2025-01-0355 Published 15 Sep 2025

Applying Specialized System Analysis for Brake by Wire in Software Defined Vehicles: The Development of a System Analysis Tool (SAT)

Edward Heil, Sean Zuzga, and Caitlin Babul General Motors LLC

Citation: Heil, E., Zuzga, S., and Babul, C., "Applying Specialized System Analysis for Brake by Wire in Software Defined Vehicles: The Development of a System Analysis Tool (SAT)," SAE Technical Paper 2025-01-0355, 2025, doi:10.4271/2025-01-0355.

Received: 13 Apr 2025 Revised: 14 Jun 2025 Accepted: 17 Jul 2025

Abstract

The emergence of Software Defined Vehicles (SDVs) has introduced significant complexity in automotive system design, particularly for safety-critical domains such as braking. A key principle of SDV architecture is the centralization of control software, decoupled from sensing and actuation. When applied to Brake-by-Wire (BbW) systems, this leads to decentralized brake actuation that demands precise coordination across numerous distributed electronic components. The absence of mechanical backup in BbW systems further necessitates fail-operational redundancy, increasing system complexity and placing greater emphasis on rigorous system-level design validation. A comprehensive understanding of component interdependencies, failure propagation, and redundancy effectiveness is essential for optimizing such systems.

This paper presents a custom-built System Analysis Tool (SAT), along with a specialized methodology tailored for modeling and analyzing BbW architectures in the context of SDVs. The operation of the SAT is described in detail, and its application is demonstrated through illustrative examples derived from a representative BbW system model. The SAT enables systematic evaluation of individual component failures, their logical and functional interdependencies, and the cumulative impact of multiple simultaneous faults. It provides structured, data-driven insights that support early trade-off decisions between cost, complexity, and safety, and facilitates the generation of robust, traceable functional requirements. Additionally, the SAT quantifies the relationship between component failure rates, expressed in Failures in Time (FIT), and system-level performance degradation across multiple defined operational states.

By enabling a rigorous, model-based approach to design exploration and fault analysis, the SAT enhances system engineers' ability to validate fail-operational behavior, identify design weaknesses, and refine brake system architecture. This supports a more efficient development process for safety-critical systems and contributes to the overall reliability and performance of BbW implementations within SDV platforms.

Introduction

The automotive industry is undergoing a significant transition, shifting from traditional vehicle architectures that depend on separate modules to a new era of tightly integrated SDVs. In conventional setups, control modules are scattered throughout the vehicle, generally with each module incorporating the necessary sensing and actuation hardware with associated mechatronic and control software into an Electronic Control Unit (ECU). These ECUs then manage the required functionality of the specific system. The move to SDVs marks a departure towards a more centralized control approach, with decentralization of sensors and actuators. This approach involves combining vehicle motion control feature functions, while decentralized sensors measure the vehicle's state and decentralized actuators provide the dynamic forces necessary for the vehicle's operation.

This shift is laying the groundwork for the possible widespread adoption of 'X by wire' technologies, such as Steer by Wire (SbW) and Brake by Wire (BbW). These innovative systems bring several advantages over traditional methods, including improved efficiency, increased safety, and more modular and scalable designs. They also reduce mechanical complexity, offer greater flexibility in design, and create opportunities for the introduction of new functionalities through software enhancements. However, integrating these systems within a distributed SDV architecture requires advanced tools and methodologies to ensure robustness and facilitate rapid iterative design.

As SDVs continue to evolve, the braking system—a vital component for ensuring vehicle safety—becomes more complex through tight integration with other vehicle systems. There's an escalating need to understand the

interrelationships between the centralized computing algorithms and the distributed networks of power, communication, sensing, and actuation. This intricate multi-layer setup emphasizes the importance of an advanced analytical approach to guarantee that all system components operate in unison and with reliability.

In response to these intricate new demands, the development and deployment of a tool to facilitate analytical analysis of a system is critical. The System Analysis Tool (SAT), developed by the authors, is specifically designed to evaluate and enhance the brake system design within the context of SDVs. It provides an in-depth understanding of how different system components interact and enables the management of overarching system modifications.

The SAT's role extends beyond an initial analysis; it is pivotal for enabling engineers to rapidly respond to the evolving needs of distributed control systems. With the necessity for parallel development of all vehicle systems, the ability to quickly iterate and fine-tune designs becomes indispensable.

This paper will provide a background on the theoretical aspects of BbW systems and address the unique challenges posed by SDV to brake systems. We will present the architecture of an example BbW system referenced in contemporary literature [3], which will serve as a model throughout this paper to illustrate the practical applications of the SAT. Our discussion will center on how the SAT facilitates the construction of a robust, fault-tolerant BbW system that integrates into an SDV architecture, including the utilization of a central computing unit with a backup ECU.

The following sections will describe the SAT's concept and design, illustrate its capabilities, and explain the methodology used to develop and analyze the system. By methodically modeling the BbW system, assessing its states, and iteratively refining the design, the SAT significantly aids in requirement development and system verification. This introduction sets the stage for further discussion on how the SAT supports these tasks, enhancing the safety and design efficiency of complex BbW systems within SDVs.

Background

A Brake by Wire (BbW) system is an advanced vehicle braking technology in which the traditional physical hydraulic connection between the brake pedal and the foundation brakes is completely replaced by electronic control systems—both during normal operation and in degraded modes. In conventional braking systems, a mechanical backup exists for degraded operation, providing a direct hydraulic pathway from the brake pedal to the foundation brakes. This backup allows the human driver to manually generate the braking force through physical pedal effort, supplying both the pedal force and pedal travel (i.e., mechanical work) necessary to produce braking torque at the wheels. Importantly, this force is not amplified by any electronic or hydraulic assist; instead, the driver alone supplies the energy required to slow the vehicle.

In a BbW system, pressing the brake pedal sends electronic signals from travel sensors via serial data to the vehicle's centralized brake controls, which commands the four independent Electro-Mechanical Brake (EMB) corner actuators to activate and generate braking torque at each wheel. This architecture demands robust redundancy in computing and communication to ensure safe operation in the event of failures. In the BbW system discussed in this paper, no degraded mode involving mechanical backup exists; therefore, all redundancy and fail-operational capabilities must be provided electronically to maintain braking performance. The use of decentralized brake corner actuators inherently supports this requirement, as the independent actuation capability at each wheel allows for distributed redundancy and continued vehicle braking even under fault conditions. Ensuring reliable operation in such architectures requires careful consideration of real-world implementation constraints. Previous real-world implementation experiences, such as those reported for the GM Sequel project [1], underscore practical challenges and the critical importance of detailed system reliability planning.

