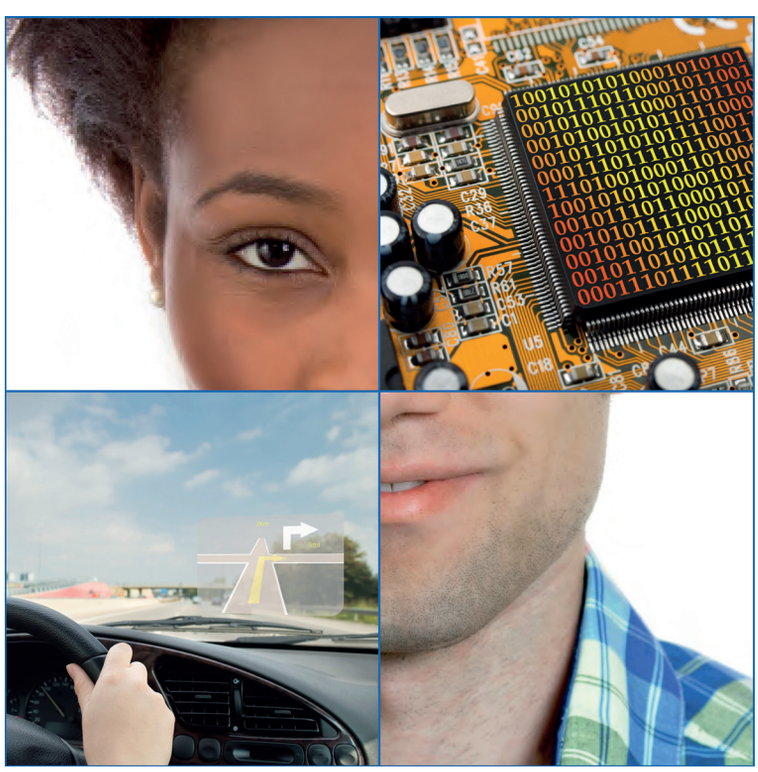
ZVEI:
Die Elektroindustrie

Introduction to the combined Application of Agile & Safety in Automotive Software Development



German Electrical and Electronic Manufacturers' Association

Imprint

Introduction to the combined Application of Agile & Safety in Automotive Software Development

Publisher:
ZVEI - German Electrical and Electronic
Manufacturers' Association
Electronic Components and Systems and
PCB and Electronic Systems Divisions
Lyoner Strasse 9
60528 Frankfurt am Main, Germany

Phone: +49 69 6302-276
Fax: +49 69 6302-407
E-mail: zvei-be@zvei.org
www.zvei.org

Responsible:
Dr. Stefan Gutschling, ZVEI

Februar 2021

Table of Contents

1 About this Document

1.1 Motivation

The future value of automobiles will undoubtedly be created through software. Whereas software contributes about 10 percent of the added value today, it is expected to rise to 40 percent by 2020 (Morgan Stanley Research, 2016). Software and the relevant electronic control units are paramount for enabling trends in the automotive industry, such as automated driving, connected car and electric mobility. With an average modern high-end car compromising up to 100 million lines of code, managing software development efficiently, while adhering to all safety related issues have grown highly in importance.

Traditional methods applied for organizing software development (e.g. planning the whole project from start to end) may not provide the flexibility needed for handling innovation.

In contrast, agile principles can stimulate efficient organization and planning of software development, also in a functional safety context.

1.2 Target Audience

This document targets people with experience in either functional safety or agile software development in the automotive industry. Furthermore, this document can be of relevance to anybody who is familiar with quality management systems. It is important to note, that a functional quality management system must already be established as a prerequisite for an agile & safety application.

1.3 Scope

This document describes how agile practices can be combined with the functional safety standard ISO 26262 when developing safety-related automotive embedded software. As such, the discussed ISO 26262 requirements are primarily focused on part 2 “Safety Management” and considerations on part 6 “Product Development at the Software Level”, part 8 on “Supporting Processes” and part 9 on “ASIL-oriented and Safety-oriented Analyses”. Scrum, its workflow and roles, will be detailed as an agile method in combining agile with safety due care.

In this document, we will focus on software development, knowing that an already large and increasing part of function development is data-driven.

ISO 26262 only mentions the handling of configuration and calibration data quite briefly and Scrum does not describe data handling approaches at all.

Also, agile system development as well as security aspects will also not be discussed here. As the importance of both areas is expected to grow, they may be included in future considerations of combining agile and safety.

1.4 Prerequisites for Combining Agile & Safety

For combining agile and safety in the institution of question, two prerequisites must be met for the application to be suitable.

1. A suitable quality management system must have already been established within the institution. In an automotive context, ASPICE Level 2 is an appropriate reference for process maturity for software development.
2. If agile methods is to be introduced in an organizational unit that already develops safety-related software, a suitable functional safety management system and development and engineering approach for safety-related software must have already been established (e.g. suitably performed process, method and tool enhancement regarding development of safety-related embedded software in accordance with the applicable ASIL).

Furthermore, it is highly recommended that all relevant employees either have experience with both function safety and agile methods

2 Introduction to Scrum and ISO 26262

2.1 Agile Values, Principles, Methods and Practices

The term agile was popularized by the Manifesto for Agile Software Development by a team of seventeen software developers in 2001. As mentioned earlier, motivation for adaptive software development arose from difficult and cost-intensive solutions when faced with spontaneous changes and requirement. The agile manifesto outlines four fundamental values and complements those with twelve additional principles.

Values and Principles

The agile manifesto is based on the following four core values:

1. Individuals and interactions over processes and tools.
2. Working software over comprehensive documentation.
3. Customer collaboration over contract negotiation.
4. Responding to change over following a plan.

