DIGITAL INDUSTRIES SOFTWARE

Automotive systems and software engineering using the Capital Software Developer solution with IBM Rhapsody

Executive summary

As the automotive industry navigates the shift from mechanical engineering to software-centric design, the transformation demands a deep integration of advanced computing technologies and a paradigm shift in how vehicles are conceptualized and built. Software developers must ensure seamless interaction between an ever-increasing number of Electronic Control Units (ECUs) and software consolidation through centralization, managing the complexities of real-time data processing, cybersecurity and system reliability. Additionally, they must address evolving expectations of consumers who seek sophisticated and personalized driving experiences, all while adhering to stringent safety and regulatory standards.

Importantly, the development process has long been characterized by disconnection and data silos. Teams must find ways to play better together and standardize processes, in order to progress faster.

Contents

Introduction

To address these challenges, Siemens and IBM have partnered to deliver an innovative software integration between Siemens Capital™ software and IBM® Engineering Systems Design Rhapsody® for systems engineering, with solutions from the Siemens Xcelerator portfolio of software and services. With Capital Software Developer solution with IBM Rhapsody, customers benefit from advanced systems engineering and asset management, and seamless workflows between mechanical, electronics, electrical and software engineering domains. The solution enables effective collaboration by eliminating disconnected processes and data silos (figure 1).

In this whitepaper, we'll examine how the integration between Capital and IBM Rhapsody can help automotive software development teams:

- De-risk complex software development by system-level early integration and testing
- Reduce time to market with end-to-end concurrent design on connected data
- Improve quality with correct-by-construction generative design
- Catch errors early using virtual ECU testing
- Quickly deploy cyber-safe and secure production code with preconfigured AUTOSAR modules

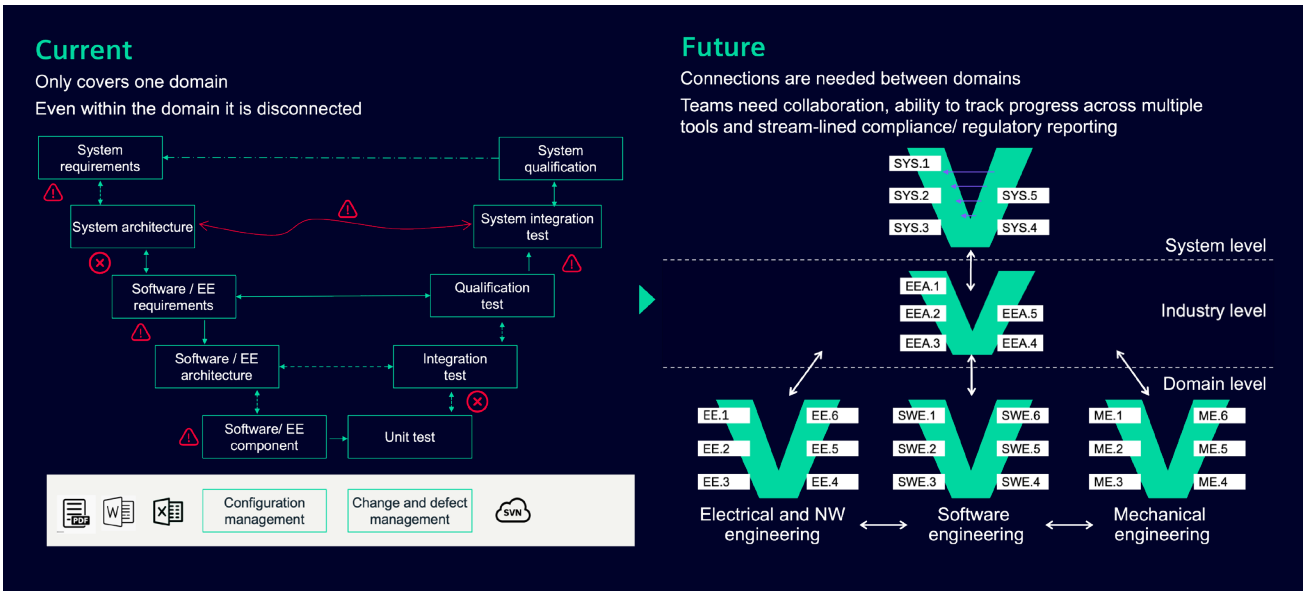

Figure 1. Improve sustainability and traceability throughout the product development lifecycle.