Centralized vs. Decentralized Braking

Centralized brake control and actuation has been the approach used in conventional vehicles and involves a single eBoost electro-hydraulic Electronic Brake Control Module (EBCM) that controls braking and delivers pressurized fluid to actuate the hydraulic brakes located at each wheel. Generally, all the sensors required for providing brake operation are directly incorporated into this EBCM. During normal braking situations, the centralized EBCM measures the driver intent for braking and uses electro-hydraulic actuation to provide common pressure to each of the hydraulic foundation brakes generating braking torque that transfers to the tire contact patches resulting in vehicle deceleration. When independent wheel torque control is necessary, for example during Antilock Braking System (ABS) mitigation of excessive wheel slips, or Electronic Stability Control (ESC) mitigation of unwanted vehicle yaw motions, the system relies on intricate hydraulic valve control operations to regulate wheel-specific brake pressures. This centralized EBCM can also control separate Electric Parking Brake (EPB) motor-on-caliper clamping actuators, which can generate and maintain wheel torque through the foundation brakes to immobilize the vehicle. Given the single centralized nature of the EBCM electronics, there likely exists single point failures where the central controls cannot provide braking functions. In these degraded states, the system would rely on mechanical backup to provide braking to the vehicle. Additional braking redundancy may be necessary when relying on mechanical backup isn't sufficient, such as in a highly

autonomous vehicle where there is no human driver that can apply a brake pedal. Even more consideration is necessary to ensure the ability for the brake system to immobilize the vehicle and may require additional electrical redundancy or another vehicle system to provide immobilization (e.g. park pawl in transmission), both of which increase system cost.

On the other hand, a BbW system represents a significant departure from this norm, offering a modern approach in which braking controls are centralized, while sensing and actuation are decentralized into independent brake corner actuators. The centralized compute achieves redundancy in brake control by applying methods commonly used in other industries that demand high computing reliability, such as Information Technology, which has developed efficient electronic design strategies for ensuring compute redundancy. This redundancy can be implemented using independent primary and backup Microcontroller Units (MCUs), either housed within the same physical Centralized Compute Unit (CCU) or distributed across separate ECUs, with appropriate interconnecting communication networks supporting either configuration.

In contrast to this centralized compute strategy, a BbW system employs decentralized actuation—often using smart or semi-smart EMB brake corner actuators [2]. These systems are distinguished by their inherent redundancy, enabled by four independent actuators that are free from common-cause failures. To support this architecture, a redundant low-voltage power distribution system is required to ensure continuous power availability to a minimum number of actuators, delivering the critical availability needed for brake systems without mechanical backup and for highly autonomous vehicles. For similar reasons, BbW systems also include sufficient redundancy to ensure reliable vehicle immobilization functionality.

A decentralized BbW system enables tight integration of vehicle motion controls (perhaps including steering, propulsion and suspension along with braking controls) in a common CCU, which allows the rapid deployment of new vehicle motion control features and enhancements with significantly less developmental and integration efforts compared to traditional EBCM systems. Further, by defining a family of brake corner actuators as common components that can be applied to many vehicle programs, costs can be optimized and complexity reduced for integration across a vehicle portfolio. EMB brake corner actuators can provide other key features such as positive control of brake retraction, which minimizes or potentially eliminates residual brake drag [3]. Unfortunately, the mark of a well-designed BbW brake system is that the customer that purchases the vehicle may not directly perceive a change to the brake system from conventional vehicles. However, the benefits of a decentralized actuation BbW system will indirectly enhance the customer experience through vehicle design; such as increased cabin space, extended driving range, a broader array of brake-based features across a larger selection of vehicle models, enhanced brake pad wear sensing, elimination of brake fluid maintenance, quieter ABS, and overall, more consistent vehicle performance and feel [3].

As Brake by Wire (BbW) systems forgo the mechanical backup safe state traditionally found in braking systems, the application of advanced system analysis methods is imperative. The objective is to ensure the system is not only maintaining, but potentially enhancing, the safety levels provided by legacy systems. Such safety concepts are particularly critical as they relate to the brake system's availability—an aspect fundamentally tied to the relationship between the system's achievable performance and the aggregate failure rates that lead to given system states [4].

Challenges with the Software-Defined Vehicle

Adopting a BbW system within SDVs introduces several new challenges, primarily arising from increased system complexity. The integration and interaction of electronic components within the distributed networks of an SDV necessitates a comprehensive approach to failure management. Comparing failure states between traditional EBCM and SDV-based BbW systems reveals that a single component failure in SDVs can have cascading impacts, highlighting the necessity of robust fail-operational designs.

Critical failures, or combinations of failures, that cause a complete loss of braking in BbW systems must meet stringent ISO 26262 (Automotive Safety Integrity Level) ASIL D quantitative safety targets, typically represented as occurrence of random hardware failures targets of 10 FIT, or the allowance of no more than 10 failures per billion hours of operation [5]. To achieve these dependability requirements, BbW systems may utilize advanced redundancy architectures, each presenting distinct trade-offs among cost, complexity, and performance [6]. These stringent redundancy requirements—spanning centralized computing and decentralized actuation—add complexity due to distributed communications among control, sensing, and actuation elements. This communication involves numerous individual modules, components, interconnecting wires, and electrical connectors, each representing potential failure points. Additionally, since all these electronic components rely on power, the power distribution system must be rigorously analyzed to identify potential failures impacting the BbW system. Furthermore, contemporary braking control functions integrated into a BbW system depend heavily on accurate vehicle state estimates derived from vehicle motion sensors and braking force measurements from both friction and regenerative braking. Each of these sensor inputs increases system complexity and introduces additional potential failure points, potentially causing partial information availability across different system components.

Cross-domain hazard analysis, as detailed by Schrade et al. [7], becomes critical in SDVs, as braking systems closely interact with steering, suspension, and propulsion domains. Each failure could negatively impact other interconnected components, and these complex interactions

might not always be immediately apparent to the engineering teams performing the analysis.

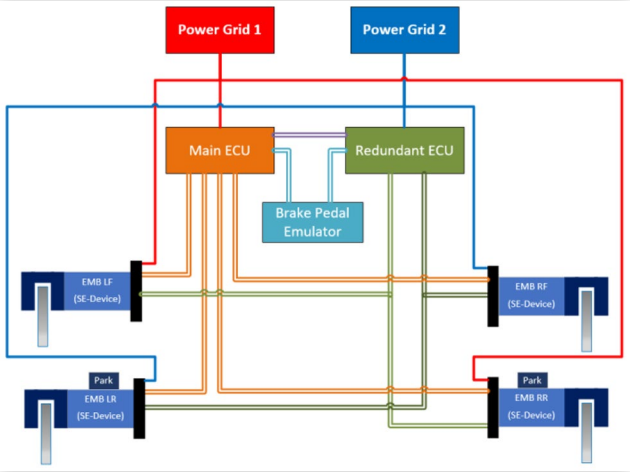
To ensure safety, vehicle operability may be limited when any subsequent failure would result in an excessively degraded performance, thereby minimizing exposure to potential vehicle level hazards. On the other hand, maintaining the high level of vehicle reliability expected by customers is crucial. Intimately understanding the failure network that could cause any restriction of vehicle operation and assessing the rates of occurrence in a fleet of vehicles are imperative. Balancing these competing requirements is an ongoing challenge, demanding effective analysis of the impacts of multiple independent failures and efficient iterative adjustments to system designs.