The four core values are supplemented by twelve principles that offer further clarification on agile practices:

1. Customer satisfaction by early and continuous delivery of valuable software.
2. Welcome changing requirements, even in late development.
3. Working software is delivered frequently (weeks rather than months).
4. Close, daily cooperation between business people and developers.
5. Projects are built around motivated individuals, who should be trusted.
6. Face-to-face conversation is the best form of communication (co-location).
7. Working software is the principal measure of progress.
8. Sustainable development, able to maintain a constant pace.
9. Continuous attention to technical excellence and good design.
10. Simplicity – the art of maximizing the amount of work not done – is essential.
11. Self-organizing teams.
12. Regular adaptation to changing circumstance.

Agile Methods

Most agile methods are derived in alignment with the values and principles of the manifesto for agile software development. The agile methods Scrum and Kanban are nowadays the most commonly known. This document will discuss Scrum, its workflow and application in a safety context in detail.

Agile Practices

Agile practices are descriptors of best practices that support the implementation of agile software development. These can cover various areas such as requirements, modelling, coding and testing. In combination, these agile practices enable the implementation of the different agile methods. Due to the large number of different agile practices, a selection of them are outlined in this document and will be discussed in detail in chapter 6:

- Refactoring
- Pair programming
- Test driven development
- Continuous integration
- User stories

2.2 Possible Misunderstandings Regarding "Agile Software Development"

Parallel to the comprehensive agile methods and practices, it is not uncommon that some agile concepts are vaguely understood or even misinterpreted. Some misunderstandings are cleared up in this section.

"Agile means that you don´t have to follow any processes." The application of agile development practices in automotive requires suitable development processes that are well-defined and monitored strictly. In order to enable agility, it should be possible to improve them easily and quickly. The processes should be defined in such a way that all teams are able to adapt and improve their individual way of working as long as no other teams are concerned.

„Agile only plans short-term." The combination of a rough long-term and a detailed short-term planning provides the necessary outlook just like traditional planning but also reduces waste in case of plan changes.

„Agile needs no documentation." Documentation is reduced to the necessary minimum, e.g. for enabling maintenance and customer guidance. Documentation intended just for short-term knowledge transfer is replaced by close human interaction.

„Agile has no automotive application." All subject matter activities necessary for compliance with quality and safety standards can be included like all other tasks. In fact, quality and safety can be improved by an agile working mode, e.g. because suitably applied agile approaches can ensure a constantly high-quality level and usually supports early verification or even validation.

„Agile can't handle unforeseen requirement changes." During an implementation iteration, requirements should not be changed, but agile practices enable very late requirement changes without rework. This enables fast adaptation to changed market needs.

2.3 Scrum

Scrum is a management approach to support the management of a cross-functional team, especially in an environment for software development. Jeff Sutherland and Ken Schwaber, inventors of the Scrum development management process, were determined to facilitate development teams to deliver working software within few weeks. To enable this objective, their management framework Scrum defines three distinct roles and describes an iterative, incremental development process (https://www.scrum.org/resources/scrum-guide). As an agile method, Scrum is primarily an attitude towards relationships between employees, managers and customers. This attitude is also reflected in the terminology of Scrum, with its origins being borrowed from rugby. In rugby, Scrum describes the situation when the ball is introduced into the play. Players from both teams are packed closely together in a circular formation, attempting to gain as much ground as possible. Translated to the management framework, it is supposed to symbolize the coherence of the Scrum team and the adherence to rigid roles and processes. The Scrum workflow and its roles are described in detail in the following chapters.

2.4 Agile Scaling

Agile practices and their benefits have traditionally been enjoyed by small, co-located teams, as agile practices were originally designed for a small team size. To leverage these good results in larger organizations and more complex environments, agile practices must be scaled. To meet challenges, including the integrating of non-development activities, different agile scaling frameworks have been proposed by experts. These agile scaling frameworks are applied to large projects with multiple cross-functional teams. In the automotive industry the most commonly used frameworks are LeSS and SAFe.

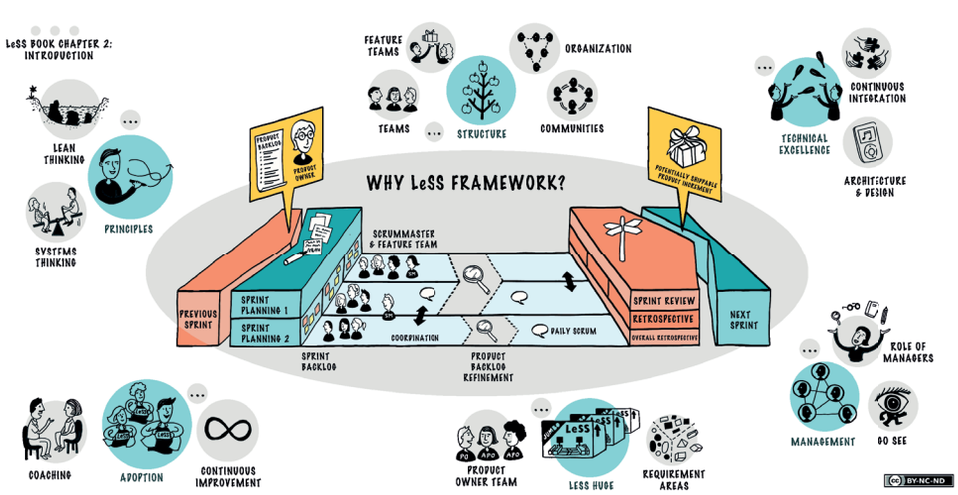


Figure 1: LeSS overview diagram (source: The LeSS Company B.V.)

