Developing SEooC -Original Concepts and Implications when Extending to ADS

Rolf Johansson, Håkan Sivencrona

Developing SEooC – Original Concepts and Implications when Extending to ADS

Rolf Johansson
Astus AB
Mölndal, Sweden
rolf@astus.se

Håkan Sivencrona
Zenseact AB
Göteborg, Sweden
hakan.sivencrona@zenseact.com

***Abstract*— Reference life-cycle models as prescribed in safety standards shall never be interpreted as a timeline, but as depicting dependencies. In this paper we describe how we many years ago made this explicit to the ISO 26262 community, by introducing the concept of safety element out of context (SEooC). As the term then has become widely used, and sometimes filled with another meaning than what is the intention, this paper elaborates what is necessary to constitute an SEooC, emphasising the importance of of semantically unambiguous safety requirements, and reminding about that SEooC never can be used as an excuse not to follow a proper development process, i.e. it is never a "26262 light". SEooC is a form of component-based safety argumentation, and this paper shows how using the SEooC concept as a pattern all over a complex product structure, it fits well in modern agile product development with continuous integration and continuous deployment (CI/CD). Looking into the future, we claim that automated driving can never become successful without a strict formalization of a fine-grained product structure realized by SEooC in every piece.**

Keywords—SEooC, component-based design, safety case, agile development, safety contracts, continuous assessment, automated driving systems

I. Introduction

During the authoring of the first version of ISO 26262 [1], it became more and more evident for us that there were different understandings among the different experts what would be the full connotation of the reference life cycle of that standard. At the time, we were working for (competing) tier2 suppliers and it became clear that the international expert committee was dominated by OEMs and Tier1 suppliers having the full "Item" in their business offer. To ensure that the component perspective was visible in the safety argumentation, and especially the situation that a supplier tries to be ahead of its customer, we introduced the concept of Safety-Element-out-of-Context (SEooC). In short this means that you describe how to perform the life cycle activities of the full reference life cycle (the V model) out of order, still guaranteeing that all dependencies between these are met.

To clarify, SEooC was never intended to be a "26262 light", rather it is by all aspects as strict when it comes to complying with all normative requirements in the standard. Throughout the years we have met many different references to the SEooC term, and not all of these have been in-line with the original intent. This paper has partly the goal to more clearly describe what SEooC actually means, and it also explains how this can be used in future challenging contexts.

The paper is organized as follows: Section II gives a background about the ISO 26262 reference life cycle and its relation to the safety argumentation. In section III we give the background to SEooC, and explain some of the fundamentals of the concept. In section IV SEooC is compared to other concepts, which sometimes might have been mixed up with SEooC. Then in section V, we describe some challenges that come with the very big software intensive automotive products of today, and we elaborate how SEooC can become an advantage in the modern world of agile development with CI/CD and the autonomous features of tomorrow. Then follows an analysis of the possibilities that can follow if the pattern of SEooC is implemented to its extreme, opening up for a formalization that can significantly reduce the complexity of generating full safety cases for very complex features. Finally, conclusions are drawn in section VII.

II. ISO 26262 Life cycle and Safey Argumentation

The reference life cycle is often referred to as the ISO 26262 V-model. For each phase of the V, there are prescribed activities generating output ("Work products") based on inputs ("Work products" from previous phases). The inputs contain, among other things, safety requirements, and each safety requirement has an integrity attribute (ASIL value). The prescribed activities in each phase have process requirements that are depending on these integrity attribute values.

The scope of ISO 26262 is to give a framework on how to argue that an automotive functional feature ("Item") is safely implemented. It is however outside the scope of ISO 26262 to argue whether the specified feature is safe as it is specified to the driver, and only deviations from this are inside the scope of functional safety. Whether it is safe to define a certain variant of an automated emergency brake (AEB) feature, is not a functional safety issue, but only if the deployed implementation deviates from the promised one (false negative, false positive, too late, too early, too little, too much etc).

For the ISO 26262 safety framework, there is a reference lifecycle prescribing a safety requirement hierarchy, where the set of safety requirements on a lower level (further down on the left leg of the V) shall completely cover the safety requirements on the level above. The complete safety of an Item is achieved by arguing that;

- All steps are complete and consistent in the safety requirement identifications (from Safety Goals down to atomic HW and SW safety requirements). *"Left leg of V has a complete and consistent set of safety requirements"*.
- All safety requirements (independent of refinement level) are verified: The restricted failure modes are shown not to occur with a confidence that is in line with the integrity attribute value of the corresponding safety requirement. "*Right leg of V has a complete set of verifications of the corresponding left leg requirements"*.
- All process requirements related to the corresponding integrity attribute values (ASIL) are fulfilled. "*All activities in the full V have been made using proper processes"*.
- All organizations involved in fulfilling the processes requirements constitute appropriate environments. "*All organizations of the full V have a safety culture"*.

