Safety Case Maintenance: A Systematic Literature Review

Carmen Cârlan[1(✉)], Barbara Gallina[2], and Liana Soima[1]

[1] fortiss GmbH, Munich, Germany
lastname@fortiss.org
[2] Mälardalen University, Västeras, Sweden
barbara.gallina@mdh.se

Abstract. Safety standards from different domains recommend the execution of a process for keeping the system safety case up to date, whenever the system undergoes a change, however, without providing any more specific guidelines on how to do this. Even if several (semi)automated safety case maintenance approaches have been proposed in the literature, currently, in the industry, the execution of this process is still manual, being error prone and expensive. To this end, we present in this paper the results of what is, to the best of our knowledge, the first Systematic Literature Review (SLR) conducted with the goal to provide a holistic overview of state-of-the-art safety case maintenance approaches. For each identified approach, we analyze its strengths and weaknesses. We observe that existing approaches are pessimistic, identifying a larger number of safety case elements as impacted by a change than the number of the actually impacted elements. Also, there is limited quantitative impact assessment. Further, existing approaches only address a few system change scenarios when providing guidelines for updating the safety case.

Keywords: safety case maintenance, systematic literature review

1 Introduction

Motivation: The system safety case can be used as a medium for assessing the impact system changes have on the system safety assurance [40]. Safety cases are explicitly required or recommended by standards from different safety critical domains, such as ISO 26262 voluntary standard [39] and UL 4600 [41] for automotive systems, the CENELEC EN 50129 standard for railway systems, the IAEA's safety standards for systems based on nuclear energy, the FAA Order 8900.1 FSIMS, Vol. 16, in the avionics or the JSP 318B standard for military aircraft systems. A safety case is a specialization of an assurance case, which is an argumentation that, based on certain evidence, a system satisfies certain system requirements, in a defined operational environment [6]. As a small change to any related safety work product may affect a large part of the safety case [25], [41], the same standards also require that the system safety case reflects the current status of the system. For example, ISO 26262 states that the safety case is a

work product (e.g., hazards list, requirements specification, system design) generated by the execution of the system safety lifecycle and that the evidence in the safety case is a compilation of the other safety work products. The same standard, in Part 10, recommends maintaining the system safety case consistent with the other safety work products. According to ISO 26262 and to UL 4600, safety case maintenance is a two-phased process. First, given a change in a safety work product, a change impact analysis (CIA) shall be conducted. Second, based on identified impact, the safety case shall be updated correspondingly. However, even if the maintenance of safety cases is a complex process, neither ISO 26262, nor any standard in other domains provide guidelines on concrete techniques for executing safety case maintenance. Currently, in practice, safety case maintenance is manually executed by safety engineers, being an error-prone and time and resource consuming process. Consequently, the inadequate management of changes in the specification of the system or its operational context has led in the past to accidents [4] or NHTSA recalls[3]. Automated change impact analysis for the system safety case and the existence of guidelines for how to update it given certain types of system changes would be beneficial. As such, safety case maintenance approaches have gained much attention in research.

Objectives and Method: The scope of this work is (1) to synthesize a comprehensive list of all automated safety case maintenance approaches proposed in the literature in the time interval 2000-2020, based on the results of a conducted systematic literature review (SLR) [26]; and (2) to report on the results of an in-depth analysis of these approaches. We are especially interested in assessing the following capabilities: 1) the degree of automation for CIA, 2) the accuracy of CIA, 3) the provision of support for quantitative CIA, 4) the provision of guidance for updating the safety case, 5) the availability of tool support. Further, we also analyze the addressed change scenarios.

Results: The SLR resulted in the selection of 65 papers, presenting 26 approaches for safety case maintenance. The results of our conducted SLR highlight three important limitations of existing approaches for safety case maintenance. First, we conclude that existing approaches are pessimistic, as their identified impact area of impact may be larger than the actual impact area. Second, we observe that, in current literature, guidelines for updating the safety case only address a few change scenarios. Another outcome of our analysis is that there is limited support for assessing the impact of a change in a quantitative manner.

