Safety Assurance of Open Adaptive Systems – A Survey

Mario Trapp and Daniel Schneider

Fraunhofer Institute for Experimental Software Engineering
Kaiserslautern, Germany
{mario.trapp,daniel.schneider}@iese.fraunhofer.de

Open adaptive systems are the basis for a promising new generation of embedded systems with huge economic potential. In many application domains, however, the systems are safety-critical and an appropriate safety assurance approach is still missing.

In recent years, models at runtime have emerged as a promising way to systematically engineer adaptive systems. This approach seems to provide the indispensable leverage for applying safety assurance techniques in adaptive systems. Therefore, this survey analyzes the state-of-the-art of models at runtime from a safety engineering point of view in order to assess the potential of this approach and to identify open gaps that have to be closed in future research to yield a safety assurance approach for open adaptive systems.

1 Introduction

The development of safety-critical embedded systems has to follow strict rules and a rigorous safety assurance case is required before a product can be introduced to the market. Developers therefore avoid using flexible and progressive concepts like dynamic adaptation in safety-critical contexts. Many safety standards such as IEC 61508[47] even prohibit the use of techniques like dynamic reconfiguration or self-healing.

Over the last decade, however, new applications have emerged, which are today often subsumed under the popular term cyber-physical systems. In some sense, cyber-physical systems are Open Adaptive Systems (OAS), i.e. systems of systems that dynamically connect to each other (openness) and adapt to a changing context at runtime (adaptive). Industry sees huge economic potential in such systems -particularly because their openness and adaptivity enables new kinds of promising applications in different application domains. Many application domains of cyber-physical systems, however, are safety-critical. This includes, for example, car2car scenarios, plug'n'play operating rooms, or collaborative autonomous mobile machines.

This means that two different worlds, which have intentionally been kept separate, have to grow together in the near future. Using the full potential of OAS without endangering a product's safety is therefore one of the primary challenges today. Regarding the state-of-the-art, however, there are only a few approaches that explicitly address the safety assurance of OAS. Whereas the adaptive systems community mostly considers safety as one of many quality properties, the safety engineering community is still mainly concerned with design time variability, and only a few groups focus on the safety of Open Adaptive Systems. Therefore, safety could easily become a bottleneck preventing the successful transition of a promising idea into business success.

From a safety point of view, there are, in fact, a few approaches that could be extended to assure safety in OAS. For example, some groups are pursuing the idea of safety bags [47], which detect and handle failures at runtime. By this means, even failures that potentially result from system adaptations would be covered so that the system adaptation as such would not be the subject of safety assurance anymore. In practice, however, the effectiveness of such approaches is still very limited. A further alternative would be to assure safety completely at design time by predicting all possible system adaptations and covering the complete adaptation space already during safety assurance at development time. Such approaches could easily run into a state space explosion problem and for open systems in particular, the structure cannot be completely predicted at development time.

Therefore, this article focuses on alternative approaches enabling safety assurance at runtime. To this end, we particularly regard Models@Runtime, which have emerged as a possible means for the systematic development and runtime management of adaptive systems. It is our perception that Models@Runtime as a new paradigm could be an appropriate catalyst for accelerating progress in the safety assurance of OAS. In particular, they seem to provide an efficient basis for the safety assurance of Open Adaptive Systems: Models@Runtime provide a kind of formal basis for reasoning about the current system state at runtime, for reasoning about necessary adaptations, and for analyzing or predicting the consequences of possible system adaptations. This makes dynamic adaptation tractable, traceable and in some sense predictable. Therefore, having explicit Models@Runtime may provide the indispensable leverage needed for applying safety assurance techniques at runtime, hence bridging the gap between traditional adaptive systems and safety engineering research. At the same time, however, a Models@Runtime framework imposes additional complexity that potentially detriments the assurance of safety. As a consequence, it will be important to find the right balance between capabilities and complexity of the Models@Runtime framework on the one hand and the corresponding complexity and feasibility of the safety assurance on the other hand. Moreover, in order to be accepted, any safety assurance concept must still fit into the safety engineers' and certification bodies' views of the world.

Using conventional safety assurance approaches as a reference, however, would immediately lead to the result that dynamic adaptation must not be applied at all. In order to identify the current position and missing steps on the way to safety assurance in OAS, it is nonetheless necessary to know the target we want to reach. Therefore, we have to look ahead in order to get an idea of what such a safety assurance framework based on Models@Runtime could look like. To this end, we use an established, conventional safety engineering lifecycle as starting point which is introduced in Chapter 2. By applying the idea of Models@Runtime to the models and activities of the safety lifecycle we create a projection of a possible future safety assurance framework in Chapter 3. In a subsequent step, we analyze the state-of-the-art with respect to adequate starting points and building blocks for our envisioned future safety assurance framework. The state-of-the-art analysis will thereby be twofold. On the one hand, in Chapter 4, the state-of-the-art of the safety engineering community will be investigated with respect to promising approaches and concepts that might be employed in the context of the envisioned framework and runtime assurance measures. On the other hand, in Chapter 5, the same will be done for the adaptive systems community. In addition, for the adaptive systems community there will also a brief overview on current Models@Runtime approaches that might serve as a technological basis or starting point for the envisioned safety assurance approach. In Chapter 6.1 the state-of-the-art is then being categorized based on the different conceptual classes of safety assurance approaches that have been identified in the context of the envisioned framework. Based thereon, open gaps are pointed out and possible future research directions are devised in Chapter 6.2.

2 Safety Engineering for Traditional Embedded Systems

2.1 Safety Engineering in a Nut-Shell

The precise definition of a safety engineering lifecycle, and particularly of the terms used, depends on the concrete application domain. The principal idea, however, is similar across all safety-related application domains. For the sake of simplicity, we therefore use the terms as defined in the ISO 26262[55], which is the relevant safety standard for automotive systems. It is at the same time one of the most recent safety standards.

The overall goal of safety engineering is to ensure 'freedom from unacceptable risk'[55]. The term risk is defined as the 'combination of the probability of occurrence of harm and the severity of that harm'[55]. Usually, however, it is not possible to directly assess the harm that is potentially caused by a system. Instead, safety managers identify the hazards of a system, i.e., 'potential sources of harm'[55]. In many domains, this vague definition is further refined. In the automotive domain, for example, 'hazards shall be defined in the terms of conditions and events that can be observed at the vehicle level'[55]. Usually, harm is only caused when a hazard, a specific environmental situation, and a specific operation mode of the system coincide. This coincidence is called 'hazardous event'.

The identification of these hazardous events and the assessment of the associated risks is the first step in any safety engineering lifecycle, namely the 'hazard analysis and risk assessment (HRA)' as shown in Figure 1. This step is performed during the very early phases of the development process, at the latest when the system requirements are available.

As a result of this step, safety goals are defined as top-level safety requirements, which have to be incrementally refined during the safety engineering lifecycle. Usually, any safety requirement consists of a functional part and an associated integrity level. The functional part defines what the system must (not) do, whereas the integrity level defines the rigor demanded for the implementation of this requirement. The integrity level depends on the risk associated with the hazardous event, which is addressed by the safety goal. For example, ISO 26262 defines so-called automotive safety integrity levels (ASIL).

Once the safety goals have been defined, the system development continues through different phases like the definition of a network of functions, the system and software architecture, the design, and finally the implementation of the system. In the same way that the validation and verification of the system should run in parallel to the development, the subsequent steps in the safety engineering process should be performed in parallel as well (though this is often not the case in practice). To this end, the available development artifacts are used as input to safety analyses in order to identify potential causes of the identified system failures. A wide range of different analysis techniques is available. Failure Modes and Effects Analysis (FMEA) and Fault Tree Analysis (FTA) are certainly the most widely used safety analysis techniques in practice.

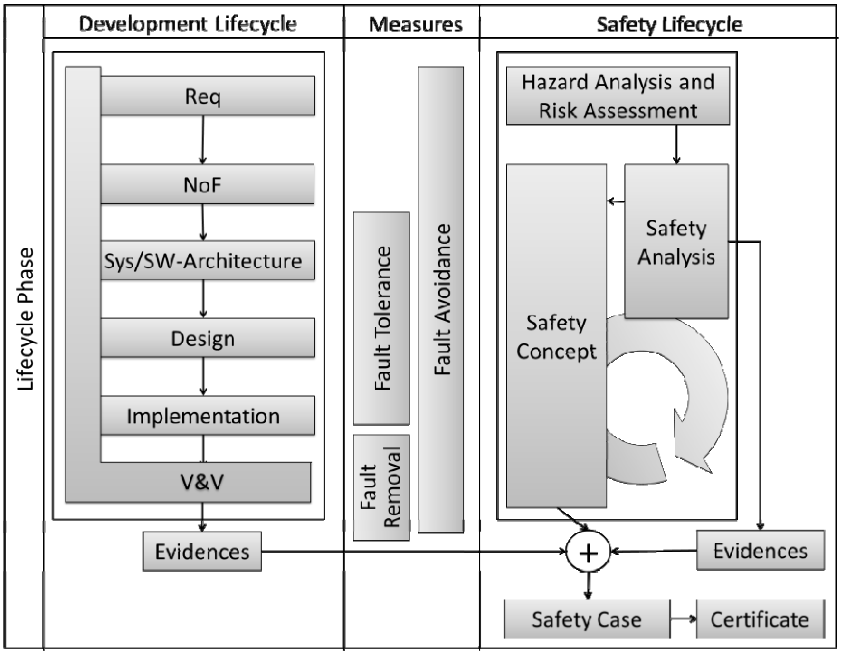

Fig. 1. Safety Engineering Lifecycle and possible countermeasures

Based on these results, a safety manager derives a safety concept. Following the idea of ISO 26262, a safety concept can be defined as a ‘specification of the safety requirements, their allocation to architectural elements and their interaction necessary to achieve the safety goals, and information associated with these requirements. In the same way as the developers incrementally refine the system over the different development phases, the safety manager analyzes the refined development artifacts step by step and refines the safety concept accordingly.

The safety concept plays a very important role in safety engineering. It defines which countermeasures have to be applied and how the measures in combination shall ensure the safety goals. Following the definition of Avižienis et al. [21], there are three principal classes of countermeasures, as shown in the middle of Figure 1. Any measure available can be assigned to one of these classes. First of all, fault avoidance measures shall mitigate the creation of faults from the very beginning. This includes measures such as strict development processes or coding rules. Usually, however, it is not possible to avoid all kinds of faults using such measures. Therefore, it is additionally necessary to apply fault removal measures. This particularly includes validation and verification activities, which try to reveal and remove faults during the development phase. Since we cannot assume that these measures are sufficient to yield a fault-free system, it is also necessary to apply fault tolerance measures. Fault tolerance measures detect and handle errors at runtime in order to prevent system failures.

Finally the safety manager has to define a safety case, which forms the basis for certification. A safety case can be defined as an 'argument why an item is safe supported by evidence compiled from work products of all safety activities during the whole lifecycle.'[55]. Evidence might be anything supporting an argument in the safety case. Evidences of particular importance are the results of validation and verification activities as well as safety analysis results. Since a safety case compiles all evidences that are relevant for proving the system's safety, it is an efficient basis for safety certification.

2.2 Modular Certification

In most domains, safety managers follow a comparable approach to assure the functional safety of systems. Usually, however, the resulting safety certificate is valid for a specific system configuration only. Even a single change requires the system to be recertified. For example, in the avionics domain, even small system changes cause recertification costs approaching or even exceeding the original costs [73]. Considering that in the avionics domain 60%-70% of the overall development costs are caused by verification and certification activities, this leads to tremendous costs for recertification.

Consequently, in the last decade, safety research has focused on approaches called modular or incremental certification, as described in more detail in Chapter 4. As illustrated in Figure 2, the idea of modular certification is that the individual subsystems are modularly certified and provide a modular safety certificate. When the system is integrated, the certification effort shall be reduced to a composition of the subsystem certificates. In fact, most of the current approaches do not consider modular certificates, but modular safety cases, which have to be composed into a safety case for the overall system. The overall system certification is then a traditional, manual process based on the composed safety case.

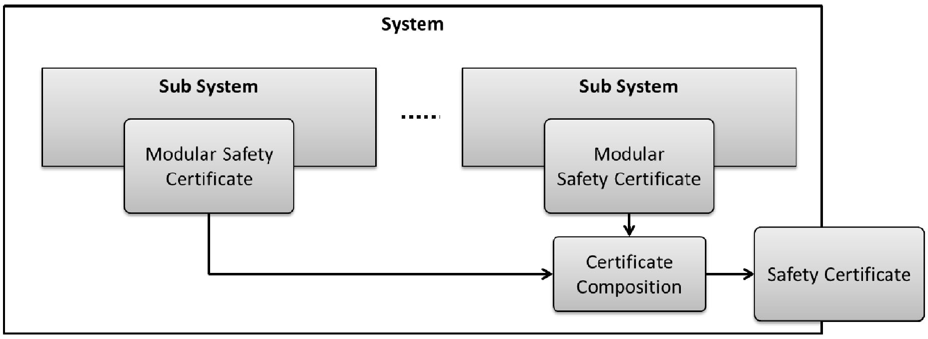

Fig. 2. Principal idea of modular certification

Nonetheless, this simplifies the certification process for complex systems, which are composed of various different systems purchased off-the-shelf or delivered by different suppliers. Even more importantly, such an approach simplifies recertification, since a change of a component only requires recertification of the changed component and re-composition of the overall system certificate.