The unique characteristics of SDV architectures result in complex networks of potential system failures, surpassing the capabilities of traditional analysis tools, such as Failure Mode and Effects Analysis (FMEA) and Fault Tree Analysis (FTA). A detailed and dynamic system model is thus essential for thoroughly comprehending the intricate relationships between failures and overall system operations, enabling engineers to design and optimize safe and reliable systems. Such a model must accurately simulate various degraded operational states. The SAT, detailed in this paper, was developed specifically to address these critical needs.

Example BbW Architecture

Figure 1 shows the example BbW architecture used throughout this paper, providing a basic representation of a decentralized SDV braking system. This architecture has been referenced in other contemporary literature on this topic [3]. Although simple, this architecture includes all essential components necessary for a BbW system and exemplifies how a distributed architecture might be conceptualized in an SDV. A system architecture for a production vehicle, however, will be more complex as there are many potential failure points that are not incorporated into this schematic, which may also drive the necessity for additional redundancies in the system.

FIGURE 1 Example architecture of a BbW system that is used in this paper.



This example architecture utilizes four smart EMB brake corner actuators, that consume braking targets via serial data and perform the control necessary to achieve the torque. The low voltage power supply to these actuators is provided with redundancy from two separate grids of power in the vehicle, such that common failures causing a loss of power to both grids is managed to a very high level of integrity and therefore can be relied upon for a braking system with no mechanical backup redundancy.

The complexity in the serial data communication is simplified in this example architecture, as the communication networks will have many other vehicle system interactions. This means in a realistic production vehicle, there would likely be additional potential failure points that exist between the Microcontroller Units (MCUs) and the EMB smart actuators.

Brake controls must be applied redundantly, with control algorithms residing in both the main and redundant ECUs. Normal service braking functionality must be maintained in the presence of all single failures, which means that both sets of brake control algorithms must consume redundantly all necessary sensor data and provide all necessary actuator commands to meet the service braking expectations. The actuators must have a strategy to consume and arbitrate between commands from main and redundant controls, and this may include ensuring that all actuators are following commands from a common controller.

Given the criticality of the service braking function, employing a tool capable of ensuring all design considerations are met throughout the system development lifecycle is essential. Such a tool, aligned with the V-model approach, can support early system concept design and requirements development on the left side of the 'V', and subsequently aid in rigorous system verification testing on the right side of the 'V'. Adopting an agile methodology with this comprehensive functional analysis and verification capability early in the development process significantly reduces costly rework, enhances project efficiency, and delivers substantial cost savings [8].

System Analysis Tool: Concept and Design

Previous work introduced a methodology for analyzing brake systems to balance reliability and performance under failure conditions. However, it also highlighted the need for more sophisticated tools to further enhance the analysis [4]. The objective of developing the SAT was to extend this prior methodology, enabling engineers to comprehensively examine system designs and verify that safety and performance requirements are met throughout the system development lifecycle. The iterative

capabilities of the SAT facilitate efficient and effective system optimization, explicitly supporting both sides of the V-model [8].

The development of the SAT complements formal safety processes like those outlined in ISO 26262 [5], providing a dynamic modeling capability that enhances traditional hazard analysis and supports systematic ASIL allocation and verification across the BbW system [9]. Traditional analysis methods, such as FTA, have been demonstrated as effective approaches for assessing BbW reliability [10]. However, unlike these traditional methods, the SAT provides dynamic modeling capabilities, allowing engineers to visualize and rapidly iterate through failure propagation scenarios. Leveraging automated dynamic safety analysis further strengthens the capability to rapidly evaluate design iterations, aligning with methodologies presented by Ebner et al. [11]. This capability significantly reduces the time required for identifying critical failure pathways and enhances accuracy in predicting complex system interactions.

The SAT was developed based on several key principles: it should require minimal programming knowledge, have a low barrier to entry (utilizing easily accessible software such as a downloadable executable program and spreadsheet files), and employ a modeling ontology (domain-specific language) familiar to automotive engineers. Engineers define system elements, their interactions, and failure logic within a spreadsheet file, which serves as the primary input to the SAT. After performing analyses, the SAT generates results and detailed verification data within the same structured spreadsheet format, enabling straightforward interpretation and efficient iterative refinement of system designs.

The process for developing and analyzing a system with the SAT closely aligns with the phases of the V-model lifecycle:

System Design: Engineers begin by creating a detailed block diagram representing the system architecture, aligning with early concept and requirements definition phases on the left side of the V-model. This diagram is translated into the SAT environment by clearly defining each system element and establishing their physical interconnections. Logical relationships are also specified to accurately capture dependencies and potential propagation of failures among system components, supporting early validation of requirements and conceptual designs.

System Requirements: Engineers define a hierarchical set of system states, each with corresponding enumerations representing varying levels of system performance. Python code is then developed by the engineer, explicitly defining the logic for determining resulting system states based on failures within the system. This code serves as a direct manifestation of the system's software functional and diagnostic requirements. Once established, the SAT iteratively executes this logic across all potential failure scenarios—single-point or combinations of multi-point failures—and documents the resulting system states in a structured spreadsheet format. Engineers refine this Python logic iteratively until the simulated system behavior precisely aligns with design expectations. After finalizing the model, engineers leverage this manifested logic to quickly and accurately establish comprehensive system requirements and detailed design specifications, effectively bridging conceptual definitions with detailed system implementation on the left side of the V-model.

System Verification: Once satisfied with the modeled system behavior, the SAT supports requirement verification by systematically comparing simulation results against established system requirements. Failure rates are estimated for each component and aggregated using Reliability Block Diagram (RBD) analyses to evaluate overall system reliability. Additionally, the SAT automatically generates structured test plans aimed at verifying degraded system functionalities. These test cases utilize Gherkin syntax (Given-When-Then) to explicitly define test scenarios, preconditions, and expected outcomes. This structured approach ensures comprehensive verification coverage, aligning seamlessly with integration, verification, and validation phases on the right side of the V-model. The SAT's flexibility provides an agile method to accommodate inevitable system modifications during development. When changes occur, engineers swiftly update and revalidate the model. After reacceptance, the SAT automatically regenerates updated verification test plans, significantly enhancing responsiveness and efficiency throughout the iterative development process.

Modeling the BbW System

Modeling the BbW system accurately in the SAT is essential to understanding how individual component failures affect the overall braking system performance. This detailed analytical approach enables engineers to effectively predict system behavior under various failure conditions and is fundamental in defining robust system requirements.

Models in the SAT consist of three main elements: ECUs, Components, and Connections.

ECUs – An ECU in the SAT represents a logical grouping of related system functions. Each ECU serves as a container for Components that define the specific functionalities it provides—such as sensing, actuation, diagnostics, or decision-making. ECUs are given user-defined names, except in certain reserved cases such as the built-in "Logic" ECU. Functionally, ECUs provide a way to logically partition the system into manageable modules that mirror real-world controller architectures. For example, a vehicle's main brake controller and redundant controller would each be modeled as distinct ECUs. The SAT does not assign inherent behavior to an ECU block itself; rather, its behavior is defined through the interaction of the Components it contains and the Connections it participates in. Grouping Components within ECUs also enables better traceability, allows easier requirement allocation, and aligns modeled structures with system architecture documents.

