ADS Safety Assurance – Future Directions

Magnus Gyllenhammar
Zenseact; KTH Royal Institute of Technology
Göteborg, Sweden
magnus.gyllenhammar@zenseact.com

Carl Bergenhem
Qamcom Research and Technology AB
Göteborg, Sweden
carl.bergenhem@qamcom.se

Fredrik Warg
RISE Research Institutes of Sweden
Borås, Sweden
fredrik.warg@ri.se

***Abstract*—More effective, efficient and flexible ways to manage safety assurance are needed for the successful development and release of Automated Driving Systems (ADSs). In this paper we propose a set of desired assurance method criteria and present an initial overview of available safety assurance methods and how they contribute to the proposed criteria. We observe that there is a significant gap between the state-of-the-art research and the state-of-practise for safety assurance of ADSs and propose to investigate reasons for this as future work. A next step will be to investigate how to merge the elements from the different assurance methods to achieve a method addressing all criteria.**

***Index Terms*—Safety assurance, Contract-based design, safety contracts, automated driving system, assurance method criteria.**

I. Introduction

Safety assurance is a challenge for development of advanced automated and connected vehicle functions, such as automated driving systems (ADSs). There is a need for more effective, efficient and flexible ways to manage assurance cases compared to currently prevailing practices. Incremental development to ease introduction of ADSs by allowing for gradual improvement of performance and gradual expansion of the operational design domain (ODD) requires support for **(a) frequent updates**, e.g. using agile development and continuous integration/continuous deployment schemes. Key to such incremental improvements is also use of targeted collection of **(b) operational data** giving feedback to development, e.g. using DevOps/DataOps methods. In addition, automated functions will have a need for **(c) monitoring** changes in the operational context to make sure the safety case stays valid, e.g. by using field data in the context of ODD verification and monitoring; as discussed by Gyllenhammar et al. [1]. Cyber-security incident management is another driving need for rapid updates. Furthermore, management of multiple **(d) variants** and integration of components from different suppliers – which requires **(e) modularity** – are prevalent issues in the automotive domain. As ADSs are complex safety-critical functions, they are likely to involve parts from multiple suppliers and exist in multiple variants. An ADS may also employ **(f) self-adaptive** system properties, i.e. a system that has to operate under uncertainty. Weyns et al. [2] coin the term perpetual assurance to capture the continuous assurance effort needed for self-adaptive systems, in contrast to a one-off effort before deployment of the system. Here, the assurance process spans the entire life-time of the system and new evidence and arguments are incorporated and integrated into the assurance case continuously also during operations. Finally, putting together an effective assurance case given the characteristics of an ADS, e.g. with non-deterministic algorithms, a complex sensor system, adequately capturing the details of the (stochastic) environment, and a large scope for the safety-critical functionality, will be challenging. We believe in basing safety assurance on a quantitative risk norm, e.g. as described by Warg et al. [3], which implies a mainly **(g) quantitative assurance case**. The goal of this approach is a more balanced and usable way of allocating necessary risk reduction; compared to the relatively coarse-grained integrity levels used in standards such as ISO 26262 [4].

In [2], a framework is presented and a set of requirements for method usefulness is given. This framework is in part relevant also for ADSs, but we suggest the *assurance method criteria **(a)-(g)*** above to more accurately capture the specific challenges in ADS assurance. A hypothesis for our work has been that safety-contract based design – a well-researched method (e.g. Bate et al. [5]) which enables design with reusable and potentially pre-certified components – would be a suitable base for ADS safety assurance. However, considering the criteria above it is not sufficient by itself, and despite the large body of existing work, we have yet to see any widespread use of safety contracts.

In this paper we discuss a few proposed safety assurance methods and map their applicability to criteria **(a)-(g)**. We find that there is a gap between state-of-the-art in research and state-of-practice, as well as a lack of tools fulfilling all criteria, something we aim to investigate further in future work. While no method seem sufficient by itself, all methods contribute to some criteria, hinting at the possibility of combining elements from different methods for an ADS safety assurance method covering all criteria. In particular, we revisit the research in safety contracts, and suggest that a way forward can be using contracts as a base to build such an assurance method.

