ADVANCED DRIVER ASSISTANCE SYSTEMS (ADAS), SAFETY, AND STATIC ANALYSIS

TRUSTED LEADERS OF SOFTWARE ASSURANCE AND ADVANCED CYBER-SECURITY SOLUTIONS
Advanced driver assistance systems (ADAS) are a key area of innovation in automotive electronics, but their potential improvement and positive impact on safety can only be realized if they undergo the same level of rigor as other safety-critical software. ISO 26262 defines the guidelines for designing and building safety functions in automobiles, which includes ADAS systems. Static analysis plays an important role in developing software to the standard. This paper discusses the role of static analysis tools within the development of an ADAS system, including the return on investment for adopting them.

CERTIFICATION AND STATIC ANALYSIS TOOLS

ISO 26262 specifies software unit design, implementation principles, and coding guidelines. Static analysis tools are particularly useful in enforcing coding standards such as MISRA C, which is just a small fraction of the capabilities of a product such as GrammaTech’s CodeSonar. CodeSonar also offers ADAS devices useful tools to achieve the high level of robustness, correctness, and consistency that requires design, coding, and testing rigor beyond the coding standards. Static analysis tools can find defects in source code before and after they are part of a project. Static analysis can also detect bugs that are hard to find in testing, such as concurrency bugs, and are expensive to debug and fix.
TOOL QUALIFICATION

Software certifications require proof of implementation to the standard, which is often manually generated, but automation greatly reduces the workload. In order for an automated tool’s results to be acceptable certification evidence, tools vendors should be certified for the products they sell. GrammaTech CodeSonar is independently certified for use in development of safety-critical software up to the highest safety integrity levels for ISO 26262, IEC 61508, and EN 50128. This means that developers can use the tools with confidence that the results produced are acceptable to approval bodies during certification. Using unqualified tools is risky and can result in further testing, documentation, and certification costs.

STATIC ANALYSIS AS PART OF ISO 26262 SYSTEM DEVELOPMENT

CodeSonar works on both source and binary versions of software, and as such is heavily used during development and testing stages, although development teams have learned to extend the value of these tools by leveraging the benefits through the lifecycle. For example, any new patch, security update, or product release should be analyzed completely before release, to ensure quality and security. The following diagram illustrates the role of static analysis in the ISO 26262 system development model (based on the well known “V”):

Figure 2: The role of static analysis in the processes defined in ISO 26262, which most ADAS systems are likely to follow.
THE ROLE OF STATIC ANALYSIS IN THE PROCESSES DEFINED IN ISO 26262

CONCEPT PHASE

3-7 Hazard Analysis & Risk Management

This stage is very heavy into risk and hazard analysis. CodeSonar can analyze binary code, so it can help evaluate existing, legacy, and third-party software for security and quality risks. Most projects aren’t greenfield, so reviewing your codebase starting point is critical for establishing a quality and security baseline.

SOFTWARE LEVEL DEVELOPMENT

6-8 Unit Design & Implementation
6-9 Unit Testing
6-10 Software Integration & Testing

The full-scale software development phase is the obvious sweet spot for static analysis. Analyzing code for security, safety, and quality on an ongoing basis is important at this stage. Integrating static analysis into every day developer workflow and their development operations provides the biggest payoff for static analysis tools. Preventing security vulnerabilities and safety defects before they become part of the system or subsystem is the goal.

SYSTEM LEVEL DEVELOPMENT

4-8 Item Integration & Testing
4-9 Safety Validation

This stage is usually testing the entire system or substantial subsystems in their physical environment. At this stage, static analysis remains critical for ensuring the code quality of fixes and patches that are required. Critical issues need to be fixed, and the software development process continues as the quality, security, and safety of the product nears acceptable standards.

PRODUCTION AND OPERATION

7-6 Operation, Service, Maintenance

Static analysis tools remain important throughout a product’s lifecycle – software development of new releases, fixes, and security patches require the same amount analysis and testing rigor as the main product. Subtle changes in production can introduce unforeseen errors that static analysis tools can often detect even when traditional testing has qualified the product.