Components – Components represent the functional building blocks inside an ECU. These may include power inputs, sensors, actuators, diagnostic states, or software processing elements. When the tool iterates through failure scenarios, the initial failure always originates at the Component level. An initial failure represents a direct malfunction of a Component, independent of any other system-level conditions or dependencies. Users can specify whether a Component is eligible to be considered as an initial failure. The SAT includes extensions for grouping Components with common naming prefixes—for example, a redundant power scheme under a given ECU might include "PWR_1" and "PWR_2." The model can be configured such that only when both power inputs fail does the rest of the ECU's functionality also fail. Another extension includes logic-only Components defined under the special "Logic" ECU. These logic gates (AND, OR, NOT) are not allowed to fail initially but are used to propagate failure conditions logically through the system.

Connections – Connections define how failures propagate between Components. Basic connection types include one-way links (Component A fails Component B) and two-way links (Component A and Component B fail each other). More advanced connections are made to Components under the "Logic" ECU, allowing failure conditions to be modeled with Boolean logic. Supported logic gates include AND, OR, and NOT. Additionally, a specialized connection type—"ECU Component to Fail"—can be used to define which Component(s) should fail when the logic conditions are satisfied.

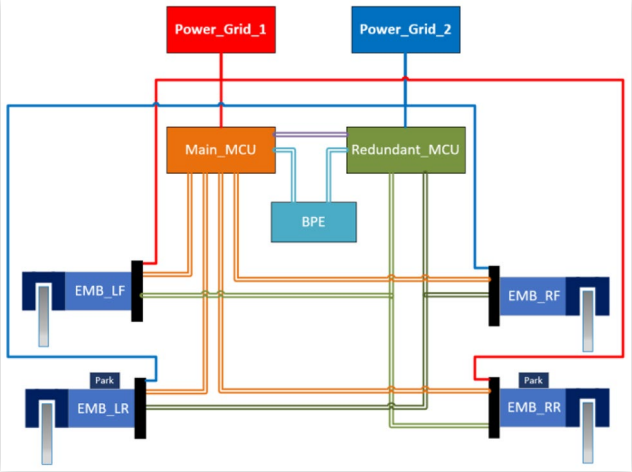
System functionality is defined through ECUs, with specific behaviors modeled at the Component level and interconnected via failure propagation logic using Connections. This structure allows users to model complex systems and accurately simulate fault scenarios. In the following section, an example modeling strategy using the SAT will be provided for the example BbW system shown in Figure 1.

Example BbW Model

Translating the BbW system architecture into an analytical model requires clearly defining the primary functional modules (ECUs) and their constituent components. Utilizing the SAT, the BbW example architecture was decomposed into distinct ECUs such as "Main_MCU", "Redundant_MCU", "Brake Pedal Emulator (BPE)," "Power_Grid_1," "Power_Grid_2," and each Electro-Mechanical Brake (EMB) brake corner actuator (EMB_LF, EMB_RF, EMB_LR, EMB_RR) for the corresponding Left, Right, Front and Rear corner. Each ECU encapsulates specific functionalities critical to brake operation, redundancy, and failure management. A block diagram architecture of the BbW system, including the SAT model ECU names, is shown in Figure 2.

The block diagram simplifies the BbW architecture for clarity, omitting certain complexities inherent in real-world systems. For example, BPE signals are sent through

FIGURE 2 Example BbW system with SAT model ECU names



two independent communication paths and derived from independent travel sensors, resulting in four distinct signals on the communication bus, but this design detail isn't visible in the block diagram. This BPE redundancy significantly enhances the robustness of the system by mitigating potential single-point communication failures.

One design goal of this example dual-ECU brake control system is to maintain active control in the Main_MCU and fail over to the Redundant_MCU only in rare instances. A possible rationale for this goal is to reduce the complexity of the redundant brake controls, potentially promoting diversity and orthogonality in the brake control software. It may be advantageous to provide only basic service braking functionality in the redundant controls, with other brake control features, such as ABS wheel slip control and ESC stability control, not being available. To support this design goal, additional signaling is incorporated to enhance the robustness of the Main_MCU. For example, the Main_MCU receives BPE signals directly from the BPE, as well as BPE signals from the Redundant_MCU, which are routed through the private communication bus between the MCUs. This signal redundancy eliminates a single point failure that would otherwise cause a premature failover to the Redundant_MCU.

Following the system design goals outlined above, the example BbW architecture was modeled in the SAT to evaluate failure propagation and system robustness. Components within the Main_MCU and Redundant_MCU ECUs represent specific functions such as power distribution, sensor inputs, actuator control, and microcontroller operations. Logical dependencies between Components were modeled using the SAT's connection definitions, with Boolean logic gates (AND, OR, NOT) applied to simulate fault propagation. Table 1 lists the Components modeled for both the Main_MCU and the Redundant_MCU. Within each ECU, Components are categorized as either initial failure points, representing independent faults, or as dependent elements that may fail based on the status of other Components.

TABLE 1 Modeled Main_MCU and Redundant_MCU with Components

Main_MCU	
Components that are Initial Failures	
Component	**Description**
PWR_1	Redundant Power Input 1
PWR_2	Redundant Power Input 2
GND	Single electrical ground connection
SerialData_LF	Serial data network connection between EMB_LF and Main_MCU
SerialData_RF	Serial data network connection between EMB_RF and Main_MCU
SerialData_LR	Serial data network connection between EMB_LR and Main_MCU
SerialData_RR	Serial data network connection between EMB_RR and Main_MCU
Micro	Single microcontroller
SerialData_Main_Red	Serial data network between Main_MCU and Redundant_MCU
SerialData_Low_1	Low speed serial data network (Main_MCU to BPE) carrying BPE_Travel_1 & BPE_Travel_2
Components that are NOT Initial Failures	
Component	**Description**
BPE_Travel_1_Signal	BPE → Main_MCU for BPE_Travel_1
BPE_Travel_2_Signal	BPE → Main_MCU for BPE_Travel_2
BPE_Travel_3_Signal	BPE → Redundant_MCU → Main_MCU for BPE_Travel_3
BPE_Travel_4_Signal	BPE → Redundant_MCU → Main_MCU for BPE_Travel_4
EMB_LF_Primary_Signal	EMB_LF → Main_MCU
EMB_LF_Secondary_Signal	EMB_LF → Redundant_MCU → Main_MCU
EMB_RF_Primary_Signal	EMB_RF → Main_MCU
EMB_RF_Secondary_Signal	EMB_RF → Redundant_MCU → Main_MCU
EMB_LR_Primary_Signal	EMB_LR → Main_MCU
EMB_LR_Secondary_Signal	EMB_LR → Redundant_MCU → Main_MCU
EMB_RR_Primary_Signal	EMB_RR → Main_MCU
EMB_RR_Secondary_Signal	EMB_RR → Redundant_MCU → Main_MCU
Redundant_MCU	
Components that are Initial Failures	
Component	**Description**
PWR	Single power input
GND	Single electrical ground connection
SerialData_Red_LF_RR	Serial data network between EMB_LF, EMB_RR, and Redundant_MCU
SerialData_Red_RF_LR	Serial data network between EMB_RF, EMB_LR, and Redundant_MCU
SerialData_Main_Red	Serial data network between Main_MCU and Redundant_MCU
Micro	Single microcontroller
SerialData_Low_2	Low speed serial data network (Redundant_MCU to BPE) carrying BPE_Travel_3 & BPE_Travel_4
Components that are NOT Initial Failures	
Component	**Description**
BPE_Travel_3_Signal	BPE → Main_MCU for BPE_Travel_3
BPE_Travel_4_Signal	BPE → Main_MCU for BPE_Travel_4
EMB_LF_Signal	EMB_LF → Redundant_MCU
EMB_RF_Signal	EMB_RF → Redundant_MCU
EMB_LR_Signal	EMB_LR → Redundant_MCU
EMB_RR_Signal	EMB_RR → Redundant_MCU
Main_MCU_Signal	Main_MCU → Redundant_MCU

The SAT explicitly distinguishes between initial and secondary failures. Initial failures represent direct, component-level malfunctions, while secondary failures result from the propagation of those malfunctions through logical or physical dependencies. This modeling approach allows engineers to accurately simulate the cascading effects of failures in complex, interconnected systems.

