International Journal of Advances in
Soft Computing and Intelligent Systems (IJASCIS)
2024, Vol 03, Issue 02, 349-361
ISSN: 3048-4987

Science Transactions © 2024

Original Paper

Maintenance in Automotive and Aerospace Applications – An Overview

Vikas Vyas[a, *], Zheyuan Xu[b, *]

[a] Department of Autonomous Driving, Mercedes-Benz Research & Development North America, Sunnyvale CA, 94087, vikas.vyas@mercedes-benz.com
[b] INSPYR Solutions, Fort Lauderdale, FL, 33334, zheyuan.xu@mercedes-benz.com, xuzheyuan961124@gmail.com

Abstract

The importance of software maintenance in safety-critical applications, such as automotive and aerospace systems, cannot be overstated. Safety-critical systems are those whose failure could result in loss of life, significant property damage, or environmental harm. Examples include aircraft, cars, medical devices, and nuclear power plants. Regular software maintenance is crucial to ensure these systems remain secure and protected against vulnerabilities. Updates address security flaws, patch known exploits, and enhance security protocols to guard against emerging threats. This paper provides an overview of the essential role of software maintenance in safety-critical systems, highlighting the necessity of prompt security updates, the challenges posed by cyber threats, and the importance of maintaining a secure software environment. It also discusses the specific challenges faced in the automotive and aerospace industries, current practices and approaches, and emerging trends in software maintenance. The paper concludes with a comparative analysis of the maintenance practices across these domains, emphasizing the commonalities and differences, and outlines future research directions to further enhance the safety and reliability of software in these critical applications.

Keywords

software maintenance, automotive, aerospace, safety applications

1. Introduction

1.1. Background

The criticality of software maintenance in safety-critical domains such as automotive and aerospace cannot be understated. Safety-critical systems, including aircraft, cars, medical devices, and nuclear power plants, are systems whose failure could result in loss of life, significant property damage, or environmental damage [4]. These systems rely heavily on software to perform essential functions, and the integrity of this software is paramount to ensuring overall system safety.

The challenges of integrating agile methodologies into the development of safety-critical systems are significant. Large-scale agile frameworks such as SAFe (Scaled Agile Framework) and LeSS (Large Scale Scrum) have been widely adopted in the automotive industry to enhance productivity and flexibility. However, these frameworks must be carefully adapted to incorporate rigorous safety practices [19]. This adaptation [10] is crucial as the automotive industry faces increasing software complexity, with premium cars now containing over ten million lines of code [19].

Incident reporting systems are crucial for maintaining safety in safety-critical applications. However, existing systems, often relying on relational databases, face problems such as data elicitation bias, precision and recall issues, data abstraction challenges, and inter-analyst reliability [1]. To address these challenges, solutions like computer-assisted interviewing, free-text retrieval and probabilistic inference, and conversational case-based reasoning have been proposed [1].

The importance of extending quality [6] assurance techniques to software development in safety-critical domains is another focal point. The paper discusses the unique challenges faced in these environments and emphasizes the need for robust quality assurance practices to ensure software safety and security. This is particularly relevant in the context of the automotive industry, where software drives innovation and enables 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 [31], driven by advancements in sensor technology, necessitates sophisticated data analysis techniques, with deep learning emerging as a promising solution [21].

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 survey aims to highlight the importance of regular software maintenance, identify common challenges, and discuss 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.

1.2. Structure of the paper

In this paper, section 2 defines software maintenance and discusses its importance in safety-critical applications, outlining 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 offers a comparative analysis of maintenance practices across the automotive and aerospace domains, highlighting both commonalities 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 a summary of key findings and 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, defined as the modification of a software product after delivery [7], is essential for correcting faults, enhancing performance, adapting to changing environments, and ensuring continued user satisfaction [8]. This involves rectifying bugs, integrating new features, and ensuring compatibility with evolving hardware and software ecosystems [7]. In safety-critical domains like automotive and aerospace, where software increasingly underpins crucial functionalities, the reliability and precision of this software are paramount to overall system safety [1, 2, 3, 4, 5].

Recent research underscores several key aspects of software maintenance in these high-stakes environments:

- Functional safety standards and fail-safe designs are non-negotiable for guaranteeing 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 unlocks complex functionalities and mitigates certain risks, it can introduce new hazards if not rigorously maintained and updated [1, 4]
- The increasing software-intensity of modern systems in these sectors amplifies the criticality of effective software maintenance. This is particularly crucial given the potential for substantial financial repercussions or even loss of life in case of system failures [5, 23].

