Chapter 25
Functional Safety and Evolvable Architectures for Autonomy

Rolf Johansson, Jonas Nilsson, Carl Bergenhem, Sagar Behere,
Jörgen Tryggvesson, Stig Ursing, Andreas Söderberg, Martin Törngren, and
Fredrik Warg

25.1 Introduction

Since October 2013, there is a national-funded research project running in Sweden
called FUSE (FUncional Safety and Evolvable architectures for autonomy). The
motivating reason for the project is that tomorrow’s vehicles should be capable to
drive autonomously and that there is a number of questions to solve before this can
happen in a safe and efficient manner. To a certain extent, this project contributes
to the Drive Me project described separately in this book. However, FUSE has a
longer time horizon and aims also to find solutions to vehicles with higher degrees
of automation than Drive Me.

The FUSE project in particular focuses on system architectures and functional
safety for autonomy. Current automotive systems and functional safety standards are
evolving but have so far not considered autonomy. This implies that the limitations
of current systems and the ISO 26262 standard [1] are currently unknown and

R. Johansson (✉) • A. Söderberg • F. Warg
SP Technical Research Institute of Sweden, Borås, Sweden
e-mail: rolf.johansson@sp.se

J. Nilsson
Volvo Car Corporation, Gothenburg, Sweden

C. Bergenhem
Qamcom Research and Technology, Gothenburg, Sweden

S. Behere • M. Törngren
KTH Royal Institute of Technology, Stockholm, Germany

J. Tryggvesson
Comentor, Gothenburg, Sweden

S. Ursing
Semcon, Gothenburg, Sweden

© Springer International Publishing Switzerland 2017
D. Watzenig, M. Horn (eds.), Automated Driving,
DOI 10.1007/978-3-319-31895-0_25
that investigations are needed regarding functional safety considerations and the scalability and cost-efficiency of architectures.

Functional safety refers to the ability to deliver services that can justifiably be trusted. In other words, failures that are more frequent and more severe than acceptable have to be avoided. While autonomy if realized correctly will improve safety, it also introduces autonomous control of the vehicle motion and thus new failure modes. The increasing complexity of the underlying embedded control systems requires new methods and architectures to be able to achieve cost-efficient and functionally safe autonomy.

Current technologies such as adaptive cruise control (ACC) always require the driver to supervise its operation. So in case something unexpected occurs, e.g., radar blockage or an animal crossing the road, then the driver has to take over control. Also in case an unlikely technical fault such as loss of braking pressure or steering assistance occurs, the driver is expected to take control. When a vehicle is driven autonomously, the driver cannot be expected to do this anymore, so alternative solutions have to be implemented and verified.

Figure 25.1 depicts three dimensions of autonomy and illustrates a challenge that is investigated in the FUSE project. The dotted line in the figure separates the left side, where there is a common understanding about functional safety, from the right side, where this understanding does not exist. In particular, there are three aspects needed to be addressed:

![Three Dimensions of Autonomy](image)

**Fig. 25.1** Autonomy dimensions
• How to define functional safety for autonomy: Not addressed by ISO 26262.
• How to achieve functional safety: There is a demand for architectural patterns and division of responsibility among parties.
• How to prove functional safety: There is a demand for new compositional safety arguing.

This project is confined to autonomy within one vehicle and the environment in which it is operated. Cooperative driving functions such as platooning are outside scope. Still the three aspects are all relevant and need to be addressed in the project.

The development of dependable, safe, and autonomous vehicles requires the integration of research results from multiple domains, including embedded systems design, component- and model-based systems engineering, robotics, and artificial intelligence (AI). Many technologies that exist in advanced modern vehicles today borrow their concepts from the robotics and AI domains. However, a similar migration in systems architecture has not happened, despite the development of a wide variety of architectures within both the robotics and AI domains. There is a need to extract relevant principles from these domains and apply them to the automotive domain, keeping in mind the specific constraints arising from standards, regulations, and legislative mechanisms. The patterns and principles for autonomous automotive architectures can be consolidated in the form of function specific as well as overall vehicle reference architectures. Such reference architectures are missing in current states of the art.

