Accelerating SDV Transition: Converting Classic Components to Services

Seth Shaji George
College of Engineering Trivandrum

Nijitha P
College of Engineering Trivandrum

Vaishnavi Ganeshan
College of Engineering Trivandrum

James Joy
Tata Elxsi
Delivery Manager

Abstract

The automotive industry is undergoing a significant transformation due to new technologies like electric cars, connected systems, autonomous driving, and a focus on sustainability. A key development is Software Defined Vehicles (SDVs), where features and functions are primarily enabled through software. This project focuses on converting software components to a service-oriented framework using a system that creates adaptive AUTOSAR templates. The system requires ARXML files as input to produce a template with pre-filled adaptive AUTOSAR code, which can be further customized. This method was tested on a door software system, which includes door hardware, door software, and vehicle components. It successfully generated about 70% of the needed code. This demonstrates that automatic code generation for service-oriented architectures can save significant development time and effort. Future improvements could include adding strong security measures to enhance the safety and reliability of SDVs.

1 Introduction

The automotive industry is changing rapidly with advancements in technology (Lopez-Vega & Moodysson, 2023), which are altering how vehicles are designed, built, and used. Important trends driving this change include the rise of electric vehicles, improved connectivity, and autonomous driving capabilities. Software Defined Vehicles (SDVs) (Goswami, 2024) are a major part of this change, shifting the focus from hardware to software. SDVs use advanced software, artificial intelligence, and connectivity to become more like electronic devices on wheels.

A critical component of SDVs is the adoption of Service-Oriented Architecture (SOA), which enhances functionality and flexibility. This is a shift from the traditionally used signal-based communication systems. However, the transition to SOA requires a significant re-engineering of traditional software development approaches. This research aims to address this challenge by proposing a novel method for converting software components to a service-oriented framework. As a proof of concept, a system that generates adaptive AUTOSAR templates is developed. The system takes necessary ARXML files as input and produces a template with pre-filled adaptive AUTOSAR code, which users can then customize for their specific needs.

This project demonstrates that automatic code generation for service-oriented architectures can significantly reduce development time and effort. Future enhancements could focus on adding strong security measures to ensure the safety and reliability of SDVs, especially as they become more connected and autonomous.

2 Literature Review

Transitioning to Software-Defined Vehicles (SDVs) is a pivotal step for the automotive industry, as it marks a shift towards software-centric innovation and connectivity-driven advancements. SDVs are transforming the automotive value chain, emphasizing software as a core driver of innovation. The integration of SDVs reshapes the industry's ecosystem by introducing new business models and fostering collaboration between traditional OEMs and tech firms. Challenges include aligning with industry standards and ensuring seamless integration. (Liu, Zhang, & Zhao, 2022)

Service-Oriented Architecture (SOA) is a design paradigm that organizes software into modular, interoperable services. SOAs are pivotal in SDVs, offering a flexible framework for integrating software modules.

◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇
Received : 28 November 2024
Revised : 01 December 2024
Accepted : 04 December 2024

This architecture supports variability and facilitates over-the-air updates, enabling continuous vehicle enhancement and reducing development costs. (Rumez, Grimm, Kriesten, & Sax, 2020)

Adaptive AUTOSAR is a modern platform designed to address the needs of high-performance computing in vehicles. It enables features like over-the-air updates, service-oriented communication, and enhanced processing power to support autonomous driving and connected systems. Unlike Classic AUTOSAR, Adaptive AUTOSAR supports parallel processing on multicore and GPU-based hardware, making it more suitable for dynamic applications. However, challenges include handling nondeterminism and ensuring real-time behavior in service-based environments (Reichart & Asmus, 2021)

3 Design

3.1 Development Stack

Programming Languages: Python for template generation.
Middleware: SOME/IP for communication between services.

3.2 System Design

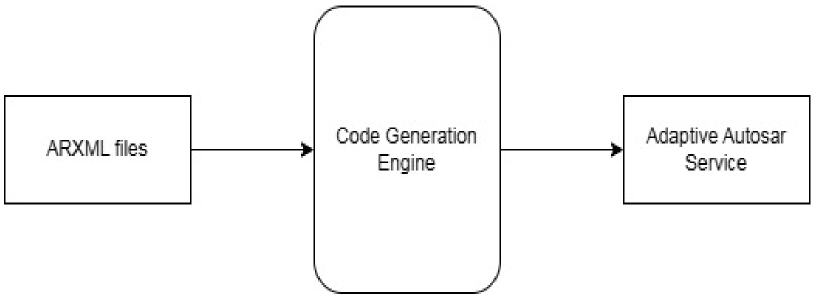

Figure 1: System design

ARXML files: They describe the adaptive applications and their services, resources, and required execution environments.
Code Generation Engine: It identifies the events, fields and methods from the ARXML files so as to generate the necessary service
Adaptive Autosar Service: The identifiers in the code used are meaningful and self-explanatory. The developers can easily edit the code to suit for their specific needs.