There is no strict ISO 26262 terminology for these four aspects, but the first aspect above is denoted design-phase verification in part six, as opposed to other verification that are mainly addressing the second aspect. The first aspect is also very clearly prescribed as essential throughout the standard, expressed as it is prescribed that every safety requirement refinement shall be complete and consistent, ISO 26262-8:2018, 6.4.3.1. c), d) [2], which is referred to from all over the LiveCycle. Birch et al [3] denotes these four aspects above as Core, Layer 1, Layer 2, and Layer 3, respectively. Their very valid point is that each layer is supporting the arguments of the inside layers, and you cannot on an outer layer compensate for incompleteness in an inner layer. An outer layer contributes by bringing confidence to the layers inside, but it cannot replace their role. This means that;

- Safety culture arguments only bring value by adding confidence that the processes are executed as stated in the process arguments. They have no other direct value to the safety argumentation.
- Process arguments only bring value by adding confidence that verification of requirements is correct and complete; and by adding confidence that all safety requirements are identified completely and correctly. They have no other direct value to the safety argumentation.
- "Right-leg" verification arguments only bring value by adding confidence that the safety requirements are fulfilled. They have no other direct value to the safety argumentation.
- "Left-leg" verification arguments are the core in the safety argumentation telling why fulfilling all these requirements, by definition makes the Item Safe.

These dependencies are important to remember and acknowledge when we more in detail analyze Safety-Element-out-of-Context in the following sections.

III. SEooC – Why and How

A. Advantages of SEooC

The reason why we don't want to execute the entire 26262 lifecycle in one long sequence, is mainly the same reason why component-based design, CBD, in general is seen as attractive. The SEooC concept of ISO 26262 can be interpreted as a component-based safety argumentation pattern. Main advantages of CBD are that the actual system design can be less complex if there are known building blocks, and one supplier can address several customers at the time even if their system designs differ from each other.

From a safety point of view, it is particularly important to point out that innovation can be stimulated among suppliers this way, as it enables the supplier to make the development before getting all safety requirements from the customer(s). A safety framework not allowing the supplier to always be ahead, would be problematic for an industry branch where innovation is key to success. Allowing any supplier to perform development compliant with ISO 26262, and fully covering all needs for the customer to build a safety case, was a key reason why SEooC was originally proposed to become part of ISO 26262.

B. Developing SEooC

When all inputs of a specific ISO 26262 phase are possible to trace all the way to the related Safety goals and Item definition, we denote this as we are developing completely in-Context. In reality, this is rare for real automotive development for the HW and SW phases. Still, it is easy to get the impression when reading some parts of the standard that the entire lifecycle is executed in order, and that the actual safety goals would be known in every life-cycle phase.

Even if we are not in an absolute and complete in-Context, we could still be locally either in-Context or out-of-Context. If we are locally in-Context, this means that the activities of the actual activity at hand is provided with real inputs produced from the activities in previous life-cycle phases. If there is at least one SEooC in the trace up to related safety goals, we say that we can be at the same time locally in-Context and globally out-of-Context. This means that even if a given activity is not handled as an SEooC, it can still be impossible to trace it to actual safety goals. In the following we are focusing on describing the local SEooC, where neither tracing to the safety goals can be done, nor to the locally assumed inputs from previous phases of the reference life cycle of ISO 26262.

If at least one assumed input is absent when starting the life cycle of a certain activity, we are developing locally out-of-Context (ooC). The missing input we compensate for by defining a place holder. Such placeholders containing safety requirements, we denote safety contracts. In general, the safety contracts consist of both Assumed input requirements, and Guaranteed output requirements. This means that the safety contracts with safety requirements in both the roles of Assume and Guarantee, respectively, are essential for constructing the Core argument why a SEooC will safely fit in an actual context. Safety contracts for safety argumentation have been around for quite a while, and can be read about in for example [4], [5], [6], [7], and [8].

When performing any life-cycle phase, there is no difference between doing this in-Context and out-of-Context. In both cases we have a full set of inputs where the applicable safety requirements are found. It is never a question of just following some "safety process" according to a certain ASIL value. The process arguments are as always, a second supporting layer, that are dependent on a specific core, and a specific first layer, for bringing value to the overall safety argumentation. This means that the safety contracts are essential for the validity of the safety assessment of an SEooC. The Guarantee part of the safety contract, tells what should be fulfilled by a layer 1 argument, which is the design verification that all Guarantee safety requirements are fulfilled by the SEooC. If we change the Safety Contract of an SEooC, we need to redo the corresponding life-cycle activities which produces the layer 1 arguments that address the particular safety requirements for the life-cycle phase. In this aspect, there is no difference compared to in-Context development, where this is also the case.

