SIEMENS
Ingenuity for life

Siemens Digital Industries Software

Orchestrating automotive embedded application development

Application development and quality assurance

Executive summary

Today's most exciting and innovative automotive features are enabled by immensely complex and sophisticated embedded software applications. Modern vehicles contain hundreds of millions of lines of code that govern dozens of systems. Automotive software engineers are being asked to manage this immense complexity to deliver simple and intuitive user experiences for the end consumer. As engineers develop application code and perform quality assurance tests, they will need a unified platform that provides clear visibility to design requirements, key system attributes and constraints, and even down to individual lines of code.

Piyush Karkare
Siemens Digital Industries Software

Introduction

Large-scale trends in the automotive industry are making embedded software development more fundamental to vehicle development overall. Vehicle automation, connectivity, electrification, and shared mobility (ACES) are driving a need for remarkably sophisticated software to enable features like advanced driver assistance systems (ADAS), battery management, vehicle-to-everything (V2X) communication, and more (figure 1).

To maintain pace with these trends, carmakers and executives are on a path of vertical integration, with the goal of bringing software development, among other key technologies, in-house. Suppliers also face stiff competition to deliver compelling vehicle features that involve substantial software development. As OEMs and suppliers attempt to evolve their core competencies to include software, they encounter significant challenges.

The software being developed for modern vehicles needs to be state-of-the-art, especially for automated safety and driver assistance features. Furthermore, the software must be nearly infallible in reliability and functionality.

Exciting automotive features require very complex software, resulting in a massive increase in the complexity of in-vehicle software overall. Today, vehicle software content commonly exceeds 150 million lines of code. The individual complexity of software applications is compounded by the interactions between vehicle systems that require these software applications to communicate and interoperate. Automotive software engineers are being asked to manage this immense complexity to deliver simple and intuitive user experiences for the end consumer.

Figure 1: The trends of autonomy, connectivity, electrification, and shared mobility are driving a demand for automotive software.

Embedded application development becomes more challenging

Software engineers are already deeply involved in the development process as they start to evaluate changes and systems updates with software architectures and construct software models to prove out the functionality required by the system level definition. Many OEMs and suppliers have adopted a model-driven software development approach for their application development. The architecting and modeling processes are intended to verify and validate the software's functionality before any code is written or modified. The key challenge is orchestrating the activities of all engineers across the data flow passively, without relying on manual effort to maintain data consistency.

Automotive embedded software engineers also must meet particularly strict safety regulations that affect both the finished software-build and the development process. It is important that engineers can audit respective design decisions, test plans, test results, and relevant data created over the development. This digital-thread is needed as evidence for software compliance with required regulations. Given the intricacy and pace of software development, it can be very easy to lose track of this information when trying to meet deadlines or resolve issues.

The development of automotive embedded software is only becoming more challenging as vehicle features rely more heavily on complex software applications. Embedded software teams need a means of managing the software development lifecycle, from planning through development and delivery, which can interact with product lifecycle management (PLM) and other needed engineering solutions to provide a unified development platform (figure 2). During application development and quality assurance (QA), teams should have clear visibility to design requirements, key system attributes and constraints, and even down to individual lines of code. Such a platform will provide full traceability to ensure regulatory compliance, and promote functional reuse and collaboration across the involved engineering domains.

Key challenges

A major challenge during application development is the inconsistency between software and product development processes. OEMs usually use vehicle milestones to track system-level vehicle features, changes, and updates. Meanwhile, software development is accomplished through fast-paced AGILE and hybrid AGILE flows. The discrepancy between these development methodologies can create checkpoint issues that hinder progress. For example, the software teams may have to wait for system-level updates that may not be available until the next milestone. Likewise, software teams may