II. Background: Contract-Based Design

Contract-based software design was pioneered by Bertrand Meyer [6] in the Eiffel programming language. The central idea, equivalent to the Hoare triple [7], is that the interaction between elements in a system are expressed as contracts. A

The research has been supported by the Strategic vehicle research and innovation (FFI) programme in Sweden via the SALIENCE4CAV project (ref 2020-02946) and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

contract describes which preconditions an element expects, and which postconditions (and invariants) it can guarantee given that those preconditions hold. Hence, the element does not have to be designed to handle cases where the preconditions are invalid, limiting both complexity and verification effort. The contract-based approach encourages **(e)** modularity, and can simplify change management as the contracts will provide a way to perform impact analysis, i.e. whenever a contract is altered it will be possible to follow how dependent elements are affected. Contract usage establishes the foundations for contract reasoning about composability and abstraction/refinement relationships. In software, contracts are often implemented as assertions resulting in an exception if a contract is broken. The paradigm has later been applied to system design, where e.g., Benveniste et al. [8] introduced and formalized *Assume-Guarantee* (A/G) contracts to describe the preconditions (assumes) and post-conditions (guarantees) for system design elements.

Nuzzo et al. [9] describe a methodology for design of complex distributed systems based on [8]. A/G contracts are used to reason about requirements and their refinement during the design process, and includes a methodology and algebra of contracts to formalize the design process. The design is carried out as a sequence of refinement steps from a high-level specification to an implementation built out of a library of components at the lower level. The high-level specification is first formalized in terms of contracts to enable requirement validation and early detection of inconsistencies. Then, at each step, contracts are refined by combining synthesis, optimization and simulation-based design space exploration methods. Refinement through different abstraction levels is also proposed by Warg et al. [10], though the way to formulate contracts is not as well defined. The main goal in [10], however, is a methodology enabling continuous deployment.

III. Safety Assurance Methods

In this section, we briefly describe safety assurance methods that fulfil some of the criteria discussed in Sec. I. Note that only the most relevant or recent paper from each author/group is included, for space reasons. A summary how the methods map to the proposed assurance method criteria can be found in Table I.

A. Safety-Contract Based Design

A use-case for contract-based design is to provide contracts aimed at quality attributes. The use of safety contracts, i.e. contracts which only encode safety-critical properties of an element, has further been proposed, e.g. Bate et al. [5] and Slijvo et al. [11]. Recently, there have also been large EU funded projects in the domain, such as the SafeCer project. This project had the goal of enabling reusable certifiable components using safety contracts and pre-qualified components. In [12], Carlson et al. (from the SafeCer project) introduce a meta model to support safety argumentation according to safety standard(s) by incorporation of the necessary elements into a component-based software engineering approach. The project AMASS [13] also included contracts in its mission to introduce a framework for multi-concern (e.g. safety and security) assurance and certification of dependable systems.

Söderberg and Johansson [14] discuss how to use a modified design methodology for contract-based design (CBD) intended for development of software and component based systems by use of safety contracts. The primary purpose is to make a proposal on how to integrate safety contracts in an implementable way (in a tool) for automatic safety contract verification.

B. Conditional Safety Certificates - ConSerts

Schneider and Trapp [16] propose Conditional Safety Certifications (ConSerts) to shift part of the certification task of open adaptive systems from design-time, where the access to detailed evidence in support for the safety analyses might be limited, to run-time, where such evidence might be more readily available. The ConSerts are constructed of a set of predefined modular **(e)** and variable **(d)** certifications for each related (sub-)system. When the system is integrated (at system start with different collection of services) or adapted (during run-time) **(f)** these modular certificates are composed and evaluated. The ConSert certifies that a number of specified safety guarantees are fulfilled with certain probability **(g)** given the precondition that the specified safety demands (cf. *assume* of CBD) are fulfilled. The demands of each modular certificate relates to the component's environment and it might thus be difficult to verify already at design-time, since the environment might change throughout the use of the system (or component). In the example given, the environment of the component is exemplified as what other services are active. Despite the lack of detailed discussions about different system environments, except the availability of different adjacent services, there seem to be nothing in the approach contradicting its use also in the context of highly dynamic environments or e.g. to cater for the complexity of different ODDs.