Modeling the System States

Clearly defining and understanding system states is essential for robust analysis of a system. These states are structured hierarchically, representing how performance degradation progresses through various levels of component and subsystem failures. This hierarchical

structure simplifies complex interdependencies, providing a clear view of how individual component issues escalate into broader impacts at the supervisory and system levels.

Figure 3 shows a simplified hierarchy organized into three distinct levels for clarity:

Component Level: States at this level are directly determined by individual component statuses, diagnostics, and sensor signals (e.g., sensor faults, ECU errors).

Supervisory Level: States here consolidate multiple component-level states into broader categories reflecting the overall health or functionality of key subsystems (e.g., power distribution system health, communication bus integrity).

System Level: This highest level integrates both supervisory-level and critical component-level states to determine overall system functionality and to identify which MCU (Main or Redundant) should assume control.

The SAT systematically evaluates component-level failures—including power supply interruptions, sensor malfunctions, and communication failures—and categorizes the resulting impacts across supervisory and system levels. This analysis highlights how individual component degradations propagate through the system, ultimately affecting overall braking functionality. Communication and power system redundancies are also evaluated, emphasizing the importance of maintaining independent pathways to preserve braking system availability during single-point or multi-point failures. For instance, failures in the Main_MCU or Redundant_MCU may yield distinct degradations in braking performance, depending on the specific fault and its propagation through the system hierarchy. System states can be structured into enumerations such as 'Full Performance,' 'Reduced Performance,' 'Low Performance,' and 'Loss of Performance,' and linked to functional reliability "buckets" to quantify system capability under various failure scenarios [4].

To further illustrate the definition of system states, Table 2 provides examples along with descriptions and their associated modeled components. These system states are grouped according to the ECU that functionally determines—or calculates—the state by executing the relevant software algorithms. This calculation process involves processing internal diagnostics and/or interpreting received signals from other components to derive the system state output. In other words, the ECU identified as calculating a given state is the one that houses and executes the logic responsible for evaluating its conditions. The states shown in Table 2 represent a simplified subset of those used in the example BbW system discussed in this paper; a production-level model would likely include a broader and more detailed set of system states.

Figure 4 presents the hierarchical relationships for the Main_MCU and Redundant_MCU, including detailed enumeration values representing the full function (no failures in the system) states.

FIGURE 3 System State Hierarchy

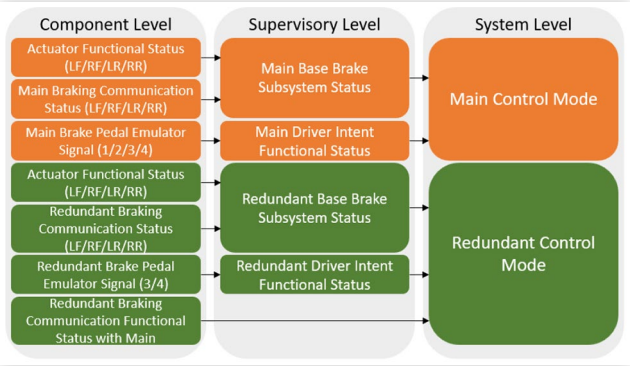
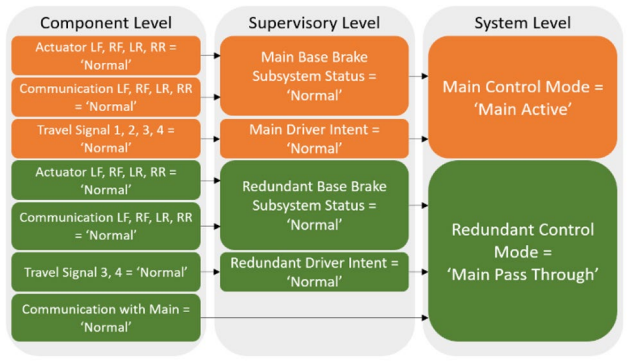


TABLE 2 State Descriptions

Actuator States Calculated	
State Name	**State Description**
Actuator Functional Status (LF/RF/LR/RR)	EMB self-assessment to whether there is capability to provide actuation.
Main_MCU States Calculated	
State Name	**State Description**
Main Braking Communication Status (LF/RF/LR/RR)	Main_MCU assessment of actuator communication paths, and whether receiving information from the EMBs
Main Base Brake Subsystem Status	Main_MCU assessment of which brake corners are available to actuate
Main Base Pedal Emulator Signal (1/2/3/4)	Main_MCU assessment of communication from the BPE_Travel_(1/2/3/4)_Signal
Main Driver Intent Functional Status	Main_MCU assessment of how many brake travel signals are available
Main Control Mode	Main_MCU assessment on the ability to command brake actuators based on driver intent
Redundant_MCU States Calculated	
State Name	**State Description**
Redundant Braking Communication Status (LF/RF/LR/RR)	Redundant_MCU assessment of actuator communication paths, and whether receiving information from the EMBs
Redundant Base Brake Subsystem Status	Redundant_MCU assessment of which brake corners are available to actuate
Redundant Brake Pedal Emulator Signal (3/4)	Redundant_MCU assessment of communication from the Travel_Signal_(3/4)
Redundant Driver Intent Functional Status	Redundant_MCU assessment of how many brake travel signals are available
Redundant Braking Communication Status with Main	Redundant_MCU assessment of communication from Main_MCU
Redundant Control Mode	Redundant_MCU assessment on the ability to command brake actuators, driver intent, and communication with Main_MCU

FIGURE 4 System State Hierarchy in Normal Full Function State



System State Propagation

To understand how failures can propagate through the hierarchical system states in the model, consider the following scenario. A critical internal component within the EMB_LF brake corner actuator fails, rendering that actuator unable to generate braking torque. This critical internal component failure causes the *Actuator Functional Status LF* component-level state within EMB_LF to transition from 'Normal' to 'Failed'. The EMB_LF actuator then communicates this failure state via serial data to other ECUs in the system. This local degradation propagates upward through the system hierarchy, resulting in a compromised status at the supervisory level: both the *Main Base Brake Subsystem Status* and *Redundant Base Brake Subsystem Status* transition from 'Normal' to a degraded state, such as 'Half System,' indicating reduced service braking functionality. This degradation affects both the Main_MCU and Redundant_MCU equally, reflecting a shared reduction in braking capability. Because both MCUs experience the same level of degradation, the system-level *Main Control Mode* and *Redundant Control Mode* states can independently determine that the Main_MCU remains suitable for active control—allowing the system to remain in the 'Main Active' state.