Such an approach is obviously very interesting for the safety assurance of Open Adaptive Systems, which are subject to continuous changes at runtime. In practice, however, safety research is still dealing with different challenges raised by modular certification at development time.

3 Models@Runtime for Safety Assurance in Open Adaptive Systems

In modular certification, a safety expert assesses the integrated system. If we want to apply such concepts for Open Adaptive Systems, however, there will be no human expert to check the system's safety. Rather, the system must assure its own safety. This leads to a series of new challenges as to how the safety-relevant information can be formalized and utilized an adequate way. Considering that safety research is still solving the problems of modular certification, safety assurance in Open Adaptive Systems seems to be a very challenging endeavor. Extrapolating the current developments of safety engineering, it would take much too long until urgently required safety assurance approaches for Open Adaptive Systems would be available. In the same way as Open Adaptive Systems form a new paradigm in system development, there must be a change of paradigms in safety assurance as well.

Regarding the future safety assurance framework for OAS, we consequently pursued the idea of combining a typical safety assurance approach with the principle of Models@Runtime, as already motivated above. Starting from traditional techniques implies the additional benefit that a clear trace can be provided from conventional safety engineering to the future concepts supporting Open Adaptive Systems, which could facilitate the acceptance of the framework. In essence, we understand the conceptual safety assurance framework as a means to:

- Raise awareness within the research communities for the specific challenges of safety assurance in OAS
- Provide orientation for researchers by interconnecting different kinds of research into a bigger picture
- Provide clear interfaces for future research

In order to create the conceptual safety assurance framework, we incrementally project elements (i.e., typical safety models) of the safety engineering lifecycle to runtime. To do so, we start with SafetyCertificates@Runtime and extend the approach backward step by step along the safety engineering lifecycle. Shifting an element into runtime always implies that corresponding runtime mechanisms need to be established that operate on the element. These are required to automate the tasks that used to be conducted by safety experts. It is obvious that the earlier the shifted element is in the lifecycle, the more engineering activities need to be automated, the more intelligence is required at runtime – and the more difficult it will be for the approach to be realized and accepted.

In accordance with the above, we first describe the ideas of SafetyCertificates@Runtime (section 3.1), then SafetyCases@Runtime (section 3.2), followed by validation and verification of Models@Runtime (section 3.3), and finally Hazard Analysis and Risk Assessment@Runtime (section 3.4). These different options are evaluated in section 3.5 before section 3.6 shows a possible safety assurance framework integrating the different approaches. The framework will finally be the basis for assessing the state-of-the-art and assigning existing work and research directions to the different classes of the framework according to their respective suitability.

3.1 SafetyCertificates@Runtime

Following the idea described above, making safety certificates available at runtime is the first option. SafetyCertificates@Runtime contain all information that is necessary to identify which safety requirements are fulfilled with which integrity by the associated system. Just like conventional safety certificates, SafetyCertificates@Runtime do not contain any white-box information on how the system was realized to yield the certification. A clear advantage of such an approach is that the runtime models and their evaluation can be quite simple and efficient as, for instance, shown by the ConSert approach [70] [71] [76]. This would also imply that an overly complex Models@Runtime framework would not be required, thus alleviating the safety assurance of the framework itself.

Classification Criteria: SafetyCertificates@Runtime are modular certificates that can be interpreted, composed, and adapted at runtime. They are dynamically adapted to represent the safety state of the system at runtime. The certificates of subsystems can be composed at runtime in order to yield an overall safety approval for a given composition.

Using SafetyCertificates@Runtime, it is particularly possible to compose systems at runtime. As illustrated in Figure 3, the individual subsystems provide a runtime representation of the modular certificates (SafetyCertificate@Runtime). In order to assess the safety of the resulting system of systems, the single certificates have to be composed. In order to yield such a Certificate@Runtime, the process is very similar to modular certification. After the subsystem has been developed, it must undergo a manual certification process at design time. Usually, however, the safety assurance of a single subsystem at design time can only yield a conditional certificate, since the certification is based on various assumptions. These assumptions might be concrete demands on other subsystems. For example, there might be a demand that the failure modes of a received signal must be mitigated by another subsystem according to a specific safety integrity level. Other assumptions might consider the integration context in general, such as the maximal number of collaborating subsystems, the type and quality of the communication system used, etc. Consequently, SafetyCertificates@Runtime often follow the idea of safety contracts defining a set of safety guarantees provided by the subsystem and a set of safety demands the subsystems require to be fulfilled by the integration context. This means that they provide runtime information on which safety properties can be guaranteed by the system under the precondition that the defined demands are fulfilled. At runtime, the fulfillment of the demands is checked and the resulting guarantees are derived. Usually, however, safety is not a completely modular property, i.e., the composition of safe components does not necessarily lead to a safe composition, even though the safety demands are fulfilled. Therefore, it is often necessary to perform additional checks in the integration context at runtime (cf. section 5.1).

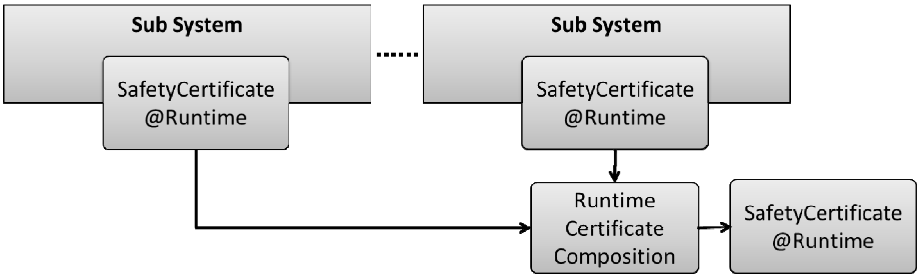
Sub System
SafetyCertificate @Runtime
Sub System
SafetyCertificate @Runtime
Runtime Certificate Composition
SafetyCertificate @Runtime

Fig. 3. SafetyCertificates@Runtime enable dynamic system composition

When subsystems are composed at runtime, it is possible to compose the Certificates@Runtime as well. To this end, the conditions defined in the runtime certificates must be checked. In the simplest approach, a system of systems is considered safe if all preconditions of all conditional certificates are true. Otherwise, the system of system must not be used. In most cases, however, the certificates of the single subsystems are not harmonized with each other. So it is very unlikely that there will be a safe match at all. In fact, such an approach is only reasonable if it is possible to adapt the Certificate@Runtime to the current integration context.

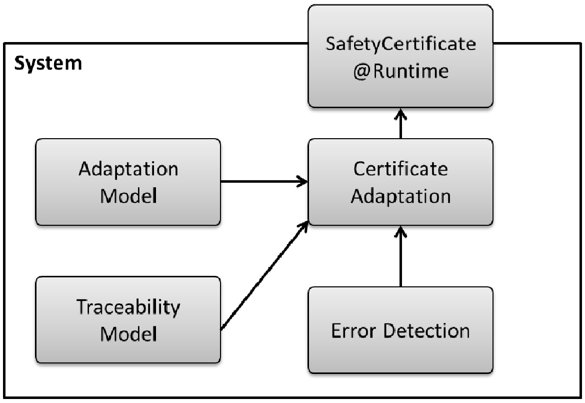

Fig. 4. Runtime adaptation of certificates provides more flexibility

Actually, the possibilities of how a runtime certificate could adapt are as versatile as adaptation in general. There could be different pre-defined variants that are dynamically selected in a given context. Or there could be more sophisticated and flexible adaptations of the certificate. From a safety point of view, however, predictable, traceable, or even provable solutions are more likely to be accepted. Some ideas are illustrated in Figure 4. If we assume, for example, that there is an adaptation model defining how the system adapts in certain situations, this information could also be used to adapt the certificate.

Alternatively or additionally, a traceability model could be used to identify those elements that are affected by system adaptations and to derive necessary certificate adaptations. In fact, traceability Models@Runtime might play an important role for the safety assurance of OAS. An efficient impact analysis is of utmost importance in traditional safety engineering in order to identify necessary changes and thus reduce the revalidation effort. As long as the effects of adaptations can be traced back to anticipated classes of system changes, even complex system adaptations could be handled by simple variants in the SafetyModel@Runtime.

As a further extension, it could be possible to use error detection mechanisms to adjust the runtime certificates using up-to-date runtime error information.

Usually, the adaptation goal for the certificates is to provide the best possible guarantees in the given context. The context, in turn, is usually given by the set of externally fulfilled safety demands and the internal state of the system.

3.2 SafetyCases@Runtime

The more adaptive a system is, the more difficult it is to consider all the different adaptations in a Certificate@Runtime. This particularly increases the effort at design time since the complete adaptation space must be considered in the certification process. Alternatively, it could therefore be another option to provide SafetyCases@Runtime. Safety cases are direct input to certification. In contrast to certificates, however, they include the complete argument of why a system is considered safe.

A good safety case model includes a complete breakdown of top-level safety goals to the detailed requirements realized in the system. And it particularly includes the evidence proving that the arguments used are sound and that the requirements have been fulfilled.

SafetyCases@Runtime therefore provide more information at runtime and enable more flexible adaptation of the system. In consequence, however, they are more complex to handle, since there is no pre-certification at design time and all the steps from a safety case to certification have to be shifted to runtime as well. As a further consequence, this will most likely reduce the acceptance of such an approach compared to SafetyCertificates@Runtime.

Classification Criteria*: A SafetyCase@Runtime is a formalized, modular safety case that can be interpreted and adapted at runtime. Based on the interpretation, it can be dynamically checked to which extent the safety goals of subsystems are met. With adaptation, the line argument can be adjusted to system adaptations. In addition, the revalidation of evidences at runtime must be supported in case system adaptations lead to the invalidation of evidences.*

As shown in Figure 5, SafetyCases@Runtime extend the idea of SafetyCertificates@Runtime. Instead of explicitly defining the adaptation of the certificates, it is possible to describe the adaptation of the safety cases and use the SafetyCases@Runtime to adapt the safety certificates automatically.

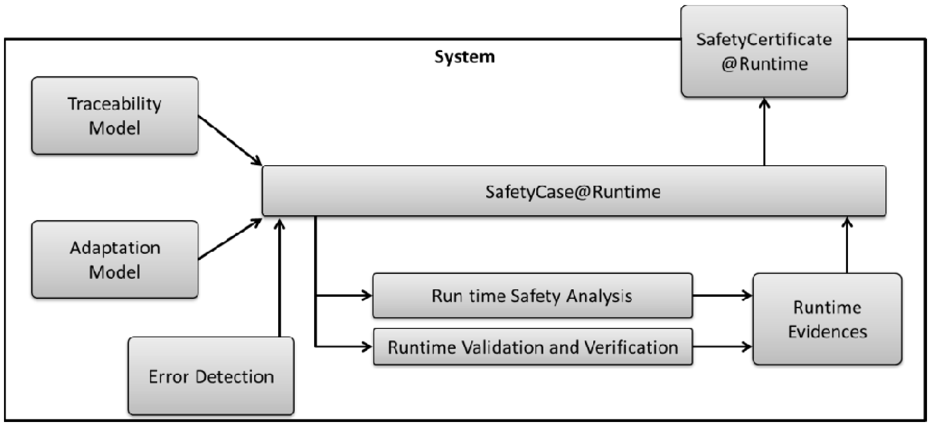

Fig. 5. Conceptual model of how safety cases could be used at runtime

A certificate certifies that certain safety guarantees are fulfilled. The safety case models the argument of why these guarantees are fulfilled. If a safety case is adapted at runtime, the resulting argumentation should enable the system to conclude autonomously which safety guarantees can still be provided at which integrity level.

A basic element of safety cases are evidences, which are, for example, verification and validation results or the results of safety analyses. By shifting safety cases to runtime, it is possible to adapt (1) the argumentation and/or (2) the evidences to the currently given context. As regards the adaptation of the argumentation, a very straightforward solution would be to include different variants of the argumentation.

In more complex versions, more intelligence might be integrated that is able to derive new lines of argumentation.

With regard to the evidences, it is necessary to attach constraints to the evidences used in the safety case. At runtime it is then necessary to evaluate whether or not these constraints are still fulfilled. If not, there are basically two combinable options. First, it is possible to find an alternative argumentation based on the remaining valid evidences – including argumentations that potentially require a reduction of the safety guarantees that can be provided in the given context. A second option would be the revalidation of evidences. This requires the capability to re-perform safety analyses as well as validation and verification activities at runtime. For SafetyCases@Runtime, let us assume that this revalidation is limited to repeating the checks defined at design time in order to provide the evidence. This presumes that the system adaptation does not lead to a change of requirements or a change of the system's interface.