Building the software-defined vehicle

Demand for smart, connected, sustainable products increasingly drives the software-defined vehicle. Consumers want vehicles that are not just modes of transportation but integrated, intelligent systems offering advanced connectivity, eco-friendly options and personalized experiences (figure 2). This shift has put software at the heart of vehicle innovation, enabling the following features and capabilities:

- **Advanced Driver Assistance Systems (ADAS) and autonomous driving:** This is one of the most software-intensive areas in modern automotive design. The development of autonomous and semi-autonomous driving systems heavily relies on software for functions like sensor data processing, decision-making and vehicle control.
- **Digital subscriptions:** Over-the-Air (OTA) updates enable automakers to remotely unlock features in the vehicle post-sale. This technology has revolutionized the way vehicle features are distributed and managed, offering a more tailored experience for consumers, and ongoing revenue opportunities for manufacturers. However, it also requires a significant shift in traditional automotive business models and manufacturing processes.
- **Electrification:** EVs require sophisticated software for managing battery systems, electric drivetrains and energy consumption. Software is essential for optimizing EV performance and range.
- **Connected vehicles and IoT:** The Internet of Things (IoT) has reached the automotive sector as vehicles increasingly become connected devices. This connectivity allows for features such as real-time traffic updates, remote diagnostics and OTA software updates.
- **Sustainability:** Software plays a crucial role in managing electric vehicle systems, such as battery management, energy efficiency and range optimization. Advanced algorithms can determine the most efficient use of battery power, extend the driving range and manage regenerative braking systems, making EVs more practical and appealing to a broader range of consumers. Software can also predict when parts of a vehicle need maintenance before they fail or become less efficient, extending the life of vehicle components and ensuring they operate in an environmentally efficient manner.

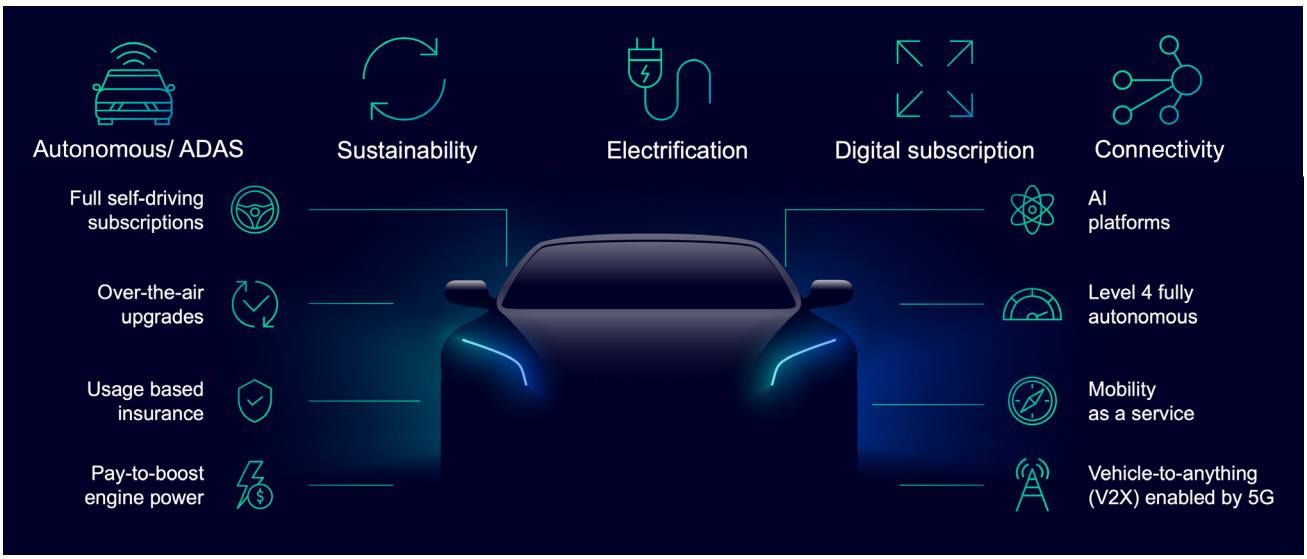

Figure 2. The automotive industry is transforming, rapidly emerging into software-driven innovations.

While consumers have come to expect these innovative capabilities, complexity keeps increasing exponentially. Maintaining performance while staying within network bandwidth budgets is an ongoing challenge.

The need for a new software design approach

As the automotive industry continues its transformation, the number of ECUs in modern vehicles proliferates, necessitating a shift in software design approaches. Traditional fragmented, decentralized software development practices – where each ECU operates independently with its own dedicated software – leads to inefficiencies in resource usage, performance and integrations. The complexity and cost of maintaining numerous individual systems in this manner makes keeping pace with the market's expectations and maintaining competitive advantage impossible.

Designing today's sophisticated software-defined vehicles requires bringing together ECUs sourced from a multitude of tier-one and tier-two suppliers, and integrating them into high-performance compute-central ECUs. Additionally, agile methodologies, continuous integration and deployment, and rigorous testing protocols are becoming commonplace, accelerating development cycles and enhancing the reliability and security of automotive software.