To summarize, both for development in-Context and out-of-Context, all required inputs of the actual life-cycle phase need to exist, including the applicable safety requirements. The life cycle activities are performed with these inputs, and if they need to be changed, the life-cycle activities need to be redone. Any process argument is related to the specific safety requirements of the corresponding life-cycle activity. Process arguments without such connection are of no value for that safety element.

C. Bring an SEooC into a Specific Context

When integrating an SEooC it goes from being out-of-context to in-Context. This integration shall fulfil all layers of arguments, as is always the case in safety argumentation.

The core argument of safely integrating an SEooC is that its safety contract fulfils the safety requirements of the higher-level context (left leg consistency and completeness). This is a bidirectional check, as the safety contract contains safety requirements both in the role of Assume and of Guarantee, respectively. The assumptions on fulfilled safety requirements on the inputs of the SEooC as expressed in the safety contract, shall cover what is expressed in the real context. The guarantees on fulfilled safety requirements on the outputs of the SEooC as expressed in the safety contract, shall cover what is needed in the context to generate a complete core argument for that level.

The layer 1 argument of safely integrating an SEooC, is the integration verification of this SEooC (right leg completeness). For the corresponding integration activity where the SEooC is one of the parts, the verification shall show that this is a safe integration of safe parts. Remember that the parts are considered safe, comes with each of the SEooC safety case fragments. And when bringing an SEooC into context, it is only the layer 1 argument of the integration itself that is to be generated.

The layer 2 argument of safely integrating an SEooC, is that both the core argument and the layer 1 argument of this integration, are produced with adequate processes for the applicable ASIL values, i.e. methods.

The layer 3 argument of safely integrating an SEooC, is that the integrating organisation producing the layer 2 argument about a safe process, has evidence of a safety culture.

IV. SEooC in relation to other concepts

In the latest version of ISO 26262-8 [2] there are the concepts of;

- Qualification of Software components
- Evaluation of hardware elements
- Proven in use argument
- Interfacing an application that is out of scope of ISO 26262
- Integration of safety-related systems not developed according to ISO 26262

Note that none of those are related to SEooC at all. SEooC is, in contrast to these, describing how to develop the safety element according to ISO 26262, and how to do the integration of that safety element also fully according to rules of ISO 26262. For SEooC all the layers of argumentation apply, and this means that it is essential that there is a core argumentation relating to the specific safety requirements both to the SEooC and to the context when integrating the SEooC.

All the five concepts listed above are instead dealing with a situation when there is no full compliance to the ordinary 26262 argumentation structure. This means that none of these listed concepts fully covers all four layers of argumentation presented above, and hence they are disqualified as SEooC.

V. SEooC, CI/CD and Automated Driving Systems

In general, the automotive industry goes in a direction of continuous integration and continuous deployment (so called CI/CD). This is especially true for automated driving systems (ADS). In CI/CD, the idea is to continuously evolve the product in frequent increments so that all these can be deployed to the end customer. CI/CD is enabling that the vehicles already being out on the roads, will get updates at a significantly higher pace than today (which mostly have contained bug fixes), enabling a real DevOps.

The change, compared to traditional development cycles, means that instead of producing safety argumentation for the start of production (SOP) of a certain platform model, there will be a need for a complete safety case at every CD release. If all the elements that are subject for the continuous integration (CI) are developed SEooC, the way to build the full safety argumentation can harmonize with the general CI pattern. Such argumentation calls for a granularity of the SEooC to be as fine-grained as is expected in the ordinary CI/CD way of building a large complex product. We could say that building component-based safety argumentation fitting CI/CD is to take the SEooC pattern to its extreme. And there is nothing problematic with that at all. It is just to say that if we want to make ground for large-scale agile development, we need to implement the safety argumentation completely component based. From an ISO 26262 perspective there is no extra problem with this. If the SEooC is used once or a million times for an Item, is still the same argumentation pattern. On the other hand, a traditional approach, starting with defining an item and then breaking down the safety requirements would be a task impossible to handle with so many different stake holders and components.

VI. SEooC Everywhere – Full component-based Safety

Similar to component-based design, CBD, safety argumentation can also be made bottom-up, i.e. component-based. This means that every life-cycle activity is made SEooC, and then everything is integrated and put in-Context. Note that there is a main difference between integrating a product in CI and integrating a safety case in CI, and that is that the latter requires all levels of abstraction to become part of the integration. For safety argumentation, it is not only to integrate the implemented product, but to bring information from all kinds of activities in the entire reference life cycle (the full V).

As pointed out in [9], a reference life cycle can be interpreted by the two dimensions of abstraction and aggregation. When taking the SEooC concept all the way to fully component-based safety argumentation, this means that for each SEooC it should be confined to one single position in such a "coordinate" system. Every applicable coordinate needs to be covered by at least one SEooC, and each SEooC stays inside exactly one coordinate.

