ANSYS

White Paper

Qualified Code Generation Greatly Reduces Cost of Safety-Critical Automotive Software

Advanced driver assistance systems (ADAS) are increasingly being used in today's automobiles to alert drivers to potential problems or even, as a last resort, to take over control of the vehicle to avoid a collision. Until recently, ADAS were primarily found in luxury vehicles, but today they are rapidly becoming mainstream due to technological advances that reduce their cost and increased interest from consumers and regulators. Manufacturers are also in the process of developing autonomous or self-driving vehicles that operate without direct driver input to control the steering, acceleration and braking and are designed so that the driver is not expected to constantly monitor the roadway while operating in self-driving mode.

While these systems offer the potential to dramatically reduce automobile-related deaths, injuries and collisions, any malfunction of an ADAS or autonomous driving system creates the potential for death or serious injury. This raises the stakes in designing embedded software for these safety-related systems to a level far above traditional automotive embedded software. In fact, testing and validation of embedded software for ADAS, autonomous driving and other safety related systems is arguably one of the greatest challenges faced by automobile original equipment manufacturers (OEMs). Before deploying a commercial ADAS or other safety- related system, the development team must test thousands of different scenarios to ensure that the system will function according to the design intent under all possible conditions, including traffic conditions, weather, obstacles, pedestrians, etc.

Increasing role of safety in product development process

Furthermore, the number of safety-critical systems in the vehicle is growing because the higher levels of integration in today's automobiles means that more and more electronic control units (ECUs) control or interact with critical systems such as brakes and steering. These trends combined with the increasing functionality, volume and complexity of safety-related software has resulted in a staggering level of complexity that most automobile OEMs have concluded is nearly impossible to negotiate using conventional manual software development methods.

In the traditional approach, specifications are conceptualized by systems architects, captured as text documents and passed to project teams that specialize in areas such as algorithm development and analog electronics. These teams write software manually in the form of source code such as C code, and perform tests to verify that the code matches the specifications. Verifying the code requires assembling the hardware, including the target computing device and the electromechanical components that are being controlled. A key problem with this approach is that errors often remain undetected until

all the modules can be tested together at the prototype stage. Because the cost of fixing a problem generally increases by an order of magnitude as the design progresses, late-stage problems quickly drive up development costs. As the size and complexity of automotive safety-critical systems continues to grow, the cost, time and risk involved in manually producing, testing and verifying automotive embedded software have increased exponentially.

Moving to a model-based design process

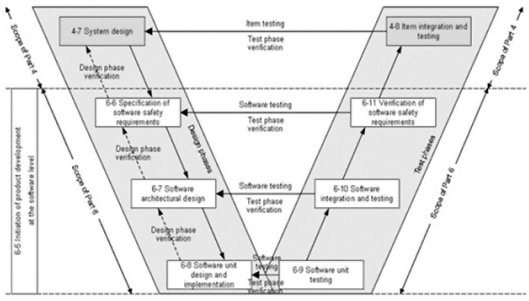
Many automotive OEMs have addressed these difficulties by changing to a model-based design process in which a graphical model of the embedded software and systems becomes the cornerstone of the development process. Engineers can simulate the behavior of the model on a PC workstation and immediately view the results, making it possible to gain critical insights early in the systems design process and to rapidly improve the model's performance. Engineers also can link the model's predicted behavior to specific customer requirements. Finally, the model can be used to automatically generate the embedded code, thus eliminating the need for manual coding.

Compliance with safety-related standards enables automobile OEMs and their suppliers to demonstrate that they use consistent, auditable processes for designing safety-related systems. ISO 26262, the safety standard that applies to embedded systems and software for automotive applications, is now the state-of-the-art for developing automotive active and passive safety systems including ADAS, autonomous driving, vehicle dynamics, braking, steering, powertrain controls, etc. Current processes for development of embedded software for automotive safety-related applications are being updated to achieve ISO 26262 compliance.

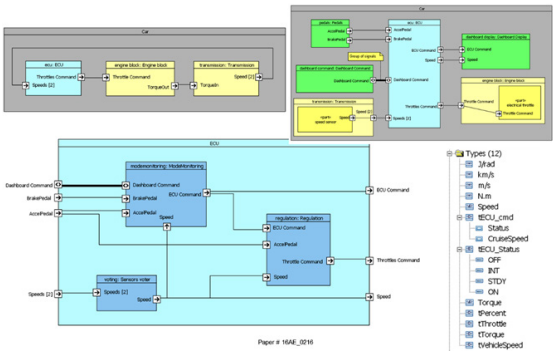
ISO 26262 includes the definition of an automotive safety lifecycle, identifying the risk of elements based on Automotive Safety Integrity Levels (ASILs) from A (lowest) to D (highest). Situation analysis and hazard identification are performed in order to identify the potential unintended behaviors of the item that could lead to a hazardous event. Part 6 of ISO 26262 describes the requirements for product development at the software level including requirements, architectural design, unit design and implementation, unit and integration testing.

