GRAMMATECH

STATIC ANALYSIS, RAILWAY SAFETY-CRITICAL SOFTWARE, AND EN 50128

TRUSTED LEADERS OF SOFTWARE ASSURANCE AND ADVANCED CYBER-SECURITY SOLUTIONS

WWW.GRAMMATECH.COM

Transportation systems (railway systems in particular) are a growing market that increasingly relies on software for command, communication, and control. Due to the impact of errors and accidents in this environment, software is developed to strict standards, including EN 50128. This standard is very specific on the use of good programming practices, tools, and techniques. This paper discusses safety-critical software affordability and how static analysis tools like GrammaTech's CodeSonar increase developer productivity and satisfy various EN 50128 requirements.

THE SAFETY-CRITICAL SOFTWARE AFFORDABILITY WALL

Software has become the leading cost of safety-critical systems. For example, one third of new airplane costs is in software and software development, and in automobiles, 25% of the capital costs of a new vehicle are now electronics (including software). Software has afforded amazing new capabilities, but its exponential growth and associated costs have made it effectively unaffordable:

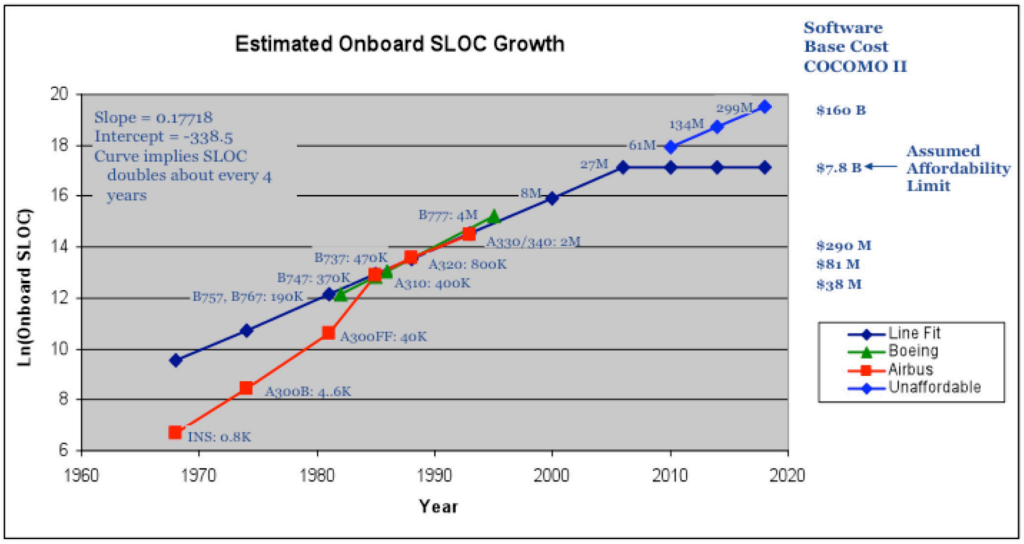

Figure 1: Source: SEI, "Virtual Integration for Improved System Design", Redman et. al, 2010

Clearly, the in-step increase in development costs for larger systems has to change. In line with this reality, various processes and techniques have been proposed. Amongst those is the use of tools – specifically static analysis – to improve the test coverage and to detect defects that traditional testing cannot. In fact, both SEI and NASA recommend static analysis as an indispensable tool in safety-critical software development.

THE EXPONENTIAL COST OF FAILURE

Safety-critical software is expensive to develop and static analysis tools are highly recommended by both certification standards and practitioners in the field, to prevent exorbitant costs. Software failures in manufactured and shipped products can drastically escalate costs, from recalls to litigation to damaged reputation. A significant software failure can have almost unbounded financial

impact – studies have shown that defects cost 100 times more to fix in production than in early phases of development.