2.2. Types of Maintenance

2.2.1. Corrective Maintenance

Corrective maintenance [30], a reactive approach to rectifying identified software problems after deployment, is indispensable for upholding the reliability and integrity of safety-critical software systems in the automotive and aerospace industries [7]. Studies reveal that software maintenance constitutes a significant portion, often 60-90%, of the total cost of ownership for an application [8]. The IEEE standards classify software maintenance into four primary categories, with corrective maintenance being one of them, supported by a range of techniques, methodologies, and tools [7]. Research indicates that a substantial portion of maintenance efforts, approximately 75%, is dedicated to adaptive and perfective maintenance [12], encompassing corrective actions [8]. Therefore, developing robust maintenance models that minimize rework expenses and enhance customer satisfaction is paramount, especially in domains where software failures carry severe consequences [7].

2.2.2. Adaptive Maintenance

Adaptive maintenance ensures software applications remain compatible and functional within evolving environments. This encompasses adapting to hardware updates, operating system changes, software dependency modifications, and evolving regulatory requirements [9]. This form of maintenance focuses on modifying software post-deployment to align with these changes, ensuring its continued usability and effectiveness [9]. Automating adaptive maintenance tasks enhances efficiency in managing the evolution of software systems, often utilizing high-level specifications to synthesize specialized code for this purpose [9].

2.2.3. Perfective Maintenance

Perfective maintenance [12] focuses on enhancing software quality by improving performance, maintainability, and overall robustness. This includes increasing efficiency, reliability, and adaptability to future modifications [11]. Research suggests that a considerable amount of development time and resources, up to 70%, is allocated to maintenance, underlining the need for effective maintenance models that prioritize software quality and customer satisfaction [7]. Techniques like visual analysis can be employed to pinpoint areas for improvement and optimize code quality during perfective maintenance, ultimately minimizing future costs and simplifying adaptations in complex software systems [11].

2.2.4. Preventative Maintenance

Preventive maintenance adopts a proactive stance, involving targeted software modifications to detect and address potential faults before they escalate into actual errors [13]. This approach aims to mitigate software failure risks, minimize downtime, and ensure the reliability and performance of critical systems [13]. Organizations can reduce planned downtime, enhance equipment reliability, and optimize system performance by implementing effective preventive maintenance strategies and scheduling [13]. A well-structured preventive maintenance program is essential in safety-critical domains, where software failures can have dire consequences, making the proactive resolution of issues before they lead to downtime critical.

2.2.4. Predictive Maintenance

Predictive maintenance harnesses advanced data analysis tools, including AI and ML, to identify potential anomalies and predict software failures before they occur [14]. This proactive strategy enables organizations to anticipate and prevent failures, minimizing downtime and boosting system reliability [14]. Research demonstrates the effectiveness of predictive maintenance in the automotive sector, where statistical inference, stochastic methods, and AI optimize maintenance strategies [14, 15]. Furthermore, ML models are valuable for detecting faults in critical components, such as engine parts, preventing failures and highlighting the potential of predictive maintenance to enhance software reliability and safety in these high-stakes domains [29].

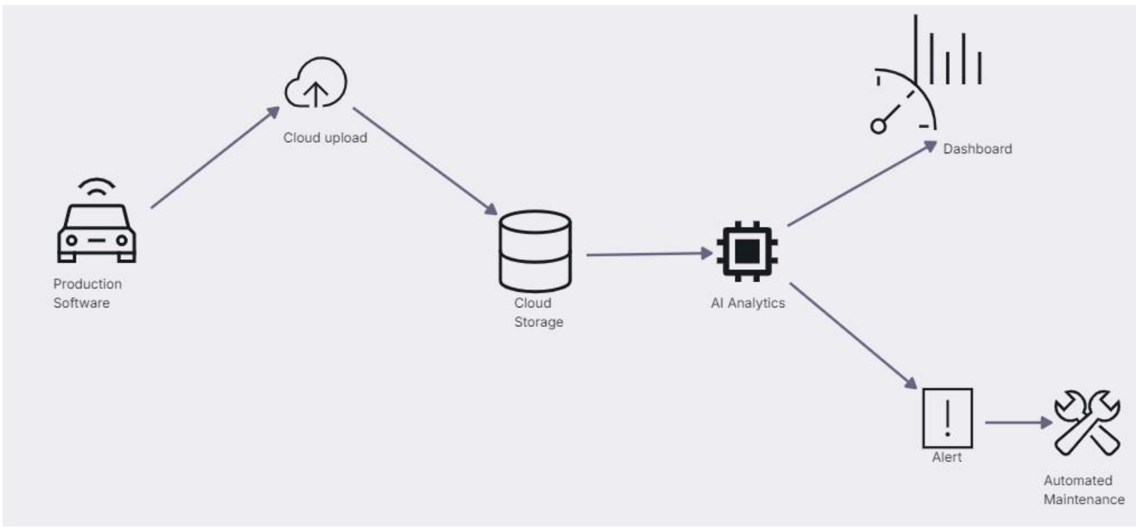

Figure 2. Predictive maintenance workflow with AI Analytics algorithms, leading to automated maintenance scheme.