The discussed example relates to adaptiveness in the sense that the system switches between different configurations during run-time. The system is able to provide certain top-level services based on cooperation between components through a service oriented architecture and dynamical service hierarchies. [16] emphasises that the ConSert should be used as a means to shift only the parts that are difficult or impossible to adequately complete during development/design-time to be evaluated in run-time, not that everything should be left for run-time certification. Only some predetermined certificate variants should be left for run-time evaluation, where their respective demands can be checked.

Contrary to CBD, the ConSert approach suggests to design the certificates composed in several paths where the most performant sub-system (path) is evaluated first and subsequently enacted if its certificates are found valid. This adaptability of the contract structure based on run-time evidence is something that seems generally missing in work in the CBD-approach. More recently, Trapp et al. [15] build on the ConSert concept and introduce a dynamic safety management framework for autonomous systems. This facilitates monitoring **(c)** as well

TABLE I
SAFETY ASSURANCE METHODS MAPPED TO ASSURANCE METHOD CRITERIA.

Assurance method	**Assurance method criteria (a) to (g)**							**References**
	Freq. upd.	***Op. data***	***Monitor***	***Variants***	***Modular***	***Self-adapt.***	***Quant. AC***	
Safety-contract based design					X			[5], [11]–[14]
Conditional safety certificates (ConSert)			(X) [15]	X	X	X	X	[15], [16]
Dynamic assurance cases			X			(X) [17]	X	[17]–[20]
Product-line contract based design				X	X			[21]
Continuous assurance cases	*X*	*X*			*X*			[10]

as to create safety-awareness, ultimately allowing the system to self-adapt based on available run-time evidence.

C. Dynamic Assurance Cases

Denney et al. [18] introduce the term Dynamic Safety cases. They deploy real-time monitors **(c)** of both the system and its environment to initiate changes to the safety case where applicable. This relies on a safety case that is possible to query and includes the appropriate metadata. Asaadi et al. [19] continues this work and propose an architecture (including monitors and decision mechanisms) as well as a methodology for dynamic assurance. The dynamic assurance case consists of five models or parts: 1) Assurance policy models, 2) Assurance architecture model, 3) Assurance quantification model **(g)**, 4) Evidence model, and 5) Assurance rationale.

Calinescu et al. [17] propose a method for dynamic generation of assurance cases; with generation of assurance evidence during both design-time and runtime. This is done for a target application that is denoted a self-adaptive software system **(f)**. Self-adaptive software handles environmental and internal uncertainties by dynamically adjusting its architecture and parameters in response to events such as workload changes and component failures.

Asaadi et al. [20] propose a method to quantify **(g)** assurance measures online, during run-time, by using Gaussian processes. They suggest that such quantification of the use case considered could consist of two aspects (i) identification of relevant dependability (e.g. safety) attributes, and (ii) quantification of these attributes and the associated uncertainty of that quantification. They propose to use the output of the quantified assurance measure to dynamically check the validity of the assurance case and thus support dynamic assurance cases.

D. Product-Line Contract Based Design

Nešić et al. [21] discuss assurance, in the context of Product-Lines (PL) in system development, based on CBD. The purpose is to handle system variants **(c)** at design-time and create valid assurance cases for all of them. This is done by extending the model for CBD (e.g. [8]) to define conditions under which all system variants can be shown to fulfil the specified requirements. These conditions are used to define an assurance case pattern that also enable step-wise assurance case creation. Further, they believe that it is feasible to express general PLs according to the proposed CBD model. Using this model of the system in the design phase would yield early insights into the system and what needs to be addressed in order for the assurance pattern to be completed and a valid assurance case compiled. The work addresses similar problems as those discussed by Schneider and Trapp (ConSerts) [16], but in slightly different contexts and also focus on different main challenges. Where [21] address the combinatorial complexity of configurations of different variants across a PL, [16] address the challenge where evidence of the system configuration and the components operating environments cannot be obtained prior to run-time. Both the certificates in ConSert and the contract structure of PL-CBD are defined during design-time. ConSerts are however evaluated in their present context during run-time and the demands are checked using the currently available run-time evidence.

