



Developing Software for Critical Systems

Christof Ebert¹ and Stefan Kriso

From the Editor

Our society and our lives depend on software and IT. As such critical systems directly affect people's lives, utmost care must be used during development. Fast-changing technologies, standards, and increasing liability risks demand state-of-the-art knowledge. Growing economic challenges demand how to do this efficiently. This article with insights from Robert Bosch, a global technology market leader, will provide an overview.—Christof Ebert

SOFTWARE SYSTEMS ARE the basis for automation, mobility, aviation, medical technology, and energy supply. Such systems contribute to social stability by making critical infrastructures such as power grids and public transportation trusted, robust, and reliable. Modern medical technology needs an increasing amount of software, such as in life-saving devices in hospitals or implants such as pacemakers or insulin pumps. In road traffic, safety-critical vehicle functions such as braking and steering systems or autonomous driving functions reduce accidents and protect human lives.¹ Such

software is critical and thus must possess a high quality level, covering reliability, availability, safety, cybersecurity, usability, and more.

Challenges with Developing Critical Software

To fulfill the expectations of users and society, critical systems require suitable development processes that remain effective even in stressful situations, cover the numerous relevant regulations and standards and can be implemented economically. Reliability and integrity of critical systems are crucial for creating trust in technologies and enabling social innovations such as autonomous driving or intelligent systems. Here are some examples:

- **Vehicles:** Complex vehicle functions require secure sensor fusion and reliable decision logic in a variety of environmental conditions.
- **Automation:** Collaborative automated and autonomous systems and production facilities must operate in a controlled manner.
- **Health Care:** Medical devices and clinical IT systems must perform with highest dependability and rule out life-threatening malfunctions.

The main threats to critical systems and infrastructure today are software errors and cyberattacks.^{1,2,3} The risk increases with the complexity of these systems.⁴ Due to

Digital Object Identifier 10.1109/MS.2025.3538258
Date of current version: 11 April 2025

competitive pressure, necessary measures for safe development are often neglected. This can imply that the safety case is inconsistent, and cybersecurity is insufficiently secured and tested. The growing number of accidents in automated vehicles or recalls in medical technology show the importance of reliable software and underlying development processes.⁵

The number of attacks on critical software systems is growing at almost 50% per year.³ Attacks are not only becoming more frequent, but also more intense and more sophisticated. Generative AI copilots not only undermine copyrights, but also create vulnerabilities in the code through malicious code snippets. In addition to functional safety, cybersecurity is therefore highly relevant in the development of critical software.^{1,3}

Risk-Oriented Development of Critical Systems

The development of critical software is a regulated process that places high

Reliability and integrity of critical systems are crucial for creating trust in technologies.

demands on reliability, availability, dependability, safety, and cybersecurity. Figure 1 provides an overview on standards related to development, management, functional safety, cybersecurity, and certification of critical systems, both impacted by product IT and enterprise IT.

Standards define the minimum level of the state of the art to be deployed. They guide the development of critical systems along the entire life-cycle. However, there is a considerable gap in the practical implementation of standards. A typical example is traceability of requirements and design decisions across work products, product

variants, and versions. In practice, we see a similar pattern across industries. Traceability, though demanded by most safety and security standards to ensure consistency, is rarely maintained throughout the product life-cycle with its many software updates. The same is true for test coverage, where many companies prefer to test a feature catalog rather than deploying systematic methods for regression test and for identifying corner cases and critical feature correlations.⁵

Development begins with risk-oriented analyses, such as hazard analysis and risk assessment (HARA) and threat analysis and risk assessment