Need for an ISO 26262 qualified code generator

A limiting factor in the use of existing model-based design environments in the development of safety-related automotive embedded code is that the code generator used in today's leading model-based design environment has not been developed in compliance with any safety standard such as ISO 26262, which means that companies following an ISO 26262-compliant process are not allowed to trust its output. The code requires an extensive manual verification and validation process which negates one of the core advantages of the model-based design concept. Typically, each model-to-code translation triggers back-to-back review and testing phases to verify that the executable compiled from the generated code matches the model used in the design process. Some parts of this testing process, such as demonstrating



The system and embedded software lifecycle V-cycle defined in ISO 26262



Software arhitectural design with SCADE System

numerical equivalence, can be automated. But other required elements of the testing process such as demonstrating traceability of software safety requirements, software development artifacts and test cases are much more difficult to automate.

Emergence of first qualified code generator

The recent qualification of the ANSYS SCADE Suite code generator for developing ISO 26262 compliant applications up to ASIL D, the highest safety requirement for automotive applications, substantially reduces the embedded software development time and cost. The code generator is offered as part of a complete end-to-end, model-based systems engineering (MBSE) solution for the development of safety-related systems. It has been proven through its use in the development of many safety-critical aeronautics systems by Airbus, Boeing, Pratt & Whitney, General Electric and many others.

The SCADE Suite KCG code generator was qualified at TCL3 by TÜV SÜD to be used to develop ASIL D automotive software. To achieve its qualification, KCG was developed according to the IEC 61508 standard. As part of this process, hazard analysis and risk assessment were performed on the KCG tool using HAZOP, a method recommended by ISO 26262. This analysis produced deviations, potential causes and mitigation actions from all tool stakeholders including the tool developer, tool installer and tool users. Dozens of failure conditions were identified and dozens of individual mitigation actions were allocated to the tool developer, tool installer and tool user. The developer actions were achieved by verification activities performed during the development cycle of the code generator, such as design and code reviews and structural coverage analysis. As a result, costly code reviews and low-level testing activities are not required to verify that the code is a correct implementation of the model. The SCADE Suite code generator has already been used to produce embedded software without the need for manual verification for many safety-critical projects, including many commercial airliners flying today produced by Airbus, Boeing and Embraer and other leading airframe manufacturers.

The SCADE KCG code generator is offered as part of a complete toolset that enables efficient implementation of the lifecycle V-model defined in ISO 26262 Part 6. The SCADE System tool is used to describe the system and software architecture, using SysML block diagrams to represent software architecture components and connect them through ports and connectors. The next step is to produce a software architectural design that implements the requirements using software blocks. Then software engineers model the software components associated with the software architecture, often using state machines and data flows to model the logic and control laws. The SCADE LifeCycle tool is used during this phase to establish traceability links from the initial system requirements to the design models, generated code and test scenarios.

The model is simulated to detect functional faults early in the design process. Test cases can be run and validated in the host environment long before they are run in the more expensive and complicated target hardware environment. Tools are provided to automate the creation and management of test cases and to analyze test coverage to ensure that the design fully complies with its software requirements. SCADE Test Environment for Host has been qualified as a verification tool for DO-178C/DO-330, so the results prove the compliance of a SCADE model with its high level requirements (HLRs). The same functional test execution is provided on host and targets independent of code generation.

Software arhitectural design with SCADE System

Suburu uses qualilfied code generator to develop HEV control software

Subaru®, the automotive brand of Fuji Heavy Industries (FHI) Ltd. — a comprehensive, multifaceted transport equipment manufacturer — recently started its own HEV and electric vehicle (EV) programs and adopted the model-based design approach for upcoming development projects, like the Subaru XV. Engineers used the SCADE System to develop the software architecture from the ground up. The software development process began with the development of test scenarios based on software requirements and the conversion of many pieces of functional Simulink® models designed by system engineers into SCADE models via a gateway. Functional models were verified by unit testing. The final stage of the process was to generate C source code from the verified models using the ISO 26262 qualified SCADE Suite KCG code generator. The verification time at code level was substantially reduced since most of the verification was completed upfront at the model level.

Conclusion

As automobile electronics become more prevalent and sophisticated, ensuring the reliability of the embedded software code within those systems becomes critical to the safety of passengers and pedestrians. The development of an ISO 26262 qualified code generator helps automotive OEMs and suppliers drastically reduce development costs while ensuring that their embedded software applications will meet stringent safety standards. This code generator is part of a model-based design solution in which modeling and simulation are used throughout the product development lifecycle as the authoritative definition and verification of the product design.

ANSYS, Inc.
Southpointe
2600 ANSYS Drive
Canonsburg, PA 15317
U.S.A.
724.746.3304
ansysinfo@ansys.com