It seems possible to adapt and configure the approach from [21] to also solve the challenges of run-time evidence addressed by [16]. All possible system configurations (including service configurations) and operational contexts need to be described according to the PL CBD structure and further analysed according to the proposed criteria in order to assert the fulfilment of the requirements. However, even though the example in [16] would be possible to fit into the framework of [21] (since the parameters considered only pertain to the system configuration and that this number of configurations clearly is finite), cases where the evidence is only obtainable during run-time (e.g. pertaining to external environment) might not be possible to frame in the PL CBD framework. Especially, the complexity of considering all possible system configurations with all possible environmental contexts would lead to a combinatorial explosion, which in general would result in the PL CBD approach being infeasible to provide completeness of the assurance (cf. Scenario-based V&V). In such a context, delaying the certification until run-time when the specific combination of parameters has been significantly reduced would perhaps yield a better confidence in the provided assurance, thus playing in favour of the ConSert approach of [16]. We also note that it is unclear how stochasticity and uncertainty is addressed in the two approaches.

E. Continuous Assurance Cases

Warg et al. [10] propose a method for updating the assurance case for a function in step with changes in an agile/iterative development cycle. The motivation is to enable steady feature growth after deployment, e.g. being able to expand the ODD or improve performance for an ADS (e.g. due to better understanding through increased data collection). The

method include using assurance case fragments for individual components which are then aggregated using safety contracts. Updates of the assurance case are planned for, and performed, in step with the feature development, and the assurance case is version managed with the product. This method has yet to be proven in a use-case and lacks tooling. However, it is based on the criteria for modularity **(e)**, frequent update **(a)** and use of operational data **(b)**, and we believe it can be a useful base for adding parts of other methods to eventually fulfill all criteria.

IV. Conclusions & Future Work

In this paper, we have proposed criteria that we believe are necessary for effective, efficient and flexible development and assurance of ADSs. Some existing techniques that may contribute to fulfilling these criteria are discussed – where a more systematic survey of the state-of-the-art remains as future work to create a complete view of existing work. An observation is that there seems to be a significant gap between state-of-the-art in research and state-of-practice. There are likely several reasons for this, and safety assurance for ADSs lack maturity in general. However, in our future work we would like to better understand the reason behind, and be able to address, this gap.

One initial impression is that there is a lack of tools compatible with our proposed criteria, at least widely deployed tools. Safety-contract tools such as CHESS [22] and CHASE [23] exist, but do not seem well suited, especially considering criteria **(a)-(c)** and **(f)**. Since both are based on formal specification, which require significant expert skills, it may also be difficult to achieve wide adaptation in an agile organization. However, a more in-depth investigation of tools in light of the proposed criteria remains as future work.

We intend to improve the continuous assurance approach [10], aiming to fill the remaining gaps identified in the assurance method criteria table. To that end, one task will be a deeper study of the methods mentioned in this paper, as well as other works related to ADS safety assurance, aiming to find additional elements to incorporate. Looking at the applicability of the ConSert approach for complex systems such as ADSs and the possibility to merge the runtime flexibility of this approach with the elements of the continuous assurance approach seems to be a promising start. Finally, answering the question: "How to compose contracts to capture the quantitative risk norm and relay this information throughout the ADS?", will be central in order to connect a contract-based assurance method to the risk norm at vehicle level.

References

[1] M. Gyllenhammar, R. Johansson, F. Warg, D. Chen, H.-M. Heyn, M. Sanfridson, J. Söderberg, A. Thorsén, and S. Ursing, "Towards an operational design domain that supports the safety argumentation of an automated driving system," in *10th European Congress on Embedded Real Time Systems (ERTS)*, Toulouse, France, Jan. 2020.

[2] D. Weyns, N. Bencomo, R. Calinescu, J. Camara, C. Ghezzi, V. Grassi, L. Grunske, P. Inverardi, J.-M. Jezequel, S. Malek *et al.*, "Perpetual assurances for self-adaptive systems," in *Software Engineering for Self-Adaptive Systems III. Assurances*. Springer, 2017, pp. 31–63.

[3] F. Warg, M. Skoglund, A. Thorsén, R. Johansson, M. Brännström, M. Gyllenhammar, and M. Sanfridson, "The quantitative risk norm-a proposed tailoring of HARA for ADS," in *50th Annual IEEE/IFIP Int. Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 2020, pp. 86–93.

[4] ISO, "ISO 26262:2018 Road vehicles – Functional safety," 2018.

