AdaCore | Build Software that Matters

Tech Paper

Investing in SPARK:

Formal methods for automotive functional safety

adacore.com

Executive Summary

Software-related issues are becoming an increasingly common cause of safety recalls in the automotive industry, with their share rising significantly in recent years. In this context, compliance with ISO 26262, the international standard for functional safety in automotive systems, is no longer merely a technical requirement but a critical business priority.

To meet the industry's growing demand for high-assurance software, some developers are turning to formal verification to improve system reliability and security significantly. To achieve the highest level of ISO 26262 assurance for DriveOS as efficiently as possible, NVIDIA adopted Ada and SPARK and recently published its reference process to support this approach.

Introduction

Software issues are increasingly causing safety recalls in the automotive industry, with the proportion of software-related recalls rising significantly in recent years. This trend is driven by the growing complexity of vehicle software, particularly with the rise of advanced driver-assistance systems (ADAS) and electric vehicles.

An article in Ars Technica states* that software fixes are now responsible for more than 1 in 5 automotive recalls and that software-related recalls often involve electrical systems, ADAS features like automatic emergency braking, and powertrain components.

Software has become the primary enabler of innovation, functionality, and safety. However, as vehicle software grows in scale and complexity, so do the challenges, particularly in ensuring safety in life-critical scenarios.

ISO 26262 is an international standard for functional safety in the automotive industry. The standard applies to electrical and electronic systems comprising vehicle hardware and software components. It defines requirements to be met by the system's safety-relevant function and processes, methods, and tools used in the development process.

Why does ISO 26262 matter?

Adherence to ISO 26262 is not just a technical obligation but a business imperative. With vehicles increasingly reliant on complex software systems, ensuring the functional safety of E/E (Electrical and Electronic) components has become essential to protecting human life and preventing accidents caused by system failures.

Ensuring the functional safety of E/E (Electrical and Electronic) components has become essential to protecting human life and preventing accidents caused by system failures.

Compliance demonstrates due diligence and accountability, which are crucial in an industry subject to intense public scrutiny and stringent liability regulations. Failure to conform to ISO 26262 can result in serious consequences, including vehicle recalls, litigation, regulatory penalties, and long-term reputational damage.

Moreover, aligning development practices with ISO 26262 can act as a catalyst for improved engineering discipline. It encourages a structured approach to system design, promotes traceability and documentation, and helps organisations maintain consistency across geographically distributed teams and suppliers. In addition, for original equipment manufacturers (OEMs) and suppliers alike, ISO 26262 compliance can be a key differentiator in competitive tenders and partnerships, signalling a commitment to safety and quality.

What does ISO 26262 cover?

The standard covers the entire lifecycle of automotive systems, from concept and design to production, operation, and decommissioning. Central to ISO 26262 are its Automotive Safety Integrity Levels (ASIL), which categorize risk into four levels—A through D—based on the severity, exposure, and controllability of potential hazards. These ASIL levels guide the safety measures required for different components, with higher levels demanding more stringent verification and validation processes.

There are four ASILs identified by ISO 26262: A, B, C, and D.

The determination of ASIL is the result of hazard analysis and risk assessment. In the context of ISO 26262, a hazard is assessed based on the relative impact of hazardous effects related to a system, as adjusted for the likelihood of the hazard manifesting those effects. Each hazard is evaluated in terms of the severity of possible injuries within the context of how often a vehicle is exposed to the possibility of the hazard happening, as well as the relative likelihood that a typical driver can act to prevent the injury.

The ASIL ranges from ASIL D, representing the highest degree of automotive hazard and the highest degree of rigor applied in ensuring the resultant safety requirements, to QM, representing an application with no automotive hazards and, therefore, no safety requirements to manage under the ISO 26262 safety processes. The intervening levels are a range of intermediate degrees of hazard and degrees of assurance required.

ASIL D, an abbreviation of Automotive Safety Integrity Level D, refers to the highest classification of initial hazard (injury risk) defined within ISO 26262 and to that standard's most stringent safety measures to apply for avoiding an unreasonable residual risk. In particular, ASIL D represents a likely potential for severely life-threatening or fatal injury in the event of a malfunction and requires the highest level of assurance that the dependent safety goals are sufficient and have been achieved. An example of a dangerous hazard that warrants the ASIL D level is loss of braking on all wheels. Any product compliant with ASIL D requirements would also comply with any lower level. ASIL C is associated with moderate to high risk, ASIL B is for moderate risk, such as headlights and brake lights, ASIL A is associated with low risk, such as rear lights, and QM relates to non-safety components, such as GPS.

Achieving compliance with ISO 26262 is not merely a regulatory checkbox but a strategic investment in product quality, customer trust, and organisational reputation. By understanding the scope of the standard, rigorously applying verification techniques, incorporating formal methods, qualifying tools, and aligning safety goals with risk levels through ASILs, automotive developers can systematically reduce hazards and enhance the dependability of their systems.

How can formal methods help with ISO 26262 certification?

To support the automotive sector's evolving needs for high-assurance software, automotive developers can significantly enhance the reliability and security of their systems by considering formal verification in their processes. Formal methods are particularly valuable in adherence to ASIL levels D and C.