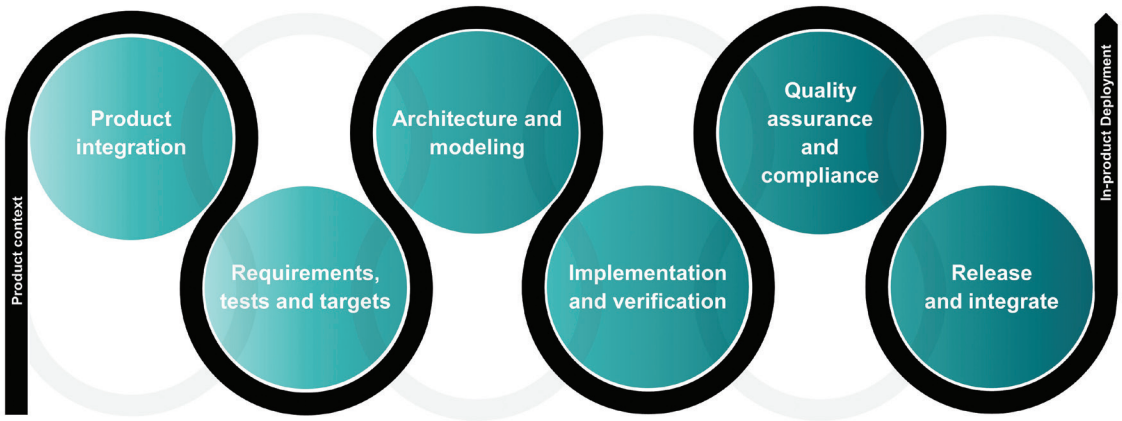

Figure 2: Automotive embedded software engineers need a unified platform to track and manage application development.

be under pressure to produce a build for the next milestone, compromising quality to meet a deadline.

Additionally, conflicting development methodologies complicate the traceability and visibility of information across teams. During testing, especially hardware-in-the-loop (HiL) or driver-in-the-loop (DiL) testing, it is critical that software teams know what they are testing and, more importantly, why it is being tested. This includes knowing what changes have been made to which data artifacts, which software-builds are ready to use, and which hardware abstraction levels are required for the software-builds being verified and validated. In the other direction, the teams responsible for implementation need visibility to testing results so they can make updates and resolve issues uncovered in testing in a timely and efficient manner.

While automated code generation can help teams become more agile by integrating software modelling and coding, a clunky and segmented data-management solution increases risk during integration and coordination, erasing any time savings realized. Engineers need code implementation to be consistent with up-to-the-minute hardware and system-level constraints. This demands an auditable change management process that allows engineers to update application builds in real-time in response to testing results.

An auditable change management process that spans product and software engineering is critically important for ensuring regulatory compliance. First, it needs to ensure that software teams can quickly respond to new or ambiguous regulations that affect software functionality. Second, change management needs to guarantee that the OEM possesses the needed evidence to demonstrate compliance with regulations.

Achieving a flexible implementation and verification process is not only up to the engineers writing code. Software testing and its coverage must be driven by requirements and triggered earlier in the development cycle. Detecting defects early across many implementations is extremely advantageous, as defects only become more expensive the later they are found. Application designs should be modelled, simulated, and executed early on to optimize and validate the concept against design requirements, architectural constraints, and system behaviour. Executing early hardware-software co-simulations in the context of the system will help the software engineers to uncover issues and defects earlier in the development process.

After a round of testing has been completed, implementing the recommended changes can be time-consuming. This is especially true if the implementation team must sort through complicated testing results to determine what they are being asked to fix. Supplying precise documentation and traceability to requirements and system constraints to the relevant teams and stakeholders will facilitate quick and accurate solutions.

Finally, comprehensive traceability, from system-level changes down to individual lines of code, is often undermined by constant changes and updates. Keeping a clear line of sight throughout the development process becomes increasingly complex as software content and dependencies across multiple features grows. This issue is only amplified if engineers must navigate several unintegrated tools to maintain traceability. Solutions that rely on spreadsheets, databases, and home-grown connections are complicated and do not scale with organizations.

Embedded application development and QA with a unified platform

Solutions are available today that help track and manage the complex and concurrent tasks involved in the application development and quality assurance process. These solutions can provide the comprehensive traceability and visibility needed for embedded application development. Let's walk through the development of an embedded application with a development coordination solution in place (figure 3).

Functional design and modeling – leveraging models and tested software components for coding

Software teams create robust software component architectures, software models that represent true functional twins of the needed behavior, and model interactions that ensure the completeness of the needed communication to define and validate the software application elements before any coding is completed. Software architects and engineers usually use dedicated tools like MATLAB or SIMULINK to complete this work. Modern software development coordination solutions feature extensions to these tools, providing a direct connection between the architecture, models, requirements, test methods, loss-analysis, safety compliance and more. As deliverables become mature, the software development solution will track the progress and consistency of models, architecture, and component interactions in relation to the application requirements and constraints. This also ensures coverage of all the requirements, specifications, and behaviors with model representation associated to a robust architecture that accurately depicts all the interactions between components.

