



Implementing Functional Safety

Christof Ebert

What are the current best practices for implementing functional safety? How can we follow the heavyweight industry standards efficiently and effectively? Many companies in critical industries such as the automotive industry, automation, and medicine approach me with such questions. To cope with emerging product liability risks and to ensure high-quality products, significant improvements to technology and processes are necessary. This column presents industry experiences in adapting development and technology to conform to safety standards. I look forward to hearing from both readers and prospective column authors about this column and the technologies you want to know more about. —*Christof Ebert*

A CAR MANUFACTURER had introduced an electronic parking-brake system to assist drivers and save on weight and mechanical overhead and cost. Its underlying principle was

fine, and, of course, the system had two redundant channels straight from the button to the brakes.

On a hot summer day, the driver stopped the car to get something

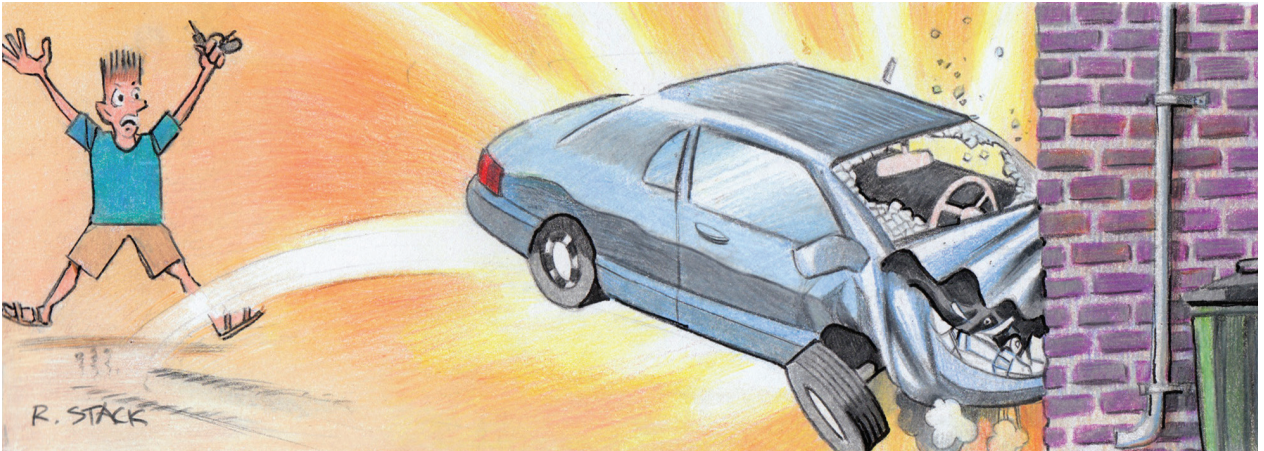
What had happened? The electronic parking-brake system worked just fine. But, when the driver left the car, he naturally opened the door. This allowed hot air into the vehicle. The air conditioner activated to sustain the desired interior temperature. Because it needed more power, it slightly increased the throttle—which released the brake.

Safety-critical systems can cause physical harm if they fail. Failures can be due to random hardware faults (for example, a broken component) or systematic design errors (for example, software defects). The risk associated with the system is reduced by minimizing the probability of a failure occurring and limiting the consequences of unavoidable failures. With the increasing complexity of systems, their electronic components, and their interworking, safety

We need to ensure safety end to end—with both a quality and a business perspective.

simple. The driver pushed a button to activate the brake, which prevented the car from rolling. When the driver stepped on the accelerator, the brake released, relieving the driver from simultaneously releasing the brake and accelerating. The concept worked

from the back trunk and activated the brake. He left the engine running because it was a short stop and he only intended to briefly leave the vehicle. The car was, after all, secured by the parking brake. Suddenly it accelerated and crashed into a wall.



concepts should be at the core of any new design, whether it's for changing an existing function, such as in my automotive example, or adding a new one. Here, I discuss some best practices in efficiently and effectively implementing functional safety in software systems.