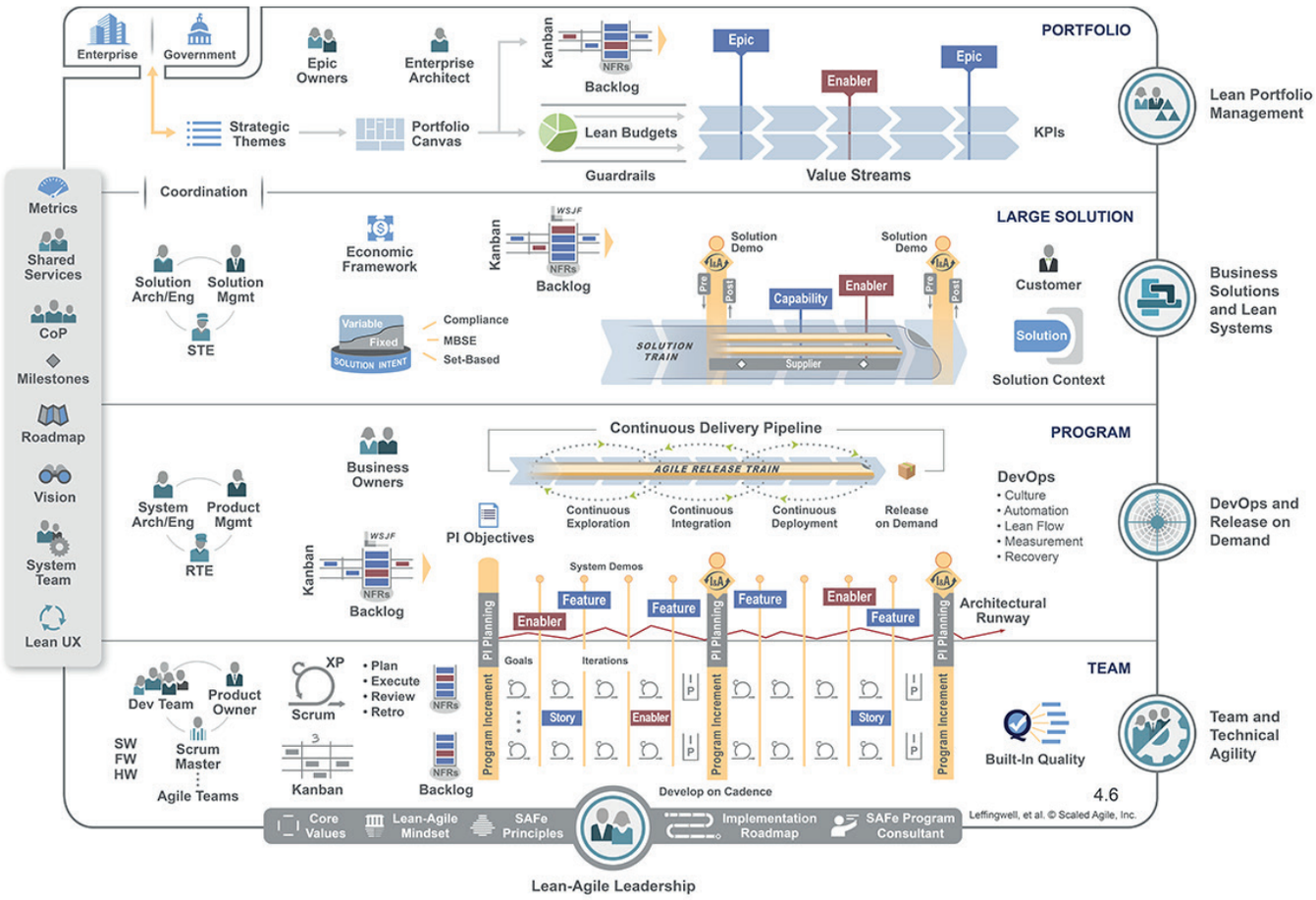


Figure 2: Full SAFe configuration (source: ©Scaled Agile, Inc.)

2.5 Functional Safety

The topics of safety and risk have been growing in importance in the public eye, as the number of electrical, electronic and programmable systems have been increasing exponentially to satisfy the demand for more automated and electrified functionalities. These range from essential vehicle functions such as steer-by-wire and brake-by-wire to more complex features such as connected car and highly automated driving. Hazards resulting from malfunctioning behavior of E/E systems caused by contained hard- and software, fall under the definition of functional safety.

To warrant functionally safe products from a legal perspective, product development and its related safety aspects are expected to be developed according to state of the art. For instance, under the German product liability act the following is mandatory: "The duty of replacement by the manufacturer is eliminated only, if the error could not have been detected according to state of the art in science and technology at the time of bringing the device into market" (Product Liability Act, §1 Cl. 2, Cypher 5). State of the art in science and technology is defined by established standards, current products on the market and other literature, such as papers and publications.

2.6 ISO 26262 – Functional Safety of Road Vehicles

To meet functional safety requirements and develop according to state of the art, it is reasonable to develop according to established international standards. These serve as proper basis for argument during product liability cases by providing evidence that all reasonable functional safety objectives were satisfied. ISO 26262 is an international standard, which covers the functional safety of electrical and electronic systems of series production road vehicles.

ISO 26262 includes guidance to mitigate risks from systematic failures and random hardware failures by providing appropriate requirements and processes. ISO 26262 provides an automotive safety life cycle by addressing the safety-related aspects of development activities and work product and an automotive-specific risk-based approach to determine integrity levels. These integrity levels are referred to as Automotive Safety Integrity Levels (ASIL). ASILs are used to specify applicable requirements of ISO 26262 to avoid unreasonable residual risk. These are further supplemented by requirements for verification, validation and confirmation measures to ensure a sufficient level of safety.