A unified platform for application development helps software engineers to ensure consistency across the functional architecture and allows different teams to make informed architectural trade-offs. Such a solution also automatically tracks that models meet application requirements and provides traceability between the models and the relevant requirements. As a result, engineers can more effectively leverage mode-based software development to identify issues early and maintain consistency during downstream processes (figure 4).

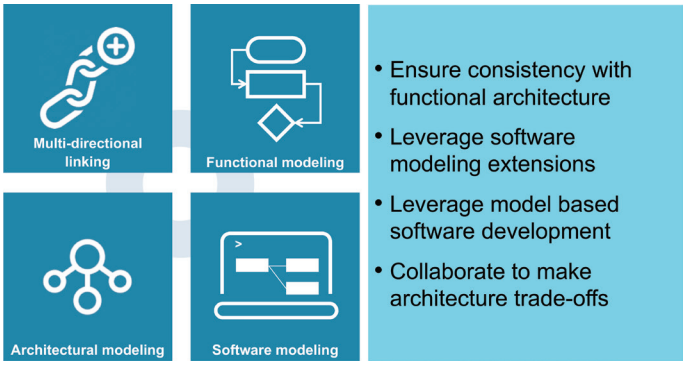

Figure 4: A unified application engineering platform supports architecture and modeling processes by ensuring traceability from requirements to software architectures, models, and more.

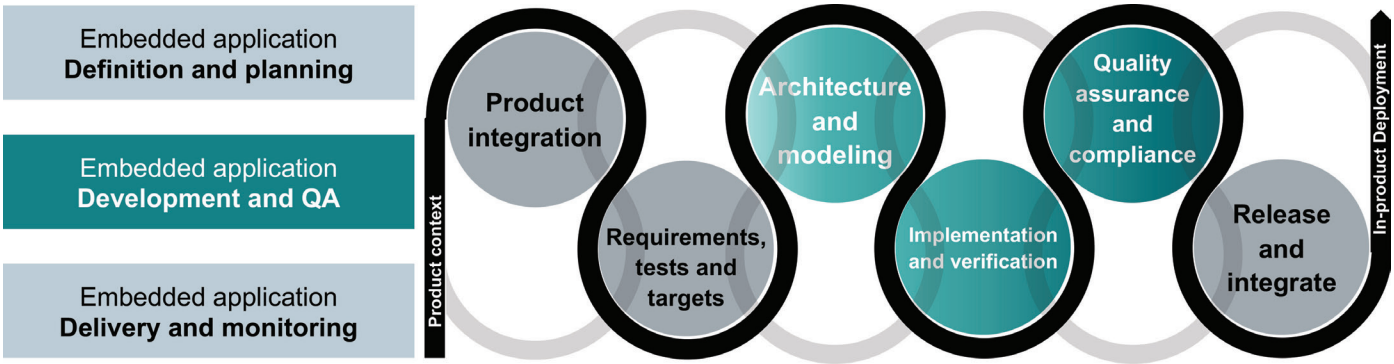

Figure 3: Application development and quality assurance in the overall application development flow.

Software modelling validates that system needs are met before engineers invest considerable time in actual coding. The application development platform can be configured to automatically assign these coding tasks to team members based on customizable conditions that can be determined on a project basis. As engineers complete these tasks, the application development platform can monitor progress with extensions to the software code management (SCM) solution and integrated development environment (IDE) of choice. The application development platform can also track adherence to non-functional requirements via integrations with various code performance (static and dynamic) and code-coverage tools.

Tightly orchestrated and consistent software modeling practices not only speed up the process, but can also instill methods such as SOTIF (Safety of the Intended Functionality) to ensure that the software is working as intended, and hazards are prevented by-design. Incorporating SOTIF methods complements standard functional safety approaches that mitigate risks by employing safety goals that assume faults will occur. This combination produces exceptionally robust automotive embedded software applications.

Code implementation and integration

The application engineering platform enables engineers to integrate modelling and code implementation activities, removing duplicated effort between model creation and coding. A direct connection from requirements, to models, to code, and to testing ensures that implementations can be verified as they mature. Such a connection also guarantees that implementations are always up-to-date with constraints and changes from the system-level definition, hardware specifications, and more. An application development platform also provides auditable change control processes, facilitating flexible code updates in response to model-, software-, hardware-, or vehicle-in-the-loop testing (figure 5).