Let's examine some of the new ways developers are addressing increased complexity as they develop software-defined vehicles:

- **Centralized architectures:** Centralized, integrated software architectures consolidate multiple functions into fewer, more powerful ECUs or even a single central computing unit. This approach allows for better management and coordination of different vehicle systems along with improved performance, efficiency and scalability. It reduces the redundancy of components, and simplifies wiring and communication between systems.
- **Domain-based architectures:** A domain-based approach groups related functions into distinct domains, each managed by a central ECU. For example, all drivetrain functions might be grouped under one ECU, while another handles all infotainment responsibilities. This approach strikes a balance between fully distributed and fully centralized architectures.
- **Zonal architectures**: Another emerging trend is the zonal architecture, where the vehicle is divided into zones (front, rear, left and right), and each zone has its own ECU controlling all the functions within that area. This approach simplifies wiring harnesses and can lead to weight reduction and improved reliability.
- **Integration of high-performance computing platforms:** These platforms can process large amounts of data from advanced sensors (like LIDAR and cameras used in ADAS systems) and support sophisticated algorithms required for autonomous driving and other advanced features.
- **Automotive ethernet:** Traditional automotive communication systems like Controller Area Network (CAN) and Local Interconnect Network (LIN) are being supplemented or replaced with Automotive Ethernet. Ethernet supports higher data transfer rates, which is essential to manage the increased data flow between ECUs in modern vehicles.
- **Modular, scalable software platforms:** As hardware architectures evolve, so does the need for more modular and scalable software platforms with more service-oriented software interfacing. Software now needs to be designed to work across different vehicle models and types, allowing for easier updates and feature enhancements. Modular software design allows for code reuse across different vehicle models and platforms, as well, reducing development time and cost.

- **Increased focus on cybersecurity:** With more ECUs and increasingly connected architectures, cybersecurity has become a critical concern. Architectural changes now often include enhanced security features at both hardware and software levels to protect against hacking and unauthorized access.

For automotive manufacturers to implement these new design approaches and software architectures successfully, they must adopt a more holistic management approach that spans the product and asset lifecycle. Teams must be able to collaborate efficiently throughout development and track progress across multiple tools and workstreams. Mitigating risk requires an actionable "digital thread" that connects all engineering domains, and as well as engineering processes across design layers and lifecycle stages (figure 3).

Stay integrated with Siemens Xcelerator

Siemens' flexible and open MBSE ecosystem

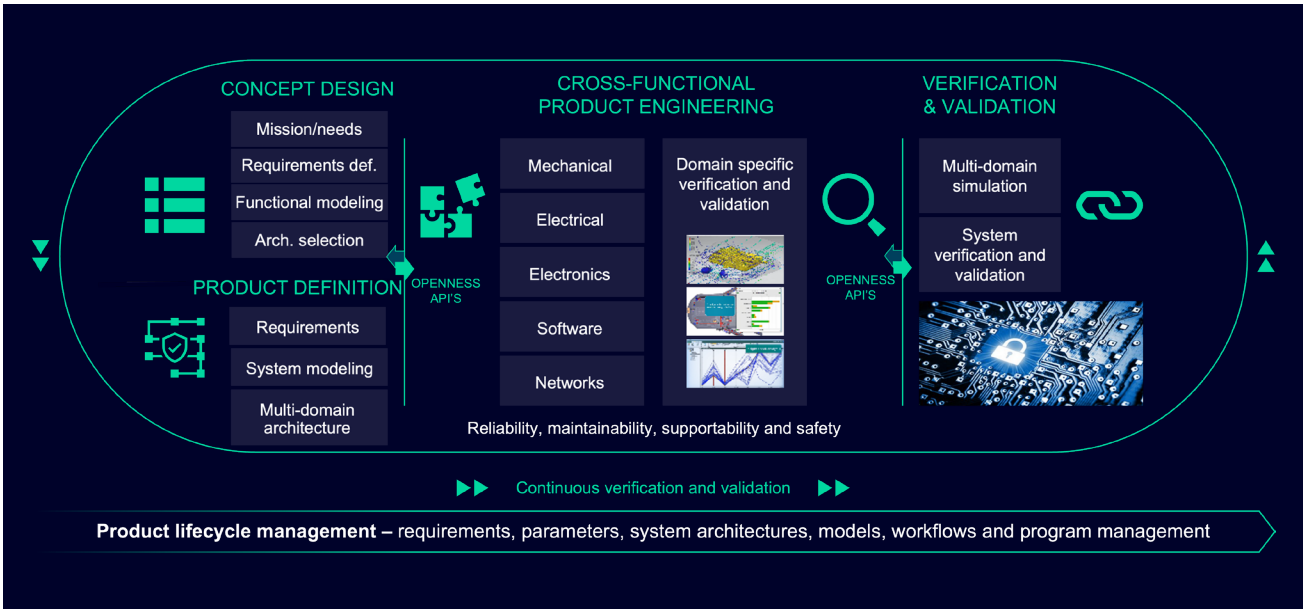

Figure 3. The digital thread integrates data from various stages of the software development lifecycle and makes that accessible across various teams and stages, ensuring that all stakeholders have access to up-to-date, accurate information.