This scenario shows how a single component-level fault clearly propagates upward through the hierarchy, ultimately affecting system-level decision-making and ensuring continued safe operation despite localized failures.

Iterative Design and System State Optimization

Early integration of fail-operational architectural considerations significantly enhances overall system robustness and aligns with ISO 26262 compliance strategies [12]. Through iterative design cycles, engineers use SAT-generated data to evaluate trade-offs between component redundancy, complexity, and reliability. This iterative approach allows design teams to quantitatively assess and justify decisions, leading to optimized solutions that meet reliability targets without incurring unnecessary complexity or cost.

The iterative design approach, supported by the SAT, is essential to optimizing BbW systems. This approach involves defining initial system configurations, calculating reliability and failure probabilities, and iteratively refining system architecture based on calculated system states. System reliability is estimated using Failure in Time (FIT) rates. Specific component failure rates can be derived from industry-standard reliability handbooks [13] and empirical warranty or component durability data of similar components already in the field. Several papers [6, 7, 10] describe active redundancy configurations and the associated failure rates of various components, providing baseline reliability data essential for SAT-based analyses.

The simplified system model architecture that is referenced in this paper resulted in 59 potential single-point and 1,654 dual-point failures. Each failure in the system will result in a propagated calculation of every system state that is defined, which in this example model is 32 unique system states. When extrapolated to realistic, production-level BbW systems, this complexity increases significantly—potentially resulting in hundreds of single-point failures, tens of thousands of dual-point failures, and several dozen distinct system states. This underscores the importance of utilizing a tool such as the SAT to systematically analyze the failure network to identify the worst-case subsequent failures and to ensure that the system functional reliability remains within tolerable limits.

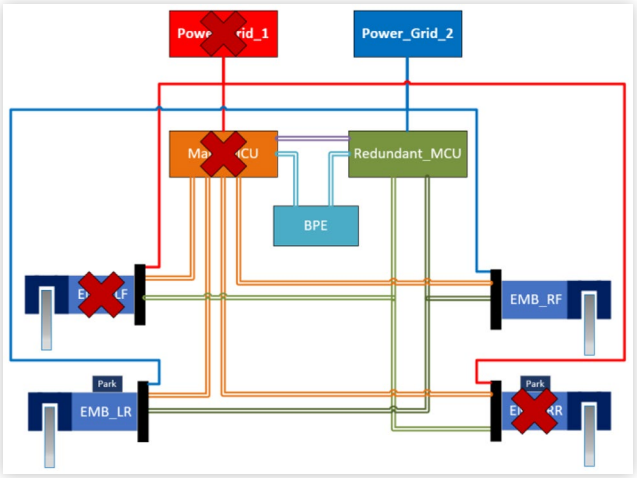
To help understand how the SAT is used, we will focus on one specific failure in the system and analyze two example architecture variations to compare the impacts to the braking system. The first failure example shows how an engineer can use the tool to identify limitations in the system design, such that system functional reliability may not achieve targets. In the second failure example we will show how a specific change to the architecture increases the redundancy and significantly impacts the resiliency of the entire system.

Example 1: Power_Grid_1 Failure with Original Architecture

For the first example, we will consider the failure mode of the low voltage power grid 1 malfunctioning, perhaps due to a detected short in the circuit that caused the entire grid to become disconnected from a power source. This type of failure will propagate to a loss of power for all system elements supplied by power grid 1, causing all connected ECUs to revert to a non-powered state. For the given BbW architecture design referenced in this paper, a failure of Power_Grid_1 causes Main_MCU, EMB_LF, and EMB_RR to lose power. Figure 5 shows this failed situation in the architecture.

Based on how the system is modeled, the Power_Grid_1:Grid Component serves as the initial failure point, with single-point, one-way logical connections to the

FIGURE 5 Example 1 – System Impact



following Components: Main_MCU:PWR_1, Main_MCU:PWR_2, EMB_LF:PWR, and EMB_RR:PWR. When the initial failure occurs, these connected Components also fail as a direct consequence of the modeled logical connections. If all the power supply Components within an ECU fail, the remaining Components in that ECU subsequently fail through additional defined logical connections, representing a complete loss of internal power. Table 3 summarizes the resulting secondary failures caused by the initial failure of Power_Grid_1:Grid. Power_Grid_2 is not impacted by the Power_Grid_1 failure considered in this example.

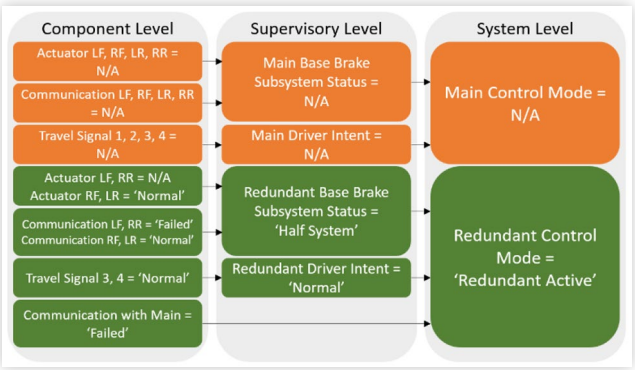
TABLE 3 Example 1 - Resulting System Component Status

Main_MCU		Redundant_MCU	
Component	**Failed**	**Component**	**Failed**
PWR_1	Yes	PWR	No
PWR_2	Yes	GND	No
GND	Yes	SerialData_Red_LF_RR	Yes
SerialData_LF	Yes	SerialData_Red_RF_LR	No
SerialData_RF	Yes	SerialData_Main_Red	Yes
SerialData_LR	Yes	Micro	No
SerialData_RR	Yes	SerialData_Low_2	No
Micro	Yes	BPE_Travel_3_Signal	No
SerialData_Main_Red	Yes	BPE_Travel_4_Signal	No
SerialData_Low_1	Yes	EMB_LF_Signal	Yes
BPE_Travel_1_Signal	Yes	EMB_RF_Signal	No
BPE_Travel_2_Signal	Yes	EMB_LR_Signal	No
BPE_Travel_3_Signal	Yes	EMB_RR_Signal	Yes
BPE_Travel_4_Signal	Yes	Main_MCU_Signal	Yes
EMB_LF_Primary_Signal	Yes		
EMB_LF_Secondary_Signal	Yes		
EMB_RF_Primary_Signal	Yes		
EMB_RF_Secondary_Signal	Yes		
EMB_LR_Primary_Signal	Yes		
EMB_LR_Secondary_Signal	Yes		
EMB_RR_Primary_Signal	Yes		
EMB_RR_Secondary_Signal	Yes		

Table 3 shows that the Redundant_MCU remains functional since it is connected to Power_Grid_2, as illustrated in Figure 5. However, the Redundant_MCU assesses EMB_LF, EMB_RR, and the Main_MCU as failed, likely based on a loss of communication diagnostic that detects the absence of expected messages over an excessive time span. At the supervisory level, the Redundant_MCU combines *Redundant Braking Communication Status (LF/RF/LR/RR)* and *Actuator Functional Status (LF/RF/LR/RR)* to determine that only two of the four brake corner actuators are available, resulting in *Redundant Base Brake System Status* = 'Half System'. Based on the failed Main_MCU_signal, the Redundant_MCU sets *Redundant Communication Functional Status with Main* = 'Failed'. No degradation is detected in the *Redundant Driver Intent Functional Status*. Using these three supervisory system states, the Redundant_MCU transitions to *Redundant Control Mode* = 'Active' due to failed communication with the Main_MCU. The corresponding signal flow is illustrated in Figure 6.