Functional-Safety Concepts

Functional safety is a property of the system as a whole rather than just a component property; that is, it depends on the integrated operation of all sensors, actuators, control devices, and so on. The goal is to reduce the residual risk associated with a functional failure of the system below a threshold given by the assessment of severity, exposure, and controllability.¹⁻³

Safety as a basic concept has a long tradition in specific disciplines, such as designing a pacemaker with high reliability or an antilock braking system. Both have strongly reduced fatalities with advanced technology, while being safe by not causing injuries by malfunction.⁴ However, safety has reached a new level of risk and impact. It's no longer just individual components that add to safety but increasingly their

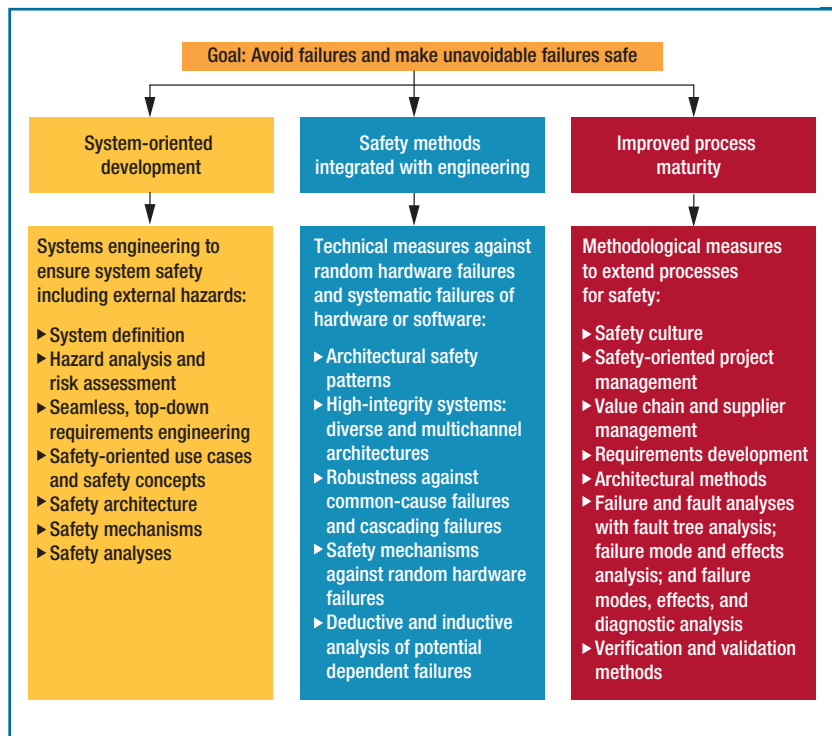


FIGURE 1. Practical safety engineering. To be successful, safety engineering must meet three needs in parallel: system-oriented development, safety as an integral part of engineering methods, and strong process maturity.

interworking at the system level—for example, a medical ecosystem with remote diagnostics or the entire car with its interworking electronic subsystems.

My experience with safety engineering in different industries shows that it's only successful when considering three needs in parallel (see Figure 1): system-oriented development,

TABLE 1

Industry standards for safety-critical products and systems.

Standard	Scope	Approach	Interpretation or use
IEC 61508	The generic standard for electronic systems' functional safety	Basic safety understanding with risk-based mitigation	The predecessor of many safety standards for electronic systems. Several industry-specific standards have been derived from it. Still widely used despite its rather high-level approach. A solid focus on hardware aspects, but less suitable for software.
ISO 26262	Road vehicles	Rather detailed and prescriptive risk-based mitigation covering product, process, and culture	Focuses on electrical and electronic subsystems for automotive vehicles. No formal certification required. Covers the complete life cycle, systems engineering, hardware, software, production, and operation. Will evolve toward all types of vehicles.
ISO 25119	Agriculture and forestry	Prescriptive risk-based mitigation	The safety standard for agriculture and off-road systems, such as industry transport, mining, and so on. Increasingly enhanced by ISO 26262 for off-road vehicles.
ISO 13849	Machinery and safety-related parts of control systems	Safety awareness for a wide range of industries	Focuses on the performance and safety of industrial machinery, such as production systems and embedded control systems.
IEC 62304	Medical-device software	Basic safety-oriented life-cycle requirements	The primary standard for implementing functional safety in medical equipment. A strong focus on certification, unlike the other standards.
DO-178C	Avionics software development	Rather strong process focus and life-cycle requirements	The standard for software development in civil aviation. The DO-254 hardware standard for complex electronics was derived from it. DO-178C is seeing increased application in defense programs. DO-178B, a rigid enforcement by certification, has proven its effectiveness over two decades, minor revision, and extension with DO-178C.
ISO 15288	Systems life-cycle processes	Generic life-cycle requirements	Provides an overarching systems-engineering life-cycle framework to which specific industry norms are adjusted.
ISO 12207	Software life-cycle processes	Generic life-cycle requirements	Provides the foundation for software product life-cycle development.

safety as an integral part of engineering methods, and strong process maturity.