If the respective pass-criteria are met, the newly created evidence can replace the invalidated original evidence and be integrated into a new argumentation. Otherwise, the evidence remains invalid and the system must either find an alternative line of argumentation or invalidate the affected safety goals.

3.3 V&V-Models@Runtime

SafetyCases@Runtime already provide a very flexible means for safety assurance at runtime. Some system adaptations, however, might require a new set of verification and validation checks to provide the evidence required for the argument. Moreover, it might be desirable to be able to remove the faults identified during runtime V&V instead of being limited to only checking the pass-criteria.

For the former aspect, it is necessary to additionally enable the system to define verification and validation suites autonomously. Realizing the latter aspect even requires systems that are able to localize the causing faults, and to isolate or even remove them. Considering how difficult this step easily becomes for developers at design time, it is obviously a very challenging task to shift these activities to runtime.

***Classification Criteria**: V&V-Models@Runtime presume that all models that are necessary to perform validation and verification activities (e.g., test cases, pass/fail-criteria etc.) can be interpreted and adapted at runtime in order to create new evidences after system adaptations.*

3.4 Hazard Analysis and Risk Assessment@Runtime (HRA@Runtime)

In the previous alternatives, we assume that the requirements and the resulting safety goals are not adapted. As a consequence, it has only been necessary to adapt the argumentation that the safety goals are still met in spite of system adaptations based on the safety case and the evidences created at runtime. Some adaptation approaches, however, also consider a change of requirements at runtime. If we apply the safety lifecycle to the idea of Models@Runtime, this means that we require a hazard and risk analysis at runtime, i.e. that the system must adapt and extend the hazard and risk analysis and potentially have to adapt and extend the set of safety goals. By doing so, the complete existing argumentation for a changed safety goal might be invalidated. For new safety goals, an argumentation is completely missing. On the one hand, this type of runtime assurance certainly provides the highest possible flexibility. On the other hand, however, it requires very intelligent mechanisms for defining a safety argumentation and generating the necessary evidence autonomously at runtime.

Classification Criteria: *HRA@Runtime implies that a hazard and risk analysis model can be interpreted and adapted at runtime. This includes the identification of new hazards and the reassessment of existing hazards after adaptations at the requirement level.*

3.5 Evaluation of the Different Approaches

Regarding the approaches described above, they obviously build upon each other. This means that a HRA@Runtime requires V&V-Models@Runtime, which in turn require SafetyCases@Runtime and so on. So it is necessary to decide to which extent we want to shift the safety lifecycle to runtime. This results in a trade-off decision. From a safety point of view, it is certainly preferable to leave as much responsibility as possible with a human expert. Consequently, it would be reasonable to have only SafetyCertificates@Runtime. From an adaptation point of view, however, it is preferable to have as much flexibility as possible in order to tap the full potential of dynamic adaptation. In consequence, this would require shifting elements of the complete safety lifecycle to runtime.

In order to further illustrate this trade-off, Figure 6 shows the relations of the different approaches to their acceptance on the one hand and to their flexibility on the other hand. Acceptance in this case refers to the probability of acceptance by safety authorities and legislation. Since there is no practical experience available, this is a qualitative estimation. First, we assume that acceptance is inversely proportional to the responsibility and intelligence given to the system. Second, the acceptance of an approach is usually inversely proportional to its complexity. Or vice versa: The simpler an approach can be realized, the more probable is its acceptance. For obvious reasons, it is very probable that the required intelligence as well as the resulting complexity will grow with the number of safety assurance steps that are shifted to runtime. Consequently, in our opinion, SafetyCertificates@Runtime have the best chances of being accepted, whereas the acceptance of an HRA@Runtime (i.e., shifting all safety assurance activities to runtime) is quite improbable. As a further aspect, acceptance will be higher if the Safety-Models@Runtime are reconfigured at runtime to predefined variants only, whereas acceptance will rapidly decrease if the safety models themselves are adapted more flexibly at runtime.

Flexibility, on the other hand, represents the degree of which different types of adaptations are supported. More precisely, in this case we refer to the type of adaptation used to adapt the system itself and not to the type of adaptation used to adapt the safety models, since different adaptation approaches might be used for the system itself on the one hand and the safety models on the other hand. In order to classify the supported flexibility of system adaptations, we differentiate between three basic classes. We first differentiate between 'known unknowns' and 'unknown unknowns'. In the former case, we assume that the system can only adapt to a runtime context that has been anticipated at design time. In the latter case, we assume that the system needs to flexibly adapt to situations not anticipated at design time. In consequence, the system structure or behavior is hard or even impossible to predict. We have further subdivided the 'unknown unknowns' into adaptations at the design level on the one hand and at the requirements level on the other hand. In the former case, we assume that the requirements can remain unchanged and an adaptation of the realization (e.g., at the architecture level) is sufficient to adapt to the context given. In the latter case, the adaptation also includes the adaptation of existing and/or the definition of new requirements.

SafetyCertificates@Runtime can only be used to address 'known unknowns' since an adaptation of certificates to an unpredicted context is not possible without considering the underlying safety case, which forms the indispensable basis for a sound argumentation of a certificate's validity. But even for 'known unknowns' the configuration space might be too large to be covered completely by variants at the certificate level. Therefore, it might be reasonable to use SafetyCases@Runtime already to efficiently support 'known unknowns'.

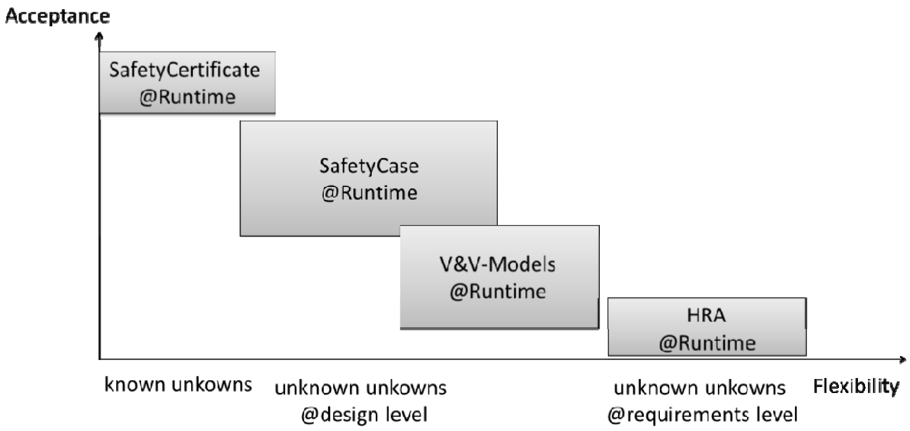
Acceptance
SafetyCertificate @Runtime
SafetyCase @Runtime
V&V-Models @Runtime
HRA @Runtime
known unkowns
unknown unkowns @design level
unknown unkowns @requirements level
Flexibility

Fig. 6. Qualitative relations between acceptance and flexibility

If we consider 'unknown unknowns' at the design level, this means especially that the requirements and thus the safety goals remain unchanged. Depending on the degree of system modifications required for the adaptation, SafetyCases@Runtime or V&V-Models@Runtime are therefore sufficient. While SafetyCases@Runtime are limited to running predefined validation and verification activities at runtime, V&V-Models@Runtime additionally support the modification of V&V models, e.g., the modification of test cases or pass/fail criteria. The more flexible the system adaptations must be, the more likely it is that V&V-Models@Runtime approaches will be required in addition to SafetyCases@Runtime.

As soon as the adaptation to ‘unknown unknowns’ also requires an adaptation of requirements, it is additionally necessary to adapt the hazard and risk analysis and the resulting safety goals at runtime. As described above, it is not sufficient to identify new hazards or to re-assess the associated risk at runtime. In fact, the system must be able to appropriately create or adapt all affected artifacts along the complete safety lifecycle.

3.6 Conceptual Safety Assurance Framework for Open Adaptive Systems

Models@Runtime obviously provide a wide range of possible approaches for the safety assurance of Open Adaptive Systems and it is certainly not possible to pick out one particular approach that leads to the best trade-off between flexibility and acceptance in general. In fact, we believe that it will be necessary to integrate different approaches into an assurance framework in order to use the advantages and compensate for the disadvantages of the different approaches.

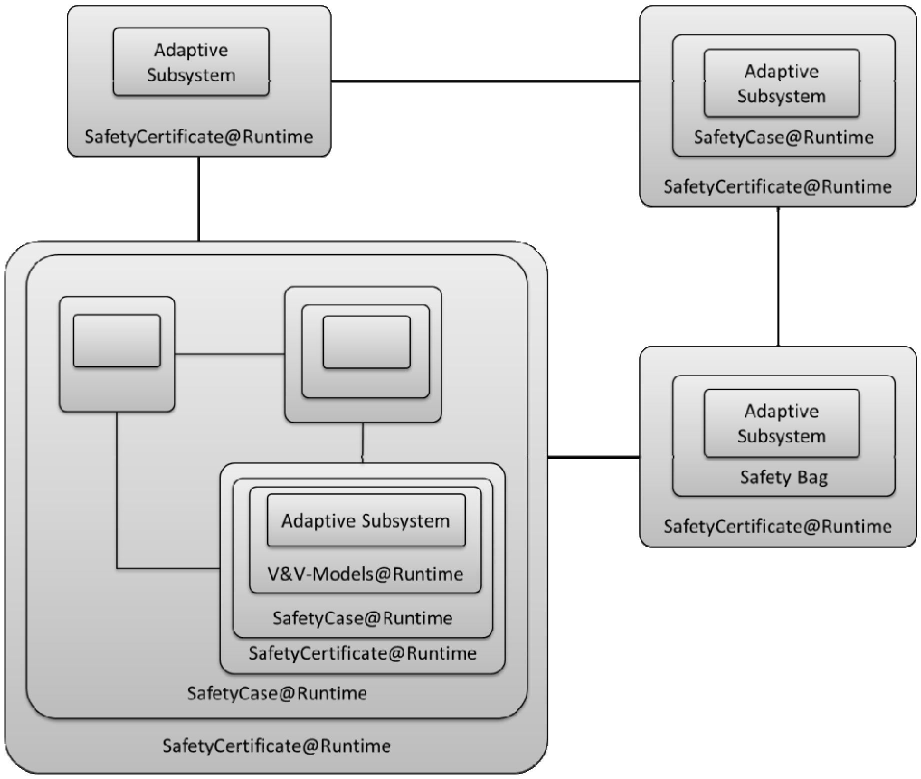

Fig. 7. Integrated Conceptual Safety Assurance Framework for OAS

Some ideas for such a framework are shown in Figure 7. Learning from traditional safety engineering, we recommend using modularity as the basic ingredient for a safety assurance framework from the very beginning. First, this obviously reduces complexity. Second, this enables us to use different assurance approaches for different modules. In this context, we use the term module very flexibly to express a modularized entity that can range from a complete system in a system of systems to a single software component. Since the required types of adaptation usually differ widely across the different modules, it reasonable to limit more complex assurance approaches to those modules that actually have to adapt very flexibly.

Following the idea of modular certification, it seems to be reasonable to use SafetyCertificates@Runtime as the basic building blocks to enable the modularization and runtime integration of different subsystems. In this case, SafetyCertificates@Runtime are the common denominator enabling the combination of a wide range of different assurance approaches used for the single modules.

Assume, for example, that we have a module that adapts to 'known unknowns' only, as shown in the upper left corner of Figure 7. Then it might be sufficient to perform the major safety assurance activities at development time and limit the runtime models to SafetyCertificates@Runtime only. If we have a module that has too large a configuration space or that also adapts to 'unknown unknowns', it might be necessary to have SafetyCases@Runtime as well, as shown in the upper right corner of Figure 3-5. As described above, SafetyCases@Runtime are an extension of SafetyCertificates@Runtime, so a runtime certificate is still available at the module's interface, facilitating the safe integration of the components. In some cases, a module might adapt so flexibly that we will need V&V-Models@Runtime or even an HRA@Runtime. However, realizing this is very complex, so it seems reasonable to keep the complexity of such modules very small. To this end, it is helpful that the modularization of the framework can be applied recursively to achieve hierarchical decomposition, as illustrated in the lower left corner of Figure 7. This decomposition additionally illustrates an alternative way of composing SafetyCertificates@Runtime. If we assume systems of systems for example, each providing a SafetyCertificate@Runtime, the single systems are usually sufficiently independent from each other that composition at the certificate level is likely to be sufficient. If we assume the runtime integration of different software modules running on the same platform, there are usually tight interdependencies. Merely the fact that they share the same resources, for example, creates a safety-relevant dependency. For this reason, it is likely that additional evidences will be required for proving that the integration of the single modules is safe as well. Therefore, it might also be reasonable to have Safety-Cases@Runtime at the integration level.