3. AUTOMOTIVE INDUSTRY

3.1. Overview of Automotive Software Systems

Modern vehicles are heavily reliant on software for various functions, ranging from basic operations to advanced driver-assistance systems. The increasing complexity of these systems and the emergence of software-defined vehicles (SDVs) present unique challenges for software maintenance in the automotive industry [21].

3.1.1. Types of Software Used in Automotive Systems

Automotive software encompasses a wide range of applications, including:

- Embedded Systems: These systems are the backbone of numerous vehicle functions, controlling critical aspects like engine operation, braking systems, and infotainment [20].
- Model-Based Development: To manage the growing complexity and shortened development cycles, the automotive industry increasingly relies on model-based software development. This approach leverages high-performance computing resources and model-based design techniques to enhance efficiency and ensure compliance with safety standards [17].

- In-Vehicle IoT and Telematics: Connected vehicles leverage IoT and telematics systems to exchange data with external servers, applications, and various vehicle components, enabling features like smart mobility and advanced in-vehicle services [17].

3.2. Key Characteristics and Requirements

Automotive software systems demand unique characteristics and must adhere to stringent requirements to ensure safety, performance, and reliability.

3.2.1. Standardization and Modularity

The AUTOSAR standard plays a critical role in developing automotive software/system architectures, ensuring modularity, variant handling, and standardized execution management:

- AUTOSAR Standard: The AUTOSAR (AUTomotive Open System ARchitecture) standard plays a critical role in defining a standardized software architecture for automotive systems. It promotes modularity, facilitates variant handling, and establishes standardized execution management, contributing to improved software quality and reusability [18].
- Safety and Security: Given the safety-critical nature of automotive systems, adherence to industry standards like MISRA and AUTOSAR, along with robust cybersecurity practices, is paramount [19].

3.3. Maintenance Challenges in Automotive

Maintaining automotive software presents significant challenges due to the intricate and safety-critical nature of the systems:

- Escalating Complexity: The exponential growth in software complexity within vehicles poses considerable challenges for traditional software engineering practices. This complexity arises from the interconnected nature of various systems and the increasing demand for advanced functionality [19, 20].

- Constraints and Pain Points: Automotive software development operates under strict constraints, including cost considerations, stringent quality requirements, tight time-to-market pressures, and the need to seamlessly integrate with manufacturing processes. These constraints necessitate efficient maintenance strategies and tools [20].

- Software-Defined Vehicles (SDVs): The paradigm shift towards SDVs introduces new challenges related to software development and maintenance. Traditional research and development models need to adapt to accommodate the continuous evolution and update cycles characteristic of SDVs [21].

3.4. Current Practices and Approaches

Several techniques and methodologies are employed to address the challenges of maintaining automotive software:

- Lifecycle Management: Managing the extended lifecycle of automotive software is crucial, considering factors like platform selection, over-the-air updates, and long-term maintainability. Addressing these aspects ensures the sustainability and reliability of automotive software throughout its operational life [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, 23, 24, 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 comprise 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
- Due to safety and security concerns, software in aerospace domains are usually accessible to the outside via a wired, ad-hoc maintenance window.
- Data Management and Analysis: Modern aircraft generate vast amounts of data. Effectively managing, analysing, and utilizing this data for maintenance purposes is crucial but complex [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. Techniques and Methodologies Used

The aerospace industry employs a range of practices and methodologies to address software maintenance challenges:

- 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 behaviour, analyse data, and predict maintenance needs [14, 30].

