Towards Dynamic Safety Management for Autonomous Systems

Mario Trapp and Gereon Weiss [1]

Fraunhofer ESK
Munich, Germany

Abstract *Safety assurance of autonomous systems is one of the current key challenges of safety engineering. Given the specific characteristics of autonomous systems, we need to deal with many uncertainties making it difficult or even impossible to predict the system's behaviour in all potential operational situations. Simply using established static safety approaches would result in very strict worst-case assumptions making the development of autonomous systems at reasonable costs impossible. This paper therefore introduces the idea of dynamic safety management. Using dynamic safety management enables a system to assess its safety and to self-optimize its performance at runtime. Considering the current risk related to the actual context at runtime instead of being bound to strict worst-case assumptions provides the essential basis for the development of safe and yet cost-efficient autonomous systems.*

1 Introduction

Safety assurance of autonomous systems is still an open challenge. Autonomous systems operate in open contexts so that it is not possible to anticipate all relevant operational situations. Moreover, safety managers are still struggling with the assurance of AI-based algorithms, which are an essential prerequisite for facilitating autonomous systems. Considering open contexts and the non-deterministic, non-linear behaviour of AI, this leads to intrinsic and extrinsic uncertainties, which make it impossible to anticipate the system's behaviour a-priori. Having said that, it is clear that applying conventional safety approaches based on static worst-case assumptions would not lead to reasonable results. Doubtlessly, there is a long list of open challenges concerning safety assurance of autonomous systems, but as long as such conservative static worst-case assumptions completely bound our space of options, we will not be able to find any reasonable solution for safety assurance of autonomous systems at all.

[1]{mario.trapp|gereon.weiss}@esk.fraunhofer.de

Therefore, it is necessary to loosen the ties of static worst-case assumptions by moving from static safety management to dynamic safety management. This means that instead of considering possible and hypothetical situations from a worst-case perspective at design time, the system evaluates its current risk at runtime for the concrete given context at this particular given point in time - considering its operational context as well as its own quality state. To this end, we combine approaches from model-based safety engineering on the one hand, and the self-adaptive systems community on the other hand. This paper briefly introduces the basic concepts of dynamic safety management. Even though the concepts are applicable to different application domains, we will refer to ideas and examples from the automotive domain to simplify matters throughout the remainder of this article.

2 Adaptive Fail-Operational Architectures

In order to have some flexibility at runtime, we need to implement adaptive fail-operational architectures as shown in Figure 1. We will not discuss these architectures in this article in detail. It is important to note that we do not have one fixed, static functionality, but that the system can adapt itself to various different configurations. Such an adaptation can encompass different types of changes in an explicit, reproducible way.

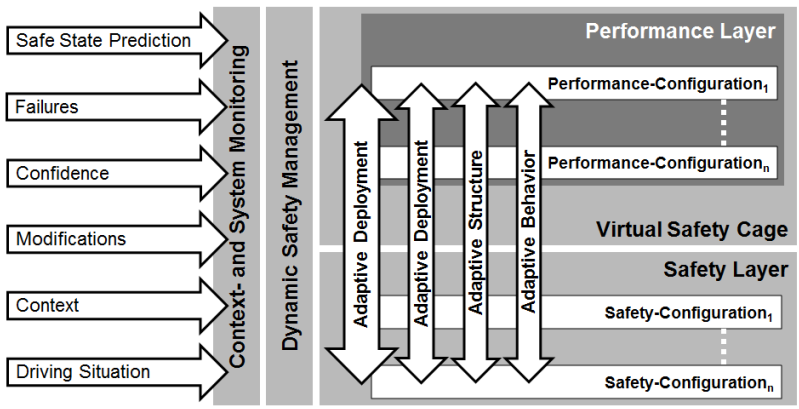

Fig. 1. Basic Concept of Fail-Operational Architectures

For instance, the system can dynamically re-deploy software components to different hardware nodes or dynamically adapt its structure (e.g. by using different sensors, different algorithms etc.). An adaptation of single parameters of the algorithms allows for fine-grain system adaptations. One major aspect of such an architecture is dynamic safety management, which includes identifying and assessing the current situation and planning adaptation strategies for optimizing the