SUPPORTING PROCESSES

8-9 Verification
8-11 Confidence in Use of Software Tools
8-12 Qualification for Software Components

Static analysis tools – and in particular, certified and qualified tools such as CodeSonar – are integrated into the documented development process. Because it’s a certified ISO 26262 development tool, CodeSonar can be used on automotive safety-critical projects without the need for additional qualification. Reports from CodeSonar are used as certification evidence as required by the standard, recognized by the certification authorities.
ADAS DEVELOPMENT ACCELERATION

The additional rigor required for ADAS systems, including risk management and functional safety defined in ISO 26262, is relatively new to automotive software development teams. Static analysis tools play an important role in the development process of all safety-critical systems, providing tangible productivity improvements to software teams seeking stringent software safety certification.

Specifically, using a qualified tool as part of the software development process from early stages of development can have significant benefits, such as those listed below:

- **Enforcing coding standards for safety, security, and style.**
  
  Automating code analysis during code development ensures quality in the development stream every day, including being used to enforce ISO 26262, MISRA C, and other safety-critical coding standards.

- **Reducing manual effort in proving software robustness and behavior.**
  
  Static analysis tools augment testing by providing more assurance of software quality.

- **Reducing number of defects throughout development.**
  
  Code that works the first time is much cheaper to test and integrate than buggy code. Bugs removed from the code before testing (or even source configuration management) reduces costs and risk.

- **Finding serious defects that elude testing.**
  
  Software testing in ADAS is exhaustive and, depending on the level of concern (ASIL), require complete statement and or decision coverage. Despite this testing rigor, static analysis tools have found defects that were missed. These are the most worrisome types of defects – is it really worth the risk of letting these bugs go into a shipping product?

- **Analyzing legacy and third-party code.**
  
  Use of third-party code such as commercial off-the-shelf software (COTS) and open-source software is a fact of life in embedded software development. CodeSonar can analyze third-party source and binaries to discover defects and security vulnerabilities in software that could be impossible to test otherwise (without including it and running it, an expensive option).

- **Accelerating certification evidence.**
  
  Documenting the results of software unit acceptance is critical to proving compliance to certification standards. Static analysis tools have rich reporting features to help support certification requirements.
THE RETURN ON INVESTMENT (ROI) FOR STATIC ANALYSIS

So what is the return on investment given these factors? Static analysis decreases the volume of defects in software under development at all stages of development. A simple analysis that only factors in defect reduction during software development, finds approximately $126 of savings per defect (using data from Capers Jones (2012)). Given an average of 15 defects per 1000 line of code (during development when defect volumes are high), yields a savings of $1,900 per KLOC. Multiply this by the number of 1000 lines of code in your project and the simple ROI is obvious.

Results will vary based on other factors such as labor rates, defect detection, repair time, and defect density. Static analysis tools provide a quick return on investment even when used in the most obvious stages of development; however, looking beyond coding into other stages of development, there’s a richer reward.

STATIC ANALYSIS IS MORE THAN JUST DEFECT REDUCTION

In addition to “simple” defect-detection, GrammaTech’s CodeSonar is used in many more valuable ways, for instance to analyze potentially hazardous dataflows (tainted data), detect complex concurrency issues, analyze third-party source and binaries, and detect other errors that traditional testing misses. These critical benefits are not factored into the simple analysis above, but clearly add to the tool’s ROI. Finding defects that “slip through the cracks” provide the additional risk and cost reduction that competitive ADAS projects need.

CONCLUSION

ADAS is growing in range and scope in automotive systems. In these systems, the required software development is taking over in terms of costs and risk. Standards such as ISO 26262 require thorough risk management. ADAS software is complex and expensive to build, but static analysis tools can boost the security, quality, robustness, and correctness needed in these systems while providing excellent return on investment. Early adoption and use, plus extending the usefulness throughout the software development lifecycle, is key to reaping the most rewards.
REFERENCES & FURTHER READING