Iterative planning tools help embedded application development teams efficiently budget their time and track when tasks are completed to keep development moving on schedule. The platform can be configured to require electronic signatures to perform certain actions or process steps, ensuring that key stakeholders review and approve decisions. The embedded application development platform can also track accountability for change analysis and implementation, allowing teams to check that a software release or build is well supported for planned updates with full traceability to all modified data-artifacts and the rationale behind each modification.

The planning solutions also feature team velocity management to provide estimations for task completion based on empirical data. Multiple estimation methodologies are supported, including time, money, and story points. Team leaders or application owners can define the conditions for a completed workflow. For example, a workflow will only be marked complete if documentation is submitted. Users can also attach supportive documentation, such as component and design requirement specifications, to ensure functional and quality consistency.

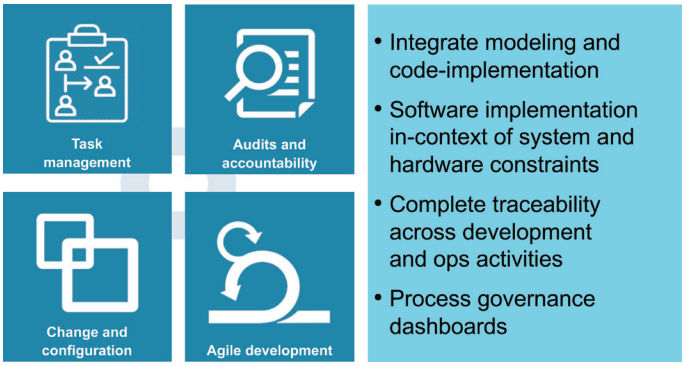

Figure 5: A unified platform for application development coordination enhances task management, accountability, and change management while supporting Agile development methods.

Quality assurance

Quality assurance is a predominantly continuous process that starts early on and is performed as requirements, architectures, models, and code implementations are developed and matured. Quality assurance is the sum of testing at multiple abstraction levels such as model-in-the-loop (MiL), software-in-the-loop (SiL), hardware-in-the-loop (HiL), and vehicle-in-the-loop (ViL), which includes both virtual HiL and physical testing at the proving ground. This iterative testing flow can be orchestrated to assess software quality at each stage of development, and as design changes are implemented. This type of testing regime also promotes model-based software design flows by enabling engineers to manipulate and verify software models early in the development process as needed.

However, as engineers develop and test software architectures, models, and actual code, they are constantly creating new data that must be tracked and associated to the appropriate software build or configuration. For each round of testing, software engineers must create or update test plans that detail the test case, test-execution strategy, test vectors, and other related data artifacts. As tests are completed, issues must be assigned

and resolved while test results are linked back to the relevant test runs. These data associations are crucially important to demonstrate test coverage and support complete traceability from requirements to implementation.

Organizing and tracking the testing value-chain (test plans, test cases, test results, related data-artifacts, and the status of identified issues and risks) escalates the challenge and importance of overall software development orchestration. There is a lot of data with intricate interrelationships that needs to be tracked and managed. Burdening the engineers with data tracking and managing prevents them from focusing on software engineering and development. Software teams must also coordinate the delivery of applications, along with all associated data, to final builds. Key capabilities in today's application development and coordination platforms can help manage this complexity to ensure the quality of delivered application binary and associated data (figure 6).

Application development and coordination platforms, such as Polarion, feature integrated test management tools. Using these tools, engineers can create and link test cases to the corresponding work item. Work items may include application requirements, change requests, other test cases, and more. The test management tools enable the steps of each test case to be modified, separating the test specification and test-item configuration to enable greater test coverage from a single procedure. These solutions can also integrate with external test-automation tools to automatically run tests and import results.

This workflow makes process compliance seamless and painless. Company processes and best practice information are available directly in the project view to ensure that all team members comply with established processes, even without complete knowledge of the process details. Polarion's workflow also integrates requirements, tasks, change requests, process management, project planning, time management, build management, source code audits and metrics, and maturity-model compliance tracking to ensure that application development is on time, and of high quality.