In the following sections, some of the challenges addressed until mid 2015 in the FUSE project are summarized.

25.2 Why It Is Harder to Show Functional Safety for Autonomous Vehicles

Generally speaking, there are two kinds of consequences when changing from a vehicle where the driver is responsible to where the vehicle is supposed to take all necessary actions autonomously. Firstly, some things may become much more difficult and complex, and, secondly, they may become significantly different. When it comes to functional safety and road vehicles, we have identified both kinds of problems. The below problems have been presented and elaborated in [2–5].

25.2.1 Item Definition

The starting point of the ISO 26262 reference life cycle is the item definition, i.e., the confinement of the functional scope on which to apply functional safety. Traditionally, it is enough to consider one item at the time, and it is of no interest to look at the entire vehicle in an aggregated way. The underlying assumption is
that whatever functionality is needed for the entire vehicle to behave safely is either performed by one of the items or by the driver. An implicit responsibility of the driver then becomes to complement the functionality of all items. As long as the items one by one are defined and assessed as safe, it is reasonable to leave to the driver to use a strategy for the driving where all unforeseen situations can be handled in a sufficiently safe way. The role for the ISO 26262 is then confined to the assessment of each individual item.

When reaching so high in the degree of automation that we no longer require the driver to be (quickly) responsive to the situations, this will imply that the responsibility to cover everything falling between the explicit items needs to be transferred from the manual driver to an autopilot. This means that all of a sudden, it has become an issue to make sure that the set of items is complete. This is because the autopilot also has to be covered by the set of items.

The item definition has both become much more complex and completely different in nature. The complexity increase is due to the much more complex task to solve. What makes the item definition completely different is the partly implicit definition. By implicit we mean that regardless of how we state the explicit item definition, it has to take care of a number of rare situations. We can no longer take the item definition as given but need to assure whether there is a need for more functionality in its scope, in order to make the behavior of the vehicle safe.

In the following section, we address the problem how to perform hazard analysis and risk assessment given that we cannot assume a complete and explicit item definition.

25.2.2 The Role of the Driver

When determining the ASIL attribute of a safety goal, we use as an input the attributes of the covered hazardous events (HEs): severity, exposure, and controllability. It is more or less obvious that we will see a change of the controllability factor when we change the role of the driver dramatically. The basic question is who is in control of the vehicle? If we only consider fully autonomous vehicles when there is no room for any person to interfere with the driving, we can simply conclude that all controllability factors should be C3, i.e., no possibility for a driver to compensate for failures. The problem is how to look at the situations when we on one hand are so high in automation that the driver is not requested to be (quickly) responsive and on the other hand we allow the driver to compensate for errors in the automation.

In traditional evaluation of the C-factor, the underlying assumption is that the activity of the driver can (more or less) compensate for problems caused by the electronic/electrical (E/E) implemented functionality. In higher degrees of automation, it might be hard for the driver to understand what a safe an efficient way to control the vehicle is. Even if the vehicle itself is doing perfectly well, the driver may think it is faulty and needs a compensating action. The consequence of
the intervention of the manual driver might be that an accident is caused instead of avoided. Our conclusion is that once we introduce higher degrees of automation, it will no longer be valid to use anything but C3 and we should consequently not allow the driver to “compensate” for the automatic vehicle actions, unless judged as safe by the autopilot.

Another issue with the role of the driver is to always ensure a consistent view between the manual driver and the vehicle, regarding who is currently responsible to react (on unforeseen events). This problem is also known as mode confusion. There are two main possible categories of confusions: either both of them think they are responsible or none. If both the manual driver and the vehicle autopilot think they are the one responsible for taking full responsibility of the driving including handle unforeseen situations, we just have to select whom to give the last word. As we argue above, it is reasonable for lower degrees of automation to give the manual driver the possibility to override the vehicle-induced actions. Then the C-factor analysis is still relevant. For the higher levels of automation, we let the vehicle get the full control regardless what the driver tries to do.