system's performance while preserving its safety. To this end, monitoring needs to derive information about the context and about the system. Examples of monitored information for a vehicle are given on the left-hand side of Figure 1. Based on this monitoring, dynamic safety management utilizes dynamic adaptation as described above for switching the system configurations. Safety-configurations ensure that the system can preserve its safety in any given situation. An isolation mechanism, such as a virtual safety cage (Heckemann et al.) or similar concepts (Weiss et al. 2018), ensures that performance configurations can be used for optimizing the systems performance without endangering its safety, as the system can move to an adequate safety configuration whenever necessary.

3 The Basic Concept

In order to facilitate safety assurance of autonomous systems, we employ the idea of safety-awareness and dynamic safety management as illustrated in Figure 2. Instead of anticipating any foreseeable situation and assuming the worst case at design time, we enable the system to reason about the actual given situation at runtime using different sources of runtime information. Perception systems and vehicle observer models of driver assistance systems provide lots of information about the system's environmental context. Internal monitors, such as error detection, provide a lot of information about the system's internal (quality) state. Usually, this information is available as an integral part of the functionality. In order to use it for dynamic safety management, we need to transform this information to a higher semantic level, which we refer to as safety awareness (cf. Fig. 2).

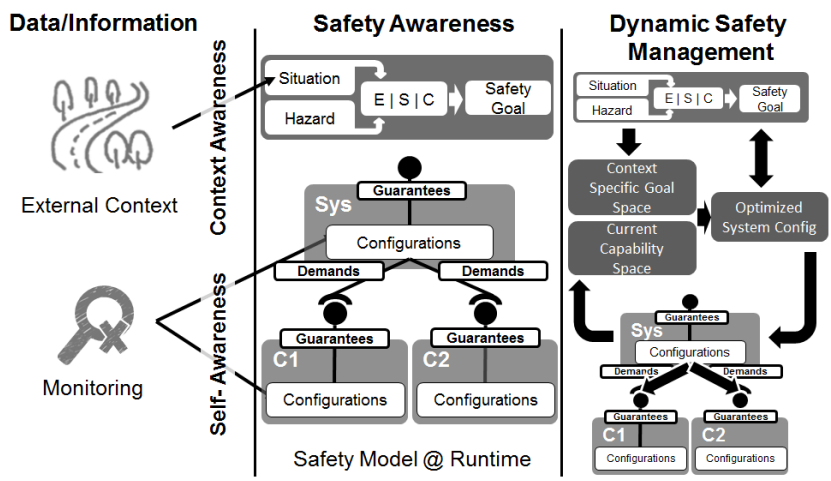

Fig. 2. Basic Idea of Dynamic Safety Management

For creating safety-awareness, we make safety models, which safety managers usually use during design time, available at runtime. Namely, as a first step, we use models of the hazard analysis and risk assessment (HARA) and safety concept models, which provide a modular, contract-based representation of safety requirements. Using external context sensing, we enable the system to reason about the current operational situation. While the default perception functionality as it is required for realizing autonomy focuses on high precision, but with a low integrity, it is sufficient to realize a low-precision but high-integrity perception as input for dynamic safety management. By mapping this perception information to HARA models at runtime, the system becomes "context-aware" enabling it to reason about its current operational situation from a safety point-of-view. In consequence, the system can identify those safety goals that are still of relevance in the current runtime situation. Since the required integrity of a safety goal results from the worst case of all possible operational situations, regarding the currently likely situations at runtime furthermore reduces the required integrity levels. By doing so, we can dynamically loosen the safety constraints to those safety goals with those integrity levels, that are actually of relevance in the current runtime situation. Of course, this requires that the decision logic must be at the highest integrity level. Compared to considering worst-case situations, this provides us with much more freedom for optimizing the system's functional performance while preserving its safety.