The acceptance of sophisticated assurance approaches, in particular, is very low. An alternative way to ensure the safety of highly adaptive systems is given by different traditional approaches, particularly in the field of fault tolerance. So-called safety bags (cf. e.g., [47]), for example, are a typical concept for monitoring a function to detect anomalies and trigger counter-reactions. Assuming that it would be possible to define a safety bag that can detect and handle any safety-related failure of an adaptive module, it would not be necessary to provide further assurance of that module. Though such approaches are based on traditional mechanisms rather than Models@Runtime, they would nonetheless fit into our conceptual framework as shown in the lower right corner of Figure 7.

Summarizing, this conceptual framework has been created based on a prognostic evolution of state-of-the-practice safety engineering lifecycles using the idea of Models@Runtime as a catalyst, which it uses to build a conceptual bridge between the world of safety engineering on the one hand and Models@Runtime on the other hand. Being based on safety engineering principles makes acceptance of the approach more likely. Yet it provides sufficient flexibility to integrate various different solution approaches based on Models@Runtime. Therefore, in the subsequent chapters we will analyze the state-of-the-art with respect to the suitability of the different approaches to fit into specific parts of the framework. We will further identify 'white spots' and interfaces for future research.

4 State-of-the-Art from the Safety Engineering Community's Point of View

As already discussed above, Open Adaptive Systems have long been beyond the scope of the safety engineering research community. However, the work that has been done in the direction of modular certification might well prove to be a sound foundation for tackling the safety-related challenges posed by Open Adaptive Systems. Moreover, there are some first approaches advocating the introduction of runtime measures. The state-of-the-art presented in this chapter consequently focuses on approaches from the safety engineering community that either belong to the aspiring research field of modular certification, or that advocate certain runtime measures for the context of OAS.

In general, modular certification can be characterized as a means for the modularization of safety cases. The safety case is modularized such that components developed by different suppliers, and components that are likely to be replaced or reused, specify a self-contained modular safety case. These modular safety cases, specified by the module developer, are connected on the system level by the integrator to build the system safety case. In order to be able to assemble the system safety case, each module must provide an interface specification containing the module's guaranteed behavior and the behavior demanded of other interacting modules. Demands are necessary since the behavior of the module at hand depends on the behavior of the other modules it is interacting with. Therefore, the module at hand is only able to give guarantees under the premise of a certain behavior of the interacting modules. These premises are called demands and, together with the afore-mentioned guarantees, shape demand/guarantee contracts.

The idea to use contracts as a metaphor for describing the interaction of components with mutual obligations and benefits can also be found in approaches that do not specifically focus on safety, such as those presented in Section 5.1.1. These approaches do, however, focus on specifying the nominal behavior and/or specific quality characteristics of components and do not consider a component's failure behavior (how does the component fail, what failures of other components can the component tolerate), which is essential for safety-related modularization.

4.1 Foundational Work on Modular Certification

To enable modularization of safety cases, it is crucial to formalize the relevant information in an appropriate way. As a first step, it is necessary to enable modular safety analyses. A corresponding starting point is given by techniques from the class of failure logic modeling (FLM) [48], where the failure logic is modeled separately for each component and the failure logic model defines how deviations at the input of a component propagate to deviations at the outputs of the component. Architecture models, which are (should be) available anyway, define how the components are connected. Based on the architecture, it is therefore possible to also connect the failure logic models of the component, and the failure propagation throughout the overall system can be analyzed automatically. Prominent solutions in this regard are the 'Hierarchically Performed Hazard Origin and Propagation Studies' – HiP-HOPS [49] and the 'Failure Propagation and Transformation Notation' – FPTN [50]. Another approach that is based on safety contracts has been proposed by Hawkins and McDermid[51]. Moreover, component fault trees [52] provide an extension for the well-known technique of fault trees that supports the modular, component-based definition of fault trees [53]. Fault trees and CFTs generally also enable probabilistic analyses by annotating faults with respective probabilities of occurrence. Since it is often not possible to determine concrete probabilities for a given event, Foerster and Schneider introduced an approach that uses intervals of probabilities to efficiently deal with such uncertainties during development [54].

4.2 Modular Certification as Represented by Current Standards

Some concepts related to modular certification have already been adopted by current standards and thus found their way into the state of the practice. This is particularly true for the fields of automotive systems and avionic systems because the trend towards modularized architectures has been particularly strong in these fields. The following paragraphs provide a brief overview of the corresponding standards and the modularization concepts they advocate.

4.2.1 ISO 26262

The international standard ISO 26262 for the functional safety of street vehicles contains the so-called concept of Safety Element out of Context (SEooC) [55]. A SEooC is defined as a component for which there is no single predestinated application in a specific system. Therefore, the SEooC developer does not know the concrete role the product has to play in the safety concept. Subsystems, hardware components, and software components may be developed as SEooCs. Typical software SEooCs are reusable, application-independent components such as operating systems, libraries, or middleware in general.

For SEooC development, the standard suggests specifying assumed safety requirements and developing the system according to these requirements. When the SEooC is to be used in a specific system, the system developer has to specify the demanded requirements, which can subsequently be checked against the assumed requirements. If there is a match between the demanded and the guaranteed (assumed) requirements, system and component are compatible. The standard does not provide any suggestions or methods on how to identify safety requirements such as to increase the chance that assumed and real requirements match. Neither does the standard provide information on how to perform the verification of the assumed requirements during integration of the SEooC. The standard specifies a relatively coarse-grained process for embedding a SEooC development into the standard's safety lifecycle. In general, SEooC integration is expected to be done at development time and thus there is no explicit support for open systems where components are to be integrated dynamically. Moreover, there is no explicit support with respect to the management of variabilities, be it at development time or at runtime.

4.2.2 DO-297

The DO-297 [56] standard regulates the modular certification of components in an Integrated Modular Avionic (IMA) system. The terminology of the standard talks of incremental acceptance instead of modular certification. Acceptance is defined as the confirmation of a certification body that a module of an IMA system (a general-purpose execution platform or an application) fulfills its specification. This acceptance can be achieved for an IMA system and is one building block of the final certification, with the latter always being in the context of a specific airplane or engine. The wording incremental has been chosen because the process of the DO-297 allows step-wise acceptance of single modules of a system and because it allows incrementally extending a system with new applications, without having to re-certify all the modules in the system.

4.3 State-of-the-Art for Modular Certification Approaches

This section briefly describes a selection of prospective modular certification approaches. All these approaches are briefly described and their applicability in the context of Open Adaptive Systems is considered.

4.3.1 Concepts for Modular Certification by Rushby

Rushby provides some theoretical considerations on the use of modular certification for software components in IMA architectures. The goal is to enable the certification of software components in order to allow them to perform their functions in a given (aircraft) context based solely on assumptions about other related software components. Three key elements were identified as the potential backbone of a corresponding approach [60]:

1. *Partitioning* creates an environment that enforces the interfaces between components; thus, the only failure modes that need be considered are those in which software components perform their function incorrectly, or deliver incorrect behavior at their interfaces.

2. *Assume-guarantee reasoning* is a technique that allows one component to be verified in the presence of assumptions about another, and vice versa.
3. *Separation of properties into normal and abnormal properties*. Abnormal properties capture behavior in the presence of failures.

To ensure that the assumptions are closed and the system is safe, three classes of properties that must be established using assume-guarantee reasoning were identified:

1. *Safe function* ensures that each component performs its function safely under all conditions consistent with its fault hypothesis;
2. *True guarantees* ensure that each component delivers its appropriate guarantees;
3. *Controlled failure* is used to prevent a 'domino effect' where the failure of one component causes others to fail, too.

It is important to note that the publication presents conceptual foundations but does not provide concrete solutions. Still, the presented concepts are clearly relevant and likely to be of avail for future work in the context of the envisioned framework.

4.3.2 Modular Goal Structuring Notation

The Goal Structuring Notation (GSN) [61] is a graphical notation for modeling a safety argument, which is the core part of every safety case. A safety case has been defined in the context of the GSN as follows:

'A safety case communicates a clear, comprehensive and defensible argument that a system is acceptably safe to operate in a particular context.'

Therefore, a safety case serves the purpose of specifying a comprehensive argument to prove the safety of a system. To this end, the GSN allows modeling tree-like arguments beginning with safety goals, and iteratively connecting them through chains of logical argumentation and sub-goals, with the evidences created during system development. Evidences can be performed tests or analysis reports from an FMEA or an FTA that are used for underpinning the fulfillment of the goals.

In order to deal with modular systems and modular certification, there is an extension to GSN that allows modularizing safety cases [62]. The interface of a safety case module is defined by a set of public items that are available for use in other safety case modules and a set of items that the safety case module at hand demands from other modules. Those items can be goals, evidences, and context.

A strategy for the construction of a modular safety case architecture is given in [63]. These guidelines are based upon the guidelines for general modular system design and comprise the following requirements:

- Modules must be as independent as possible.
- Modules must exhibit high cohesion and low coupling.
- Modular safety cases and safety case architectures must be constructed top-down.
- Modules must have well-defined interfaces.
- All modular dependencies must be captured.

In summary, modular GSN is a graphical notation that allows modeling modular safety arguments. As described above, there are also product-related guidelines for the specification of modular safety arguments. Openness and adaptivity are not explicitly addressed, whereas the modularization concepts would at least provide a starting point for corresponding augmentations. Apart from that, it has been shown that the GSN can be utilized in conjunction with a software product line approach [64]. Considering SafetyCases@Runtime, a GSN-like notation might be a possible starting point. Usually, however, the single elements of a GSN-based safety case are described in natural language. Using GSN at runtime will require an appropriate means for formalizing the notation in order to enable runtime evaluation and adaptation.

4.3.3 The Generic Safety Case in DECOS

The DECOS (Dependable Embedded Components and Systems) project [65] was a European Integrated Project in the FP6 Embedded Systems area which ran from 2004 to 2007. The main objective of the project was to make a significant contribution to the safety of dependable embedded systems by facilitating the systematic design and deployment of integrated systems [66]. In order to reach this objective, a generic safety case approach for incremental certification was developed, which improves the efficiency of the certification process and thus shall facilitate significant cost savings during the development of safety-critical systems.

According to [66] and [67], modularity is achieved by separating the certification of core services and architectural services from applications (enabling generic application safety cases (for the class of applications) and individual (specific) safety cases by supporting independent safety arguments for different distributed application subsystems).

1. *Separating certification of architectural services from certification of applications:* The clear interfaces between the platform and the applications provided via the platform interface are a prerequisite for the separation of the certification of architectural services from the certification of applications.
2. *Separating certification of different distributed application subsystems:* The integrated architecture allows the independent certification of different application subsystems, instead of considering the system as an indivisible whole in the certification process. The safety argument for each subsystem is provided to the integrator by the suppliers along with the compiled application code of the jobs in the corresponding subsystem. In order to construct the safety argument for the overall system, the system integrator combines the safety arguments of the independently developed subsystems and acquires additional evidence, such as the results of a formal verification of the architectural services. The decomposition of the overall system into encapsulated subsystems with different criticality levels reduces the overall certification efforts and allows focusing on the most critical parts. Furthermore, the separate certification of subsystems is beneficial if functionality is reused in different systems. In this case, the safety argument for the functionality needs to be constructed only once.

Like the approaches above, DECOS supports the modularization of development time safety artifacts. Openness and adaptivity are not explicitly supported and all certification activities are to be conducted at development time. However, the incremental approach adopted by DECOS seems to be well suited to handling variability at development time, maybe in conjunction with an adequate software product line approach as it has already been explored for the GSN.

4.3.4 Vertical Safety Interfaces

The goal of the VerSaI (*Ver*tical *Sa*fety *I*nterfaces) method is to assist the integrator of an integrated architecture in checking whether the application software components are able to run safely on the execution platforms of the system, and if so, provide assistance in generating appropriate evidence [72].

Before *safety compatibility* between the application and the platform can be checked with the VerSaI approach, demands and guarantees have to be specified. Demands are typically used to express all the properties a platform needs to have for an application to be executed safely, whereas guarantees represent the safety-related properties the platform possesses. A compatibility check is successful if a sound argument for the fulfillment of the demands with the available guarantees can be established. To enable tool-supported integration, the VerSaI approach offers a semi-formal language for modeling these demands and guarantees. The language consists of a number of elements, each representing a certain type of demand or guarantee exchanged by an application and a platform. This implies the noteworthy fact that there is a finite number of language elements and, therefore, also a finite number of dependencies that can be expressed with the language. First evaluations have shown that this is suitable, because the typical service relationships between an application and a platform are finite and regular, too, which is also the reason why platforms have been standardized in the first place.

The final step of the method is to check whether each demand can be met with the guarantees identified as relevant in the previous step. In contrast to conventional interfaces, it is usually not possible to simply match demands and guarantees, respectively. In fact, it is necessary to generate an additional fragment in the safety case providing the arguments and evidences that the demands of the platform are met by the guarantees given by the platform. To this end, this step is supported by a so-called strategy repository. The repository contains expert strategies that are selected and presented to the integrator and describe what guarantees are needed to fulfill the current type of demand and how to generate a piece of evidence containing a sound argument.

