperature differences pose unique challenges for aerospace software maintenance, mandating robust solutions that can withstand the worst conditions.
- Human Factor Considerations: While both industries emphasize safety, the aerospace industry prioritizes more on the human factor. Pilots and astronauts rely on the integrity of software systems for safety-critical tasks, thus human-machine interface design in the aerospace domain focuses more on reliability and robustness, rather than fanciness or advanced features.

6. EMERGING TRENDS AND FUTURE DIRECTIONS

6.1 Trends in Software Maintenance

The software maintenance landscape, especially the one pertaining to safety-critical industries such as automotive and aviation, is constantly changing. Several notable trends are reshaping software maintenance in these domains:

- Artificial Intelligence and Machine Learning (AI/ML): The adoption of artificial intelligence and machine learning techniques is reshaping predictive maintenance land- scape [14], [29] by introducing automation in the pro- cess. By analyzing empirical data and past performance, AI/ML algorithms can identify patterns, predict potential failures, and enable preemptive maintenance [27], [28]. While AI/ML demonstrates potential in predictive main- tenance, maintenaing the reliability, trustworthiness, and interpretability of these models is crucial.
- Digital twins, which are virtual representations of physical assets, hold great promise in revolutionizing engineering practices across industries [14], [30]. By replicating real-world operational conditions within interactive simulators, digital twins allow engineers to test and validate design modifications without risking costly failures. Whereas, the adoption of digital twins requires standardized data format and communication protocols to enable platform-

agnostic data exchange and interoperability. Establishment of such standards will accelerate adoption and maximize the benefits of digital twins in maintenance applications.

- Over-the-Air (OTA) Updates: OTA software updates are gaining popularity, especially in the automotive industry [21]. The ability to deliver software updates remotely and seamlessly allows manufacturers to achieve faster software iteration and deliver timely performance improvement and security fixes to the vehicle firmware. However, ensuring the safety and reliability of OTA updates in environments with constrained hardware re- sources remains a tough software engineering challenge.
- Cyber Security: As vehicles and aircraft become increasingly connected, cybersecurity threats become a more prominent issue. As a result, there is an increasing need for robust cybersecurity measures integrated into software maintenance practices, which include secure-aware software development practices, periodic security audits, and timely patching of exploits and vulnerabilities to mitigate the risk of cyberattacks.

7. CONCLUSION

As an ongoing process essential to the operational safety, re- liability, and security of safety-critical systems, continued soft- ware maintenance is crucial in the automotive and aerospace industries. As an inevitable trend, AI/ML, digital twins, OTA updates, and increased automation, hold great potential for revolutionizing software maintenance practices. However, they introduce new challenges, such as security concerns, model interpretability and explainability, and ethical considerations. As software continues to be deployed on more vehicular components, the importance of effective software maintenance will keep growing. Research, collaboration, and innovation across multiple domains are key to meeting the evolving demands of these safety-critical domains and ensuring road safety and well-being of system operators and passengers.

Authors Contributions (Compulsory)

All authors have equally contributed.

REFERENCES

1. Johnson, C. (2000). Software Support for Incident Reporting Systems in Safety-Critical Applications. In: Koornneef, F., van der Meulen, M. (eds) Computer Safety, Reliability and Security. SAFECOMP 2000. Lecture Notes in Computer Science, vol 1943. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-40891-6 9
2. Shaw, R. ”Safety-critical software and current standards initiatives,” Computer Methods and Programs in Biomedicine, vol. 44, no. 1, pp. 5-22, 1994, doi: 10.1016/0169-2607(94)90143-0.
3. Martins, L. E. G. and T. Gorschek,” Requirements Engineering for Safety-Critical Systems: Overview and Challenges,” in IEEE Software, vol. 34, no. 4, pp. 49-57, 2017, doi: 10.1109/MS.2017.94.
4. John C. Knight. 2002. Safety critical systems: challenges and directions. In Proceedings of the 24th