As pointed out by Capers Jones (2009), looking at a cost-per-defect metric alone is misleading since it doesn't factor in the volume of defects and the fact that the cost to find and repair a defect is often the same over time (something developers are quick to point out). For safety-critical embedded systems, however, the cost of repair is higher than other industries. If a safety-critical defect isn't fixed on time (or worse, purposely hidden), the financial impact can escalate to legal liability and impact of future revenue.

Considering the typical software development lifecycle illustrated by the V-model below, we can consider the relative benefits of static analysis at each phase of development. The V-model is a good example here, being featured in many of the safety-critical software certification standards (e.g. ISO 16508 and 26262).

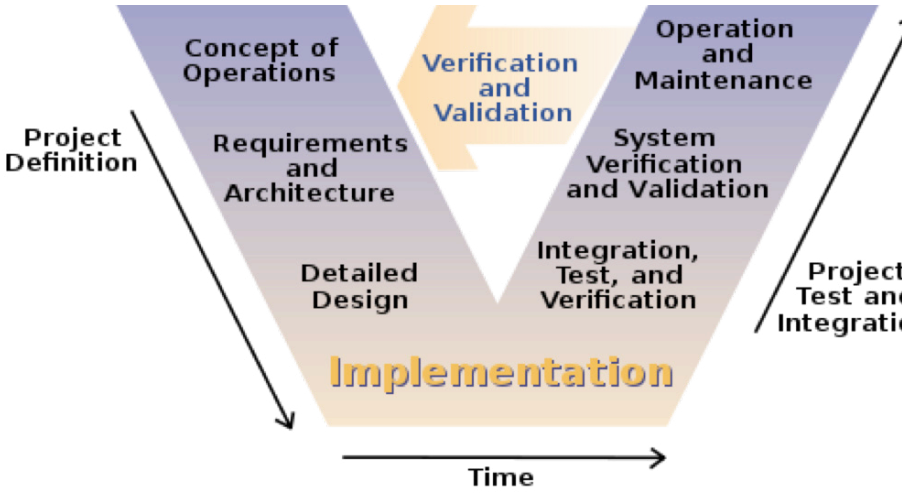

Figure 2: The V-model of system development, often referenced in safety critical standards such as ISO 16508 and 26262.

Rather than looking at the traditional cost-per-defect over time or per phase, which Jones (2009) argues is true mathematically but doesn't reflect what is seen in practice, the more revealing data is the cost-per-defect per relative volume of defects (as volume of defects decreases over time and each phase of development).