The remainder of this paper is organized as follows. In Section 2, we provide essential background on safety case maintenance and an overview of related SLRs and mapping studies. In Section 3, we describe the protocol we used for this SLR. Then, in Section 4, we synthesize and analyze existing work regarding maintaining the system safety case consistent with other system and safety engineering artifacts in the literature. Towards the end, in Section 5, we summarize the results of our SLR, while highlighting the limitations of current approaches, and, in Section 6, we conclude by proposing possible research directions.

[3] `https://betterembsw.blogspot.com/p/potentially-deadly-automotive-software.html`

2 Background and Related Work

Safety case maintenance - basic concepts. Kelly and McDermid [25] elaborate on the two–phased safety case maintenance process recommended by ISO 26262. They classify *impacted* elements in two categories: elements *directly impacted* by a system change (also called *challenged* safety case elements) and *indirectly impacted* elements, which are only impacted due to the "ripple effect" of the impact propagation. Kelly and McDermid also differentiate between CIAs that only identify *potentially* impacted elements, which still need to be manually checked by the safety engineer, and accurate CIA that only identify *actually* impacted elements, which are surely invalidated by the change. Potentially impacted elements may be either actually impacted elements or false positives, meaning that they might not actually be invalidated by the change.

Related literature studies. While there are two works reporting on the state of the art in safety case tools [29], and in safety case languages [21], to the best of our knowledge, there is no review of current safety case maintenance approaches. Maksimov et al. [29] report the results of a survey concerning tool support for the creation and management of safety cases, while also analyzing certain tool functionalities, among which the support provided for maintaining safety case models consistent with other work products. Govardhanrao [21], in her master thesis, presents the results of a comparative analysis scoping a selection of argumentation languages. Among others, she assesses the support for consistency checks between system design and developing safety cases and the support offered to automatically update the safety argumentation, given system changes. The two related reviews have certain important limitations. On the one hand, the list of tools identified by Maksimov et al. is outdated – five relevant tools have been reported in the literature after the publication of this survey, namely in 2019 and 2020. On the other hand, the analysis performed by Govardhanrao does not cover all the existing approaches, but a selection of those. Further, none of the two works differentiates between the capability of identifying directly impacted elements, and the capability of automatically computing impact propagation, nor between accurate and inaccurate change impact analyses. Finally, Maksimov et al. and Govardhanrao do not provide details regarding the system change scenarios that are regarded by the maintenance approach.

3 Review Protocol

3.1 Establishing the Quasi-gold Standards by Manual Search

We first selected several publications as our quasi-gold standards (QGSs) [45]. We base our SLR on the results of the SLR conducted by Maksimov et al. [29]. As such, to establish our QGSs, we first started by manually selecting publications, which, according to Maksimov et al., present tools that have medium or strong support for safety case maintenance [29]. While Maksimov et al. identify 17 publications describing 17 different tools implementing approaches for safety case maintenance, after applying our filtering criteria (see Subsection 3.3), we

only selected 13 from these publications. Based on our expert knowledge, to these 13 selected publications we also added 3 publications, each discussing another approach different to the ones identified by Maksimov et al. [29] and 2 deliverables presenting different capabilities of the AMASS platform [14].

We complemented the aforementioned QGSs with publications we manually selected. To this end, we conducted a manual search through the proceedings published in the time frame 2000-2020 of a selection of venues that we identified as highly relevant for the safety engineering research: International Conference on Dependable Systems and Networks (DSN), International Conference on Reliable Software Technologies (Ada-Europe), European Dependable Computing Conference (EDCC), International Conference on Computer Safety, Reliability and Security (SAFECOMP), Pacific Rim International Symposium on Dependable Computing (PRDC), The International Conference/Workshop on High-Assurance Systems Engineering (HASE), International Symposium on Software Reliability Engineering (ISSRE), International Symposium on Model-Based Safety and Assessment (IMBSA), International Conference on Software Engineering (ICSE), International Conference on Model Driven Engineering Languages and Systems (MODELS) and their satellite workshops. After filtering out based on the criteria presented in Subsection 3.3, we identified 19 publications presenting additional capabilities of already identified approaches, but also introducing 6 new approaches.