Besides this context-awareness, it is important to consider the system's internal state. In order to realize this "self-awareness", we map available monitoring information to modular safety concepts, i.e. the safety requirements, at runtime. By this means, the system can reflect monitoring information at the level of safety requirements, i.e. which safety requirements are fulfillable in the current runtime situation. In order to enable dynamic adaptation, we model safety concepts using safety-contracts. Using safety-contracts, single modules specify which safety guarantees they are able to provide under the assumption that their context (e.g. other components, platforms etc.) fulfills the module's safety demands. Instead of a single static concept, the modules may have different configurations. Depending on which demands are fulfilled, the module selects the best possible configuration and will provide the corresponding set of safety guarantees. Therefore, the system has several degrees of freedom to adapt itself for optimizing its performance while preserving safety.

This optimization is the core of dynamic safety management. Using the HARA at runtime, the system gets the currently relevant goal space, i.e. which safety goals are actually relevant in the actual given runtime situation. Using self-awareness by mapping monitors to safety requirements, the system gets its current capability space. Using the goal space and the capability space as available optimization space, the system can optimize itself and assure its safety at runtime.

4 Dynamic Safety Management

If we use this general idea as a basis to derive a framework for dynamic safety management, we derive the elements illustrated in Figure 3. Dynamic safety management follows the basic ideas of a MAPE-K cycle known from self-adaptive systems (Kephart and Chess 2003). A MAPE-K cycle consists of five basic steps: *M*onitor, *A*nalyze, *P*lan, and *E*xecute. All of these steps use shared *K*nowledge, which – in our case – is represented by safety models at runtime (SM@RT) (Trapp and Schneider 2014), namely HARA-models at runtime (H@RT) and safety concept models at runtime (SC@RT). By doing so we follow the basic ideas of models@run.time (Blair et al. 2009) (Cheng et al. 2014). Using these models as our representation of shared knowledge, monitoring enables the system to become aware of its current quality state and the state of its environment. As mentioned above, we consider a) *self-awareness*, i.e. the system uses internal monitors such as error detections and maps the results to the SC@RT for reflecting the current state in terms of fulfillment of safety requirements. Moreover, we facilitate b) *context-awareness*, i.e. the system uses safe perception mechanisms to monitor its operational situation and to map it to the H@RT for reflecting the current situation in terms of safety-related risk.

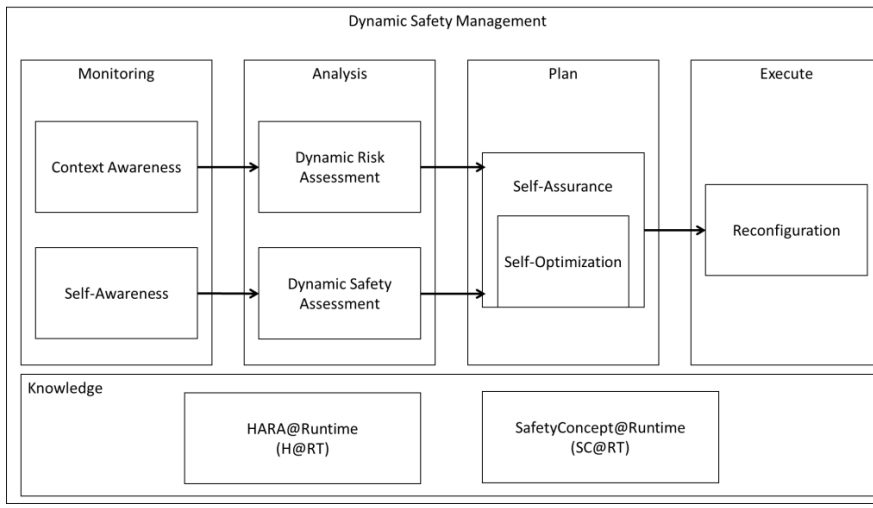

Fig. 3. Dynamic Safety Management Framework

In the second step, the analysis, the system uses the results observed in the monitoring step for assessing the current safety risk using a dynamic risk assessment approach based on the H@RT (cf. Section 4.1), and for assessing the system's current safety state using a dynamic safety assessment approach based on the SC@RT (cf. Section 4.2). In the planning step, the system uses the available