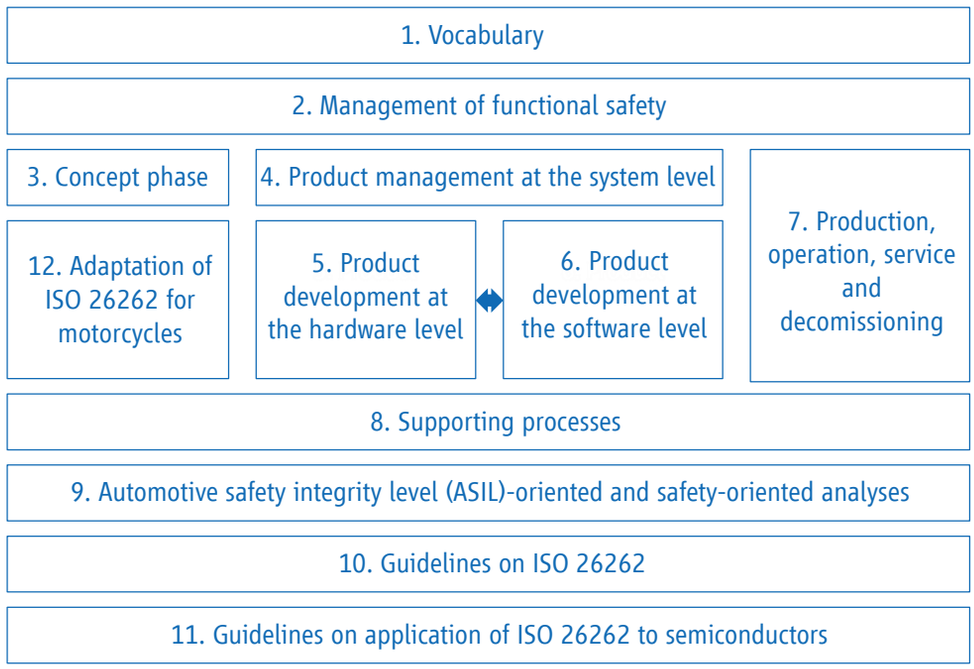


Figure 3: Source Structure of ISO 26262: 2018 (source: Elektrobit Automotive)

Figure 3 below shows the structure of ISO 26262. It consists of twelve parts and is based upon a V-model as a reference process model for the different phases of product development. The "V" represents the interconnection between parts 3, 4, 5, 6, 7 and 12.

For application of agile software development approaches, part 2 on "Management of functional safety", "part 6 on 'Product development at the software level", part 8 on "Supporting processes", and part 9 on "ASIL-oriented and safety-oriented analyses" are of primary relevance.

2.7 Interrelationship between Scrum and ISO 26262

ISO 26262 mentions Automotive SPICE® (ASPICE) as a means to achieve a working quality management and to establish suitable basic software development processes (QM to ASIL D). ASPICE is a framework for improving and evaluating processes within the automotive industry. ASPICE targets repeatable project success through sufficient process quality. ASPICE does not however specify in what ways these requirements must be met. Agile methods support to meet ASPICE collaboration-related requirements by, for example. assigning roles and facilitating interaction between stake holders. Figure 4 shows the interrelationship between functional safety, ASPICE and agile in this case the agile method Scrum.

Scrum, as seen in Figure 4, can support meeting ASPICE requirements regarding Project Management and therefore also assists in meeting corresponding requirements in ISO 26262.

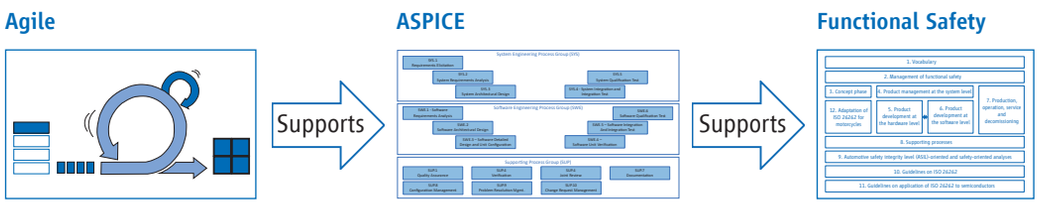


Figure 4: Interrelationship between Functional Safety, Automotive SPICE®, and Scrum (source: Elektrobit Automotive)

Scrum Supports Compliance to ASPICE

In one sprint (nearly) all ASPICE process areas may be processed. This makes it possible to assess and improve the process maturity regularly after each sprint in the project. Thus process changes can almost immediately be tested in the project regarding improvement impact and efforts. This increases quality, efficiency and acceptance of the resulting processes. As ASPICE does not assess the theoretical processes, but their practical application in a project, the acceptance of a defined process by the project team is essential for achieving a good ASPICE rating.

Doing all activities related to one function in a very short time frame within one team makes it much easier to keep, for example. requirements, design, implementation and tests consistent and to link them in a traceable way, which is a central requirement of ASPICE.

ASPICE Compliance Supports Compliance to ISO 26262

ISO 26262 part 8 (“Supporting processes”) demands the usage of configuration and change management. ASPICE is explicitly mentioned as an example of a possibly applicable standard in this context.

ISO 26262 part 3 (“Concept phase”) describes how technical risks are classified according to so called Safety Integrity Levels (ASIL) or “QM”. QM indicates that quality processes are sufficient to manage the identified risk. For risks classified with an ASIL, requirements from ISO 26262 come on top of the normal quality processes. Therefore software development compliant to ASPICE can be considered as the necessary basis for compliance with ISO 26262.

3 Workflow

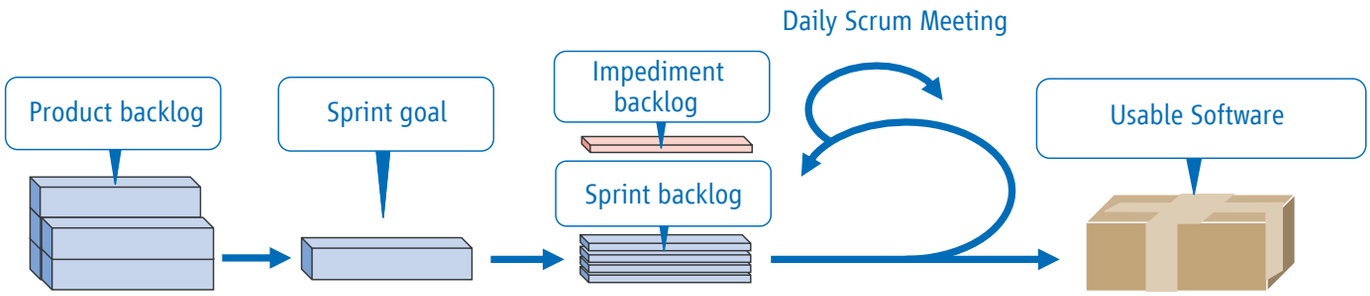


Figure 5: Scrum Workflow (source: Elektrobit Automotive)

3.1 Scrum Workflow

Scrum is one of the most widely applied agile methods. Scrum describes an iterative, incremental development process. The Scrum process is illustrated below in Figure 5.