Safety Standards

Safety standards and methods are increasingly converging across industries. This is driven partly by ISO's efforts to provide coherent standard frameworks such as ISO 15288 for systems life-cycle processes, ISO 12207 for software development life-cycle processes, ISO 250x0 for quality attributes and their validation, ISO 15504 to guide process and life-cycle assessments, and a set of standards for func-

tional safety such as ISO 26262 and ISO 13849.

Suppliers for safety-critical products and systems as well as component suppliers must comply with formal processes such as those summarized in Table 1. These standards have different origins and address different safety cultures. Nevertheless, I see converging approaches toward hierarchically breaking down system-level safety requirements to safety cases with full traceability to the product and its use and maintenance.³

The current safety standards prescribe a top-down refinement of the

technical safety requirements based on an initial hazard and risk analysis and system safety concept. This refinement employs techniques such as failure mode and effects analysis (FMEA) and fault tree analysis (FTA). Traceability of the safety requirements throughout development, up to and including a systematic testing approach, is required. In addition, the standards emphasize supporting processes such as supplier management, configuration management, and quality management to ensure that failures in basic project management activities don't introduce errors. Typically,

the product supplier or OEM develops an overall system safety concept and associated safety case. These are refined on the basis of specific contributions from the component and subsystem suppliers.

The current safety standards are rather abstract and don't give much concrete guidance on how to efficiently integrate the required measures into existing processes.³ To provide more guidance, recent safety standard evolution tends to provide tons of prescriptive details on how to treat engineering processes, safety requirements and their traceability and validation, tool use, and safety case documentation and maintenance. The risk with this approach is that engineers and their management are overwhelmed and just follow what's described while not designing according to safety needs, actual risks, and economic trade-offs. Also, with either the abstract or prescriptive standards, safety engineering often is expensive and complex but doesn't necessarily exhibit improved safety to the user.

The best way to understand this is through the examples of the software failures of the London ambulance system (in which software reconfiguration led to service disruption) or the Therac-25 radiotherapy machine (which caused deaths from overdoses).⁴ Both systems followed the required standards but still were unsafe because they lacked usability. With untrained users who don't understand the underlying devices' complexity (as in my automotive example), this problem will worsen. The message clearly is that safety standards aren't sufficient if developers pair outdated criteria of systems engineering and thinking with disciplined high-maturity processes.

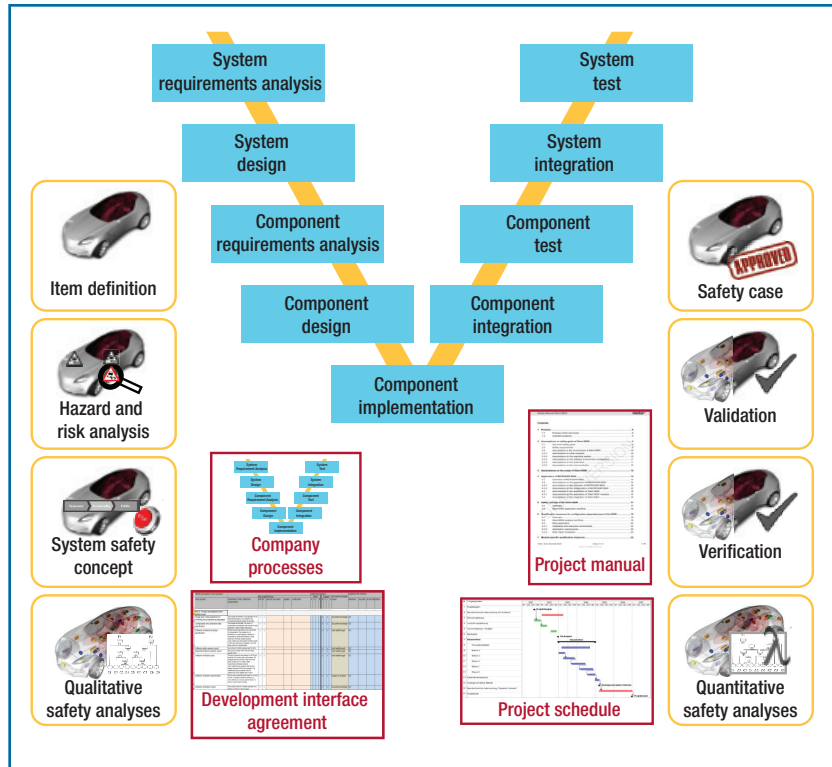


FIGURE 2. Safety concepts evolve top-down. The “V” relates the different abstraction layers to major safety concepts and deliverables, such as a safety concept on the left as a result of hazard analysis and item definition, or the safety case at the end of development, which documents all safety measures.