Advanced application development and coordination platforms also feature defect and risk management capabilities. These capabilities can automatically create issue and risk reports from test failures, and assign tasks to software engineers to reduce the time to solution implementation. The embedded application development platform automatically tracks issues, recording all subsequent related activities, including implemented fixes, and additional tests. As additional tests are performed, it is critical that hardware and software are co-simulated to improve overall quality and facilitate software delivery to vehicle programs. A unified platform for embedded application development orchestration can ensure that each software build is tested in all of the relevant hardware configurations using both emulated and physical hardware. All test information will be tracked whether the test was manually or automatically executed, or performed in a third party tool or within the application development platform environment. This will ensure that a complete report of all issues, active and resolved, is maintained throughout development.

With these capabilities, automotive embedded software engineers will realize several key benefits. They can prototype, simulate, and execute designs to validate that they meet requirements, optimize architectures, and verify and validate correct behavior. The engineers can also perform hardware-software co-simulations to evaluate the designs in the context of the system hardware and peripherals. This enables the engineers to uncover issues or risks, either in the software or the hardware mapping, that arise in a system context. In the background, the application development platform solution will track requirements, models, architectures, test plans, test results and more to ensure traceability and provide software engineers with the right information at the right time. This will streamline issue resolution by making it straightforward to supply precise and complete documentation to the concerned party. This comprehensive level of traceability also helps to prevent critical bugs from being released into the field.

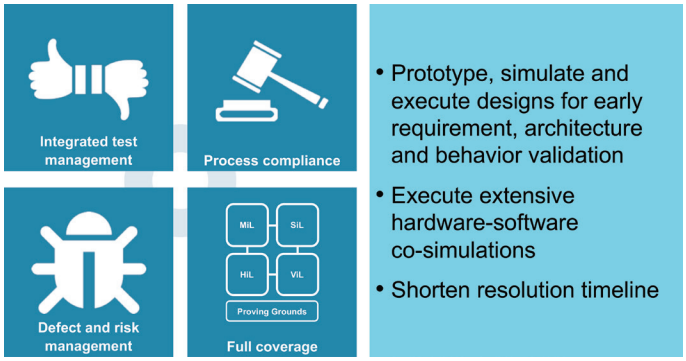

Figure 6: Test management, process compliance, and risk management tools help ensure full test coverage.

Conclusion

Automotive embedded software application development and QA is an intricate and involved process, as it must constantly consider implementation hardware (including peripherals like sensors and actuators), and system considerations and constraints (including vehicle integration and geometrical dependencies). A unified software engineering platform provides the engineering environment that ensures data consistency despite constant changes, keeping all the parties in the development process continuously involved in delivering quality software applications that are compatible with the full range of vehicle variability. Such a platform achieves this data coherency through robust integrations with the various tools used in application development, and powerful change management capabilities.

These capabilities are critical to the delivery of increasingly complex software features and functions needed to compete in the digitalizing automotive market. As ADAS, infotainment, and a widening array of electrical, electronic, and software-based features continue to drive value for the consumer, the average lines of code in vehicles will climb. Today, it is common for a luxury vehicle to surpass one-hundred and fifty million lines of code.

In the future, the development of automated driving, vehicle connectivity, electric powertrains, and shared mobility (ACES) will drive this complexity even higher, and bring software to the top of the automotive value chain. Self-driving vehicles are predicted to require billions of lines of code to operate. This level of complexity will make the coordination of embedded application development even more critical, as the creation of innovative and powerful automotive software becomes the key differentiator for automotive and mobility companies.

To read more about how Siemens solutions can help you, please visit siemens.com/aes where you will find additional content such as blogs, whitepapers, podcasts, product videos, webinars, solution capabilities and infographics.

Siemens Digital Industries Software

Headquarters
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 972 987 3000

Americas
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 314 264 8499

Europe
Stephenson House
Sir William Siemens Square
Frimley, Camberley
Surrey, GU16 8QD
+44 (0) 1276 413200

Asia-Pacific
Unit 901-902, 9/F
Tower B, Manulife Financial Centre
223-231 Wai Yip Street, Kwun Tong
Kowloon, Hong Kong
+852 2230 3333

About Siemens Digital Industries Software

Siemens Digital Industries Software is driving transformation to enable a digital enterprise where engineering, manufacturing and electronics design meet tomorrow. Our solutions help companies of all sizes create and leverage digital twins that provide organizations with new insights, opportunities and levels of automation to drive innovation. For more information on Siemens Digital Industries Software products and services, visit siemens.com/software or follow us on LinkedIn, Twitter, Facebook and Instagram. Siemens Digital Industries Software – Where today meets tomorrow.

siemens.com/software

81256-C1 12/19 C