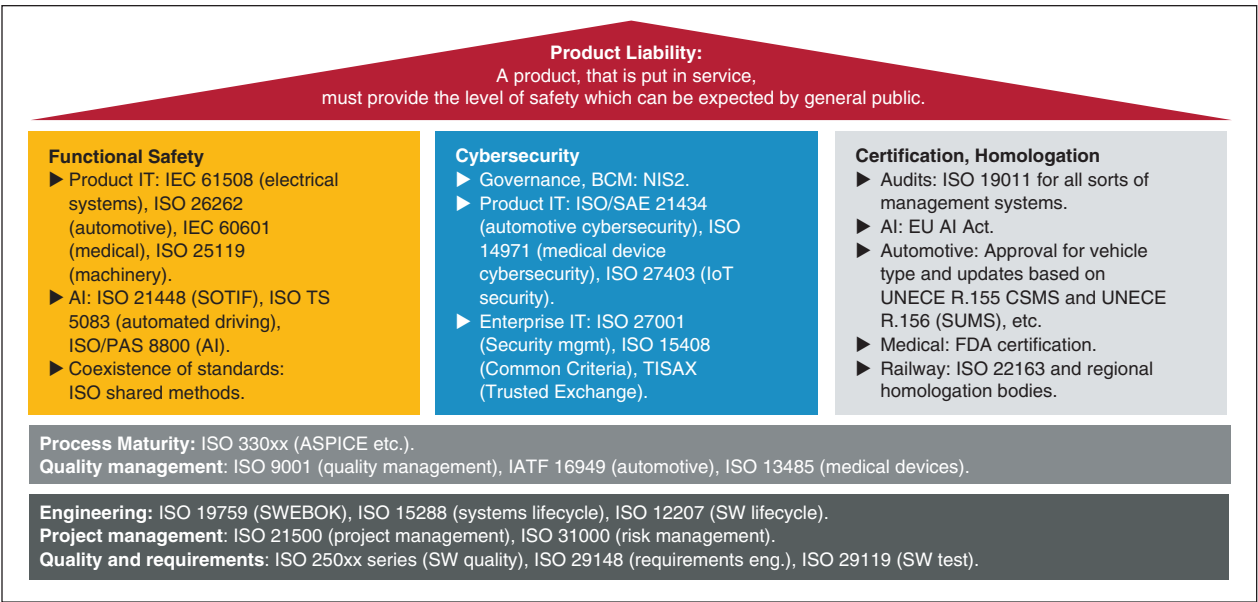


FIGURE 1. Product liability demands using standards.

(TARA), which are used to identify and evaluate potential hazards. Figure 2 shows the risk-oriented evaluation of requirements in a system context. Based on this, an integrity level is defined, which determines the safety risk potential of a function and specifies the requirements for risk mitigation, e.g., fault prevention methods to be applied during development and operational failure detection.^{1,2} Higher integrity levels require stricter measures for fault prevention and error detection.

Continuous development of critical systems requires systematic and automated techniques to ensure end-to-end consistency from functional and quality requirements to design, test, and for documentation.⁴ Model-driven development (MDD) enables functions such as control systems or object recognition to be modeled and simulated at an early stage. Such models can be automatically converted into code, which reduces implementation errors. For critical systems, methods such as formal verification are also used to

mathematically prove that the system meets certain safety requirements.

Test-driven requirements engineering (TDRE) links requirements with testing and enables early validation.⁵ The requirements are specified directly as a test case to provide a basis for later regression tests. At least one direct test case should be specified for each function, the so-called “sunny day scenario,” for later regressions of this function. In addition, essential correlations of functions and the quality requirements are validated with further tests. Agile TDRE applies the idea of test-driven development to requirements engineering and testing. It thus connects the agile flexibility with systematic methods and processes as demanded for software in critical systems development.

The development of critical software is a regulated process that places high demands on reliability, availability, dependability, safety, and cybersecurity.

Where Do We Go from Here?

Good balance of innovation, quality, and costs are the prerequisite for

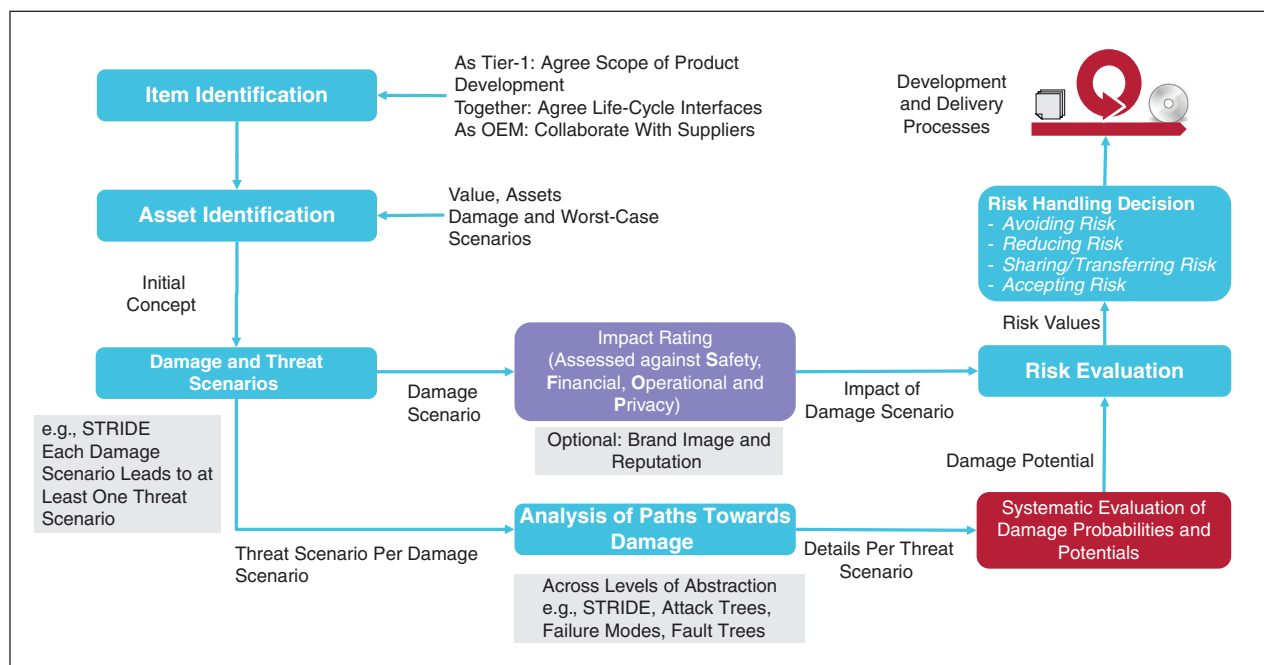


FIGURE 2. Risk-oriented development of critical systems.

sustainable development of critical systems. Various best practices exist that must be carefully selected and consistently applied.