optimization space derived during the risk and safety assessment steps for optimizing its performance (self-optimization) without violating any safety goals (self-assurance). Finally, the system reconfigures itself in the execute step (Ruiz et al. 2015).
The remainder of this paper focuses on the analysis and the planning part. To this end, we focus on the concepts used in the automotive domain even though the approach can be applied to other domains as well.

4.1 Dynamic Risk Assessment

As mentioned before, dynamic risk assessment is a core element of dynamic safety management as part of the analysis step. Using perception information about the system's context and HARA models at runtime, we enable the system to analyze the current risk of the actual situation at runtime, instead of being bound to static worst-case assumptions.

Regarding a hazard analysis and risk assessment according to ISO 26262 (ISO 2018), we are interested in hazardous events, i.e. the coincidence of a hazard (e.g. "an unintended acceleration of more than 2m/s² for more than 1s") and an operational situation (e.g. "wet winding country road"). That means that the operational situation has a tremendous impact on the risk. In a HARA, safety managers consider

- the exposure E, i.e. the likelihood that the system might be exposed to an operational situation,
- the controllability C, i.e. the likelihood that the driver or any other person could mitigate a hazardous event in spite of a system failure, and
- the severity S, i.e. the severity of harm the safety manager expects in case that the hazardous event occurs.

Based on these hazardous events, they derive safety goals, i.e. the top-level safety requirements, whose integrity level depends on the risk assessment of all hazardous events covered by the goal.

Actually, we have different possibilities for shifting the HARA to runtime to yield a H@RT (cf. Figure 4). Selecting one of these possibilities is usually a trade-off decision between flexibility on the one hand and "assurability" on the other hand. For all possible approaches, first, the system must become aware of its current context (context awareness). To this end, we use the situations defined in the HARA during design time as starting point and characterize them by different parameters that can be monitored at runtime - such as speed, weather conditions, traffic context (urban, country road, highway, etc.), or road structure (straight, winding, etc.). There are various different ways to determine the situation using those parameters.

A possible approach is using Fuzzy logic (Klir and Yuan 1995) (Zadeh 1996). Using different measureable values, we map the current situation to linguistic terms. Using different inference rules, we can then derive a likelihood that the system is in a specific situation at the current point in time at runtime. It is not necessary to get precise probabilities. Instead, it is sufficient to use classes of likelihood as we use them in design time analyses, too. This simplification is particularly noteworthy, as it is important to limit the necessary context information to a set of required and sufficient information since we need this information with a sufficiently high integrity. Since autonomous systems are already equipped with numerous sensors for much higher precision, such a coarse-grained derivation of situations with enough confidence seems feasible, e.g. through cross-validation.