3.2 Automated Search and Snowballing

Next, while using a search string, we automatically search through the following databases: ACM Digital library, IEEE, Springer, Elsevier, Google Scholar and dblp. Based on the most frequent words found in the publications we included in our QGS, we specified the following search string: ("safety case" AND "maintenance") OR ("assurance case" AND "maintenance") OR ("safety case" AND "change") OR ("assurance case" AND "change") OR ("safety case" AND "evolution") OR ("assurance cases" AND "evolution"). The search resulted in 3 selected publications. To mitigate the potential limitations due to blurry terminology, we then applied snowballing [44], which resulted in the selection of 10 more relevant publications describing approaches already identified during the other search phases.

3.3 Exclusion and Inclusion Criteria

During our search, we only selected publications whose title and abstract made it explicit that the publication was presenting the results of primary research on approaches for safety case maintenance, or at least for safety case change impact analysis. Further, we also used some other inclusion criteria. First, we only regarded the publications that appeared in 2000 or after. We chose 2000 to be the earliest date for our search, since this was the earliest publication year of one paper which we identified as QGS. We searched all publications that appeared until December 2020 - as the automatic and manual searches were

finished in January 2021. Second, given a number of different papers presenting the same approach, we considered all the papers, in order to ensure that we do not miss any information regarding safety case maintenance support. We excluded publications matching any of the following criteria: 1) publications presenting approaches for assurance case modeling, but not having at least a minimum support for assurance case maintenance; 2) publications describing maintenance approaches for other types of assurance cases (e.g., security or trust cases), 3) publications presenting support for safety case maintenance only as future work; 4) books, tutorials or poster publications; 5) publications that have not been peer-reviewed; 6) publications that are only available in the form of abstracts/posters and presentations, 7) publications not written in English.

3.4 Evaluation Criteria

After identifying all existing approaches for safety case maintenance, we carried out an in-depth analysis of these approaches, while using a set of evaluation criteria (see Table 1-a). In conformance with the work of Kelly and McDermid [25], we differentiate between approaches that support the automated identification of challenges, i.e., of the safety case elements directly impacted by a system change (**EC1**) and the automated identification of indirectly impacted elements due to impact propagation (**EC2**). **EC3** addresses the accuracy of CIA (i.e., freedom of false positives). Inspired by one work from our QGS, namely the one of Jaradat and Bate [24], who propose a quantitative assessment of the change impact, we defined evaluation criterion **EC4**. In accordance to the requirements of ISO 26262 and UL4600, evaluation criterion **EC5** assesses the capability of approaches to provide guidance for updating the safety case in accordance to the CIA results. Further, as UL4600 highly recommends the usage of tools to execute impact analysis, we are also interested whether the identified approaches have tool support (**EC6**).

According to ISO 26262 and UL 4600, given a change in the system specification, the safety case needs to be re-evaluated. Different system change scenarios have a different impact on the system safety case [1]. As such, we analyze the change scenarios addressed by the identified safety case maintenance approaches, especially considering the scenarios in Table 1-b. **CS1**-**CS3** are general change scenarios. However, since addressing more concrete change scenarios increases the accuracy of CIA [27], we also address more concrete scenarios. A report on an industrial survey conducted by de la Vara et al. [13] presents the state of the practice with respect to safety evidence change impact analysis. The survey reports that requirement specifications are the artifacts most exposed to changes during the entire system lifecycle (**CS4**). UL4600 requires that safety case maintenance is executed given any change in the system design (**CS5**). ISO 26262 mandates the demonstration that all the safety critical requirements specified for the system under consideration have been designed, implemented and tested. This is usually established by traceability links. The report of de la Vara et al. emphasizes the fact that traceability links are bound to frequently undergo changes during the entire system lifecycle. This is because changes in different

engineering artifacts also trigger changes in the traceability links (**CS6**). The results of the industrial survey conducted by de la Vara et al. also indicate that safety analysis is frequently re-executed during the system lifecycle, outputing new analysis results (**CS7**). Another type of engineering artifacts reported by the survey conducted by de la Vara et al. as undergoing frequent changes are verification and validation results (**CS8**). Whenever the system requirements, design or source code change, re-verification shall be executed, in order to detect early specification violations. Further, UL4600 requires the execution of safety case maintenance given changes to the intended operational environment (i.e., contextual assumptions) (**CS9**). UL4600 also recommends that the impact a system reconfiguration has on the system safety case needs to analyzed (**CS10**).