As discussed earlier in this paper, some design strategies may choose to limit the redundant set of brake controls to service braking functionality only, excluding features such as ABS and ESC. This approach can help reduce overall system complexity and may lessen the validation effort associated with duplicating advanced control features in redundant hardware. One possible strategy is to increase redundancy within the Main_MCU to mitigate high failure rate single-point failures, while configuring the Redundant_MCU to provide only the essential service braking functionality. Under this approach, it is generally desirable for the Main_MCU to remain in control as much as feasible, as it offers higher fidelity control and access to a broader range of brake system features—provided it remains operational, maintains sufficient control interfaces, and continues to meet the required safety integrity for its functions. However, since the Main_MCU is powered solely by Power_Grid_1, a failure of this grid renders the main controls inoperable and necessitates a transition to the redundant brake controls housed in the Redundant_MCU.

FIGURE 6 Example 1 - Resulting System States



Using the model output, an engineer can identify this challenge by assessing the functional reliability of ABS and ESC features in the presence of failures. By assigning an estimated failure rate of the power grid 1 and assessing the complete aggregate failure rates to loss of these features, engineers can have a data-driven debate whether to add redundancy to the hardware, or to add complexity to the controls and testing regiments.

Example 2: Power_Grid_1 Failure with Architecture Change

For this second example, we consider adding hardware redundancy by connecting the Main_MCU to both power grid 1 and power grid 2 while incorporating fault-tolerant power management circuitry into the Main_MCU. This circuitry can be designed such that, in the event of a power grid 1 failure, it automatically supplies sufficient power from power grid 2 to the main brake controls MCU, maintaining processing functionality and avoiding the need to transition to redundant brake controls. This architecture change and failed situation is illustrated in Figure 7.

Based on these new changes to the system architecture, the system model maintains the Power_Grid_1:Grid component with single point, one-way logical connections to the following components: Main_MCU:PWR_1, EMB_LF:PWR and EMB_RR:PWR. However, the system model adds a new single point, one-way logical connection between Power_Grid_2:Grid and Main_MCU:PWR_2. As a result, both PWR_1 (power grid 1) and PWR_2 (power grid 2) must fail for the Main_MCU ECU to fail.

This small and relatively easy change to the SAT model functionally increases the Main_MCU availability due to single power grid failures. As shown in Table 4, the Main_MCU maintains functionality despite the power grid failure.

Based on the failure of power to two actuators, the Main_MCU evaluates the system state and sets *Main Brake Subsystem Status* = 'Half System'. There are no failures in the BPE_Travel_(1/2/3/4)_Signal, so the *Main Driver Intent Functional Status* remains 'Normal'. As a result, the Main_MCU sets *Main Control Mode* = 'Main Active'. The Redundant_MCU evaluates the same system conditions and sets *Redundant Brake Subsystem Status* = 'Half System' but determines that *Redundant Control Mode* = 'Main Pass Through', passing through main commands along a redundant communication path. This results in the main brake controls continuing to command the remaining two actuators, potentially preserving ABS wheel slip control on the wheels that are still providing braking. The resulting system states are shown in Figure 8.

TABLE 4 Example 2 - Resulting System Component Status

Main_MCU		Redundant_MCU	
Component	**Failed**	**Component**	**Failed**
PWR_1	Yes	PWR	No
PWR_2	No	GND	No
GND	No	SerialData_Red_LF_RR	Yes
SerialData_LF	Yes	SerialData_Red_RF_LR	No
SerialData_RF	No	SerialData_Main_Red	No
SerialData_LR	No	Micro	No
SerialData_RR	Yes	SerialData_Low_2	No
Micro	No	BPE_Travel_3_Signal	No
SerialData_Main_Red	No	BPE_Travel_4_Signal	No
SerialData_Low_1	No	EMB_LF_Signal	Yes
BPE_Travel_1_Signal	No	EMB_RF_Signal	No
BPE_Travel_2_Signal	No	EMB_LR_Signal	No
BPE_Travel_3_Signal	No	EMB_RR_Signal	Yes
BPE_Travel_4_Signal	No	Main_MCU_Signal	No
EMB_LF_Primary_Signal	Yes		
EMB_LF_Secondary_Signal	Yes		
EMB_RF_Primary_Signal	No		
EMB_RF_Secondary_Signal	No		
EMB_LR_Primary_Signal	No		
EMB_LR_Secondary_Signal	No		
EMB_RR_Primary_Signal	Yes		
EMB_RR_Secondary_Signal	Yes		

FIGURE 7 Example 2 – System Impact

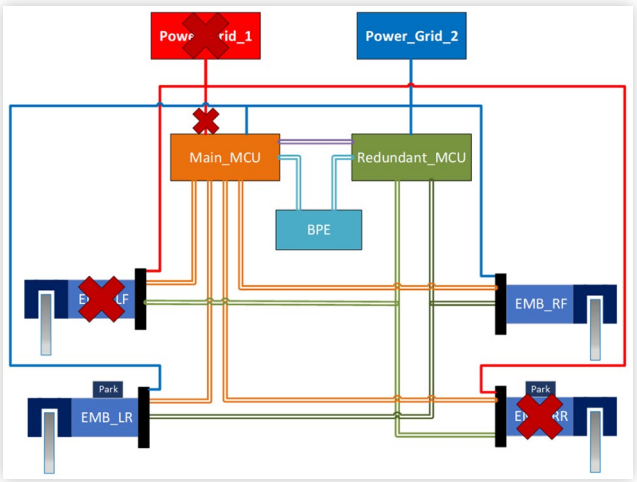
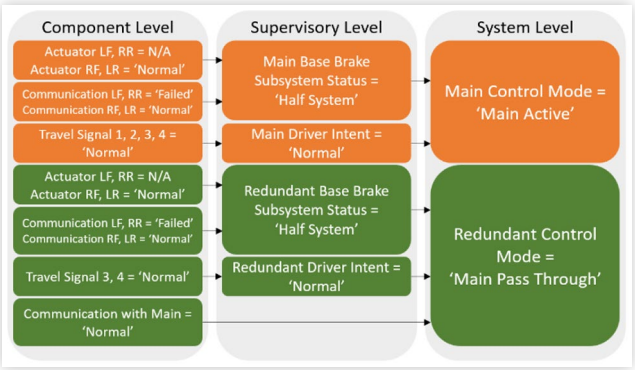


FIGURE 8 Example 2 - Resulting System States



This example illustrates how modifying the architecture to connect redundant power grids to the Main_MCU increases the availability of the main brake controls. However, this improvement comes at the cost of added complexity to the Main_MCU hardware. Increased complexity introduces additional potential failure points, which may, in turn, lead to a higher likelihood of hardware-related failures. Engineering teams must continuously balance added complexity with overall system reliability to arrive at an optimized final product.