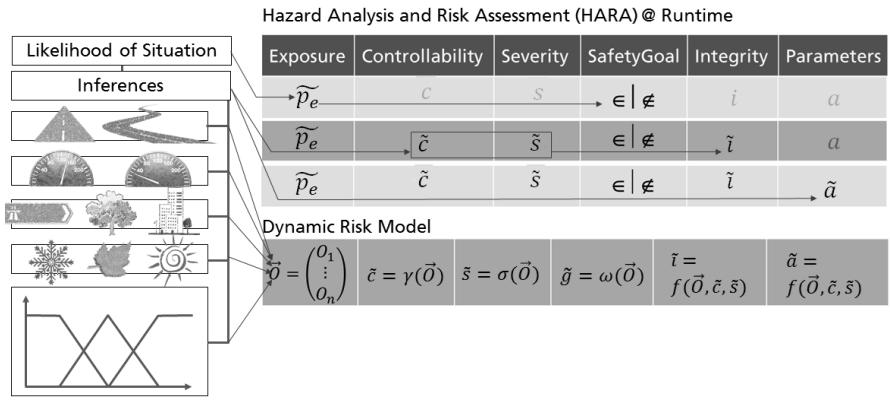

Fig. 4. Dynamic Risk Assessment

The table on the right hand side of Figure 4 shows different alternatives and how the system can use this context awareness for a dynamic risk assessment. In the simplest case shown in the first row, we simply replace the exposure parameter by the runtime likelihood to be in an operational situation considered in a hazardous event. If this likelihood is below a certain threshold, the system neglects this hazardous event during dynamic risk assessment. By doing so, many safety goals can be considered irrelevant for the current situation or, at least, their integrity level can be reduced.

If we accept more freedom, the system can additionally reassess the controllability as well as the severity parameters depending on the situation-specific parameters such as speed or weather conditions (cf. line 2). By this means, the system might further reduce the integrity level. As a further degree of freedom, it is additionally possible to adapt parameters of safety requirements (cf. line 3).

Eventually, as a final, alternative step, it is possible to leave the established way of creating HARAs. Instead of identifying and assessing single hazardous events in the sense of a combination of hazards and operational situations, a situation can be described as a vector of single characteristic parameters. Instead of fixed, static assessments for each hazardous event, (continuous) functions deter-

mine the parameters. By this means, we yield a high degree of flexibility. On the other hand, however, this makes safety assurance much more difficult compared to the minor modifications of established HARAs as described before.

4.2 Dynamic Safety Assessment

Besides the dynamic risk assessment, which primarily focuses on the system's external context, it is additionally important to regard the system's internal safety state in a dynamic safety assessment. To this end, we employ dynamic variabilities defined in dynamic safety concepts. Instead of defining one single solution of how to assure the system's safety, a dynamic safety concept captures different alternatives. However, it is important that the safety concept is not decoupled from the system design.

We use the mechanisms derived from engineering self-adaptive systems. Each component may take various different configurations. A configuration may represent different algorithmic or structural variants of the component's realization at runtime. Additionally, adaptive parameters provide some additional freedom at runtime. From safety engineering, we use the ideas of model-integrated safety engineering (Domis and Trapp 2009) (Domis et al. 2009). This means that the safety models such as safety concepts are directly integrated into functional models. Thus, each component of a system's architecture has its own modular safety concept. When the engineer composes the single components to a system, the underlying tools (semi-) automatically compose modular safety concepts to an overall system-level safety concept as well. Combining both approaches, we define modular safety concepts for each configuration of each component. In fact, defining variability in our safety concept might lead to additional configurations in a component since different safety concepts might require different realizations of a component. In principle, this extends the idea of modular, conditional runtime safety certificates ConSerts (Schneider and Trapp 2010) and Digital Dependability Identities DDI (Schneider et al. 2015). Each modular safety concept therefore defines a safety contract consisting of safety guarantees and safety demands. Since adding configurations to modules increases the complexity and impedes an efficient safety analysis, it is important to provide probabilistic analysis techniques considering system configurations (Adler et al. 2007).