Like the other modular certification approaches, Versa focuses on development time integration. However, it provides some interesting aspects that could be of relevance for SafetyModels@Runtime. First, it already provides a formalization of the interface language, thus facilitating automated checks of interface consistency. Second, it introduces first ideas of how missing fragments of a safety case could be generated automatically. Though this is currently not possible without human interaction, some ideas could be a starting point for extending/modifying safety case argumentations at runtime. However, VerSaI is limited to the vertical interface between application and platform software. This has the advantage that the typical safety requirements concerning this vertical interface are quite limited - thus simplifying the formalization of the interface language. For OAS, this approach would have to be extended to horizontal interfaces as well. However, those interfaces are usually application dependent so that the formalization approach used in VerSaI cannot be easily extended to support horizontal interfaces as well.

4.4 Runtime Certification

First ideas with respect to runtime certification have been introduced by Rushby[68], [69]. In contrast to most of the other approaches presented in this section (which are already quite mature and have partly even been proven in use), Rushby`s work remains on a rather conceptual level. However, considering its motivation and the solution concepts presented, it is very important in the context of safety assurance of OAS.

In the first publication, Rushby presents the general idea that certain elements of a conventional certification case could be transferred to runtime. The focus is on those elements that apply formal analyses (e.g., automated verification) to representations of a software component and its local safety or other critical requirements. Formal analyses are usually employed at development time to formally verify that a component follows a certain prescribed behavior. At runtime it would be possible to employ monitors to control the component's behavior during execution and to trigger adequate measures when deviations occur. Such monitors might be synthesized from the model that specifies the component's behavior using very similar—and equally trustworthy—techniques as those used in formal verification.

In the second publication, Rushby outlines a framework in which the basis for certification is changed from compliance with standards to the construction of explicit goals, evidences, and arguments (generally called an 'assurance case'). He then describes how runtime verification can be used within this framework, thereby allowing certification to be partly performed at runtime. The core of this approach is again the usage of runtime monitors, which have been defined outside the context of an assurance case in order to dynamically monitor assumptions, anomalies, and safety, respectively.

Overall, the presented work is still very conceptual but nevertheless provides a good starting point for future work in the context of the envisioned framework. One of the main ideas advocated by Rushby, namely to shift parts of the safety assurance measures into runtime to cater to the specific challenges within OAS, has also been adopted by us in the framework presented here.

4.5 Discussion

From the state-of-the-art in safety engineering approaches that support modularization it becomes apparent that openness and adaptivity have been largely out of scope and thus are not explicitly supported by most approaches. Moreover, even though the umbrella term 'modular certification' seems to suggest otherwise, all of the considered approaches and standards rather focus on the modularization of pre-certification safety artifacts, particularly safety cases. The only exceptions are the approaches on runtime certification, which build on pre-certification of the system. Since most approaches have been designed to support engineers during their development time activities, they lack an adequate degree of formalization, which would be required for automated runtime evaluations. All of these approaches nevertheless provide sound conceptual starting points for new safety engineering approaches for Open Adaptive Systems. As for supporting adaptivity, some of the presented modular certification approaches (such as the GSN) have at least been used in conjunction with software product lines. Others, such as the approach introduced by Rushby, DECOS and VerSaI, seem to be well-suited in this regard as well.

As the considered approaches are more or less established in the safety engineering community, using them as a starting point for Models@Runtime certainly increases the probability of acceptance. Since the approaches are mainly based on safety cases, they would provide a good starting point for research in the direction of SafetyCertificates@Runtime or for SafetyCases@Runtime.

Apart from the modular certification approaches discussed above, the runtime certification approach presented by Rushby builds on dynamic monitoring (and repair) of the systems'/components' behavior. This approach could fit into the category of V&V-Models@Runtime. Based on the conceptual descriptions, however, it seems that mainly predefined verifications can be executed at runtime. So depending on the concrete realization of these concepts, they will rather support the re-validation of evidences as part of SafetyCases@Runtime.

5 State-of-the-Art from the Adaptive Systems Community's Point of View

Some of the first significant research efforts for adaptive systems emerged from the middleware community, where adaptive middleware platforms have been designed to meet the new demands of flexible, distributed heterogeneous systems. Examples in this regard are the solutions proposed by Blair et al. [4], Kon et al. [5], Capra et al. [6], and Truyen[7]. These solutions were mainly designed to enable adaptability (i.e., reconfiguration of the middleware or platform to fit a given setting) or even self-adaptation (i.e., an adaptive middleware or platform that dynamically adapts itself to provide optimized service functionality and quality in any situation). A related field of research, where the topic of self-adaptivity also gained momentum quite early, is the field of adaptive quality of service (QoS) assurance. Corresponding research has mostly focused on communication systems and end-to-end consideration of QoS. The results have been platforms, middleware, and frameworks enabling adaptive QoS.

It was soon recognized that quality assurance for adaptive systems is an important topic with significant scientific challenges. Initial corresponding research efforts have mostly focused on the issues of validation and verification (V&V) of adaptive systems. First results were based on development time V&V, but recently we have seen that V&V measures are being increasingly shifted into runtime. The upcoming topic of Models@Runtime seems to be a catalyst in this regard. Thus, even more capable Models@Runtime-based approaches for runtime V&V can be expected in the future.

In recent years, one main research focus of the community has been to investigate sound engineering methodologies for adaptive systems. Such methodologies ideally span all typical phases of software development (from requirements engineering to the validation of the final product) and explicitly consider important non-functional properties. This methodological research focus has been pushed by community research roadmaps [1]and has been advocated strongly by conferences in the area of adaptive systems, e.g., the SEAMS symposium [8] and the SASO conference [9]. In the context of engineering frameworks, the different fields of adaptive systems research are growing together ever more. The current Models@Runtime research landscape underlines this trend, since researchers from the fields of adaptive middleware, V&V, and engineering methodologies are working together to develop seamless approaches combining all these important aspects under the umbrella of the Models@Runtime topic[2][3]. Relatedly, Baresi and Ghezzi argue that the clear separation between development-time and run-time is blurring and is probably doing so even further in future [74].

From the perspective of the envisioned safety assurance framework, there are consequently two categories of approaches that will be considered in more detail in the following:

1. Approaches concentrating on V&V in the context of adaptive systems. V&V is here not necessarily aimed at safety assurance. Nevertheless, the approaches can be valuable input for future approaches in the context of the envisioned framework. A short overview of the state-of-the-art will be provided and the assurance scope of the different approaches will be considered. Note that completeness cannot be a goal for this article, thus we rather tried to identify a representative set of approaches covering the most important different classes.
2. Frameworks and approaches for adaptive systems that enable the utilization of Models@Runtime for different relevant concerns. Such approaches provide a possible technological basis and therefore define the frame the envisioned safety assurance framework would have to be integrated into. The approaches will be briefly presented and analyzed with respect to their runtime assurance capabilities and their usage of Models@Runtime. Again, completeness was not the goal. For this part of the state of the art we also compiled a possibly representative set of approaches to indicate the current status quo of Models@Runtime approaches in relation of assurances – and safety in particular.

5.1 Approaches Using Validation and Verification as a Means for Assurances

The approaches considered in this section focus on ensuring certain properties through the application of adequate V&V techniques. Some approaches rely on development time measures alone, whereas others utilize runtime measures or a combination of both. For both cases, this section will provide an overview of the respective state-of-the-art. Prior to that, however, there will be a paragraph on contract-based design, since this is an enabling technology for efficient V&V. Moreover, safety contracts and assume-guarantee reasoning are likely to be enabling technologies for important parts of the envisioned framework.

5.1.1 Design by Contract

About twenty years ago, Meyer introduced a set of basic principles of Design by Contract in the context of his Eiffel language [23]. Since then, a wide range of related approaches have been developed for the specification and utilization of different kinds of functional and non-functional contracts. Beugnard et al. provide a recent overview of the general use of Design by Contract concepts in the domains of embedded systems, component architectures, and service oriented architectures [24]. The work in the respective domains is classified according to a scheme introduced in an earlier publication by the authors [25]. Essentially, the types of contracts are classified into four levels:

1. Syntactic (or basic): The goal is to make the system work. It is generally specified with Interface Definition Languages (IDLs), as well as typed object-based or object-oriented languages. It ensures the components can be assembled.
2. Behavioral: The goal is to specify each operation. It is generally specified with a couple of assertions: a precondition and a post-condition. It ensures the operations offered and required are not only syntactically compatible but also semantically.
3. Synchronization: The goal is to specify the coordination of operations. It can be specified with an automaton labeled with operations. It ensures the operations are used in the proper order.
4. Quality of Service: The goal is to quantify a few features associated with operations. Performance, availability, and quality of result can be specified and negotiated at that level.

An interesting and widely recognized approach for contract-based design (even though not specifically addressing adaptive systems) is the Rich Component Model (RCM). The RCM is the backbone of the embedded systems design approach developed in the SPEEDS project (Speculative and Exploratory Design in Systems Engineering) [26]. One primary goal of the RCM is to optimize the reuse of embedded applications. Safety-relevant applications are explicitly included. The main ideas forming the foundation of the approach are described in [28].

The language typically used to describe such contracts is hybrid automata as shown in [27], [28] and [29]. There are formal definitions for the semantics of the hierarchical and horizontal composition of the contracts, which allows checking the fulfillment of system-level requirements after the system has been integrated, using a model checker for example. The formality of the approach increases the achievable degree of automation while equally increasing the upfront effort for modeling the system. The RCM is therefore a modeling paradigm that allows specifying the contract interface of a modular safety argument.

In relation to assurances and adaptable systems, Inverardi et al. recently presented a theoretical assume-guarantee framework for adaptable systems [30] that can be used as a basis for establishing runtime contracts and thus also for V&V in adaptive systems. The major aim of this framework is to define efficient conditions to be proved at runtime to guarantee the correctness of the adaptation of a composed adaptive system.

Conditional Safety Certificates (ConSerts) are a means for facilitating safety certification in the context of OAS [70] [71] [76]. This is one of the approaches explicitly addressing Open Adaptive Systems. There are three main differences between ConSerts and standard certificates that are owed to the nature of open systems: A ConSert is not static but conditional; it usually comprises a number of variants; and it must be available in an executable (and composable) form at runtime. Conditions within a ConSert manifest in relations between potentially guaranteed safety requirements (denoted as guarantees for the remainder of this article) and the corresponding demanded safety requirements (i.e., demands). The demands always represent safety requirements relating to the environment of a component, which consequently cannot be verified yet at design time. A ConSert therefore certifies that the guarantees will hold with acceptable probability under the precondition that the specified safety demands are fulfilled by the environment. Variants come into play because ConSerts usually comprise not only one but a series of different potential guarantees. Eventually, the ConSerts must be available at runtime in an executable representation and the systems need to possess mechanisms for composing and analyzing these runtime models. Using these means makes it possible to establish and maintain safety contracts at runtime that span all levels of a composition hierarchy through pairs of ConSert-based guarantees and demands.

In the same way as standard certificates, ConSerts shall be issued by safety experts, independent organizations, or authorized bodies after a stringent manual check of the system. To this end, it is mandatory to prove all claims regarding the fulfillment of safety requirements by means of suitable evidence. The guarantees that can be provided by a system usually depend on the fulfillment of demands. On the one hand, these demands might directly relate to the required functionalities of other systems. In other cases, some evidences must be acquired at the integration level, since safety is not completely composable. To this end, ConSerts support the concept of so-called runtime evidences. The resulting variability (of the fulfillment of demands) ultimately leads to variants and conditions within the safety case, which are the basis for the definition of ConSerts.

In terms of the conceptual assurance framework, ConSerts belong to the class of SafetyCertificates@Runtime. But they also support single elements of SafetyCases@Runtime through the instrument of runtime evidences.

5.1.2 Approaches Utilizing Development Time V&V for Assurances

In [31], Zhang and Cheng introduce a method for constructing and verifying adaptation models using Petri nets. In [32], linear temporal logic is extended with an ‘adapt’ operator for specifying requirements that a given system must match before, during, and after adaptation. An approach for ensuring the correctness of component-based adaptation was presented in [33], where theorem proving techniques are used to show that a program is always in a correct state in terms of invariants. [34]introduces a formal model of reconfiguration and an associated set of high-level system dependability properties that can be verified. Giese and Tichy introduced a development-time hazard analysis approach for analyzing all configurations a self-adaptive system can reach during runtime [35]. In [75], Becker et al. present a further development time verification technique for the invariant verification of structural properties. This technique has been designed to be appropriate for large multi-agent systems that are subject to structural adaptations at runtime.

Mohammad and Alagar recently introduced a formal approach for the specification and verification of trustworthy component-based systems [36] that advocates formal specifications and dedicated safety properties as a basis for V&V. The properties can be defined as constraints (such as time or data constraints) at the component level and are to be understood as invariants over the component behavior. The behavior can be defined using timed automata. Eventually, the specifications enable automated analysis and verification (through model checking) of the considered properties.

All of the above approaches have in common that they try to analyze (with respect to safety or other specific properties) all possible variants that a given system might assume during runtime. Based on the analysis results, engineers can implement adequate measures to improve or ensure the considered properties.