The other case is trickier; when neither the manual driver nor the vehicle regards themselves as the finally responsible part. This kind of “underride” is foreseen to have a similar solution. The HMI should be designed in such a way that the manual driver stays responsible for the vehicle until there is a very explicit handover to the automatic system. Then the vehicle stays responsible until again there is a very explicit acknowledgment from the driver that she has accepted to get the responsibility back. The implication of this becomes that once a vehicle has got the responsibility for the driving, it needs to be capable to safely handle any situation without assistance of the driver for any decision or maneuver.

### 25.3 How to Perform Hazard Analysis and Risk Assessment

As concluded in the previous section, it will become more or less impossible to use a set of well-defined explicit items as the starting point for the ISO26262 activities when it comes to highly automated vehicles. From the FUSE project, we are proposing a slight modification of the early stages of the ISO 26262 reference life cycle. Instead of a firm item definition, we introduce a more loosely defined preliminary feature description (PFD) to act as input to the first step in an iterative process. This is the starting point of an iterative process as depicted in Fig. 25.2 below. In the following, we briefly introduce the steps.

#### 25.3.1 Preliminary Feature Description

The difference between the preliminary feature description, PFD, and an item definition is that it does not need to contain the details required by an item definition
Fig. 25.2 Iterative process to conclude concept phase

according to ISO 26262. Rather, the aim is to describe the expected (customer) benefit of the proposed feature. From this starting point, the feature is iteratively refined until a point where it is possible to enunciate an item definition.

The key point of the PFD is to give a first workable description of what the proposed feature is supposed to do. The material could, for instance, be based on input from market research, previous projects, or whatever else might be available and deemed relevant to define the nominal function. It is an open question to which extent the quality and size of this material will affect the remaining steps in the process. On the one hand, the scope of the feature must become clearly defined during the iterative refinement process regardless of the initial input. On the other hand, it seems reasonable that more relevant input will make the rest of the process easier. A few possibilities for information in a PFD could be:

- A collection of a few use cases (high detail)
- A collection of a larger number of user stories (low detail)
- Very simple feature description

A use case describes a product use scenario with relatively high detail. It is described as a main scenario, possibly branching out in a number of alternate scenarios (variants of the main scenario). This content makes a use case, or a
collection of use cases (depending on the complexity of the proposed feature), a promising candidate for information in a PFD. Since use cases have long been a popular tool, it might have the additional advantage of already being used in existing product research and development processes, which would make adoption of this format straightforward.

User stories are typically favored in agile development processes. The distinguishing features compared to a use case is that stories are much more lightweight and therefore quicker to develop. Instead, they need to be elaborated by the developers at some later development stage in order to become implementable.

We hypothesize that these features make user stories even better material for a PFD than the use case. The lightweight format makes it feasible to spend effort on creating a larger number of stories compared to use cases. In the context of autonomous vehicles, where the exact scope of a feature is not known at this point in development, this larger number of stories can aid in covering a larger part of the problem space compared to the use cases. This is illustrated in Fig. 25.3 where the user stories are smaller (i.e., less detailed) but has a better coverage of the space containing hazards relevant for autonomous driving.

Note that neither user stories nor use cases in any way guarantee coverage of the relevant hazards or provide any tools for this. Therefore, describing the function with user stories has the potential of providing a better starting point for hazard analysis that will help in the subsequent steps. But it will not in itself provide any proof of coverage.

**Fig. 25.3** Comparison of use case and user story format to describe a function
25.3.2 Situation Analysis and Hazard Identification

Situation analysis and hazard identification has a slightly different purpose when starting with a preliminary feature description compared with the traditional process where an item is already defined. It becomes part of an iterative process (steps 2–4) where the intended functionality and hazardous events are jointly elicited by stepwise refinement.

Compared to conventional analysis, there are two additional challenges: the function is not yet clearly defined, and for autonomous vehicles, the hazard analysis must assume the function is self-sufficient, i.e., does not rely on a driver as backup.