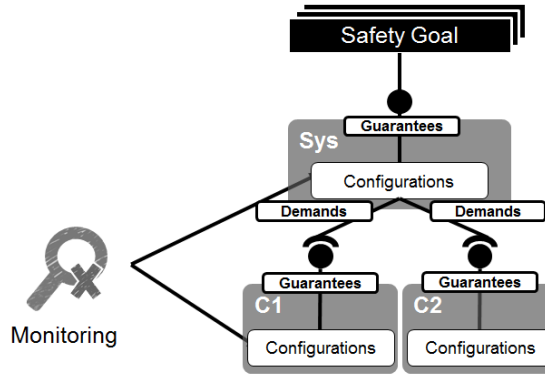

Fig. 5. Mapping of monitors to contract-based safety concepts

As shown in Figure 5, we map system monitors and diagnostics such as error detections, power supply monitors, or any other available runtime status monitor to the components' configurations. Depending on these monitors, the system evaluates which configurations are available from a functional point of view and from a safety point of view. To this end, the system primarily evaluates the demands defined in the safety contracts. Additionally, it is necessary to evaluate any assumption defined in the safety concepts (Wei et al.). This requires a formalization of demands and assumptions. For this particularly purpose, it is however sufficient to provide a mapping function to runtime monitors. By step-wise aggregation of the components' safety contracts, the system becomes aware which safety requirements of the underlying safety concepts and eventually which safety goals are still fulfillable. By this means, we consequently yield a dynamic capability space.

4.3 Self-Assurance and Self-Optimization

By combining both, the risk assessment results and the safety assessment results, the system gets a dynamic safety space. The dynamic risk assessment provides a list of safety goals that need to be fulfilled at the current point in time. Using this input, the system can reduce the capability space provided by the safety assessment to those configuration sets that fulfill the currently demanded safety goals with the required integrity. Within this safety space, the system can optimize its performance by selecting a safe configuration set that optimizes the system's performance goals.

If we summarize this (cf. Figure 6), on the left hand side, we have the goal space, comprising the required safety goals as well as the performance goals. On the right hand side, we have the system's current capability space.

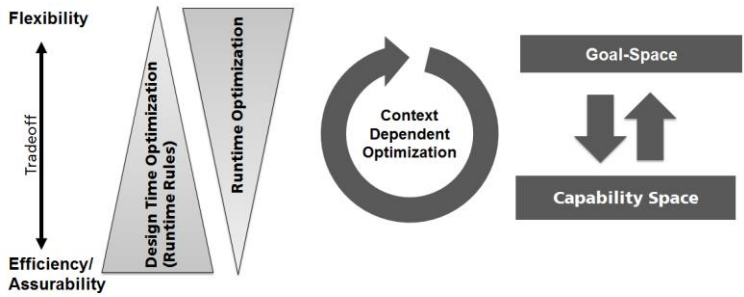

Fig. 6. Optimization trade-off

Those spaces span the frame for a context dependent optimization. This optimization can be focused on design time or runtime. The more aspects we (pre-) optimize during design time, the more efficient the runtime framework will be and the simpler will be the assurance of the runtime framework. However, this purely static approach significantly decreases the flexibility because a design time optimization does not cover unanticipated situations. To increase flexibility, we need to shift the optimization or parts of the optimization to runtime. Runtime optimization is very flexible but decreases system performance and makes safety assurance more difficult. Therefore, a reasonable mix between design time and runtime optimization is a trade-off decision that might be taken differently for different systems and settings.

An example is the definition of pre-defined contexts the system will operate in. One can model and pre-verify these context models at design time. Thus, a designer only needs to define the system's required software applications and their requirements, e.g. with respect to timing behaviour and reliability. The resulting model can be formulated as an optimization problem, which can be solved efficiently during design time yielding optimized configurations, which in turn meet the defined requirements. In this case, the optimization task of the runtime system mainly is to determine the present context and selecting the appropriate configuration. Since this mapping and the selection can be implemented very efficiently, it enables a fast adaptation that is essential for real-time cyber-physical systems.

5 Conclusion

By integrating self-adaptation and safety management, dynamic safety management paves the way to consider the actual case instead of the worst case. This provides the necessary freedom to build safe and yet cost-efficient autonomous systems. One may assume that this approach may reduce the safety margin. However, if we assume that a sound situation detection is given with a sufficient integrity and that static analyses ensure that it is valid to neglect a safety goal in specif-

ic situations, then we will only neglect irrelevant safety goals that are not of relevance for the actual situation at runtime.