Capital and IBM Rhapsody: Optimizing automotive product lifecycles

Siemens is working with IBM to deliver a combined software solution that helps organizations optimize automotive product lifecycles (figure 4). This solution includes:

- Capital Software Developer with IBM Rhapsody for software and systems engineering
- Systems Modeler with IBM Rhapsody for systems modeling based on SysML
- IBM Maximo Application Suite for asset management with Siemens Teamcenter software to support an integrated digital thread between service engineering, asset management and services execution

The new integrated engineering software suite supports traceability and sustainable product development with a digital thread that links mechanical, electronics, electrical engineering and software design and implementation. The solution spans the product lifecycle, from early design and manufacturing to operations, maintenance, update and end-of-life management, improving traceability across processes, prototypes and test concepts for sustainable product development. The solution complies with standards for open integration, including OMG SysML, UML, UTP, AUTOSAR, Linux, OSLC, C++ and SystemC.

Let's take a look at the various components of this design flow in greater detail.

Capital and Rhapsody for E/E systems development

An end-to-end software design flow in E/E context for AUTOSAR and beyond

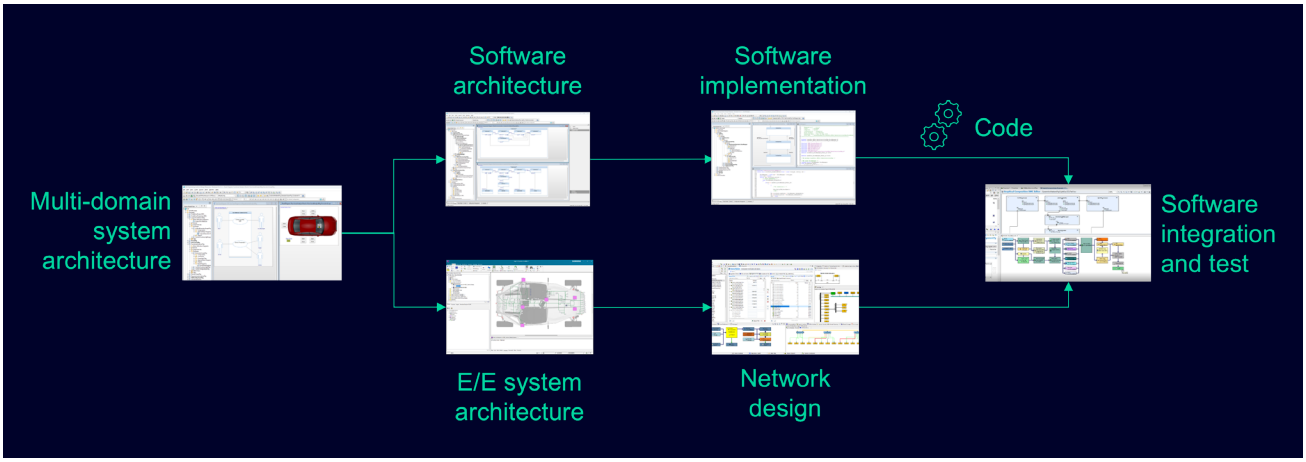

Figure 4. The multi-domain system modeling feeds both E/E systems architecture and software architecture, followed by software implementation within IBM Rhapsody. Outcomes flow into Capital Embedded for deployment into the AUTOSAR compliance ECU basic software during integration and testing.

Multi-domain system architecture

To begin the design flow, inputs for building a multi-domain system architecture include stakeholder requirements, business goals, scope definitions and constraint definitions. The stakeholder requirements are captured as models and validated by simulating and analyzing use case activities, sequence and state machine diagrams. This happens before conducting the functional analysis and transitioning to the creation of the cross-domain, multi-domain architecture models – which are expressed in terms of block diagrams and interconnections of blocks on block diagrams. The entire process is OMG SysML compliant and adaptable to the needs of the enterprise via simple configuration of custom profiles.

Next, the outcomes flow into the E/E and software architecture steps:

1. The E/E architecture and the system multifunctional internal block diagrams from IBM Rhapsody are key inputs. These include additional requirements, for example, bill of signals and carriers or functions. During this step, the platform designs are created, consisting mostly of components and their interfaces, and network pathways. This is made possible with Capital Systems Architecture capabilities.
2. The early KPI design rule check (DRC) analysis provides early insight into the resource consumption of CPU memory and network bandwidth requirements from a software perspective, as well as the electrical key performance indicators (KPIs), reducing the risk of rework costs late in development. These and additional DRCs help to improve design quality and enable right-the-first-time design. The DRCs also provide check points for design readiness for hand-off to the next technical steps.

Deliverables for this step in the development process include the electrical connectivity proposal, and the network topology or system description. The ability to auto-generate these descriptions eliminates the need to re-enter data and prevents the introduction of errors.

Here is an example of an Adaptive Cruise Control system designed in IBM Rhapsody (figure 5).