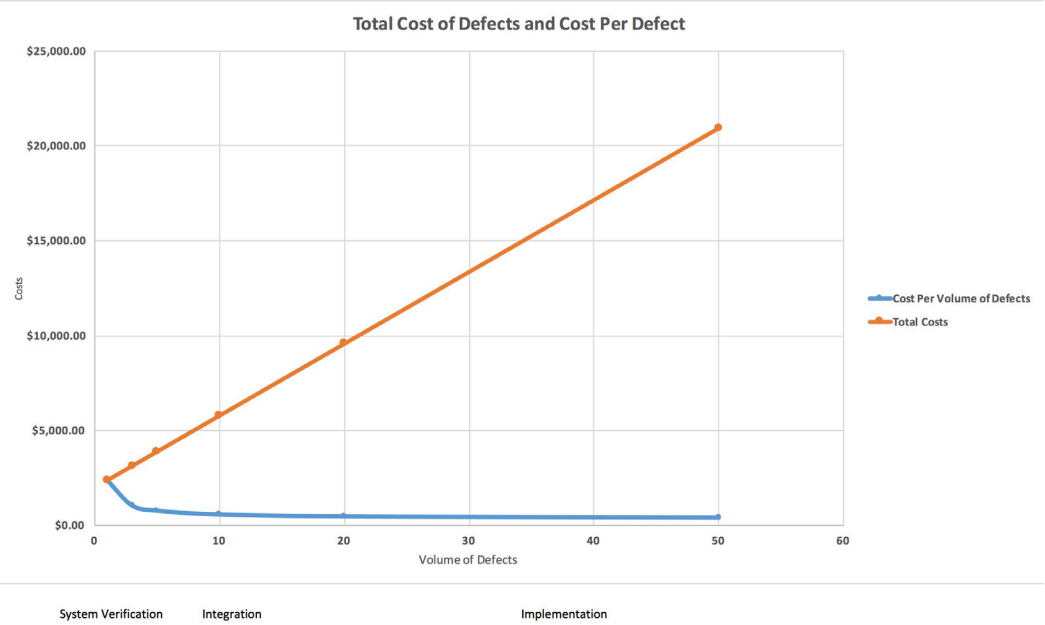

Figure 3: The total cost and cost per defect as the volume of defects diminishes. Source: Capers Jones (2009)

What is interesting about Figure 3 (note the development phase order), is that the cost per defect at each phase goes go up as expected, but total costs are going down due to the decreased volume of defects. In practice, it doesn't take longer to find and fix bugs at each phase, but the costs are still there despite diminished volume. It's worth noting, also, that as a product matures into operation and maintenance (not covered in the chart), cost-per-defect is much higher due to the impact of servicing a fielded product. The other intangible costs such as damage to brand and loss of future customers and income, are still additional factors to consider.

THE ROLE OF STATIC ANALYSIS IN MAKING RAILWAY SOFTWARE SAFER AND AFFORDABLE

The EN 50128 standard is specific about the use of static analysis tools "using a customizable set of Coding Standards, Control Flow and Data Flow Analysis Rules" and is highly recommended for SIL 1 to 4. Interestingly, the standard calls to "Use the inter-procedural Control Flow Analysis module to find variables in use before being initialized, buffer overflows, resource leaks etc." As this is a highly recommended practice, it's clear that static analysis is an important part of the safety-critical development toolkit.

Static analysis tools provide significant productivity gains to software teams seeking stringent software safety certification. Many safety standards require high levels of code coverage (proof that tests executed most, if not all, statements and conditions). Although this is very exhaustive, it's very expensive to do and must be repeated in each major phase of development (unit, integration and system testing). The criticality of the software dictates the level of coverage with some less-critical software requiring no formal test coverage (e.g. aircraft on-board entertainment). Testing code coverage is one metric to evaluate software quality by, but there are cases where it doesn't catch

everything. Using a qualified tool as part of the software development process from early stages of development can have significant benefits:

- Bugs that coverage-based testing miss: Testing software based on coverage metrics is inherently unit-based (although coverage is evaluated at a system level as well). Concurrency errors and security vulnerabilities are two key instances of defects that can be missed even during rigorous testing. Concurrency is often tough to program correctly and can yield errors (e.g. race conditions) that are undetected until some unforeseen condition during operation. Security vulnerabilities do manifest as bugs in the code – the conditions creating the error are often due to types of input not considered during testing. Static analysis can discover these errors early and prevent them from being show-stoppers late in the development cycle.

- Detecting defects early: Rigorous testing can discover most defects in software, but it's expensive and extremely time-consuming. Discovering and fixing these bugs when writing the code is also considerably cheaper than later in the development cycle (defect discovery is exponentially more expensive over time). Static analysis can detect bugs in the code as it is written, as part of a developer's development environment, greatly reducing the downstream cost of defects.

- Managing the software supply chain: 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. Some safety standards consider any software that isn't developed to the specific standard as Software Of Unknown Pedigree (SOUP) -- software that needs to be looked at carefully for inclusion in the system. Static analysis tools 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: Static analysis tools (and many other testing and lifecycle management tools) provide automated documentation to support testing, coding standard, and quality/robustness evidence. Much of the manpower used in safety certifications is documentation, evidence production, and automation. Static analysis reduces this burden significantly.

SOURCE CODE COMPLIANCE

The EN-50128 standard is very clear on using good programming techniques, such as modularity, components, structured and object-oriented programming. It also requires the use of design and coding standards, and language subsets such as MISRA C. In fact, these coding standards are mandatory for higher safety integrity levels SIL 3 and 4. Static analysis tools like CodeSonar are useful and effective in enforcing coding standards, whether they are widely-used standards such as MISRA C, or customized versions specific to your application.