In the initial step, the role of the product owner lists a priority of user stories, which are coherent requirements for the overall software product. This set of user stories is referred to as the product backlog. All following goals and tasks are pulled from the product backlog. A subset of these user stories is pulled in each iteration, referred to as a sprint. For each sprint, usually with a fixed duration of one to four weeks, a sprint goal is defined during the sprint planning. The subset of user stories during each sprint is called the sprint backlog. The development team should meet the requirements set by the sprint goal and ideally deliver a usable software product at the end of each sprint. The development team is continuously assisted by the Scrum master, who manages the process and eliminates all occurring obstacles along the way. Daily Scrum meetings are held, primarily for the development team to interchange and discuss what has been done so far, what is planned until the next daily Scrum meeting and what impediments they have come across. Daily Scrum meetings also give the product owner the possibility to stay up to date and answer questions when required.

Figure 6 shows the iterative Scrum development management process. Each sprint concludes with a sprint review, where the accomplished user stories are demonstrated. The development team receives feedback from the product owner and all present stakeholders. During the subsequent Scrum retrospective, the development team, assisted by the Scrum master, reflects upon the achievements of the sprint and discusses commitments to improve the next sprint. A new sprint follows, by pulling new user stories from the product backlog. This cycle continues until the final software product can be delivered.

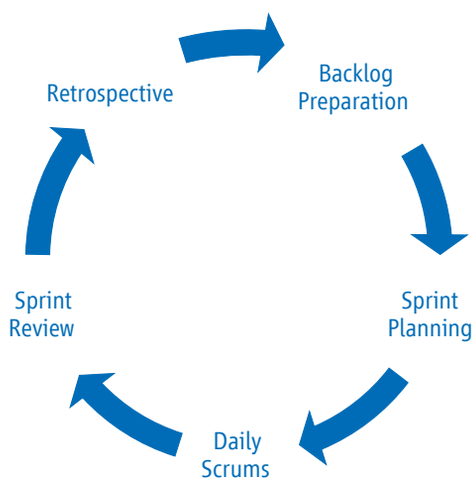


Figure 6: The Iterative Process of Scrum (source: Elektrobit Automotive)

3.2 ISO 26262 Safety Life Cycle

ISO 26262 describes a safety life cycle of automotive E/E systems. The safety life cycle consists of three phases, from the product idea to its final decommissioning, in which all relevant safety activities are embedded in. The three individual phases are divided in Figure 7 and are classified as “Concept phase”, “Product development” and “After the release for production”. All phases of ISO 26262 require defining persons, departments and organizations responsible for the individual safety activities and the confirmation that the items under considerations are developed in accordance with ISO 26262. The concept phase initiates the safety life cycle. The objective of the concept phase is to define and describe the functionality of the item, its dependencies and interactions with the environment and other items. Adequate understanding of the item functionality then supports the completion of activities in subsequent phases. Central to the concept phase is the hazard analysis and risk assessment (HARA). The HARA is a method to identify and categorize potential vehicle-level hazards due to malfunctioning behavior of the item and formulate safety goals to prevent or mitigate the hazardous events.