International Conference on Software Engi- neering (ICSE ’02). Association for Computing Machinery, New York,
NY, USA, 547–550. https://doi.org/10.1145/581339.581406
5. Pietrantuono, R., & Russo, S. (2013). Introduction to Safety Critical Systems.
6. Antinyan, V. (2023). Seven Lessons Learned From Automotive Software Supplier Collaborations. IEEE Software, 40, 77-85.
7. Gorla, N. ”Techniques for application software maintenance,” Informa- tion and Software Technology, vol. 33, no. 1, pp. 65-73, 1991, doi: 10.1016/0950-5849(91)90025-7.
8. Uttamjit Kaur, Gagandeep Singh . A Review on Software Maintenance Issues and How to Reduce Maintenance Efforts. International Journal of Computer Applications. 118, 1 (May 2015), 6-11. DOI=10.5120/20707-
3021
9. Tansey, W. (2008). Automated Adaptive Software Maintenance: A Methodology and Its Applications.
10. Lapouchnian, A. (2011). Exploiting Requirements Variability for Soft- ware Customization and Adaptation.
11. Tru¨mper, J . , M. Beck and J. Do¨llner, ”A Visual Analysis Approach to Support Perfective Software Maintenance,” 2012 16th International Conference on Information Visualisation, Montpellier, France, 2012, pp. 308-315, doi: 10.1109/IV.2012.59. keywords: Visualization;Measurement;Software systems;Maintenance engineering;Complexity theory;Context;Software maintenance;Visualization;Quality management
12. Rine, D. ”Software perfective maintenance: Including retrainable soft- ware in software reuse,” Information Sciences, vol. 75, no. 1, pp. 109- 132, 1993, doi: 10.1016/0020-0255(93)90116-4
13. Ab-Samat, H., Jeikumar, L.N., Basri, E.I., Harun, N.A., & Kamaruddin, S. (2012). Effective Preventive Maintenance Scheduling : A Case Study
14. Arena, F., Collotta, M., Luca, L., Ruggieri, M., & Termine, F. (2021). Predictive Maintenance in the Automotive Sector: A Literature Review. Mathematical and Computational Applications
15. Tessaro I, Mariani VC, Coelho LdS. Machine Learning Models Applied to Predictive Maintenance in Automotive Engine Components. Proceed- ings. 2020; 64(1):26. https://doi.org/10.3390/IeCAT2020-08508
16. Schaefer, J., Christlbauer, H., Schreiber, A., Reith, G., Jonker, M., Potman, J., Dannebaum, U., & Eissfeldt, T. (2021). Future Automo- tive Embedded Systems Enabled by Efficient Model-Based Software Development.
17. Dajsuren, Y., & Brand, M.V. (2019). Automotive Software Engineering: Past, Present, and Future. Automotive Systems and Software Engineer- ing.
18. Staron, M. (2021). AUTOSAR (AUTomotive Open System ARchitec- ture). In: Automotive Software Architectures. Springer, Cham. https:
//doi.org/10.1007/978-3-030-65939-4 5
19. Hanselmann, H. (2008). Challenges in automotive software engineering. ICSE Companion ’08.
20. Zhai, Y., Vetter, A., & Sax, E. (2023). Analysis of Current Challenges of Automotive Software in the View of Manufacturing. SAE Technical Paper Series.
21. Liu, Z., Zhang, W., & Zhao, F. (2022). Impact, Challenges and Prospect of Software-Defined Vehicles. Automotive Innovation, 5, 180 - 194.
22. Lundqvist, K., and J. Srinivasan, ”A First Course in Software Engineering for Aerospace Engineers,” 19th Conference on Software Engineering Education & Training (CSEET’06), Turtle Bay, HI, USA, 2006, pp. 77- 86, doi: 10.1109/CSEET.2006.5.
23. Filho, P.S. (2018). The growing level of aircraft systems complexity and software investigation.
24. Vassev, E., Hinchey, M. (2014). Software Engineering for Aerospace: State of the Art. In: Autonomy Requirements Engineering for Space Missions. NASA Monographs in Systems and Software Engineering. Springer, Cham. https://doi.org/10.1007/978-3-319-09816-6 1
25. Vassev, E., Hinchey, M. (2012). Fundamentals of Designing Complex Aerospace Software Systems. In: Hammami, O., Krob, D., Voirin, JL. (eds) Complex Systems Design & Management. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-25203-7 4
26. Aubrey, D. (2023). The Future of Avionics: High Performance, Machine- Learned and Certified.
27. Rengasamy, D., Morvan, H.P., & Figueredo, G.P. (2018). Deep Learning Approaches to Aircraft Maintenance, Repair and Overhaul: A Review. 2018 21st International Conference on Intelligent Transportation Sys- tems (ITSC), 150-156.
28. Hassan, K., Thakur, A.K., Singh, G. et al. Application of Artificial Intelligence in Aerospace Engineering and Its Future Directions: A Systematic Quantitative Literature Review. Arch Computat Methods Eng (2024). https://doi.org/10.1007/s11831-024-10105-7
29. Ucar A, Karakose M, Kırımc¸a N. Artificial Intelligence for Predictive Maintenance Applications: Key Components, Trustworthiness, and Fu- ture Trends. Applied Sciences. 2024; 14(2):898. https://doi.org/10.3390/ app14020898
30. Moleda M et al. From Corrective to Predictive Maintenance—A Review of Maintenance Approaches for the Power Industry. Sensors. 2023; 23(13):5970. https://doi.org/10.3390/s23135970

31. Taheri, E., Kolmanovsky, I.V., & Gusikhin, O. (2019). Survey of prognostics methods for condition-based maintenance in engineering systems. ArXiv, abs/1912.02708.

Vikas Vyas has experience from both the automotive and aerospace industries with a proven track of successful record. With over 15 years of experience, his expertise is in program and product management and the development of cutting-edge technologies. Serving as a technical lead at Mercedes-Benz Research & Development North America, Vikas stands at the frontier of autonomous driving, oversees market analysis, research, and development of advanced automotive solutions, and shapes the future of mobility. Prior to his role at Mercedes-Benz, he led global teams at Bosch USA, where he participated in the development of safety- critical EV steering systems with integrated cybersecurity. His career began at Rockwell Collins, focusing on software for aerospace display systems. Vikas holds a certificate from the University of California, Berkeley, Haas School of Business, and an engineering degree from Rajasthan Technical University, Kota, India. He has four pending US patents and has authored multiple articles and chapters. As a senior member of IEEE and an active member in SAE, Vikas is dedicated to advancing industry standards and knowledge.

Zheyuan Xu works in the autonomous driving field. His expertise is in mechatronics, ADAS software development and over-the-air (OTA) update algo- rithm innovation, and works as a software solution provider with automotive clients such as Mercedes- Benz Research & Development North America. Zheyuan holds bachelor's and master's degrees from Georgia Institute of Technology and University of Washington, is a co-inventor on two US patents, and co-authored an award-winning research paper on indoor autopilot systems.