Formal verification frameworks are instrumental in assessing the correctness of hardware and software design operations by applying formal mathematical proofs. Unlike traditional methods, which focus on testing, reviews, static analysis, and in-depth coding standards such as MISRA-C, formal verification seeks to provide mathematical assurances regarding the adherence of a system to specified safety and/or security requirements. Whereas testing

depends on extrapolating from a limited sample of inputs and inferring that the software meets its required properties, formal methods can demonstrate, with mathematical rigor, that these properties hold for all possible inputs. Formal methods can help verify properties such as secure information flows, absence of run-time errors, and functional correctness concerning formally specified requirements. With advances in proof technology and hardware support, formal methods are a practical component of an organization's software production infrastructure for high-assurance systems.

NVIDIA introduces Formal Methods

The demand for ever more capable ADAS and autonomy in cars has led to a proliferation of ECUs (Electronic Control Units) and an associated increase in software complexity, communication, and wiring. NVIDIA addresses this challenge with DRIVE® AGX, which combines the functionality of multiple ECUs into a full-stack hardware and software

To reach the highest level of assurance under ISO 26262 for DriveOS as quickly and efficiently as possible, NVIDIA selected Ada and SPARK.

platform supporting ADAS and autonomous driving. NVIDIA DriveOS is responsible for ensuring the isolation of integrated software components of differing criticality levels, enabling this reduction in the number of ECUs and simplification of communication and wiring. To reach the highest level of assurance under ISO 26262 for DriveOS as quickly and efficiently as possible, NVIDIA selected Ada and SPARK.

About Ada and SPARK

The Ada programming language, together with SPARK (deductive formal verification for the Ada language), is among the most reliable programming languages available for developing high-integrity applications.

The SPARK language and SPARK analyses provided by SPARK Pro work together to automatically verify correctness of software properties at the source code level, such as component requirements and absence of weaknesses, before you even build and test your software.

SPARK Pro brings the power of deductive program verification through formal proof to your development team, allowing you to scale smoothly from memory ownership checking and dataflow analysis through absence of runtime errors to proofs of functional correctness. This is the only language, tool, or methodology that can compete at this scale.

The result is software with an exceptionally low defect density, leading to significant cost savings post-deployment.

SPARK Ada reference process for ISO-26262 development

In collaboration with AdaCore, NVIDIA has published an off-the-shelf reference process that enables the automotive community to replicate its success in using Ada and SPARK to develop software certified under ISO 26262.

NVIDIA selected these languages to develop some of the most critical components of its software stack. This required establishing a development process that takes advantage of formal methods and other safety characteristics of Ada and SPARK, thus fully leveraging their capabilities.

AdaCore and NVIDIA have decided to publish this reference process freely as an open-source and evolving document, allowing the industry to adopt Ada and SPARK. And you can find the documentation via these links:

https://github.com/NVIDIA/spark-process
https://nvidia.github.io/spark-process/

Leveraging the Ada and SPARK beyond Safety - the NVIDIA Firmware Security Caset

In an increasingly hostile cybersecurity environment, NVIDIA examined all aspects of its software development methodology, asking themselves which parts needed to evolve. They began questioning the cost of using the traditional languages and toolsets they had in place for their critical embedded applications.

NVIDIA's software security team examined various methods and strategies that offered measurability. They soon recognized that the bases of many of these methods were mathematical formal methods and the use of formal provers. They also discovered that these tools had undergone a major evolution over the past decade or so.

They asked themselves, "What alternative languages and tools are available to support the use of formal methods?" In trying to answer this, NVIDIA discovered SPARK.

"You tend to have more confidence when coding in SPARK because the language itself guards against the common, easily made mistakes people make when writing in C.

James Xu, Senior Manager
GPU Software Security, NVIDIA

James Xu is the senior manager of GPU software security at NVIDIA.

"The main reason why we use SPARK is for the guarantees it provides," said Xu. "One of the key values we wanted to get out of this language was the absence of runtime errors. It's very attractive to know your code avoids most of the common pitfalls. You tend to have more confidence when coding in SPARK because the language itself guards against the common, easily made mistakes people make when writing in C". "It's very nice to know that once you're done writing an app in SPARK—even without doing a lot of testing or line-by-line review—things like memory errors, off-by-one errors, type mismatches, overflows, underflows and stuff like that simply aren't there," Xu said. "It's also very nice to see that when we list our tables of common errors, like those in MITRE's CWE list, large swaths of them are just crossed out. They're not possible to make using this language."

NVIDIA has demonstrated groundbreaking leadership and innovation in the software security domain. They had a very challenging goal and took an ambitious path to accomplish something that had never been done before in the semiconductor industry. For several years now, they have successfully demonstrated time and time again that their choice to adopt SPARK was the right one, and they have paved the way for anyone interested in following NVIDIA's lead.

Conclusion

SPARK offers a practical, scalable pathway to verifiable software assurance. By eliminating classes of vulnerabilities at the language level and providing mathematical proof of correctness, SPARK allows organisations to move beyond reactive testing and towards measurable trust in their systems.

NVIDIA's adoption of SPARK and decision to openly share the associated ISO 26262 reference process marks a significant step forward for the industry. It demonstrates that formal methods are not only viable but invaluable in achieving the high levels of safety demanded by today's automotive systems.

By investing in SPARK, organizations position themselves to meet regulatory expectations, reduce development risk, and deliver robust, future-ready software, not through guesswork or tradition, but through proof.

References

https://arstechnica.com/cars/2024/09/more-than-20-of-vehicle-recalls-are-software-fixes-now/#:~:text=In%202022%2C%20almost%2022%20percent,fully%20charging%20to%20100%20percent.

AdaCore | Build Software that Matters

adacore.com