2-5 Overall safety management

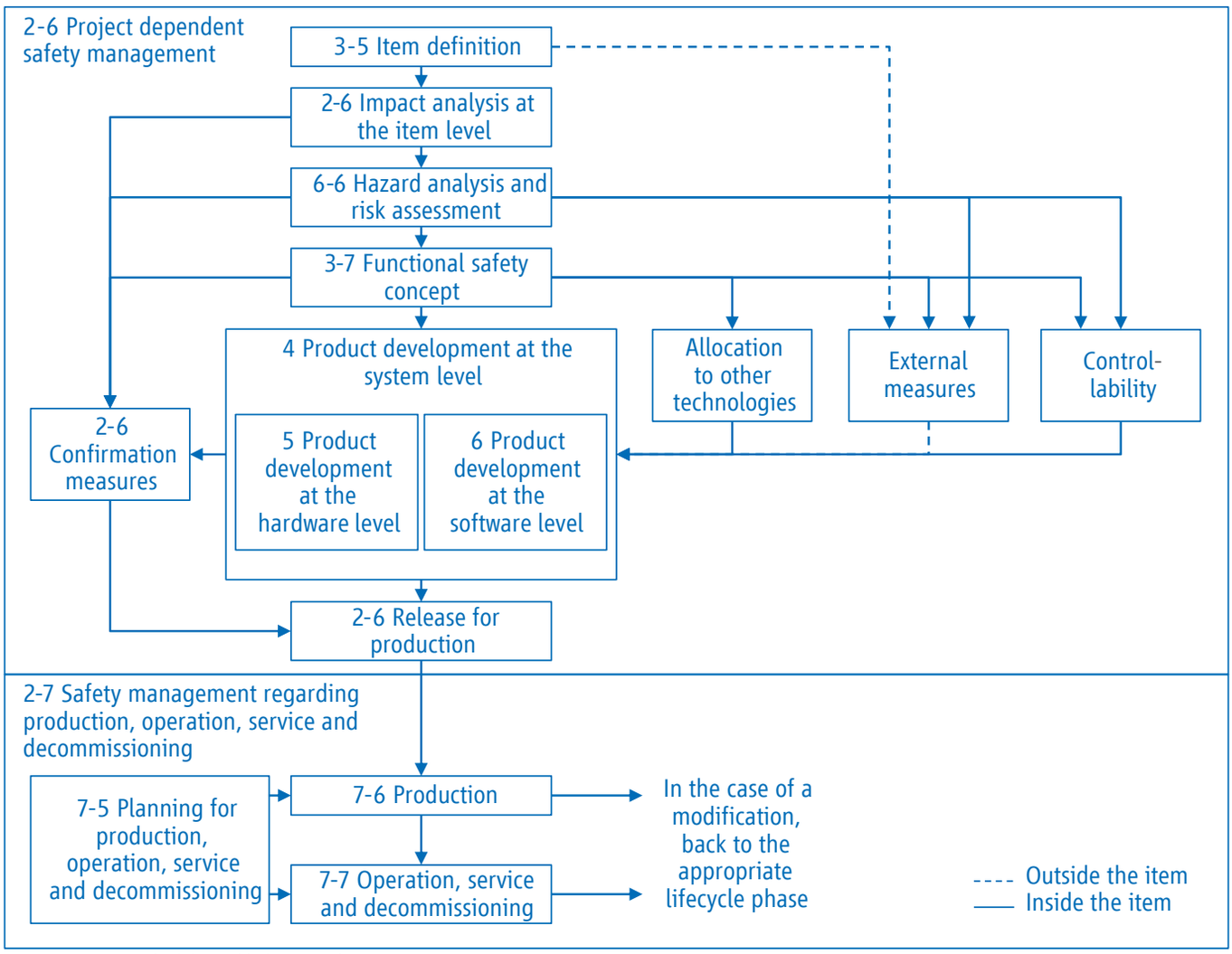


Figure 7: Source ISO 26262 Lifecycle (source: Elektrobit Automotive)

The product development phase of the safety life cycle offers guidance on the product development and the corresponding functional safety activities at the system level, the hardware level, and the software level.

Functional safety confirmation measures are performed to provide additional evidence for the achievement of functional safety by the item and its elements during the development phase.

3.3 Challenges

Scrum and Scrum sprints with a fixed duration of one to four weeks are based on an iterative approach. At first glance, it may seem that Scrum sprints are not easily matched with the hierarchically modeled structure of ISO 26262. The major objective at the end of each sprint is to deliver working software.

This does not contradict with the requirements of ISO 26262 as long as it can be ensured that each sprint is based on adequately mature input work products needed for the development of safety-related software and that the resulting software is used in a suitable way considering the achieved level of safety maturity. Therefore, no software that is suitable for vehicle testing may be generated in preliminary sprints.

Working increments delivered with every sprint may achieve a certain functionality, however from a safety perspective all increments must also fulfill corresponding safety requirements. Thus besides offering working software, additional effort is required to also achieve the required level of safety with each sprint.

3.4 Solutions

To maintain an organization-wide safety culture, additional safety activities and mechanisms can be included in the product backlog (e.g. definition of the required minimum safety maturity depending on the intended use of the software, such as vehicle testing on proving ground versus testing on public roads). By suitably integrating safety aspects into the backlog and the implemented software, a Scrum-based development management supports adherence to safety requirements. A dynamic sprint backlog also welcomes changing requirements, even in late development phases, unless there is an unreasonable negative impact on safety. From a Scrum perspective, teams must be aware that no unrestricted usable software product may be generated in early sprints during a product development. Therefore, project planning must not be reduced to the planning of each sprint, but must consider the milestones of the entire project.

Additionally, both Scrum and ISO 26262 support a maturity model with corresponding maturity levels. Maturity levels define a degree of completion. By introducing a maturity model, efforts regarding safety can be divided into maturity levels and distributed over multiple sprints. The necessary quality and maturity of work products (e.g. software, documents, and, if applicable, hardware) to meet safety-related due care, can be achieved with suitable "Definition of Done" and the corresponding acceptance criteria (acceptance review).

Detected safety issues are included into the backlog, prioritized and solved in sprints using a risk-based approach. A subset of all safety requirements may be postponed and implemented later depending on factors such as milestones defined by stakeholders (e.g. customer or yourself) or required safety maturity of the software depending on the intended use of the product. It is not necessary to achieve full safety maturity at the end of each sprint.

4 Roles

4.1 Scrum Roles

Scrum defines three roles with distinctive responsibilities, the product owner (PO), the development team (DT), and the Scrum master (SM). Together these roles form the Scrum team (ST).

The product owner is responsible for the product backlog. Based on product functionality, the product owner determines the requirements, formulates, and provides a uniform view of the final product and its intended use. The product owner is responsible for providing and maintaining the product backlog and for periodization of the contained product requirements (e.g. user stories) to maximize satisfaction of all relevant product stakeholders. After each sprint, the product owner may change the priorities and functions, as well as accept or reject the delivered functionalities (e.g. implemented user stories).

The development team is responsible for product implementation by using suitable technologies (e.g. software). A Scrum development team, comprised of 4 to 10 self-organizing members, realizes the final product using an incremental approach. The development team adheres to the tasks set during each sprint backlog to achieve the goal of each development cycle. The development team is responsible for carefully executing the development activities required, to provide the defined product in time and quality considering the state of the art for the technologies used (e.g. state of the art for software engineering). The work results of the development team are demonstrated regularly to the product owner and other relevant stakeholders.

The Scrum master is responsible for Scrum process adherence. The Scrum master assumes a supportive role for the team, by ensuring readiness and productivity of the team. The Scrum master further facilitates close collaboration between roles and functions and coaches the team when necessary. To keep the team goal-oriented and on course, the Scrum master clears up impediments and protects the team from interference on the way.

A definition of Scrum including a more detailed description of Scrum roles/events/artifacts is given by the "Scrum Guide" (https://www.scrum.org/resources/scrum-guide).

4.2 ISO 26262 Roles

ISO 26262 primarily defines the safety manager (SaM) role. The safety manager is responsible for planning and coordinating the functional safety activities in the development phase (i.e. define and maintain the safety plan and monitor the progress of the safety activities against the safety plan). The safety manager ensures adherence to ISO 26262.

For product development at software level the safety activities include:

- Development or refinement of software safety requirements.
- Development and analyses of the software architectural design.
- Development and implementation of the software units.
- Software integration at different integration levels.
- Verification across the development cycle.
- Activities to ensure confidence in used software tools.
- Applicable functional safety confirmation measures.