In order to start mapping the problem space, the idea is to use trees of generic operational situations and hazards as a starting point. For the first iteration, select a subset of situations from the table of generic operational situations and a subset of hazards from the table of generic hazards. The trees are organized after classes of factors, and in several levels where each sublevel has more detail than the previous one, making it possible to select the level of detail deemed necessary for each factor. This initial selection is based on what is reasonable to believe will affect the function based on the PFD. The operational situations and hazards are then combined to make up an initial set of hazardous events before moving on to step 3.

For each iteration, the situations and hazards can be made more detailed, where this is useful. That is, as the function becomes more clearly defined, it should also become more clear which situations need to be investigated in more detail.

Guiding factors for finding new or more detailed situations and hazards in order to advance the refinement process:

1. If it is hard to determine whether a previously defined HE falls within the scope of the function or not (see step 4), it needs elaboration to remove the uncertainty.
2. If the aspects of the function are not yet defined clearly enough to be able to write an item definition, these aspects need more analysis (see step 4).
3. Situations or hazards identified as potentially relevant in the previous iteration but left unused (i.e., did not become part of a HE) might indicate an omission. Otherwise, there should be a rationale why it was discarded.
4. Parts of the generic trees that have not yet been used should be revisited to see if there are any new situations or hazards that have become relevant.
5. Use rules for dominance and non-dominance of HEs (see step 3) to find potential new HE candidates.

For each iteration, new situations, hazards, and hazardous events are created in this manner. It should be stressed that the generic situation and hazard lists are a starting point but not an exhaustive list for all items and all contexts. As the function description matures and more known details about the context (vehicle) are taken into account, the generic lists have to be supplemented with context-aware specializations and additions, for instance, impact of performance characteristics of the target vehicle.
25.3.3 Find Dimensioning Hazardous Events

In order to keep the number of hazardous events manageable, HEs that will not contribute to the list of unique safety goals can be culled from the list. First, classification of hazardous events is performed according to ISO 26262, 3–7. During classification, each HE is assigned values for the exposure (E), controllability (C), and severity (S) factors, as well as a resulting integrity level (ASIL). Then, the list is processed using the rules for identification of dominance and non-dominance. The method is fully described in [5].

25.3.4 Function Refinement

The list of hazardous events becomes supporting material when trying to find requirements describing the nominal functionality. Normal requirements engineering is used to define the nominal functionality. The HEs from the previous step are considered. Any HE that may fall outside the scope of the function (perhaps because it is already handled by an existing item) is removed from the list of dimensioning HEs but kept as an aid to help define the boundaries of the function. A rationale for why the HE is outside the scope is noted.

After the first iteration, there should be a rough idea of what has to be included, and this can be assembled as a first step toward an item definition. Even if there are still many unknowns, this description can be used as input to the next iteration.

The process is illustrated in Fig. 25.4. After the first iteration, the function is delimited by a number of HEs, but the exact capabilities are still uncertain (illustrated by the fuzzy border separating the HEs within the function and those outside). For each successive iteration, it becomes more and more clear what the capabilities of the function must be.

When no more dimensioning HEs can be found using the guiding principles for finding HEs described in step 2, exit the iteration and continue to step 5. At this point, the function description should also have become detailed enough to write an item definition, and the assembled list of hazardous events will be transformed into safety goals.

25.3.5 Item Definition

The item definition is written according to ISO 26262. If this step uncovers new missing hazards, they shall be added to the hazard analysis and steps 2–4 repeated to include the new hazard.
25.3.6 Safety Goals, FSC, and TSC

From here on, the traditional V-model is followed. Safety goal, functional safety concept, and technical safety concept are constructed according to ISO 26262. In every step, if the refined design reveals any new hazards, the process is repeated from step 2 in order to see if any modifications are needed.

To further strengthen the confidence in coverage of the relevant hazardous events, the completeness can be challenged according to the idea in [4]. That is, to let a review team challenge the list of HEs looking for a candidate that is not covered in the list of safety goals, i.e., to find a new dimensioning HE.