5.1.3 Approaches Utilizing Runtime V&V for Assurances

Runtime V&V measures are typically applied in a complementary way together with corresponding development-time activities. On the one hand, there are runtime verification techniques that utilize runtime monitoring to record software execution traces that can then be analyzed [37]. On the other hand, there are approaches that employ quantitative model checking at runtime as an assurance technique for the context of adaptive systems (e.g., [38], [39], and [40]). In [43], Goldsby et al. present AMOEBA-RT, a run-time monitoring and verification technique that provides assurance (based on dynamic model checking) that dynamically adaptive software satisfies its requirements. Calinscu and Grunske introduced the QoSMOS (QoS Management and Optimization of Service-based systems) framework for the development of adaptive service-based systems that are able to manage their QoS adaptively and predictably [44]. QoSMOS utilizes probabilistic model checking at runtime to evaluate if the system satisfies the given QoS requirements. In the traditional development-time versions of these kinds of approaches, the analysis of temporal-logic properties (including probabilities, costs, and rewards) is commonly used to assess relevant non-functional properties of a system. At runtime, such analyses can be performed on a model base that is continually updated as the underlying system evolves. In general, this introduction of runtime measures for the context of adaptive systems is particularly promising since traditional development-time techniques do not scale sufficiently well. Moreover, at runtime, detected issues can be addressed directly with adequate adaptations (i.e., countermeasures). A short related survey (which is not limited to V&V) considering runtime assurance techniques for adaptive systems has recently been published by Calinescu[42]. A further approach that is particularly focused on safety has been proposed by Priesterjahn et al. in [41]. The main idea of this approach is to ensure the safety of adaptive systems during runtime by checking whether reconfiguration is allowed based on associated hazard probabilities and potential damage that would be imminent after the reconfiguration. To this end, adapted hazard and risk analysis techniques are applied during runtime.

5.2 Frameworks for Adaptive Systems and Models@Runtime

5.2.1 MADAM and MUSIC

The MADAM (Mobility and Adaptation Enabling Middleware) European project and its follow-up MUSIC (Self-Adapting Applications for Mobile USers In Ubiquitous Computing Environments) aimed at providing techniques and tools for reducing the time and effort needed to develop self-adaptive mobile applications [10][11]. To this end, these projects propose an architecture-centric approach where dynamic adaptation is realized in an application-independent adaptation middleware. Architectural models of the applications are made available at runtime and serve as a basis for reasoning about and controlling the adaptation. Meta-models for the specification of these models are provided by means of a dedicated component framework.

In order to realize runtime adaptation, MADAM and MUSIC employ an application-independent adaptation middleware that is implementing a typical adaptation control loop with the following responsibilities:

1. Monitor both system and user context. The system context consists of system resources such as battery level, CPU utilization, memory usage, and network resources. The user context subsumes information on the environment and on the user's (maybe correlated) needs.
2. Analyze the context and the context changes that occur and plan reasonable changes of the system. To this end, utility functions are used to assess which implementation variant of a certain component type would fit the given adaptation goals best. On the system level, global utility functions are used (which can aggregate the component-level utility functions) to compute the overall utility of an application. This allows evaluating all the different configuration possibilities (i.e., it is a brute-force approach) and the most useful one in the given circumstances can be chosen at the end.
3. Implement the changes – preferably without noticeably interrupting the operation of the system.

Regarding assurances, MADAM and MUSIC explicitly address the management of functional and non-functional properties. However, the properties are only addressed in a generic way and managed via 'best-effort' without 'hard' guarantees.

5.2.2 DiVA – Dynamic Variability in Complex, Adaptive Systems

The European DiVA project can be considered as a predecessor of the MADAM/MUSIC series. In detail, the project had the following main research objectives [45]:

- To provide both build-time and runtime management of the adaptive system (re)configuration of co-existing, co-dependent configurations that can span across several administrative boundaries in a distributed, heterogeneous environment.
- To provide efficient management of the number of potential configurations that may grow exponentially with each new variability dimension.
- To increase the quality and productivity of adaptive system development and help the designers to model, control, and validate adaptation policies as well as the trajectory from one safe configuration to another.

DiVA tackles these challenges by applying and combining techniques from the fields of software product lines (SPL), model-driven engineering (MDE), and aspect-oriented modeling (AOM). Moreover, DiVA has a strong focus on utilizing such Models@Runtime, in accordance with the Models@Runtime paradigm. In [46], the DiVA contribution is summarized as follows:

At design time, engineers can avoid manually designing all of the system's possible configurations and transitions by explicitly defining an adaptive system as a Dynamic Software Product Line (DSPL). At runtime, the system analyzes the context and explicitly constructs a suitable configuration using AOM techniques. It also validates this configuration using traditional MDE techniques: invariant checking, simulation, and so on. Finally, the system automatically generates a safe reconfiguration script to actually adapt the running business system. If the produced configuration is not consistent, the system simply discards the configuration and derives a new one. Since the running business system has not been adapted yet, it is not necessary to perform a rollback. This process is open to evolution—designers can make the DSPL evolve by seamlessly adding or removing variants, constraints, rules, and so on.

Note that assurances were not the focus of DiVA and non-functional properties were only considered in a generic way. Still, the management of generic properties through models at runtime and runtime self-adaptation was foreseen.

5.2.3 Robocop, Space4U and Trust4ALL

The main goal of the ROBOCOP, Space4U, and Trust4ALL [12][13][14] series of European projects was to establish an adequate component-based architecture and middleware for OAS. According to [15], Robocop introduced a component-based framework for high-volume embedded devices with a focus on robust and reliable operation, upgrading, and component trading, while the focus of Space4U was on the validation, maturation, and extension of the Robocop architecture by introducing fault management, power management, and terminal management. Trust4All essentially extended the component-based middleware developed in the course of its two predecessors with respect to a trust management framework.

Correspondingly, according to the Trust4All innovation report [16], the project 'has defined, designed and developed a middleware software architecture specifically targeted at embedded systems that require a predefined level of trust, due to the nature of the services they provide. The project focuses on the trustworthiness-related aspects of the middleware software architecture in domains such as home medical care, security and automation, as well as on-the-move applications, for which dependability is particularly important'. A further important result of the project is the ISO/IEC 23004 standard on middleware, where seven of the eight parts of the standard were contributed by Trust4All (Architecture, Component Model, Resource and Quality Management, Component Download, Fault Management, System Integrity Management, and Reference Software).

In essence, the main scientific contribution of Trust4All, the trustworthiness management approach, is enabled through a trustworthiness model and a trust management framework model. The assurance scope of Trust4All can be classified as 'assurance of trust-related properties', although the reputation- and recommendation-based approach is not compatible with safety assurance in a traditional sense (i.e., certification would not be possible on that basis). Trust4All explicitly supports self-adaptation for assurance purposes, utilizing a runtime configurable fault management mechanism [14].

5.3 Discussion

Adaptive systems and Models@Runtime frameworks and approaches contribute the technological basis and knowledge for representing and utilizing runtime models for different concerns. Regarding the assurance and management of non-functional properties, however, these approaches remain very generic and are not designed to provide 'hard' guarantees. Accordingly, these approaches do not provide a sufficient methodological backbone, which is indispensable for safety assurance and certification.

Due to reasons of complexity, development time V&V as the sole measure for ensuring important properties of an adaptive system is only really feasible for closed adaptive systems. In contrast to OAS, for closed adaptive systems it is generally possible (although potentially very complex, depending on the applied adaptation concepts) to conduct sufficient safety analysis based on holistic system models already at development time. Therefore, one commonality of these approaches is that they focus on closed systems and on specific adaptation concepts that facilitate controlling the size of the adaptation space.

The runtime V&V approaches provide specific concepts for dynamically obtaining and evaluating V&V-related information in an adaptive systems context. These techniques would obviously be well suited for tackling challenges related to the runtime V&V parts of the envisioned framework. However, there is no conceptual integration with existing safety engineering approaches up to now. Nor is there support with respect to variability within the certificates, the safety case, and correspondingly the dynamic V&V measures. In other words, there can only be one 'static' certificate that is to be validated and verified, which consequently limits the flexibility of the open adaptive system, as elaborated before in this article. Nevertheless, in conjunction with a sound and comprehensive safety engineering backbone, these approaches would be a good starting point for future research and could play a vital role in safety assurance for OAS.

An approach that has an explicit focus on safety and is thus particularly relevant for this article has been proposed by Priesterjahn et al. in [41]. This approach is well suited to exemplify what has been stated above. The main idea of this approach is to ensure the safety of adaptive systems during runtime by checking whether reconfiguration is allowed based on associated hazard probabilities and potential damage that would be imminent after the reconfiguration. To this end, a compositional hazard and risk analysis technique is applied during runtime. However, all the safety engineering activities that are typically applied in addition to the safety analyses in order to get a system certified are omitted. Under the premise that safety-critical applications need to be certified, these steps would still be required. Assuming that corresponding safety engineering and certification were done at development time already, this would constrain the flexibility of the approach since a given system would need to be pre-analyzed comprehensively with respect to the acceptability of the failure probabilities of its configurations. A further potential problem of the approach is that emergent safety properties within a system of systems, such as common cause failures, feature interactions, and emergent dysfunctions, are not addressed.

The ConSerts approach directly addresses the idea of SafetyCertificates@Runtime. It is therefore one possible starting point for a safety assurance framework. Additional ConSerts support runtime evidences, which are a first step towards SafetyCases@Runtime. The approach has been successfully applied in different industry applications, which underscore the principal applicability of the idea of SafetyCertificates@Runtime.

6 Evaluation

6.1 Status Quo

Obviously, there are different kinds of approaches that address different aspects of safety assurance at runtime. The following tables summarize the main findings in the different communities.

Approach	**Safety Engineering**			
	supported	**foundation**	**status quo**	**open issues**
Certificate @Runtime	∅	[60]	- no established approach - modular certification provides a sound basis	- formalization - variability - runtime representation
SafetyCase @Runtime	([68], [69])	[60], [61], [62], [64], [65], [66], [67], [VerSaI]	- many design approaches supports modular safety cases - safety case models available - assumes human interaction (no formalization) - first ideas on runtime certification at conceptual level only	- formalization - runtime representation - adaptation of argumentations - realization of runtime evidences
V&V @Runtime	∅	∅	not considered	
HRA @Runtime	∅	∅	not considered	

For the safety engineering community, it is obvious that runtime assurance has not been in the focus of research. Actually, there is no approach that deals with modular certificate models. The reason for this could be that certificates as such do not play an important role at development time. In fact, they are not more than a piece of paper issued at the end of an assessment, which can be used as evidence in a super-ordinate safety case. This means that certificates are not direct working artifacts for safety engineers. The importance of certificate models mainly arises from the need to dynamically compose systems, which requires formal representation at the information level of the certificates (and an explicit specification of the variation points) that can be evaluated at runtime.

Nonetheless, there is a series of approaches that provide valuable starting points and that could be extended to SafetyCases@Runtime. Most of these approaches need to be further formalized in order to be used at runtime. Many safety case notations are still based on informal textual information as they are intended to be used by a human safety expert. Based on such formalization, it would be possible to evaluate SafetyCases@Runtime and identify invalidated evidences, for example. In order to use the full potential of SafetyCases@Runtime, appropriate approaches are required to dynamically adapt the line of argumentation used in the safety case at runtime. However, there are currently no approaches that consider doing that.

There exist first ideas on how to use runtime verification to support certification at runtime [68], [69]. However, these approaches still remain at the superficial level of concepts and ideas. The dynamic adaptation of V&V models, such as test cases or pass/fail criteria, or even adaptation of the hazard analysis and risk assessment (HRA) is completely outside the scope of the safety engineering community. In other words, the V&V measures that are shifted into runtime are always completely predefined at development time already.

Approach	**Models@Runtime**			
	supported	**foundation**	**status quo**	**open issues**
Development-Time Assurance of Adaptive Systems	[31], [32], [33], [34], [35], [36]		- promising results available - limited to few groups	- maturing approaches towards applicability and acceptance - integration with concepts like SafetyCertificates@Runtime to support open systems as well
Certificate @Runtime	[ConSert]	∅	- a first approach is available, utilizing variable certificates and Models@Runtime	- could provide a good add-on to design time assurance approaches for supporting open systems

SafetyCase @Runtime		[37], [38], [39], [40], [41], [42], [43], [44]	- some research has focused on runtime execution of predefined V&V steps - currently independent solutions that can hardly be combined - no complete coverage of safety assurance	- integration of different approaches to support complete safety assurance - currently no direct support to SafetyCases@Runtime
V&V @Runtime	∅	∅	- currently no approaches available	
HRA @Runtime	∅	∅	- currently no approaches available	

Regarding the adaptive systems community, a lot of work has been done regarding the development time verification of adaptive systems and runtime execution of predefined verification steps. Also from a safety point of view, a focus on development time verification is certainly preferable. Regarding the typical characteristics of OAS, however, such an approach appears not to be sufficient. Therefore, the idea of having runtime verification is a good extension. However, the different approaches seem to be quite independent from each other. Each of the single approaches covers only one aspect of runtime safety assurance, and it is mostly unclear how the different approaches could be combined into an integrated framework. Nonetheless, they provide a very good basis for providing evidences in the context of SafetyCases@Runtime. Some work is also available on SafetyCertificates@Runtime, which already considers aspects such as runtime evidences. Obviously, there seems to be a good basis and a lot of potential could be tapped by a more efficient combination and integration of the different approaches.