4.3.2. Tools and Frameworks

- Integrated Development Environments (IDEs): Specialized IDEs for aerospace software development and debugging, 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. Commonalities Across Domains

While the automotive and aerospace industries have distinct characteristics, they share several commonalities in their approach to software maintenance for safety-critical applications:

- Emphasis on Safety and Reliability: Both domains prioritize the safety and reliability of their systems above all else. Software failures in either industry can have catastrophic consequences, necessitating rigorous development, testing, and maintenance practices.
- Rigorous standards and certification: Both industries adhere to stringent safety standards and regulations. Automotive software must comply with standards such as ISO 26262, while aerospace software follows guidelines like DO-178C. Compliance with these standards is essential for ensuring system safety and obtaining necessary certifications.
- Increasing Software Complexity: Both automotive and aerospace systems are experiencing a rapid increase in software complexity. The growing demand for advanced features, connectivity, and automation leads to more complex software systems that require sophisticated maintenance strategies.
- Adoption of Model-Based Development: Both industries increasingly leverage model-based development techniques to manage complexity and enhance the efficiency of software development and maintenance processes.
- Importance of Data Analytics: The increasing use of sensors and data logging in both automotive and aerospace systems generates a vast amount of data. Both industries are exploring the potential of data analytics and machine learning [15] to improve maintenance practices and predict potential failures.

5.2. Domain Specific Differences

Despite the shared emphasis on safety, several key differences exist between the automotive and aerospace industries regarding software maintenance:

- System Lifecycles: Aerospace systems typically have significantly longer lifecycles than automotive systems. Aircraft often remain in operation for decades, requiring long-term maintenance and support for software systems. In contrast, automotive software lifecycles are shorter due to rapid technological advancements and model updates [16]. This difference impacts maintenance strategies, with aerospace systems requiring greater emphasis on long-term sustainability and obsolescence management.

- Development and Certification Costs: The aerospace industry generally incurs higher development and certification costs compared to the automotive industry. The stringent safety requirements and complexity of aerospace systems necessitate extensive testing, verification, and formal certification processes, driving up costs.
- Production Volumes and Update Deployment: The automotive industry deals with significantly higher production volumes compared to the aerospace industry. This difference affects how software updates are deployed and managed. Automotive manufacturers increasingly utilize over-the-air (OTA) updates for efficient software distribution and installation across large fleets. In contrast, aerospace software updates typically involve more complex procedures due to the criticality of the systems and rigorous safety regulations.
- Environmental Conditions: Aerospace systems operate in harsher and more varied environmental conditions than automotive systems. Factors like extreme temperatures, pressure changes, and radiation exposure pose unique challenges for aerospace software maintenance, demanding robust solutions that can withstand demanding conditions.
- Human Factor Considerations: While both industries prioritize safety, the aerospace domain places a heightened emphasis on the human factor. Pilots and astronauts heavily rely on software systems for critical tasks, making human-machine interface design and user training crucial aspects of maintenance and safety considerations in aerospace systems.

6. Emerging Trends and Future Directions

6.1. Trends in Software Maintenance

The field of software maintenance, particularly within safety-critical domains like automotive and aerospace, is continuously evolving. Several prominent trends are shaping the future of software maintenance in these sectors:

- Artificial Intelligence [29] and Machine Learning (AI/ML): The use of AI/ML is revolutionizing predictive maintenance strategies [14, 29]. By analyzing vast amounts of sensor data, AI/ML algorithms can identify patterns, predict potential failures, and enable proactive maintenance, minimizing downtime and enhancing system reliability [27, 28].
- Digital Twins: Digital twins, virtual representations of physical assets, are gaining traction in both automotive and aerospace industries [14, 30]. By simulating real-world operating conditions and leveraging sensor data, digital twins enable engineers to monitor system health, predict maintenance needs, and optimize maintenance schedules, leading to significant cost savings and improved safety.
- Over-the-Air (OTA) Updates: OTA software updates are becoming increasingly prevalent, especially in the automotive industry [21]. The ability to deliver software updates remotely and seamlessly allows manufacturers to address bugs, enhance functionality, and improve security without requiring physical access to the vehicles. However, ensuring the safe and reliable execution of OTA updates in safety-critical systems remains a crucial consideration.
- Increased Automation: Automating maintenance tasks is a growing trend aimed at improving efficiency and reducing human error. This includes automating tasks like code analysis, testing, and deployment. As software complexity continues to grow, automation will play an increasingly important role in ensuring the timely and effective maintenance of safety-critical systems.

- Cybersecurity Enhancements: As vehicles and aircraft become increasingly connected, cybersecurity threats are becoming more sophisticated. Consequently, there is a growing need for robust cybersecurity measures integrated into software maintenance practices. This includes secure software development practices, regular security audits, and timely patching of vulnerabilities to mitigate the risk of cyberattacks.