Success Factors for Safety Engineering

Let's look in more detail at the three safety-engineering needs related to systems, methods, and maturity.

System-Oriented Development

Safety demands a systems-engineering perspective. For example, the growing complexity of embedded electrical subsystems and their interconnections together with the increasing variability of system features make embedded electric and electronics systems more and more vulnerable. Such risks demand strong protection on various levels throughout the life cycle. The most relevant goal is to identify the per-

minent safety risks and how to avoid or mitigate them at the system and software levels.

A structured life cycle must be available—and followed—not from a piecemeal perspective in which verification happens on a rather low abstraction level, but as an integrated framework in which requirements and specifications for safety design, implementation, and verification are derived top-down from systems requirements and design principles. Figure 2 provides examples of safety-engineering techniques and how they're driven from a systems-engineering perspective.

Safety concepts must be defined at the system architecture level.

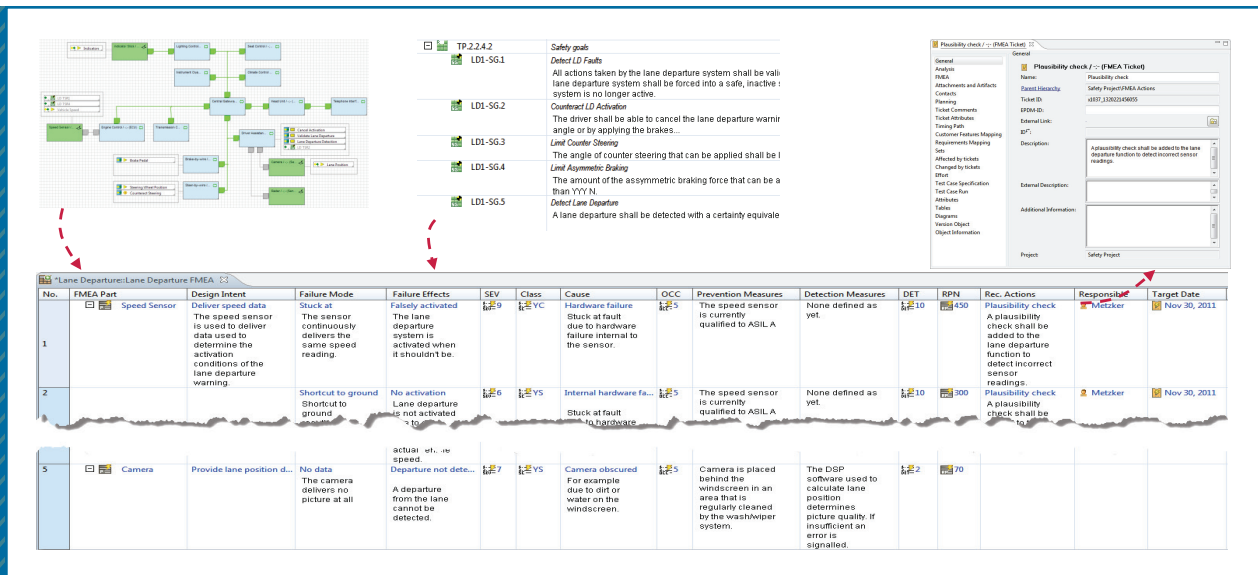


FIGURE 3. Implementing functional safety needs systematic engineering. This requires a coherent set of methods supported by engineering tools. The picture shows deliverables such as software design, safety requirements, and test cases, which must all be connected to achieve traceability and coverage.

Systematic requirements management is needed to document the safety requirements derived from the system-level safety analyses, using methods such as FMEA and FTA; allocate those requirements to individual components; and, after successive safety analyses at the component (hardware and software) level, formulate specific safety-related technical requirements for each component. A specific focus is on obtaining a *development interface agreement* to ensure a clear responsibility assignment along the supply chain. On the right side of the “V” in Figure 2, the related verification and validation mechanisms are applied to the respective abstraction level from components to the entire system. The “V” doesn’t at all suggest a sequential approach to design and verification, but rather an integrated approach such as those of agile methods. For example, test-driven development supports safety

engineering while addressing the need for different coverage schemes.