6.2 A Possible Roadmap to Safety Assurance for OAS Using Models@Runtime

Summarizing the status quo, there is already a lot of work available that directly or indirectly supports the safety assurance of OAS. However, most approaches seem to be quite independent from each other. None of the approaches alone is sufficient and complete to assure safety in OAS, but all of them provide individual puzzle pieces for a safety assurance approach. Since they have been developed in isolation, it is however not possible to simply combine them. Nonetheless, the efficient combination of existing approaches would already lead to significant progress.

From our point of view, a first step towards an efficient safety assurance approach for OAS therefore seems to be to consider the big picture of safety assurance instead of regarding single elements in isolation. To this end, a safety assurance framework, comparable to the one used in this article would be required, but it certainly needs to be more mature. Such a framework would provide the big picture the single puzzle pieces have to fit into – thus simplifying classification and combination of the different approaches. Moreover, it would define a principal understanding of what safety assurance for OAS could look like. Such a commonly accepted foundation is a prerequisite to obtaining acceptance of the assurance approaches by certification bodies and safety assessors.

Taking the framework defined in this article as a starting point, a possible roadmap to safety assurance is illustrated in Figure 8.
From an industry point of view, the most urgent need for safety assurance is certainly for open systems in which the single systems only adapt to anticipated situations. Therefore, assurance of the single systems could be achieved using available assurance approaches applied at development time. If these approaches are tightly integrated into traditional safety engineering lifecycles, safety assurance could happen completely at development time. All remaining assumptions and variabilities that must be resolved at runtime could be modeled using SafetyCertificates@Runtime, which would also enable safe composition of systems of systems at runtime.

Such an approach is also very likely to be accepted by safety assessors. Design time assurance of adaptive systems is in some sense already considered in safety standards. For example, ISO26262 explicitly defines how assurance has to deal with large configuration and parameter spaces. Alternatively, from a safety engineering point of view, adaptation is nothing but an indistinguishable part of the functionality extending the system's state space, which must be completely covered by all safety assurance activities. The available development time assurance approaches tackle the resulting challenges. SafetyCertificates@Runtime are very similar to modular certification approaches. Definition and assurance of the certificates take place at development time, and only the composition of certificates is shifted to runtime. In order to be accepted, the verification of the composition mechanisms must become an additional element of the development time verification activities. This is of course also true in general for all runtime mechanisms that are introduced as part of the safety assurance framework. As described in the previous chapter, this scenario can also be extended with alternative approaches, such as extended safety bags.

If we regard open systems that require more flexible adaptations including adaptations to unanticipated situations, or if the dynamic composition happens at the level of software components instead of systems, it is additionally necessary to provide SafetyCases@Runtime. To this end, approaches facilitating the modular specification of safety cases, as they exist in the safety engineering community, could be used as a starting point. As mentioned above, this requires formalization of the notations in the first step. For many application scenarios, however, the capability to dynamically adapt the line of argumentation could be optional. Instead, it might be sufficient to integrate different variants into the safety case at design time and to reduce runtime responsibility to the resolution of these variabilities. This would require further extension of existing safety case approaches. As an additional aspect, it is necessary to provide evidences at runtime. This step can be supported by different existing runtime V&V approaches as described above. Nonetheless, some extensions are required in order to transfer the existing approaches from the idea of a stand-alone solution to an integrated part of SafetyCases@Runtime.

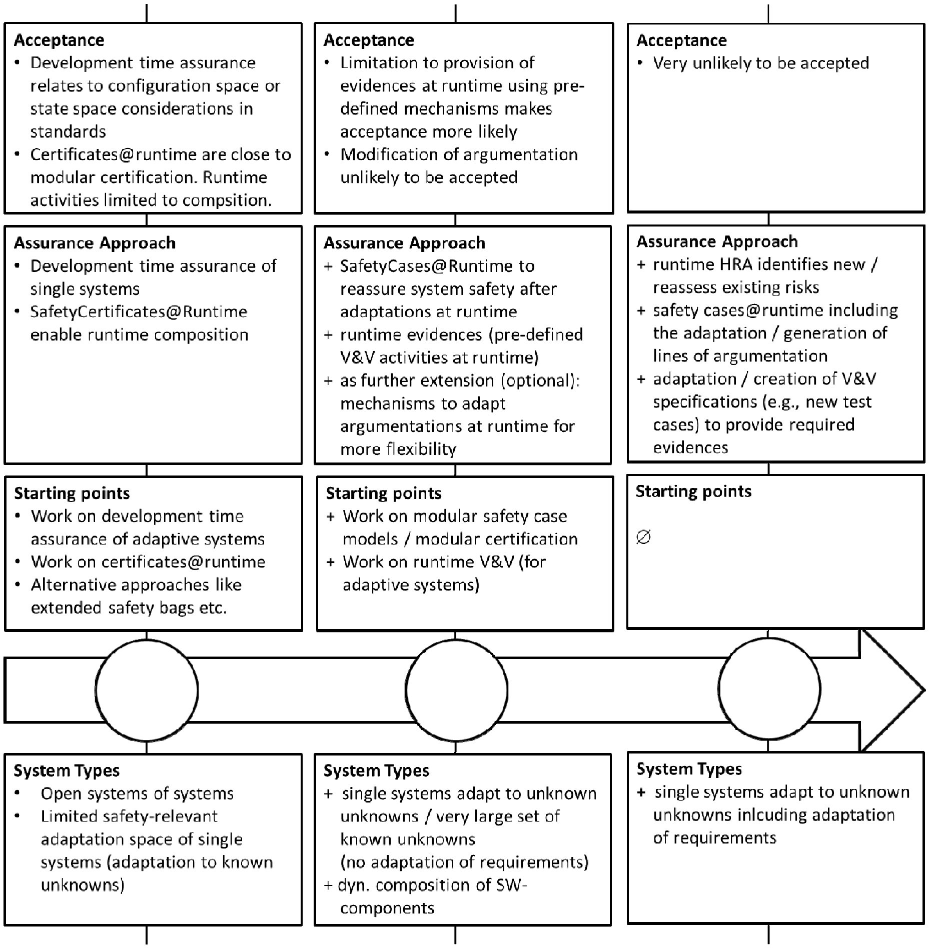
Acceptance
• Development time assurance relates to configuration space or state space considerations in standards
• Certificates@runtime are close to modular certification. Runtime activities limited to compsition.
Acceptance
• Limitation to provision of evidences at runtime using pre-defined mechanisms makes acceptance more likely
• Modification of argumentation unlikely to be accepted
Acceptance
• Very unlikely to be accepted
Assurance Approach
• Development time assurance of single systems
• SafetyCertificates@Runtime enable runtime composition
Assurance Approach
+ SafetyCases@Runtime to reassure system safety after adaptations at runtime
+ runtime evidences (pre-defined V&V activities at runtime)
+ as further extension (optional): mechanisms to adapt argumentations at runtime for more flexibility
Assurance Approach
+ runtime HRA identifies new / reassess existing risks
+ safety cases@runtime including the adaptation / generation of lines of argumentation
+ adaptation / creation of V&V specifications (e.g., new test cases) to provide required evidences
Starting points
• Work on development time assurance of adaptive systems
• Work on certificates@runtime
• Alternative approaches like extended safety bags etc.
Starting points
+ Work on modular safety case models / modular certification
+ Work on runtime V&V (for adaptive systems)
Starting points
∅
System Types
• Open systems of systems
• Limited safety-relevant adaptation space of single systems (adaptation to known unknowns)
System Types
+ single systems adapt to unknown unknowns / very large set of known unknowns (no adaptation of requirements)
+ dyn. composition of SW-components
System Types
+ single systems adapt to unknown unknowns inlcuding adaptation of requirements

Fig. 8. Possible roadmap to safety assurance of OAS using Models@Runtime

The main aspect in this approach that complicates acceptance is the runtime provision of evidences. However, assuming that the verification activities are already defined at development time and 'only' executed at runtime, this is likely to be accepted. In order to argue the appropriateness of the runtime V&V approaches, the similarity to established concepts like built-in tests could be used as a starting point.

As soon as requirements are to be adapted as well, there are no established approaches available that could be used as a starting point. The required V&V-Models@Runtime and HRA@Runtime are currently neither supported by existing work nor is a good basis available. Moreover, it is very unlikely that such approaches will be accepted in the near future. So obviously, a significant gap exists here in the state-of-the-art. In the long run, we expect that adaptations of requirements will enable new business cases – also for safety-critical systems. Therefore, we recommend reasoning about possible assurance approaches right from the beginning.

This is particularly true since the acceptance of such an approach will require a sufficiently long history of experience and empirical evidence.

7 Summary and Conclusion

In recent years, we have witnessed a strong trend towards open adaptive systems in research and industry. Meanwhile it is quite clear that new kinds of corresponding applications promise huge benefits for end-users and for businesses. The lack of suitable safety assurance approaches for OAS is increasingly turning out to be a limiting factor in this development. Models at runtime, however, could well prove to be a potent means for overcoming these problems.

Although the approaches available were not developed with an integrated safety assurance framework in mind, a promising foundation already exists. The main application scenario for the near future is characterized by open systems of systems with subsystems that only adapt to anticipated situations. Combining and advancing existing work on SafetyCeritificates@Runtime, development time assurance, and runtime V&V could already provide a sound basic solution for this scenario.

Existing safety case models in the field of safety engineering provide a sound basis for further extending the idea to SafetyCases@Runtime in order to support more flexible system adaptations. SafetyCases@Runtime appear to be sufficient to support the assurance of a wide range of application scenarios of OAS in safety-critical applications. The largest gap obviously exists if the adaptation includes the requirements. However, we expect that the application of Requirements@Runtime in safety-critical applications will only happen in the long run – leaving sufficient time to mature the safety assurance approaches in parallel.

Summarizing the results, we can safely state that Models@Runtime seem to have great potential for being successfully used as a basis for safety assurance of OAS. Since they provide a means for creating a clear trace to established safety assurance approaches, the resulting assurance approaches are likely to be accepted by safety assessors. Regarding the current state-of-the-art, there is already a good basis providing first evidence that a safety assurance framework (comparable to the one used in this article) is technically feasible.

References

[1] Cheng, B.H.C., et al.: Software engineering for self-adaptive systems: A research roadmap. In: Cheng, B.H.C., de Lemos, R., Giese, H., Inverardi, P., Magee, J. (eds.) Software Engineering for Self-Adaptive Systems. LNCS, vol. 5525, pp. 1–26. Springer, Heidelberg (2009)

[2] Blair, G., et al.: Models@Run.Time. IEEE Computer (November 2010)

[3] Dagstuhl Seminar on Models@run.time, `http://www.dagstuhl.de/en/program/calendar/semhp/?semnr=11481` (last visited June 2012)

[4] Blair, G., Coulson, G., Robin, P., Papathomas, M.: An architecture for next generation middleware. In: S.J. Davies, N.A.J., Raymond, K. (eds.) IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing, Middleware 1998 (1998)

[5] Kon, F., Roman, M., Liu, P., Mao, J., Yamane, T., Magalhaes, L., Campbell, R.: Monitoring, security, and dynamic configuration withthe dynamic tao reflective orb. In: 2nd ACM/IFIP International Conference on Middleware, New York, pp. 121–143 (2000)

[6] Capra, L., Blair, G., Mascolo, C., Emmerich, W., Grace, P.: Exploiting reflection in mobile computing middleware. ACM SIGMOBILE Mobile Computing and Communications Review 6, 34–44 (2002)

[7] Truyen, E.: Dynamic and Context-Sensitive Composition in Distributed Systems. Ph.D. thesis, K.U.Leuven (2004)

[8] `http://www.self-adaptive.org/` (last visited in June 2012)

[9] `http://www.saso-conference.org/` (last visited in June 2012)

[10] Floch, J., Hallsteinsen, S., Stav, E., Eliassen, F., Lund, K., Gjorven, E.: Using Architecture Models for Runtime Adaptability. IEEE Software 23, 62–70 (2006)

[11] Rouvoy, R., Barone, P., Ding, Y., Eliassen, F., Hallsteinsen, S., Lorenzo, J., Mamelli, A., Scholz, U.: MUSIC: Middleware Support for Self-Adaptation in Ubiquitous and Service-Oriented Environments. In: Cheng, B.H.C., de Lemos, R., Giese, H., Inverardi, P., Magee, J. (eds.) Software Engineering for Self-Adaptive Systems. LNCS, vol. 5525, pp. 164–182. Springer, Heidelberg (2009)