4 Discussion

4.1 Signal Based Communication Vs Service-Oriented Communication

Whether to opt for Signal-Based or Service-Oriented Communication in automotive systems depends on the specific needs of the domain. These options represent different communication systems employed in the software and systems design of automobiles.

Signal-based communication, which is employed in Classic AUTOSAR, is facilitated by communication bus networks such as CAN, FlexRay, MOST, and LIN. It is best suited for applications where software remains relatively static throughout its operational life cycle, like engine control, braking systems, and airbag control units. It prioritizes stability and deterministic behavior, leveraging the AUTOSAR Runtime Environment (RTE) for scheduling software components and managing communication between them without the need for an operating system (OS). This simplicity and predictability make it ideal for safety-critical functions that demand reliability over the long term.

On the contrary, Service-Oriented Communication, used by Adaptive AUTOSAR, adopts Ethernet and SOME/IP to enhance communication by utilizing advanced middleware functionalities such as serialization and Remote Procedure Call (RPC). This setup enables more dynamic interaction among ECU software components, facilitating efficient data exchange and coordination. This architecture is designed to handle automotive applications that require flexibility, scalability, and the ability to undergo frequent updates and modifications. Systems such as infotainment, telematics, advanced driver assistance systems (ADAS), and over-the-air

(OTA) updates benefit significantly from this model. It supports high-performance and computation-intensive functionalities. This is facilitated by the POSIX operating system that handles scheduling and manages communication between software components. Service-oriented communication enables automotive systems to transform rapidly in response to technological advancements and changing consumer demands, making it well-suited for applications where agility and adaptability are critical.

Adaptive and Classic AUTOSAR coexist in automotive systems. The Classic ECU serves as a gateway, converting bus signals for the Adaptive ECU. Adaptive AUTOSAR enables dynamic upgrades and flexible communication through Proxy/Skeleton and Publish/Subscribe approaches. Classic AUTOSAR ensures reliability for real-time tasks and safety applications. Together, they offer complementary solutions to enhance automotive systems.

a) Proxy/Skeleton:
This pattern involves generating two essential components from a formal service definition. The Service Proxy serves as a facade for the service consumer, representing the service's functionalities at the code level. It simplifies interaction by managing communication with the remote service implementation through a local interface. Conversely, the Service Skeleton connects the service implementation to the communication management transport layer, enabling distributed service consumers to access the service. This component facilitates requests from remote clients and integrates the service implementation via a structured relationship, providing a clear pathway for application code to interact with the middleware.

b) Publish/Subscribe:
In Service-Oriented Architecture (SOA), the publish/subscribe pattern is an event-driven messaging paradigm. It enables loose coupling between services, allowing them to communicate with each other in a decoupled manner. The producers of information (publishers) are decoupled from the consumers (subscribers).

4.2 Implementation

The system developed for this project automates the conversion of software components to a service-oriented framework. As proof-of-concept, we have taken Adaptive Autosar as the Service-Oriented framework. The primary objective is to take ARXML files as input and generate a significant portion of the necessary code automatically. The following steps outline the implementation process:

1. Input Preparation
   a) ARXML Files: Collect the necessary ARXML files containing the descriptions of the software components, their interfaces, and communication patterns. These files are essential for defining the structure and behaviour of the components in the adaptive AUTOSAR framework.
   b) Component Identification: Identify the software components that need to be converted. In this case, we focused on a door software system, which includes door hardware, door software, and vehicle components.
2. Parsing ARXML Files
   The parser extracts relevant information such as component names, interfaces, and communication signals. In this project we have used Python ElementTree XML API (xml.etree.ElementTree).
3. Template Generation
   Design a set of adaptive AUTOSAR templates that can be pre-filled with the extracted data. These templates include basic structures for components,services, and communication mechanisms.
4. Customization
   Developers can customize the generated code to meet specific requirements. This step involves adding system-specific logic and fine-tuning the behaviour of the components.

5 Results

The tool developed was successfully tested on a door software system (given in Figure 2), including door hardware, door software, and vehicle components. Actions like locking and unlocking the door or raising and lowering the window are generated based on events like rain or a crash, demonstrating the system's practical use.