Safety as an Integral Part of Engineering Methods

The development of safety-critical software requires more emphasis on excluding systematic errors through software architecture design, software implementation guidelines, and software verification and validation. Specific safety methods and techniques must be fully integrated within engineering. This is obvious when it comes to using coding standards and rules for programming safety-critical software. But coding isn’t enough. Only systematic engineering processes can meet modern electronic systems’ demands for high quality. Starting late and looking only to individual functions and their mapping to components, as developers used to do, isn’t the answer. A high overall quality level and thus early defect detection and

removal are feasible only by combining a multitude of cascaded design, verification, and validation activities. These activities start with requirements engineering and stop only with the end of service and retirement of a product and service.

Although defect avoidance should be the first and most relevant step, experience shows that people repeatedly struggle with it simply because their processes won’t support it. To effectively avoid defects, developers must define, systematically apply, and quantitatively manage engineering processes. With that overall process in place, defect prevention can cost-effectively boost customer satisfaction and business performance, as many high-maturity organizations have shown.

Safety engineering during design demands systematic defect prediction, detection, correction, and prevention. The first step is to identify

risks and hazards as they would relate to malfunctions and how defects would be critical to performance. The underlying techniques are statistical methods of defect estimation, reliability prediction, and criticality assessment. Systematic defects in the design or implementation must be detected by quality control activities, such as inspections, reviews, and requirements-based tests. Because each of these techniques has its strengths and weaknesses, they should be combined to maximize efficiency.

Strong Process Maturity


Safety engineering needs a thorough, systematic development process (see Figure 3). This is necessary for certification, such as in regulated medical or aerospace industries; for managing safety throughout product versions and variants; and certainly for visibility in case of liability lawsuits. A low-maturity organization wanting to ramp up toward a medium safety integrity level typically pays 30 to 50 percent more for engineering. Combining the introduction of safety engineering with improved engineering processes such as CMMI Level 3 would still create this cost. However, it would also yield immediate returns through defect phase containment, easier change management, and better project performance—each reducing engineering costs by approximately 5 to 10 percent.

Systematically implementing a safety culture needs professional organizational change management. Often, I've faced organizations that apply a safety standard in a bureaucratic, formal way ("we need to get certified"), which creates unnecessary high costs and demotivates engineers owing to the formalism. In such environments, basic safety

principles aren't understood, and the move toward sufficient process maturity, both lean and effective, isn't trivial. For the reasons I just described (the move toward system-oriented development and the need to establish basic process capabilities), adapting development processes to conform to safety standards nearly always must be part of a systematic, wider-ranging process improvement effort.

Safety engineering, and thus the adoption of safety standards in product development, isn't rocket science and will be absolutely necessary for many engineers to mitigate product liability risks. Let's go back to the parking-brake example to show what would have been different with proper safety engineering. System-oriented development would have considered the entire system and its signals. So, the system wouldn't have used the throttle signal; it would have used the accelerator pedal position, which was in the unused state. Integrating safety methods across the life cycle would have, for instance, created test cases in parallel with establishing the safety case. The test cases would have been broken down to ensure on the component and product levels full traceability from safety requirements and safety cases to design decisions and their verification. Finally, a higher process maturity would have reviewed operational scenarios from different perspectives and evaluated alternative designs before implementing one that looked the most cost-effective but didn't fulfill safety standards.

Safety engineering will be successful only if stakeholders perceive and perform it as a cultural change.

However, too often, safety is considered primarily a technical challenge that adds a few requirements to an already overly long specification. When supporting clients on functional safety, I emphasize establishing a thorough safety culture adjusted to the client and its products. Management and engineering staff need to understand the challenge of safety as a multidimensional need that impacts management processes, responsibilities, and engineering methods. Functional safety must be seen as a critical product liability issue with consequences for disciplined, formalized development. Engineers must understand safety needs on the system level and adopt their engineering methods toward systematic, traceable decision making from the architectural level, to the functional level, to the component level. With our modern society's increasing dependency on embedded software, we need to ensure safety end to end—with both a quality and a business perspective. 

References

1. S. Anderson and M. Felici, *Emerging Technological Risk: Underpinning the Risk of Technology Innovation*, Springer, 2014.
2. D.J. Smith and K.G.L. Simpson, *Safety Critical Systems Handbook*, Elsevier, 2010.
3. C. Ebert, *Functional Safety Overview Webinar and Podcast*, Vector Consulting Services, 2015; https://www.vector.com/vc_download_en.html?product=safety.
4. N.G. Leveson, *Safeware: System Safety and the Computer Age*, Addison-Wesley, 1995.

CHRISTOF EBERT is the managing director of Vector Consulting Services. He serves on the *IEEE Software* editorial board. Contact him at christof.ebert@vector.com



Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.