25.3.7 Consequences for Safety Case and Assessment

The record and results of this iterative process can be used to strengthen the completeness argument for the HARA phase and should be especially useful in the context of autonomous functions. While not a formal proof of completeness, it is a systematic approach where it can be shown what issues have been taken into
account in the analysis. The strength of the argument can be further improved by documenting a rationale for key decisions during the process, such as:

- The selection of operational situations and hazards and the level of detail used.
- Which classes of generic situations and hazards are determined not applicable for the function and why?
- If seemingly relevant (i.e., initially singled out), situations or hazards end up not being included in any HE.
- If it is decided that an initially selected hazardous event fall outside the scope of the final item.

The records and rationales from the iterative process, together with the results from the review team, should provide a solid base for a safety case.

25.4 How to Refine Safety Requirements

As mentioned in a previous section, there are things that become much more complicated when going up in degree of automation. In the previous section, we addressed the problem of identifying explicit item definitions and a complete and efficient set of safety goals. Even if these phases are mastered, we still have the problem of refining the safety requirements and allocate them onto the elements of the decided architectures. This problem is closely connected to the one of finding architectures on the different levels of abstraction, which is further addressed in a later section. The FUSE project proposes means for mastering the general problem of verifying that all steps of the safety requirement refinement are complete and correct.

In ISO 26262, there are general requirements that any lower level safety concept (safety requirements allocated onto elements of an architecture) needs to be verified and shown to be complete w.r.t. the higher level safety concept. Having a number of “partial” safety concepts spread among a number of companies, this verification task is certainly challenging. This is especially true for autonomous vehicles where the complexity of the functionalities is significantly higher than for manually driven vehicles. On the lower level, the safety requirements are assumed to be formulated similar to case when having manually driven vehicles. This means that there is a larger span of complexity to handle when the top-level complexity is higher, as for autonomous vehicles. Each step to be handled in the safety requirement breakdown chain is denoted the semantic gap. In Fig. 25.5 below, we depict the semantic gap between a safety goal and the underlying functional safety concept. Furthermore, the backward arrow that denotes refinement verification is depicted. This gap is present and calls for a refinement verification in phases of ISO 26262 refinement of safety requirements. During refinement, rationale, known as satisfaction arguments, shall be collected for the resulting composition, e.g., why is the refinement valid. The argument can contain, e.g., domain knowledge and design patterns. This is essential
in almost every nontrivial refinement. The rationale justifies the “refinement path taken” through the semantic gap and improves traceability.

In the FUSE project, we propose two main means for handling the complexity of the semantic gaps.

Firstly, we propose the possibility to introduce more steps in the safety requirement refinement chain. This is of course not forbidden today, but what we say is that ISO 26262 should explicitly introduce subphases for which the activities start and end on the same level of abstraction. This means that we can directly refer to work products that present the refinement (or the refinement verification) between a functional safety concept (FSC) and another FSC or between a TSC and another TSC. Introducing the formal ability of more steps would enable the lowering of the size of each semantic gap to overcome. In addition to this, we know that already today, for lower degrees of automation, a typical use case for a tier 1 supplier is to receive a TSC from the OEM customer and then break this down to an internal TSC. For both reasons, we need to complement the reference life cycle of ISO 26262 with subphases starting and ending on the same level of abstraction.

The second conclusion for how to safely master the refinement of safety requirements is that we need to be formal in the refinement verification. We argue the importance of having strong evidence, e.g., proof in a formal syntax for the correctness and completeness of every refinement verification. Furthermore, the assumptions and the domain knowledge acting as satisfaction arguments need to be explicit enough to serve as a part of this formalism. Large semantic gaps imply complex satisfaction arguments, which are hard to use in a convincing proof. These questions are further elaborated in [6].

25.5 What Functional Architectures Fit Autonomous Driving

When going up in degree of automation, we can regard this as the manual driver becoming more and more replaced by an autopilot. In the previous section, we addressed some of the responsibilities in terms of functional safety, which are
transferred from the manual driver to the autopilot. If this was the question “what does the autopilot do?,” in this section we address the question: “where does the autopilot hide?” As observed before, the functional safety issues are very much connected to the architectural ones, as the semantic of each safety requirement is partly dependent on the allocation on an architectural element. Beside this problem of enabling functional safety, there are a number of additional issues to cover when identifying what E/E architectures are efficient for autonomous vehicles.

