Combined application of agile practices and functional safety in automotive software development

How Scrum and other agile practices can support ISO 26262 compliance

Steffen Kuhn, Elektrobit Automotive GmbH, Erlangen

Abstract

Despite some common misconceptions, agile practices can be used when developing safety-related automotive embedded software. In a working group of the ZVEI (Zentralverband Elektrotechnik und Elektronikindustrie), multiple companies exchanged their experiences especially with Scrum (as one prominent example for agile methods) and ISO 26262 (the automotive functional safety standard) and came to the following conclusions:

1. Agile methods like Scrum can be combined with functional safety.
2. It is beneficial to use agile methods like Scrum in complex safety-related developments.
3. In safety-related development, agile teams must consider additional activities and process requirements.
4. The role of the safety manager can be integrated into agile teams.
5. The agile practices refactoring, pair programming and continuous integration are recommended for use in safety-related development

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) are not suitable for handling such a complexity. Agile principles can help to reduce complexity of organization and planning for software development, which is especially important in a functional safety context, because high complexity increases the risk of errors.

Clarification of possible misunderstandings regarding “Agile Software Development“

Misunderstanding: “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.

Misunderstanding: „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.

Misunderstanding: „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.

Misunderstanding: „Agile methods cannot be applied in the automotive industry.“
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.

Misunderstanding: „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.
Interrelationship between Scrum, ASPICE 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 stakeholders. Scrum can support meeting ASPICE requirements regarding project management and therefore also assists in meeting corresponding requirements in ISO 26262.

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.

Scrum workflow and ISO 26262

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 modelled 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 fulfil corresponding safety
requirements. Thus, besides offering working software, additional effort is required to also achieve the required level of safety with each sprint.

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.

**Roles in Scrum and ISO 26262**

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.

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 reflect suitable agile practices when defining the safety plan:
• 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.

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.

**Scrum artifacts and ISO 26262**

The following challenges regarding artifacts of Scrum and ISO 26262 have been discussed:

• Necessary differentiation between safety-related and non-safety-related content of artifacts (e. g. ASIL attributes for requirements).
• Additional artifacts for achieving functional safety.
• Additional planning and tracking of safety artifacts and safety measures.
• 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).

**Solutions:**

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

**Additional agile practices**

Multiple agile practices can be combined with Scrum, such as

• Refactoring
• Pair Programming
• Continuous Integration
Refactoring is the process of restructuring existing code without changing its external behavior. Refactoring is intended to improve non-functional 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]. 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 fulfilment of safety timing constraints.
- Less prone to errors => overarching goal of safety.
- Lower degree of complexity => supports avoidance of systematic faults and simplifies testability.

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. Advantages in a functional safety context:

- 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.

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]. Functional safety related benefits:

- 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.
Automated testing as part of Continuous Integration supports ISO 26262 as long as suitable tools are used.

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).

References