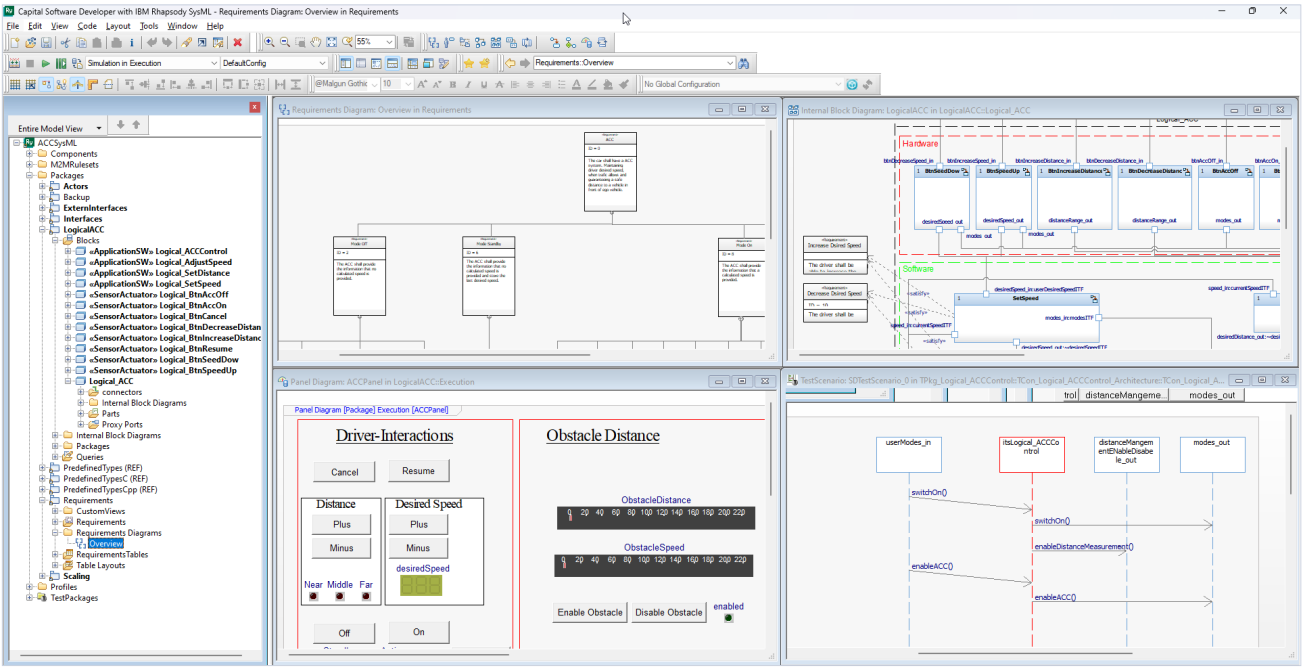

Figure 5. Multi-domain system modeling with stakeholder requirements validation and early model-based testing.

Capital can leverage the design specifications to generate functional diagrams that are ready to be integrated onto platform definitions. The new functions specified by the Adaptive Cruise Control multi-domain system model are allocated to a new ECU component, which will run those software functions. Additional design rule checks help to make sure that there are no misallocations or missing connectivities. Capital Systems Architecture then generates the network topology description, which flows directly into Capital Networks for optimal routing and scheduling across all network segments.

Software architecture definition

Now, let's take a look at how the SysML functions flow to software architecture in Capital Software Developer with IBM Rhapsody.

The software path from the SysML high-level system design is the key input in this step. Automatically generating the initial software architecture proposal in SysML accelerates the process, and a built-in framework supports iterative design and collaboration. The resulting software design is then embellished with interface-level behavior specifications, and the code interface APIs are generated. The accompanying AUTOSAR run-time environment APIs are handed over to implementation suppliers who may continue within IBM Rhapsody, but may also leverage a third-party integrated development environment (IDE) or model-based implementation tools based on AUTOSAR XML.

IBM Rhapsody supports multi-standard and extensible software interface design, including support for AUTOSAR Classic, with AUTOSAR Adaptive on the roadmap. The flow is equally applicable to arbitrary software described based on the Unified Modeling Language (UML). However, if AUTOSAR is relevant, apart from generating code interface APIs and accompanying AUTOSAR XML models, having these artifacts together make the handover to implementation suppliers more reliable and efficient (figure 6).