4.3 Challenges

Merging safety aspects with Scrum faces two challenges. The first is to ensure that the Scrum team is adequately aware of the safety-relevant due care for each Scrum role. Secondly, the question arises how the role of the safety manager (or its tasks) can be addressed in Scrum.

4.4 Solutions

Handling of Roles

For safety-related due care, each Scrum role must additionally cover the following tasks:

- Product owner:
  - Ensures that the product backlog also contains the required safety-related content (e.g. technical safety requirements allocated to software or corresponding DIA elements).
  - Requests achievement of functional safety for the developed product in accordance with ISO 26262 and the applicable state of the art for the subject matter (e.g. vehicle domain).
  - Provides the required infrastructure and resources for a safety-related development.
  - May also be assigned the role of the safety manager.

- Development team:
  - Considers higher effort for the development of safety-related products in its effort estimation (e.g. additional effort for implementing safety measures).
  - May also include a dedicated safety manager as development team member.
  - Performs the development with the required due care (e.g. considering the DIA and requirements of ISO 26262-2 or ISO 26262-6 for the development of safety-related software).
  - Ensures that the applied development processes, methods and tools or design and coding guidelines are suitable (e.g. considering the requirements of ISO 26262-6, ISO 26262-8, ISO 26262-9).
  - Informs the safety manager and product owner about the achievement of functional safety for each sprint based on the corresponding objectives (e.g. set of safety requirements to be implemented that are required for the intended use of the generated product version).
- Scrum master:
  - Supports adherence to safety-related due care (e.g. considering the DIA and requirements of ISO 26262-2 or ISO 26262-6 for the development of safety-related software).
  - Ensures the timely provisioning of the required infrastructure and resources for a safety-related development.
  - May also be assigned the role of the safety manager.

The safety manager must also reflect suitable agile practices when defining the safety plan. These are:

- Supplement the "Definition of Done" with safety-related content.
- Support the product owner in the creation of safety-related backlog.
- Coordination with other stakeholders (e.g. with OEM or suppliers according to DIA).
- Coordination of the required additional functional safety confirmation measures.

Combining Roles

There are multiple different ways to combine the ISO 26262 roles with the Scrum roles. The combination is largely affected by how the role of the safety manager is handled. It is important to note that the safety manager does not have to be organizationally independent from the team.

Figure 9 shows a Scrum team in which the role of the safety manager is done part-time. The part-time role of the safety manager can be assumed by either combining it with the product owner, Scrum master, or a member of the development team.

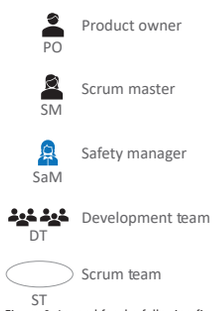


Figure 8: Legend for the following figures (source: Elektrobit Automotive)

Figure 10 illustrates a larger Scrum project. A full-time safety manager is part of the Scrum team.

A constellation of multiple Scrum teams is shown in Figure 11. One full-time safety manager is supporting multiple Scrum teams. Within each Scrum team, the relevant members must have suitable safety expertise.

Small team/complexity

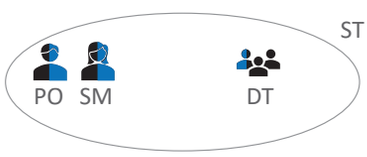


Figure 9: Small project/complexity (source: Elektrobit Automotive)

Larger team/complexity

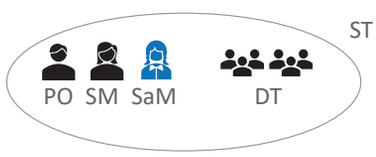


Figure 10: Larger project/complexity (source: Elektrobit Automotive)

Multiple teams

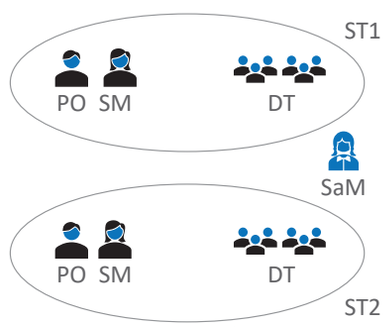


Figure 11: Project with multiple teams (source: Elektrobit Automotive)

5 Artifacts

5.1 Scrum Artifacts

- Product backlog
- Sprint backlog
- Task board
- Sprint result
- Definition of Done (DoD)

5.2 ISO 26262 Artifacts

Depending on the scope of the development and the project setup multiple instances of work products/ artifacts may be needed (e.g. for application software or basic software and/or operating system contained in an ECU).

General Documents, e.g.:

- Safety plan
- Change management plan
- Configuration management plan
- Documentation of the software development environment, e.g. modelling and coding guidelines
- Documentation guidelines
- Documentation management plan

Specifications and Designs, e.g.:

- Software safety requirement specification
- Software architectural design specification
- Software unit design specification
- Configuration or calibration data specification

Implementation, e.g.:

- Software unit implementation (e.g. as source code or binaries)
- Embedded software with calibration data

Analysis at the Software Architectural Level, e.g.:

- Dependent failures analysis report
- Safety analyses report

Verification, e.g.:

- Software verification specification
- Software verification report

Further reports, e.g.:

- Software tool criteria evaluation report
- Software tool qualification report
- Functional safety assessment report for software
- Functional safety Aaudit report for evaluated software development
- Release for production report for the embedded software with calibration data

5.3 Challenges

1. Differentiation necessary between safety-related and non-safety-related content of artifacts (e.g. ASIL attributes for requirements).
2. Additional artifacts for achieving functional safety.
3. Additional planning and tracking of safety artifacts and safety measures.
4. Agile artifacts often don't match traditional types of documents (e.g. word document), and are fine granular information instead (e.g. single requirement or tickets).