Even though there has been work on runtime certification and dynamic safety assurance for quite a while now (Trapp and Schürmann 2003) (Rushby 2007), the field is still at the very beginning. Particularly the assurance of adaptive systems still poses many open challenges (de Lemos et al. 2013) (de Lemos et al. 2017). For successfully using dynamic safety management in the future, it is necessary to rethink established safety principles. However, we are convinced that simply trying to evolve conventional approaches is not an alternative since they will not scale and would require too expensive and/or technically infeasible solutions. Simply relaxing safety demands is no alternative either. Although autonomous systems such as self-driving cars might – in total – reduce the number of accidents, it is in our opinion not an option to accept that lives depend on systems whose safety cannot be guaranteed just for the sake of technical progress.

Dynamic safety management might provide a reasonable means to combine intelligence and safety in spite of their apparently contradicting nature.

References

Heckemann et al. (2011). Safe Automotive Software. In Proceedings of International Conference on Knowledge-Based and Intelligent Information and Engineering Systems. Springer.

Weiss G, Schleiss P, Schneider D, Trapp M (2018) Towards integrating undependable self-adaptive systems in safety-critical environments. Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems. ACM.

Kephart J.O. and Chess D.M. (2003). The vision of autonomic computing. IEEE Computer 36, 1 (Jan 2003), 41–50. https://doi.org/10.1109/MC.2003.1160055

Trapp M, Schneider D. (2014). Safety assurance of open adaptive systems–a survey. In: Models@Run.Time (pp. 279-318). Springer.

Blair G, Bencomo N., France R. B. (2009). Models@run.time. IEEE Computer, vol. 42, no. 10, (pp. 22-27), IEEE.

Cheng B et al. (2014) Using Models at Runtime to Address Assurance for Self-Adaptive Systems. Models@run.time, 101–136. Switzerland: Springer International Publishing.

Ruiz A, Juez G, Schleiss P, Weiss G. (2015). A safe generic adaptation mechanism for smart cars. In proceedings of IEEE 26th International Symposium on Software Reliability Engineering (ISSRE) (pp. 161-171), IEEE.

ISO/FDIS 26262-8:2018(E), Road vehicles — Functional safety, ISO

Klir G and Yuan B (1995). Fuzzy sets and fuzzy logic (Vol. 4). New Jersey: Prentice hall.

Zadeh L. A. (1996). Fuzzy logic= computing with words. IEEE transactions on fuzzy systems, 4(2), 103-111. IEEE.

Domis D and Trapp M (2009). Component-Based Abstraction in Fault Tree Analysis. In Proceedings of International Conference on Computer Safety, Reliability, and Security, pp. 297-310, Springer, Berlin, Heidelberg.

Domis et al. (2009). Safety Concept Trees. In Proceedings of Reliability and Maintainability Symposium. IEEE.

Schneider D and Trapp M (2010). Conditional Safety Certificates in Open Systems. In Proceedings of the first workshop on critical automotive applications: robustness & safety. ACM.

Schneider et al. (2015). WAP: Digital Dependability Identities. In proceedings of IEEE 26th International Symposium on Software Reliability Engineering ISSRE. IEEE.

Wei R et al. (2018). On the need for transitioning model-based assurance cases from design time to runtime. In proceedings of 13th Workshop on Models@run.time. CEUR-WS.org

Adler R et al. (2007). Determining configuration probabilities of safety-critical adaptive systems. In Proceedings of Advanced Information Networking and Application Workshops. IEEE.

Trapp M and Schürmann B (2003) On the Modeling of Adaptive Systems. International Workshop on Dependable Embedded Systems. CMU

Rushby J (2007) "Just-in-Time Certification," 12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS). IEEE.

de Lemos R et al. (2013) Software engineering for self-adaptive systems: a second research roadmap. In: de Lemos R, Giese H, Müller HA, Shaw M, editors. Software engineering for self-adaptive systems II. Lecture notes in computer science, vol. 7475. Springer

de Lemos R et al. (2017): Software Engineering for Self-Adaptive Systems. Re-search Challenges in the Provision of Assurances. In Rogério de Lemos, David Garlan, Carlo Ghezzi, Holger Giese (Eds.): Software Engineering for Self-Adaptive Systems III. Assurances. Cham: Springer International Publishing.