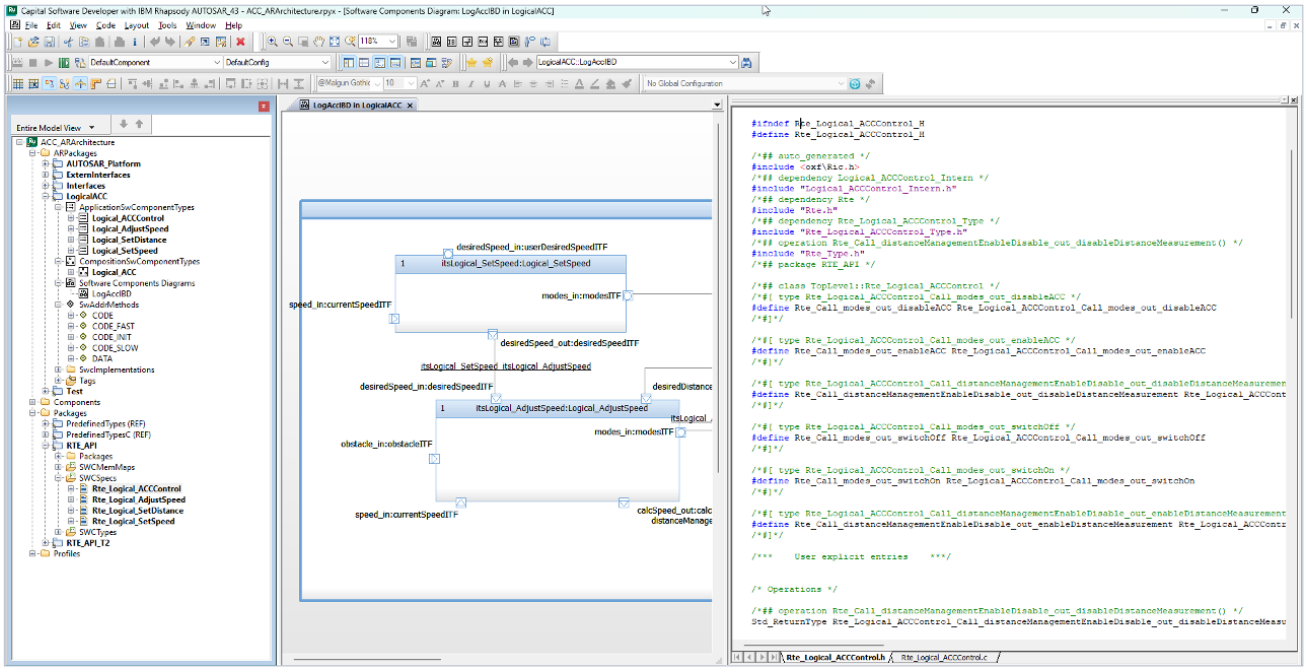

Figure 6. UML and AUTOSAR help in visualizing, specifying, analyzing and verifying the various components and interactions within complex systems. This is crucial in the early stages of development to ensure that all system requirements are met and integrated.

Software implementation in IBM Rhapsody

The starting point for implementing software in IBM Rhapsody is the software architecture. This includes the software interfaces that have been defined for the various components, along with the software requirements for internal behavior, which consists of executable functions, events and triggers, memory mapping constraints and so on. UML supports model-driven software implementation through state machine and sequence diagrams, along with code generation into multiple programming languages such as C, C++ and Java. Integrated model-based testing within IBM Rhapsody provides early verification and improves software quality to the highest levels.

The outcome of this activity is completely implemented, unit-tested and ready-to-integrate software component packages, including soft code and XML-based model descriptions for the AUTOSAR Classic platform, or, soon, the AUTOSAR Adaptive platform. The software component source code and XML models can then be integrated with embedded basic software and middleware environments, and tested on a virtual ECU environment (figure 7). Capital Embedded ties directly into the software-driven ECU development flow, accelerating verification and integration of applications into a production-quality AUTOSAR software platform, with state-of-the-art safety and cybersecurity features.

The input to this step is the software component description consisting of ARXML as well as source code and the ECU extract – the former coming from IBM Rhapsody, the latter coming from Capital – along with additional integration requirements. Next, generative tools automate integration and reduce program risk. They also help to shorten design cycles and enable developers to pinpoint and address problems early, when they're easiest and less expensive to fix (figure 7).