Evaluation and refinement of a functional architecture for autonomous driving has occurred under the aegis of the FUSE project. The reference architecture includes (1) a categorization and description of the key functional components needed for autonomous driving, (2) rationale for the distribution of these functional components across the architecture, and (3) a three-layer architecture incorporating the described components. A comparison of the reference architecture with three representative architectures for autonomous driving has also been made, with a view of highlighting the similarities and differences.

The key functional components of the architecture are divided into three categories, (1) perception, (2) decision and control, and (3) vehicle platform manipulation. It is proposed that the perception and decision and control categories should be grouped into a “cognitive driving intelligence” layer of the reference architecture, whereas the vehicle platform manipulation category components should be included in a distinct “vehicle platform” layer. Such a split acknowledges the need for reusability of the architecture across different vehicle platforms (product portfolios), as well as the characteristics and development practices of the various research domains involved. The architecture also incorporates a layer for tele-operation or remote monitoring and management of autonomous vehicles. The identified functional architectural components have a substantial overlap with those found in other successful autonomous driving architectures. However, components for world modeling and semantic understanding are unique to the proposed architecture. The same holds true for an explicit component for abstracting the vehicle platform. The reference architecture has been applied to different vehicle categories and provides freedom to the component developers to test and deploy new algorithms while localizing the effect of any needed changes.

Details of the architecture are provided in chapter “Systems Engineering and Architecting for Intelligent Autonomous Systems.” These questions are further elaborated in [7].

25.6 Conclusion

This paper presents the ongoing activities in the Swedish national-funded project FUSE. This project address a few of the questions needed to be addressed in the area of functional safety and E/E architectures, in order to enable autonomous vehicles. We conclude that the international standard for functional safety for road vehicles ISO 26262 needs to be updated in order to cover autonomous vehicles, and we give
some suggestions to improvement. We also conclude that the issues of functional safety and E/E architectures are highly interconnected especially on the higher levels of abstraction, and we give recommendations for a functional architecture pattern suitable for highly automated vehicles. The project is ongoing, but some of the results discussed here have already been published. More details of the proposed solutions have been presented at a number of conferences as listed in the section of references [2–8] below.

Acknowledgment This work has been financed by the Swedish government agency for innovation systems VINNOVA in the FUSE project (ref 2013-02650). The participating companies in the project are Volvo Car Corporation, SP Technical Research Institute of Sweden, Semcon, Qamcom Research and Technology, KTH Royal Institute of Technology, and Comentor.

References

1. ISO, International Standard 26262 Road Vehicles—Functional Safety, ed, 2011
2. C. Bergenhem, R. Johansson, H. Sivencrona, Challenges in Applying Functional Safety for ADAS. Presented at SAE International Congress, 2014
3. R. Johansson, Automated Functions and Autonomous Vehicles. Presented at IQPC 5th International Conference ISO26262, 2015
4. R. Johansson, The Importance of Active Choices in Hazard Analysis and Risk Assessment. Presented at Third Workshop on Critical Automotive applications Robustness & Safety (CARS), 2015
5. R. Johansson, Efficient Identification of Safety Goals in the Automotive E/E Domain, Presented at ERTS², 2016
6. C. Bergenhem, R. Johansson, A. Söderberg, J. Nilsson, J. Tryggvesson, M. Törngren, S. Ursing, How to Reach Complete Safety Requirement Refinement for Autonomous Vehicles. Presented at Third Workshop on Critical Automotive applications Robustness & Safety (CARS), 2015
7. S. Behere, M. Törngren, A Functional Architecture for Autonomous Driving. Presented at the First International Workshop on Automotive Software Architecture (WASA ’15), 2015
8. S. Behere, F. Asplund, A. Söderberg, M. Törngren, Architecture Challenges for Intelligent Autonomous Machines: An Industrial Perspective. Presented at The 13th International Conference on Intelligent Autonomous Systems (IAS-13), 2014