6.2. Future Research Directions

Despite advancements in software maintenance practices, several research areas require further exploration to address emerging challenges and enhance the reliability and safety of software in automotive and aerospace applications:

- Developing robust and interpretable AI/ML models for predictive maintenance: While AI/ML shows promise in predictive maintenance, ensuring the reliability, trustworthiness, and interpretability of these models is crucial. Further research is needed to develop methods for validating AI/ML predictions, understanding model behaviour, and building trust in AI-driven maintenance decisions.

- Standardizing data formats and communication protocols for digital twins: The adoption of digital twins necessitates standardized data formats and communication protocols to enable seamless data exchange and interoperability between different systems and stakeholders. Establishing such standards will facilitate wider adoption and maximize the benefits of digital twins in maintenance applications.

- Ensuring the security of OTA updates in safety-critical systems: As OTA updates become more prevalent, ensuring their secure delivery and execution is paramount. Research is needed to develop robust security mechanisms, such as secure boot processes and tamper-proof software updates, to prevent unauthorized modifications and ensure the integrity of safety-critical systems.

- Developing automated testing and verification techniques for complex software systems: The increasing complexity of software systems demands more sophisticated testing and verification methods. Research into automated testing techniques, formal verification methods, and runtime monitoring tools is essential to ensure the reliability and safety of these complex systems.

- Investigating the ethical implications of AI and automation in software maintenance: As AI and automation play a larger role in software maintenance, it is crucial to address the ethical implications. Research is needed to develop guidelines and best practices for responsible AI development and deployment, ensuring human oversight and accountability in maintenance decisions.

7. CONCLUSIONS

Software maintenance is not merely an afterthought in the development lifecycle; it is an ongoing, critical process essential for the safe, reliable, and secure operation of safety-critical systems in the automotive and aerospace industries. This paper explored the multifaceted landscape of software maintenance in these domains, highlighting the unique challenges, current practices, and emerging trends shaping the field.

The automotive and aerospace industries share common ground in prioritizing safety and grappling with increasing software complexity. Both sectors are turning to model-based development, rigorous testing, and data analytics to manage this complexity and ensure system reliability. However, domain-specific differences like system lifecycles, development costs, and operational environments necessitate tailored approaches to software maintenance.

Emerging trends, including AI/ML, digital twins, OTA updates, and increased automation, hold immense promise for revolutionizing software maintenance practices, making them more proactive, efficient, and effective. However, these advancements also introduce new challenges, requiring further research to address security concerns, ensure model interpretability, and navigate ethical considerations.

As software continues to permeate every facet of automotive and aerospace systems, effective software maintenance will only grow in importance. Continued research, collaboration, and innovation are crucial to meeting the evolving demands of these safety-critical domains and ensuring the well-being of those who rely on these systems.

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.

[2] R. Shaw, "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] L. E. G. Martins 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 Engineering (ICSE '02). Association for Computing Machinery, New York, NY, USA, 547–550.

[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] N. Gorla, "Techniques for application software maintenance," Information 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 Software Customization and Adaptation.

[11] J. Trümper, M. Beck and J. Döllner, "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.

[12] D. Rine, "Software perfective maintenance: Including retrainable software 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. Proceedings. 2020; 64(1):26.

[16] Schaefer, J., Christlbauer, H., Schreiber, A., Reith, G., Jonker, M., Potman, J., Dannebaum, U., & Eissfeldt, T. (2021). Future Automotive 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 Engineering.

[18] Staron, M. (2021). AUTOSAR (AUTomotive Open System ARchitecture). In: Automotive Software Architectures. Springer, Cham.

[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] K. Lundqvist 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.

[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.

[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 Systems (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).

[29] Ucar A, Karakose M, Kırımça N. Artificial Intelligence for Predictive Maintenance Applications: Key Components, Trustworthiness, and Future Trends. Applied Sciences. 2024; 14(2):898.

[30] Moleda M et al. From Corrective to Predictive Maintenance—A Review of Maintenance Approaches for the Power Industry. Sensors. 2023; 23(13):5970.

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

Authors

Vikas Vyas is a visionary leader with a proven track record of driving innovation in the automotive and aerospace industries. With over 15 years of experience, he specializes in program and product management and development of cutting-edge technologies. Currently serving as a technical lead at Mercedes-Benz Research & Development North America, Vikas is at the forefront of autonomous driving, overseeing market analysis, research, and development of advanced automotive solutions, shaping the future of mobility. Prior to his role at Mercedes-Benz, he led global teams at Bosch USA, where he spearheaded 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 specializes in autonomous driving, mechatronics, ADAS development and over-the-air (OTA) algorithm development 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.