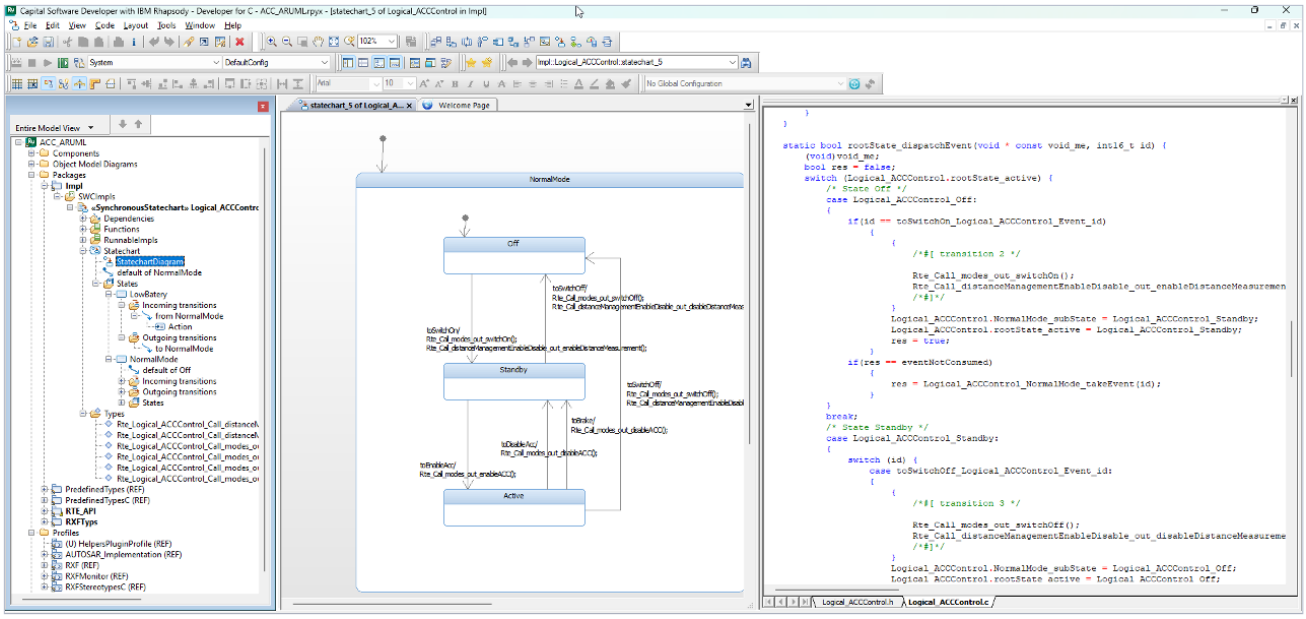

Figure 7. Model-based component behavior implementation using state machines for production code generation.

ECU integration and virtual testing

Finally, it is time to integrate the new software components to the ECU based on the extended ECU extract.

Capital Embedded ties directly into the software-driven ECU development flow, offering out-of-the-box basic software modules for ISO 26262 up to ASIL D and compliant with cybersecurity needs. Early verification and validation of applications on our production-quality AUTOSAR software platform based on virtual ECUs reduces integration risk before hardware ECU samples are available.

Following are the key steps for integrating and testing the ECU on virtual hardware:

- Highly automated tools accelerate the configuration of the AUTOSAR basic software module stack, shortening development cycles.
- Model-based testing is deeply embedded into our solution and the continuous extension of test cases is encouraged.
- Model-based test scenarios described in Capital Software Developer with IBM Rhapsody are executed directly in the virtual ECU platform. Test outcomes are reported back and recorded along with test reports in Capital Software Developer with IBM Rhapsody.
- Witnesses enable test engineers to revisit the evidence of the test verdict at a later time point.
- Finally, trace data that has been collected during test execution on the virtual ECU platform is available for inspection using the built0in trace inspection capabilities of Capital Embedded Virtualizer.

Virtual testing of the ECU concludes the flow for adding a system to the E/E architecture. Capital and IBM Rhapsody for E/E systems development provide an end-to-end software design flow with two parallel streams that come together for a fully connected, verified system (figure 8).

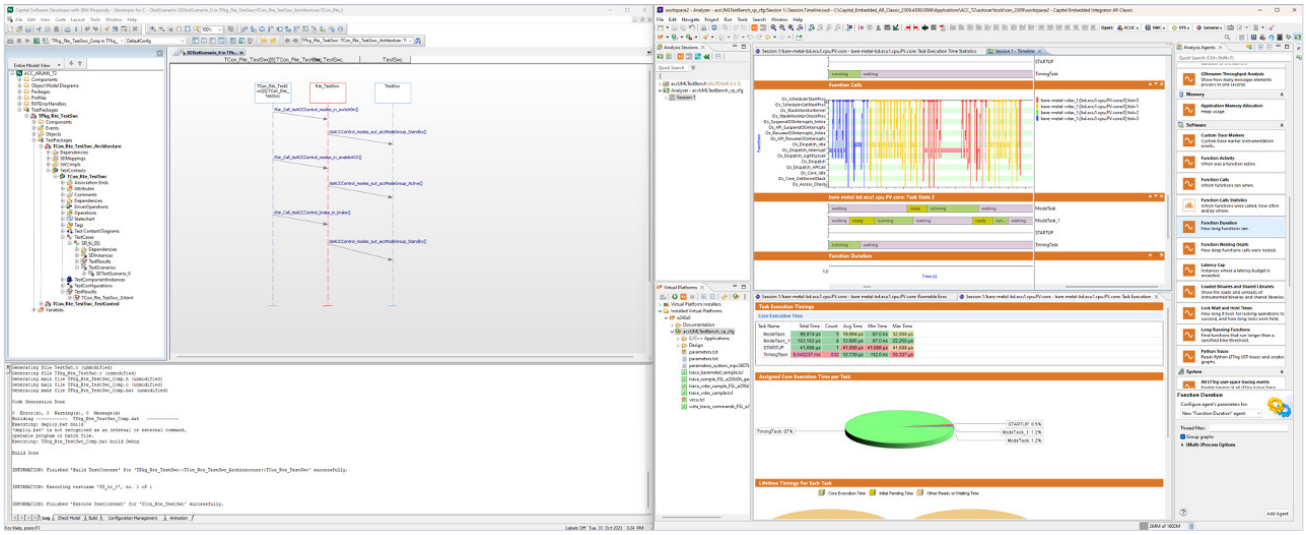

Figure 8. Siemens generative tools reduce program risk, shortening cycles and allowing developers to find problems early when they are easiest to solve and least expensive to fix.