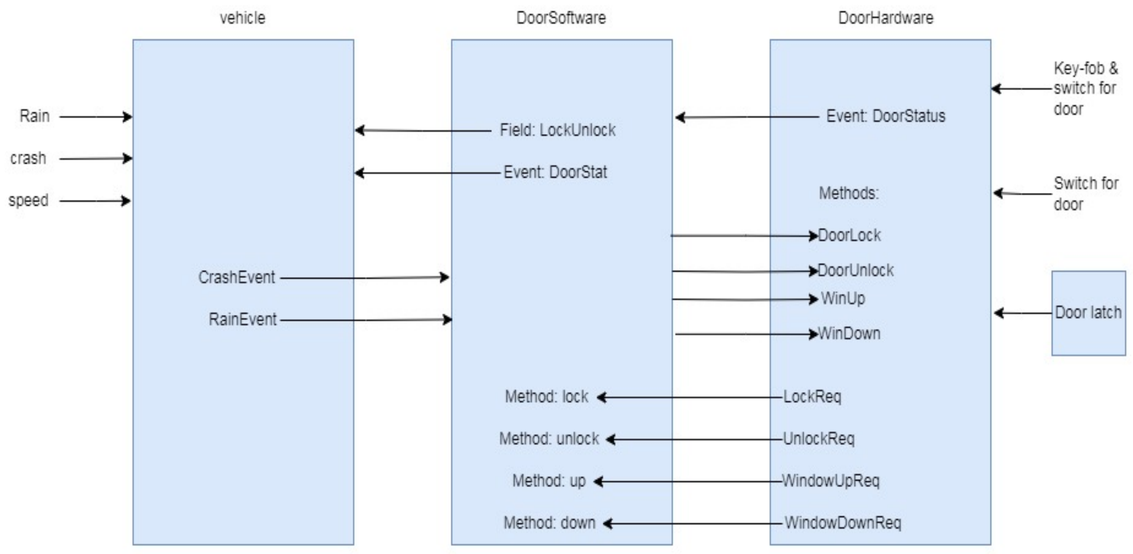

Figure 2: DoorSoftware system for the generated code.

Around 70% of adaptive AUTOSAR code is found to have generated successfully. The rest of the code can be completed by the developer according to their system-specific logic. This technique significantly reduces development time, enables code reusability, and reduces complexity.

6 Conclusion

The transition from traditional to software-defined vehicles (SDVs) represents a significant leap forward in automotive technology. This transformation is enabled by service-oriented architecture (SOA), which converts classic components into modular, interoperable services, thereby enhancing flexibility, scalability, and innovation within the automotive industry. However, this transition from existing signal-based communication systems and classic automotive platforms could be time and labour-intensive. This work presents a method for the automatic code generation for SOA.

7 Future Scope

The proposed method suggests a novel technique for automatic code generation for SOA. However, further research is needed to improve the efficiency and security of the system. As vehicles become increasingly connected and reliant on complex software ecosystems, they also become more vulnerable to cyber threats. Robust security measures are essential to protect against potential breaches that could compromise vehicle functionality, safety, and user data. Implementing strong encryption, secure communication protocols, and continuous monitoring is vital to prevent unauthorized access and ensure the integrity of the system. Additionally, as SOAs enable modular and interconnected services, securing each individual component and the interfaces between them is crucial to maintain the overall security posture. Prioritizing security in SDVs and SOAs not only safeguards the vehicle and its occupants but also builds consumer trust, paving the way for the widespread adoption of advanced automotive technologies.

References

Goswami, P. (2024). The software-defined vehicle and its engineering evolution: Balancing issues and challenges in a new paradigm of product development.

Liu, Z., Zhang, W., & Zhao, F. (2022, 03). Impact, challenges and prospect of software-defined vehicles. , 3.

Lopez-Vega, H., & Moodysson, J. (2023). Digital transformation of the automotive industry: An integrating framework to analyse technological novelty and breadth. *Industry and Innovation*, *30*(1), 67–102.

Reichart, G., & Asmus, R. (2021). Progress on the autosar adaptive platform for intelligent vehicles. In T. Bertram (Ed.), *Automatisiertes fahren 2020* (pp. 67–75). Wiesbaden: Springer Fachmedien Wiesbaden.

Rumez, M., Grimm, D., Kriesten, R., & Sax, E. (2020). An overview of automotive service-oriented architectures and implications for security countermeasures. *IEEE Access*, *8*, 221852-221870.