[5] I. Bate, R. Hawkins, and J. McDermid, "A contract-based approach to designing safe systems," in *8th Australian workshop on Safety critical systems and software (SCS) - Volume 33*, 2003, pp. 25–36.

[6] B. Meyer, "Design by contract and the component revolution," in *Technology of Object-Oriented Languages and Systems (TOOLS-34)*, 2000.

[7] C. A. R. Hoare, "An axiomatic basis for computer programming," *Communications of the ACM*, vol. 12, no. 10, pp. 576–580, 1969.

[8] A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. Sangiovanni-Vincentelli, W. Damm, T. Henzinger, and K. G. Larsen, "Contracts for system design," Ph.D. dissertation, Inria, Rapport de recherche RR-8147, 2012.

[9] P. Nuzzo, A. L. Sangiovanni-Vincentelli, D. Bresolin, L. Geretti, and T. Villa, "A platform-based design methodology with contracts and related tools for the design of cyber-physical systems," *Proceedings of the IEEE*, vol. 103, no. 11, pp. 2104–2132, 2015.

[10] F. Warg, H. Blom, J. Borg, and R. Johansson, "Continuous deployment for dependable systems with continuous assurance cases," in *2019 IEEE Int. Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2019, pp. 318–325.

[11] I. Sljivo, B. Gallina, J. Carlson, and H. Hansson, "Strong and weak contract formalism for third-party component reuse," in *IEEE Int. Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2013, pp. 359–364.

[12] J. Carlson, C. Ekelin, B. Gallina, R. Johansson, S. Puri, F. Rossi, M. Skoglund, and S. Tonetta, "Delivery d132.2, generic component meta-model' of the safecer project," *SafeCer project, Deliverable D*, vol. D132.2, pp. 1–100, 2014.

[13] J. L. de la Vara, A. Ruiz, B. Gallina, G. Blondelle, E. Alaña, J. Herrero, F. Warg, M. Skoglund, and R. Bramberger, "The AMASS approach for assurance and certification of critical systems," in *Embedded World*, 2019.

[14] A. Söderberg and R. Johansson, "Safety contract based design of software components," in *IEEE Int. symposium on software reliability engineering workshops (ISSREW)*. IEEE, 2013, pp. 365–370.

[15] M. Trapp, D. Schneider, and G. Weiss, "Towards safety-awareness and dynamic safety management," in *14th European Dependable Computing Conference (EDCC)*. IEEE, 2018, pp. 107–111.

[16] D. Schneider and M. Trapp, "Conditional safety certification of open adaptive systems," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 8, no. 2, pp. 1–20, 2013.

[17] R. Calinescu, D. Weyns, S. Gerasimou, M. U. Iftikhar, I. Habli, and T. Kelly, "Engineering trustworthy self-adaptive software with dynamic assurance cases," *IEEE Transactions on Software Engineering*, vol. 44, no. 11, pp. 1039–1069, 2017.

[18] E. Denney, G. Pai, and I. Habli, "Dynamic safety cases for through-life safety assurance," in *37th IEEE/ACM Int. Conference on Software Engineering*, vol. 2. IEEE, 2015, pp. 587–590.

[19] E. Asaadi, E. Denney, J. Menzies, G. J. Pai, and D. Petroff, "Dynamic assurance cases: A pathway to trusted autonomy," *IEEE Computer*, vol. 53, no. 12, pp. 35–46, 2020.

[20] E. Asaadi, E. Denney, and G. Pai, "Towards quantification of assurance for learning-enabled components," in *15th European Dependable Computing Conference (EDCC)*. IEEE, 2019, pp. 55–62.

[21] D. Nešić, M. Nyberg, and B. Gallina, "Product-line assurance cases from contract-based design," *Journal of Systems and Software*, vol. 176, p. 110922, 2021.

[22] A. Cicchetti, F. Ciccozzi, S. Mazzini, S. Puri, M. Panunzio, A. Zovi, and T. Vardanega, "CHESS: a model-driven engineering tool environment for aiding the development of complex industrial systems," in *27th IEEE/ACM Int. Conference on Automated Software Engineering*, 2012, pp. 362–365.

[23] P. Nuzzo, M. Lora, Y. A. Feldman, and A. L. Sangiovanni-Vincentelli, "CHASE: Contract-based requirement engineering for cyber-physical system design," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 839–844.