Here some practical guidance for software developers of critical systems:

- *Requirements analysis and traceability:* Start with a risk assessment (HARA and TARA) to identify potential risks and to develop corresponding safety and security concepts. Ensure that functional and quality requirements are traceable to design, test and documentation.
- *Cybersecurity:* Identify attack surfaces and implement tailored protective measures such as encryption, access controls, and secure update mechanisms. Keep in mind that cybersecurity must be practiced consistently and verifiably from development to maintenance.
- *Effective and efficient processes:* Orchestrated and systematic Agile processes allow efficient implementation of necessary standards. Use a lean standard approach to balance the necessary quality in terms of product liability while at the same time limiting the otherwise excessive costs of complying with the standards.
- *Development tools:* Keep artifacts consistent across the product lifecycle and its many software updates and defect corrections. Use suitable tools to model, simulate, and test functions at an early stage. Automated code generation and regression testing help to reduce implementation errors.
- *Design for safety:* Integrate fault tolerance, e.g., fail-operational and redundancy into the design. Create a robust

ABOUT THE AUTHORS



CHRISTOF EBERT is the managing director of Vector Consulting Services, 70499 Stuttgart, Germany. He serves on the editorial board of *IEEE Software* and is a Senior Member of IEEE. Contact him at christof.ebert@vector.com.



STEFAN KRISO is the functional safety lead at Bosch Center of Competence Vehicle Safety, Robert Bosch GmbH, Stuttgart, 70442, Germany. Contact him at stefan.kriso@de.bosch.com.

Critical systems development means continuous and systematic risk management.

- foundation for safety-critical applications.
- *Comprehensive verification:* Combine coordinated verification methods with hardware-in-the-loop tests and limit value tests. Validate the software in real application scenarios, e.g., through simulations, penetration tests, and robustness checks.
 - *Constantly updated safety and security case:* Automatically document conformity of safety and security requirements along the lifecycle and frequent software deliveries. Lean document generation for analyses, verification and test results facilitate consistency.

- *Ensure usability:* Many errors occur due to poor usability, such as complex interfaces. Develop intuitive interfaces and clear feedback systems, especially for emergency scenarios, e.g., in medical technology in accordance with ISO 62366.

Famous American futurologist Alvin Toffler remarked “You may forget some critical factors, but they won’t forget you.” Critical systems development means continuous and systematic risk management. To achieve the right quality level and mitigate liability risks, a multitude of standards must be followed. At the same time, global competition asks for continuous efficiency improvement.

Agile development of critical systems reduces the overhead of heavy standards and brings direct benefits—both technically and economically. It reduces liability risks through traceability and transparency. And it is extensible and scalable for increasingly AI-instrumented development.⁵ The described guidance for developing critical systems supports project and risk management and facilitates lean implementation of functional safety and cybersecurity. This directly reduces the cost

of rework and indirectly reduces the risk of product liability. 

References

1. M. Weyrich, *Industrial Automation and Information Technology*. Berlin, Germany: Springer, 2024.
2. D. J. Smith and K. G. L. Simpson, *The Safety Critical Systems Handbook*, 2nd ed. London, U.K.: Butterworth-Heinemann, 2020.
3. C. Dimitriadis, “Underfunded, under pressure: We must act to support cyber teams.” *Computer Weekly*, Nov. 19, 2024, Accessed: Jan. 1, 2025. [Online]. Available: <https://www.computerweekly.com/opinion/Underfunded-under-pressure-We-must-act-to-support-cyber-teams>
4. *Software Engineering Body of Knowledge*, vol. 40, Washington, DC, USA: IEEE Computer Society, 2024. [Online]. Available: <https://www.computer.org/education/bodies-of-knowledge/software-engineering>
5. C. Ebert and M. Weyrich, “Validation of autonomous systems,” *IEEE Softw.*, vol. 36, no. 5, pp. 15–23, Sep./Oct. 2019. doi: [10.1109/MS.2019.2921037](https://doi.org/10.1109/MS.2019.2921037).

Unlock Your Potential

WORLD-CLASS CONFERENCES — Over 195 globally recognized conferences.

DIGITAL LIBRARY — Over 900k articles covering world-class peer-reviewed content.

CALLS FOR PAPERS — Write and present your ground-breaking accomplishments.

EDUCATION — Strengthen your resume with the IEEE Computer Society Course Catalog.

ADVANCE YOUR CAREER — Search new positions in the IEEE Computer Society Jobs Board.

NETWORK — Make connections in local Region, Section, and Chapter activities.



Explore membership today
at the IEEE Computer Society
www.computer.org