When integrating an SEooC in a CI process, there is an essential merge condition related to the core safety argumentation. This implies that the safety contracts need to be checked in both dimensions of aggregation and abstraction, respectively. For an SEooC to be allowed to get integrated to the main branch, it is a necessary, but not a sufficient, condition that its safety contracts in this integration would fulfil the core arguments of completeness and consistency among safety requirements. If this cannot be proven, that SEooC will not become allowed to get integrated in the CI. By assuring the core safety argumentation this way, every update of the main branch, can guarantee that the safety case fragments of each SEooC, together always can build a complete safety case, as a result of the CI.

While the core argument of bringing the SEooC in-Context is completely produced in the CI itself, for the safety arguments of layers 1, 2 and 3, the CI has two different tasks. The one is to check their existence for each of the SEooC. And the second is to produce them, but only for the integration. This means that the layer 1, 2, and 3 arguments of the SEooC themselves are produced out-of-Context, but for bringing them in-Context these arguments are produced in the CI. Note that what is said about producing the arguments, is including the full implication of these arguments, including the safety assessment of the SEooC in relation to its contract.

Taking the SEooC concept to its full potential implies that a SEooC also can bring its own safety case fragment, which means that integrating all SEooC, also would construct a complete safety case from the safety case fragments together with the safety case fragment from the CI itself. With carefully chosen formalism in the safety contracts and the safety case fragments, such a generation of the full safety case can be automatised in the CI framework.

VII. Conclusions And Future Work

We have described the origin and the future of Safety-Element-out-of-Context, which is a pattern enabling component-based safety argumentation, separating the dependencies between customer and suppliers or between teams in a large organisation. The concept of SEooC is very well suited to form the base for automatically generating complete safety cases at a high pace for very complex features, as is the automotive needs today and in the future. Especially the automated driving systems, ADS, are dependent on high rigour of safety cases that can be produced at high pace in a development environment of CI/CD (continuous integration / continuous deployment). A challenge for the future is to find a high enough rigour in expressing safety contracts and safety-case fragments, to enable an automatization of generating safety cases in the modern development process of CI/CD.

In the domain of road vehicles equipped with automated driving systems (ADS), there is an ongoing ISO activity to define an application standard for safety (TS 5083), considering all root causes for becoming unsafe. Even if this will go beyond what is today prescribed by ISO 26262, the same pattern of SEooC and component-based safety argumentation may still apply. As long as this standard will prescribe a core argumentation of completeness and consistency in refinement of requirements, and give guidance what level 2 and 3 arguments that are applicable for a certain level 1 claim, the pattern of SEooC can be adapted, thus enabling using CI/CD when bringing automated driving to the market.

VIII. Acknowledgement

The authors would like to thank the other creators of the SEEooC concept, coined in Munich in April 2009, especially Simon Fürst, Matthias Maihöfer and Jürgen Sauler. We also would like to thank Jonas Borg for many fruitful discussions on this topic.

References

[1] 'ISO 26262:2011 - Road vehicles -- Functional safety'.

[2] 'ISO 26262:2018 - Road vehicles -- Functional safety'.

[3] J. Birch *et al.*, 'A Layered Model for Structuring Automotive Safety Arguments', in *Proceedings of the Tenth European Dependable Computing Conference (EDCC)*, 2014.

[4] I. Bate, R. Hawkins, and J. McDermid, 'A Contract-based Approach to Designing Safe Systems', in *Proceedings of the 8th Australian Workshop on Safety Critical Systems and Software - Volume 33*, 2003, pp. 25–36.

[5] J. Westman, M. Nyberg, and M. Törngren, 'Structuring Safety Requirements in ISO 26262 Using Contract Theory', in *Computer Safety, Reliability, and Security*, 2013, pp. 166–177.

[6] A. Benveniste *et al.*, 'Contracts for Systems Design', RR-8147, INRIA, 2012.

[7] E. Denney, G. Pai, and I. Habli, 'Dynamic Safety Cases for Through-Life Safety Assurance', in *Proceeding of the 37th IEEE International Conference on Software Engineering*, 2015, pp. 587–590.

[8] I. Sljivo *et al.*, 'A Method to Generate Reusable Safety Case Fragments from Compositional Safety Analysis', in *Software Reuse for Dynamic Systems in the Cloud and Beyond*, vol. 8919, I. Schaefer and I. Stamelos, Eds. Cham: Springer International Publishing, 2014, pp. 253–268.

[9] F. Warg *et al.*, 'A Continuous Deployment for Dependable Systems with Continuous Assurance Cases', in *Proceedings of the 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*.