[12] Muskens, J., Chaudron, M.: Integrity Management in Component Based Systems. In: Proc. of 30th EUROMICRO Conference (EUROMICRO 2004), pp. 611–619 (2004)

[13] Lenzini, G., Tokmakoff, A., Muskens, J.: Managing Trustworthiness in Component-based Embedded Systems. Electron. Notes Theor. Comput. Sci. 179, 143–155 (2007)

[14] Su, R., Chaudron, M.R.V., Lukkien, J.J.: Adaptive runtime fault management for service instances in component-based software applications. IET Software 1(1), 18–28 (2007)

[15] `http://www.hitech-projects.com/euprojects/trust4all/results.htm` (last visited in June 2012)

[16] `http://www.itea2.org/project/result/download/result/5585` (last visited in June 2012)

[17] `http://ercim-news.ercim.eu/adaptable-and-context-aware-trustworthiness-evaluation` (last visited in June 2012)

[18] Wang, Y., Vassileva, J.: A review on trust and reputation for web service selection. In: Proceeding of the 1st Int. Workshop on Trust and Reputation Management in Massively Distributed Computing Systems (2007)

[19] Alnemr, R., Quasthoff, M., Meinel, C.: Taking Trust Management to the Next Level. In: Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models. IGI Global, Hershey (2010)

[20] `https://swt.informatik.uni-augsburg.de/tsos/` (last visited in June 2012)

[21] Avižienis, A., Laprie, J., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE Transactions on Dependable and Secure Computing 1, 11–33 (2004)

[22] Schneider, D., Becker, M., Trapp, M.: Approaching Runtime Trust Assurance in Open Adaptive Systems. In: Proceeding of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2011), pp. 196–201. ACM, New York (2011)

[23] Meyer, B.: Applying 'design by contract'. IEEE Computer 25(10), 40–51 (1992)

[24] Beugnard, A., Jézéquel, J.-M., Plouzeau, N.: Contract aware components, 10 years after. Electronic Proceedings in Theoretical Computer Science, 1–11 (2010)
[25] Beugnard, A., Jezéquel, J.-M., Plouzeau, N.: Making components contract aware. IEEE Computer 32(7), 38–45 (1999)
[26] Website of the SPEEDS project, `http://www.speeds.eu.com/` (last visited June 2012)
[27] Benveniste, A., Caillaud, B., Ferrari, A., Mangeruca, L., Passerone, R., Sofronis, C.: Multiple viewpoint contract-based specification and design. In: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roever, W.-P. (eds.) FMCO 2007. LNCS, vol. 5382, pp. 200–225. Springer, Heidelberg (2008)
[28] Damm, W., Metzner, A., Peikenkamp, T., Votintseva, A.: Boosting Re-use of Embedded Automotive Applications Through Rich Components. In: Proceedings of the Workshop on Foundations of Interface Technologies 2005, FIT 2005 (2005)
[29] Benvenuti, L., Ferrari, A., Mangeruca, L., Mazzi, E., Passerone, R., Sofronis, C.: A Contract-Based Formalism for the Specification of Heterogeneous Systems. In: Proceedings of the Forum on Specification, Verification and Design Languages (FDL 2008), pp. 142–147. IEEE (2008)
[30] Inverardi, P., Pelliccione, P., Tivoli, M.: Towards an assume-guarantee theory for adaptable systems. In: ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2009 (2009)
[31] Zhang, J., Cheng, B.H.C.: Model-based development of dynami-cally adaptive software. In: International Conference on Software Engineering (ICSE 2006), Shanghai, China, pp. 371–380. ACM (2006)
[32] Zhang, J., Cheng, B.H.C.: Specifying adaptation semantics. In: Workshop on Architecting Dependable Systems (WADS 2005), St. Louis, USA, pp. 1–7. ACM (2005)
[33] Kulkarni, S.S., Biyani, K.N.: Correctness of Component-Based Adaptation. In: Crnković, I., Stafford, J.A., Schmidt, H.W., Wallnau, K. (eds.) CBSE 2004. LNCS, vol. 3054, pp. 48–58. Springer, Heidelberg (2004)
[34] Strunk, E.A.: Reconfiguration Assurance in Embedded System Software, Ph.D. thesis, University of Virginia
[35] Giese, H., Tichy, M.: Component-based hazard analysis: Optimal designs, product lines, and online-reconfiguration. In: Górski, J. (ed.) SAFECOMP 2006. LNCS, vol. 4166, pp. 156–169. Springer, Heidelberg (2006)
[36] Mohammad, M., Alagar, V.: A formal approach for the specification and verification of trustworthy component-based systems. J. Syst. Softw. 84(1), 77–104 (2011)
[37] Leucker, M., Schallhart, C.: A brief account of runtime verification. Journal of Logic and Algebraic Programming 78(5), 293–303 (2009)
[38] Calinescu, R., Kwiatkowska, M.: CADS*: Computer-Aided Development of Self-* Systems. In: Chechik, M., Wirsing, M. (eds.) FASE 2009. LNCS, vol. 5503, pp. 421–424. Springer, Heidelberg (2009)
[39] Calinescu, R., Kwiatkowska, M.: Using quantitative analysis to implement autonomic IT systems. In: Proceedings of the 31st International Conference on Software Engineering (ICSE 2009), pp. 100–110 (2009)
[40] Epifani, I., Ghezzi, C., Mirandola, R., Tamburrelli, G.: Model evolution by runtime adaptation. In: Proceedings of the 31st International Conference on Software Engineering (ICSE 2009), pp. 111–121 (2009)

[41] Priesterjahn, C., Heinzemann, C., Schäfer, W., Tichy, M.: Runtime Safety Analysis for Safe Reconfiguration. In: IEEE International Conference on Industrial Informatics Proceedings of the 3rd Workshop Self -X and Autonomous Control in Engineering Applications, Beijing, China (2012) (accepted)

[42] Calinescu, R.: When the requirements for adaptation and high integrity meet. In: Proceedings of the 8th Workshop on Assurances for Self-Adaptive Systems (ASAS 2011), pp. 1–4. ACM, New York (2011)

[43] Goldsby, H.J., Cheng, B.H.C., Zhang, J.: AMOEBA-RT: Run-Time Verification of Adaptive Software. In: Giese, H. (ed.) MODELS 2008. LNCS, vol. 5002, pp. 212–224. Springer, Heidelberg (2008)

[44] Calinescu, R., Grunske, L., Kwiatkowska, M., Mirandola, R., Tamburrelli, G.: Dynamic QoS Management and Optimization in Service-Based Systems. IEEE Transactions on Software Engineering, 387–409 (May/June 2011)

[45] `http://www.ict-diva.eu/DiVA/results/diva-promo-material/DiVA-Overview-Feb2009.pdf` (last visited June 2012)

[46] Morin, B., Barais, O., Jezequel, J.-M., Fleurey, F., Solberg, A.: Models@ Run.time to Support Dynamic Adaptation. Computer 42(10), 44–51 (2009)

[47] IEC 61508: Functional safety of electrical/electronic/programmable electronic safety related systems, International Electrotechnical Commission (1999)

[48] Lisagor, O., McDermid, J.A., Pumfrey, D.J.: Towards a Practicable Process for Automated Safety Analysis. In: 24th International System Safety Conference, pp. 596–607 (2006)

[49] Papadopoulos, Y., McDermid, J.: Hierarchically Performed Hazard Origin and Propagation Studies. In: Swierstra, S.D., Oliveira, J.N. (eds.) AFP 1998. LNCS, vol. 1608, pp. 139–152. Springer, Heidelberg (1999)

[50] Fenelon, P., et al.: Towards Integrated Safety Analysis and Design. ACM Applied Computing Review 2(1), 21–32 (1994)

[51] Hawkins, R., McDermid, J.A.: Performing Hazard and Safety Analysis of Object oriented Systems. In: Proc. of ISSC 2002. System Safety Society, Denver (2002)

[52] Kaiser, B., Liggesmeyer, P., Mäckel, O.: A New Component Concept for Fault Trees. In: Lindsay, P., Cant, T. (eds.) Proc. Conferences in Research and Practice in Information Technology. ACS, vol. 33, pp. 37–46 (2004)

[53] Domis, D., Trapp, M.: Integrating Safety Analyses and Component-Based Design. In: Harrison, M.D., Sujan, M.-A. (eds.) SAFECOMP 2008. LNCS, vol. 5219, pp. 58–71. Springer, Heidelberg (2008)

[54] Förster, M., Schneider, D.: Flexible, any-time FTA with component logic models. In: International Symposium on Software Reliability Engineering, ISSRE (2010)

[55] ISO/CD 26262: Road vehicles, Functional Safety Part 6: Product development at the software level, Part 10 – 'Guidelines' (2011)

[56] DO-297: Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations, Radio Technical Commision for Aeronautics (RTCA) SC-200, (2005)

[57] Eveleens, R.L.: Integrated Modular Avionics - Development Guidance and Certification Considerations. In: RTO-EN-SCI-176 Mission Systems Engineering (2006)

[58] AC 20-148: Reusable Software Components, AC 20-148 (2004)

[59] Software Consideration in Airborne Systems and Equipment Certification, DO-178B (1993)

[60] Rushby, J.: Modular Certification. NASA Contractor Report CR-2002-212130, NASA Langley Research Center (2002)

[61] Kelly, T., Weaver, R.: The Goal Structuring Notation – A Safety Argument Notation. In: Proceedings of the 34th International Conference on Dependable Systems and Networks, DSN 2004 (2004)
[62] Kelly, T.: Concepts and Principles of Compositional Safety Case Construction. University of York, sfh (2001)
[63] Bate, I., Bates, S., Hawkins, R., Kelly, T., McDermid, J.: Safety case architectures to complement a contract-based approach to designing safe systems. In: Proceedings of the 21st International System Safety Conference (ISSC 2003): System Safety Society, pp. 182–192 (2003)
[64] Habli, I., Kelly, T.: A Safety Case Approach to Assuring Configurable Architectures of Safety-Critical Product Lines. In: The Proceedings of the International Symposium on Architecting Critical Systems (ISARCS), Prague. Czech Republic (2010)
[65] DECOS: Dependable Embedded Components and Systems, Inte-grated Project within the EU Framework Programme 6, `http://www.decos.at` (last visited June 2012)
[66] Kopetz, H., Obermaisser, R., Peti, P., Suri, N.: From a Federated to an Integrated Architecture for Dependable Embedded Real-Time Systems. TU Vienna University of Technology, Austria, and Darmstadt University of Technology, Germany (2004)
[67] Althammer, E., Schoitsch, E., Sonneck, G., Eriksson, H., Vinter, J.: Modular certification support — the DECOS concept of generic safety cases. In: 6th IEEE International Conference on Industrial Informatics (INDIN), pp. 258–263 (2008)
[68] Rushby, J.: Just-in-Time Certification. In: Proceedings of the 12th IEEE International Conference on the Engineering of Complex Computer Systems (ICECCS), Auckland, New Zealand, pp. 15–24 (2007)
[69] Rushby, J.: Runtime Certification. In: Leucker, M. (ed.) RV 2008. LNCS, vol. 5289, pp. 21–35. Springer, Heidelberg (2008)
[70] Schneider, D., Trapp, M.: A Safety Engineering Framework for Open Adaptive Systems. In: Proceedings of the Fifth IEEE International Conference on Self-Adaptive and Self-Organizing Systems, Ann Arbor, Michigan, USA, October 3-7 (2011)
[71] Schneider, D., Trapp, M.: Conditional Safety Certificates in Open Systems. In: Proceedings of the 1st Workshop on Critical Automotive applications: Robustness & Safety (CARS), pp. 57–60. ACM, New York (2010)
[72] Zimmer, B., Bürklen, S., Knoop, M., Höfflinger, J., Trapp, M.: Vertical Safety Interfaces - Improving the Efficiency of Modular Certification. In: Proc. of the 30th International Conference of Computer Safety, Reliability, and Security (SAFECOMP 2011) (2011)
[73] Fenn, J.L., Hawkins, R.D., Williams, P.J., Kelly, T.P., Banner, M.G., Oakshott, Y.: The Who, Where, How, Why And When of Modular and Incremental Certification. In: 2007 2nd Institution of Engineering and Technology International Conference on System Safety, October 22-24, pp. 135–140 (2007)
[74] Baresi, L., Ghezzi, C.: The disappearing boundary between de-velopment-time and run-time. In: Proceedings Workshop on Future of Software Engineering Research (FoSER 2010), pp. 17–22. ACM (2010)
[75] Becker, B., Beyer, D., Giese, H., Klein, F., Schilling, D.: Symbolic invariant verification for systems with dynamic structural adaptation. In: Int. Conf. on Software Engineering (ICSE). ACM Press (2006)
[76] Schneider, D., Trapp, M.: Conditional Safety Certification of Open Adaptive Systems. ACM Trans. Auton. Adapt. Syst. 8(2), Article 8, 20 pages (July 2013)