SATISFYING EN 50128 REQUIREMENTS

The following table illustrates how specific EN 50128 requirements are met with a static analysis tool such as GrammaTech's CodeSonar. In many cases, the techniques/practices are highly recommended, if not mandatory, at the most critical levels.

Technique	Reference	SIL 0	SIL 1	SIL 2	SIL3	SIL4
Design and Coding Standards	Table A.4	HR	HR	HR	M	M
Language Subset	Table A.4				HR	HR
Static Analysis	Table A.5, A.8	R	HR	HR	HR	HR
Metrics	Table A.5		R	R	R	R
Coding Standard	Table A.12	HR	HR	HR	M	M
Coding Style Guide	Table A.12	HR	HR	HR	HR	HR
No Dynamic Objects/Variables	Table A.12		R	R	HR	HR
Limited Use of Pointers	Table A.12		R	R	R	R
Limited Use of Recursion	Table A.12		R	R	HR	HR
No Unconditional Jumps	Table A.12		HR	HR	HR	HR
Limited Size and Complexity of Functions/Methods	Table A.12	HR	HR	HR	HR	HR
Single Entry/Exit Point for Functions/ Methods	Table A.12	R	HR	HR	HR	HR
Limited Number of Parameters	Table A.12	R	R	R	R	R
Limited Use of Global Variables	Table A.12	HR	HR	HR	M	M
Control Flow Analysis	Table A.19		HR	HR	HR	HR
Data Flow Analysis	Table A.19		HR	HR	HR	HR
Walkthrough/Design Reviews	Table A.19	HR	HR	HR	HR	HR

Table 1: EN 50128 requirements specifically met y static analysis tools and the recommendation level. References are to specific clauses in EN 50128. Legend: R = recommended, HR = highly recommended, M = mandatory

EN 50128 CERTIFIED SOFTWARE DEVELOPMENT TOOLS

CodeSonar is an EN 50128 certified tool which means that a certification body (TÜV SÜD Saar GmbH in this case) has analyzed the functionality of the tool and its development process and certified that it satisfies the requirements necessary for usage in developing safety-critical software. Why is this important? Tools that are used in the development of safety-critical software must be documented and their results analyzed. Tools that are not certified require further scrutiny from certification bodies, possibly increasing workload and risk on the development team.

CONCLUSION

Static analysis tools have a critical role to play in railway safety-critical software development. The EN 50128 standard for railway software systems is clear in its requirements, highly recommending static analysis for any system SIL 1 or above. Supporting the certification process with certified tools reduces risk, costs, and time.

Despite rigorous testing, failures still occur in safety-critical software, with catastrophic effects in human and economic terms. Static analysis tools are essential to ensure the development of software that is secure, safe, and high quality. In some cases, concurrency errors and tainted data vulnerabilities are difficult to detect with traditional functional testing. Finding and resolving these defects before manufacturing pays huge dividends in time and costs.

By increasing development and testing productivity and finding bugs that are missed by regular testing, static analysis plays a key part in breaking through the safety-critical software affordability crisis.

REFERENCES

Virtual Integration for Improved System Design, Redman et. al, 2010

Safety critical software and development productivity, O. Benediktsson, 2000

Four Pillars for Improving the Quality of Safety-Critical Software- Reliant Systems, SEI, 2013

The Power of Ten – Rules for Developing Safety Critical Code, Gerald Holzmann, Jet Propulsion Laboratory

The Economic Impacts of Inadequate Infrastructure for Software Testing, NIST, 2002

A Short History of the Cost per Defect Metric, Capers Jones, 2009

GrammaTech, Inc. is a leading developer of software-assurance tools and advanced cyber-security solutions. GrammaTech helps organizations develop and release high quality software, free of harmful defects that cause system failures, enable data breaches, and increase corporate liabilities in today's connected world. GrammaTech's CodeSonar is used by embedded developers worldwide.