Requirement Development and Verification

Developing accurate software control functional requirements from modeled system states and iterative analyses is crucial. The SAT directly informs requirement definitions by identifying critical failure modes, functional redundancies, and acceptable performance degradations. Python code within the SAT explicitly defines the logic used to calculate system states, serving as a direct, executable representation of the intended system behavior. By clearly specifying how each system state is derived—and explicitly defining which ECU is responsible for performing these calculations—engineers can rapidly derive detailed, precise, and verifiable system requirements. Knowing exactly which ECU calculates a given system state enables efficient allocation of derived requirements to the appropriate elements within the overall system architecture. In some cases, these clearly defined requirements are directly communicated to external suppliers, ensuring the delivered software meets the necessary functional behavior. This structured and systematic allocation significantly streamlines requirement decomposition, facilitating efficient translation of high-level system behaviors into specific, implementable software functionalities. Ultimately, this structured approach enables faster, more precise, and robust development of software solutions, significantly enhancing the efficiency and effectiveness of the overall system development process.

Additionally, the SAT generates targeted test cases based on defined system states and their associated failure conditions. This systematic generation of test cases streamlines verification processes, ensuring comprehensive system testing and validation. The resultant test cases form an essential part of the vehicle's functional safety verification in compliance with ISO 26262 standards [5].

Certain key system states can be associated with specific performance "buckets," effectively representing the expected braking capability under defined conditions [4]. The SAT model is further enriched by incorporating component failure rates and performing Reliability Block Diagram (RBD) analyses to calculate overall system reliability. The tool quantifies relationships between the system performance levels (represented by calculated system states) and the aggregate failure rates causing those specific performance degradations [4].

Finally, the SAT facilitates the development of robust production vehicle verification plans for degraded system functionalities by assigning Diagnostic Trouble Codes (DTCs) to specific component failures. This enables efficient and automated generation of software test cases expressed in Gherkin syntax (Given-When-Then), ensuring thorough testing and precise verification of all defined system states.

Summary/Conclusions

The development and utilization of the SAT has significantly advanced the understanding and optimization of Brake by Wire systems within SDV architectures. The SAT's comprehensive modeling approach clarifies the intricate interdependencies among components, enabling precise identification and mitigation of potential system failures.

The methodology described provides an effective means of managing system complexity inherent in SDVs, optimizing both system design and iterative refinements. The SAT facilitates an effective balance between reliability, safety, and cost, providing quantifiable metrics that are essential for informed decision-making throughout the entire system development lifecycle.

Future work will include expanding the SAT's modeling capabilities, incorporating more advanced aggregate failure rate analysis algorithms, and enhancing its integration into automated design and verification workflows. Additionally, the team is working to expand the usage of the SAT to other safety critical vehicle systems, which will continually improve the reliability and safety of next-generation SDVs.

References

1. Sundar, M. and Plunkett, D., "Brake-by-Wire, Motivation and Engineering - GM Sequel," SAE Technical Paper 2006-01-3194 (2006), doi:10.4271/2006-01-3194.
2. Schrade, S., Röhler, A., Nowak, X., Verhagen, A. et al., "Safety Concepts for Future Electromechanical Brake Actuators," *SAE Int. J. Passeng. Veh. Syst.* 17, no. 3 (2024), doi:10.4271/15-17-03-0012.
3. Antanaitis, D., "A Renewed Look at Centralized vs. Decentralized Actuation for Braking Systems," SAE Technical Paper 2023-01-1865 (2023), doi:10.4271/2023-01-1865.
4. Antanaitis, D. and Heil, E., "Application of Brake System Failed State Performance and Reliability Requirements to Brake System Architecting," *SAE Int. J. Adv. & Curr. Prac. in Mobility* 4, no. 3 (2022): 973-983, doi:10.4271/2021-01-1267.

5. International Standards Organization (ISO), "Road Vehicles – Functional Safety," ISO 26262:2018, Second Edition, 2018.
6. Hammett, R.C. and Babcock, P.S., "Achieving 10^{-9} Dependability with Drive-by-Wire Systems," SAE Technical Paper 2003-01-1290 (2003), doi:10.4271/2003-01-1290.
7. Schrade, S., Nowak, X., Verhagen, A., and Schramm, D., "Generic X-Domain Hazard Analysis and Risk Assessment," SAE Technical Paper 2023-01-0580 (2023), doi:10.4271/2023-01-0580.
8. Douglass, B.P., *Agile Systems Engineering* (Waltham, MA: Morgan Kaufmann, 2016)
9. Cheon, J., Kim, J., Jeon, J., and Lee, S., "Brake By Wire Functional Safety Concept Design for ISO/DIS 26262," SAE Technical Paper 2011-01-2357 (2011), doi:10.4271/2011-01-2357.
10. Zhang, C., Han, Y., Lin, Y., and Wang, D., "Reliability Analysis of Brake-by-Wire Systems on Fault Tree," *Journal of Physics: Conference Series* 2029 (2021): 012135, doi:10.1088/1742-6596/2029/1/012135.
11. Ebner, C., Gorelik, K., and Zimmermann, A., "Automated Design Exploration and Dynamic Safety Analysis for Optimization of Mechatronic Systems in Safety-Critical Automotive Applications," *IEEE Systems Journal* 17, no. 4 (2023): 5357-5368, doi:10.1109/JSYST.2023.3324850.
12. Sinha, P., "Architectural Design and Reliability Analysis of a Fail-Operational Brake-by-Wire System from ISO 26262 Perspectives," *Reliability Engineering and System Safety* 96, no. 10 (2011): 1349-1359, doi:10.1016/j.ress.2011.05.008.
13. U.S. Department of Defense, "Reliability Prediction of Electronic Equipment, MIL-HDBK-217F, Notice 2," Department of Defense, Washington, DC, 1995.

Definitions/Abbreviations

ABS - Antilock Braking System
ASIL - Automotive Safety Integrity Level
BbW - Brake by Wire
BPE - Brake Pedal Emulator
CCU - Centralized Compute Unit
DTC - Diagnostic Trouble Code
EBCM - Electronic Brake Control Module
ECU - Electronic Control Unit
EMB - Electro-Mechanical Brake
EPB - Electric Parking Brake
ESC - Electronic Stability Control
FIT - Failures In Time
FMEA - Failure Mode Element Analysis
FTA - Fault Tree Analysis
GND - Electrical Ground
MCU - Microcontroller Unit
PWR - Electrical Power
RBD - Reliability Block Diagram
SAT - System Analysis Tool
SbW - Steer by Wire
SE Device - Smart Electrical Device
SDV - Software Defined Vehicle

ISSN 0148-7191