5.4 Solutions

1. Tagging of backlog content as safety-relevant (e.g. assigning ASIL to content).
2. Consideration of safety activities in DoD.
3. Employed tools must support the linking of items and offer different perspectives for base lining, versioning and traceability.
4. Generated documents must be feasible and suitable (e.g. for the creation of the safety case).

6 Other Agile Practices

6.1 Refactoring

Abstract

Refactoring is the process of restructuring existing code without changing its external behavior. Refactoring is intended to improve nonfunctional attributes of the software. Advantages include improved code readability and reduced complexity; these can improve source-code maintainability and create a more expressive internal architecture or object model to improve extensibility [source: https://en.wikipedia.org/wiki/Code_refactoring].

Assessment of the Agile Practice from a Functional Safety Perspective

1. Refactoring also offers benefits regarding the reduction of safety risks in safety-related projects through:
   - Easier maintenance ➡ supports avoidance of systematic faults.
   - Higher performance ➡ supports fulfillment of safety timing constraints.
   - Less prone to errors ➡ overarching goal of safety.
   - Lower degree of complexity ➡ supports avoidance of systematic faults and simplifies testability.
2. As part of the iterations, reviews of software architecture, detailed design, and code should be performed regularly.
3. Refactoring should be a deliberate, documented team decision.
4. Basis for decisions must be:
   - Analysis of the impact of the intended refactoring regarding safety.
   - Project-Risk-Analysis:
     In a safety context more effort regarding documentation, testing, etc. is required in comparison with QM-software. Automated tests with high coverage help limit the additional work. The earlier refactoring is employed in a project, the smaller is the resulting effort, as safety requirements do not have to be met during early product phases ("Fit for purpose").
5. In case refactoring increases the safety risks, refactoring must not be performed.
   Example: Due to time constraints, sufficient verification and validation after the refactoring cannot be ensured:
   - Refactoring and incremental reviews do not replace complete reviews.

Reference: [9]

6.2 Pair Programming

Abstract

Pair programming is an agile software development technique in which two programmers work together at one workstation. One, the driver, writes code while the other, the observer or navigator, reviews each line of code as it is typed in. The two programmers switch roles frequently.

Assessment of the Agile Practice from a Functional Safety Perspective

- The development of errors is prevented early, and the development of intelligible and efficient code is supported.
- While reviewing, the observer also considers the "strategic" direction of the work, coming up with ideas for improvements and likely future problems to address. This is intended to ensure that the driver stays focused on the "tactical" aspects of completing the current task, using the observer as a safety net and guide.
- Pair programming may replace or reduce the need for reviews during development.
- The suitability of pair programming must be considered when defining the verification approach in accordance with ISO 26262.

References:

- [8], chapter 16
- [3], Part 6: Table 7, Methods for software verification

6.3 Continuous Integration

Abstract

Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.

By integrating code regularly, you can detect errors quickly, and locate them more easily [source: https://en.wikipedia.org/wiki/Continuous_integration#cite_note-:0-1 resp. https://www.thoughtworks.com/continuous-integration].

Assessment of the Agile Practice from a Functional Safety Perspective

- The tool chain used for continuous integration must be evaluated regarding tool confidence level as well as qualified if applicable.
- Tool chain users must be sufficiently qualified.
- Software from continuous integration may not be directly released for production, additional measures are necessary.
- Results from continuous integration can be given earlier into test (e.g. HIL/SIL test).
- Test during development, not at the end of a project => Under time pressure (e.g. when close to SOP), it cannot happen that tests are reduced to save time.
- Continuous integration improves development efficiency.
- Automated testing as part of Continuous Integration supports ISO 26262 as long as suitable tools are used.

Reference: [8]

6.4 User Stories

Abstract

Epics and User Stories as rough functional descriptions (high level requirements) are helpful for early and iterative planning and negotiation. Detailed requirements have to be defined additionally.

Assessment of the Agile Practice from a Functional Safety Perspective

- At least at the end of a sprint, detailed requirements must be documented and linked with test cases and implementation.
- With respect to requirements, safety-related agile development does not differ basically from agile automotive development with ASPICE.

Reference: [8]

7 Recommendations for Audits/Assessments

Involve auditors/assessors early and carry out assessments alongside the project, especially if it´s the first agile safety-related project in the company. It is safe to assume that auditors and assessors have little understanding of agile practices, as these practices are not established widely.

A good time for a first feedback from an auditor/assessor could be when the agile safety-related process has been defined. Even though agile development strives for early feedback, feedback must not be reduced to functionality that is experienceable and from the end user only.

Development in short iterations has the big advantage, that the whole process can be assessed at any time of the project, as each iteration uses the whole process. In projects using a sequential process or long iterations (e.g. sample phases), an assessment during a project can only assess the process steps that were used up to this time (e.g. in the implementation phase, the validation process can only be assessed theoretically).

8 References to other Documents

1. Agile Principles and ISO 26262 (ZVEI): https://agile.zvei.org/agilitaet-in-branchen/agile-in-automotive/
2. Agile in Automotive: Pocket Guide Scrum (Kugler Maag)
3. ISO 26262:2018
4. VDA Guideline "Agile Collaboration"
5. https://www.scrum.org/resources/scrum-guide
6. https://www.scaledagileframework.com/
7. https://less.works/
8. Kent Beck: Extreme Programming
9. Martin Fowler: Refactoring
10. Martin Fowler: Continuous Integration
11. Agile Alliance - Agile Glossary: https://www.agilealliance.org/agile101/agile-glossary/

9 Participating Companies

Elektrobit Automotive GmbH

Endress+Hauser SE+Co. KG

Hella GmbH & Co. KGaA

Infoteam Software GmbH

Innoventis GmbH

Kugler Maag CIE GmbH

Robert Bosch GmbH

ZVEI - German Electrical and Electronic
Manufacturers' Association
Lyoner Strasse 9
60528 Frankfurt am Main, Germany

Phone: +49 69 6302-0
Fax: +49 69 6302-317
E-mail: zvei@zvei.org
www.zvei.org