> Together, Siemens and IBM will deliver a simulation-driven systems and software engineering solution that is designed to cover the full operational lifecycle. This can empower our customers to innovate by helping to reduce product development costs, drive continuous improvement and create operational efficiencies across the extended enterprise throughout the product's operation lifecycle."
>
> Tony Hemmelgarn, President and Chief Executive Officer
> Siemens Digital Industries Software

Working across domains to drive innovation

Software-centric vehicles are redefining mobility and aligning automotive innovation with broader sustainability goals, as well as the desire for a more connected, data-driven world. Automakers must transition from traditional mechanical engineering to a focus on software solutions that can adapt and evolve with consumer expectations. Cross-domain integration and collaboration leads to more comprehensive solutions, driving innovation across industries.

As such, incorporating software development as an integrated part of the development flow is imperative. This is a paradigm shift in how the software-defined vehicle is created, requiring teams to regard software as part of the EE development environment and make sure it operates as intended within the hardware architecture (figure 9).

The integration between Capital and IBM Rhapsody is helping organizations optimize product development by improving traceability across processes, and prototype and test concepts for early insights into the development lifecycle. The collaboration supports sustainable product design initiatives, while helping to accelerate time-to-market with high-quality innovative offerings.

Learn more about the collaboration between Capital and IBM, or visit our website to discover how Siemens Digital Industries Software is supporting leading automotive manufacturers.

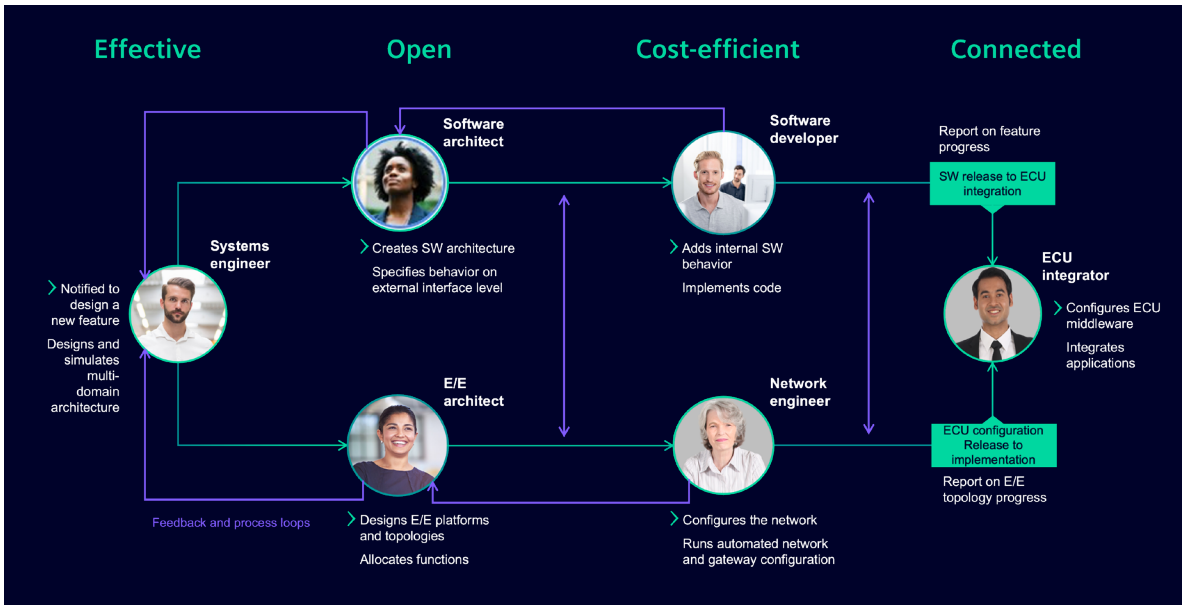

Figure 9. Capital and IBM Rhapsody provide an end-to-end software design flow in E/E context for AUTOSAR and beyond.

Siemens Digital Industries Software

Americas: 1 800 498 5351

EMEA: 00 800 70002222

Asia-Pacific: 001 800 03061910

For additional numbers, click here.

Siemens Digital Industries Software helps organizations of all sizes digitally transform using software, hardware and services from the Siemens Xcelerator business platform. Siemens' software and the comprehensive digital twin enable companies to optimize their design, engineering and manufacturing processes to turn today's ideas into the sustainable products of the future. From chips to entire systems, from product to process, across all industries, Siemens Digital Industries Software – Accelerating transformation.

siemens.com/software

85789-D3 5/24 C