a) Evaluation Criteria

ID	Evalation Criteria
EC1	Support for automated challenge detection
EC2	Support for automated impact propagation
EC3	Accuracy of CIA (i.e., freedom of false positives)
EC4	Support for quantitative impact assessment
EC5	Support for updating the safety case
EC6	Tool support

b) Addressed change scenarios

ID	Change Scenario
CS1	Deletion of any system artifact
CS2	Addition of any system artifact
CS3	Modification of any system artifact
CS4	Modification of a requirement
CS5	Modification of the system architecture
CS6	Modification of traceability links within system artifacts
CS7	Modification of risk assessment (i.e., ASIL assignments, according to ISO 26262)
CS8	Addition of new verification evidence
CS9	Modification of operational environment
CS10	Modification of system parameter values

Table 1. Overview of the used evaluation criteria and the addressed change scenarios.

4 Review Results

Our SLR resulted in 65 selected publications, presenting 26 different approaches for safety case maintenance. We present the identified approaches in Table 2, together with an overview of their capabilities for keeping the system safety case consistent with other safety artifacts. While for the analysis of each approach we used all the publications we found during our literature search, in the table we only reference one or two most relevant publications, due to space restrictions.

Commercial approaches. Our SLR revealed the existence of two safety case maintenance approaches implemented in commercial tools. Both approaches support the traceability between safety cases and other safety artifacts. NOR-STA [43] addresses change scenario **CS1**, by identifying missing traceability links. Further, NOR-STA recommends, given the addition of system model elements, the addition of argumentation legs concerning the newly added model

Approach	Automated Challenge Detection	Automated Impact Propagation	False Positives	Quantitative CIA	Update Guidance	Tool Support
AC-ROS [10]	yes [CS3]	no	no	no	yes	yes
AdvoCATE [15]	no (safety case regeneration) [CS2]	-	no	no	no	yes
AF3 [8]	yes [CS1, CS3]	yes	yes	no	no	yes
AMASS Platform [14], [22], [20]	yes [CS1, CS2, CS3, CS4]	no	yes	no	no	yes
ASCE [17], [32]	yes [CS1, CS3]	no	yes	no	no	yes
Checkable Safety Cases [9]	yes [CS1, CS2, CS5, CS6, CS7]	yes	no	no	yes	yes
D-CASE [18], [31]	yes [CS3, CS4, CS10]	no	yes	no	yes	yes
DMILS [11]	no (safety case regeneration) [CS2]	-	no	no	no	yes
Dynamic Safety Cases [2]	yes [CS3]	yes	no	yes	yes	no
ENTRUST [7]	yes [CS3, CS4, CS5]	no	yes	no	no	yes
ETB [12]	yes [CS4, CS5]	no	yes	no	yes	yes
Event-B Extension [35]	yes [CS4, CS5]	no	yes	no	yes	yes
GAGE [5]	yes [CS5]	no	yes	no	yes	yes
HIP-HOPS extension [36]	yes [CS4, CS5]	no	yes	no	yes	yes
Interlocking Safety Cases [42]	yes [CS3]	no	no	no	yes	yes
Isabelle/SACM [33]	yes [CS1, CS2, CS3, CS4, CS5]	no	yes	no	yes	yes
MMINT-A [27], [37]	yes [CS1, CS3, CS7]	yes	yes	no	no	yes
NOR-STA [43]	yes [CS1, CS2, CS3]	no	no	no	yes	yes
Resolute [19]	yes [CS5]	no	yes	no	yes	yes
SAFA [1]	yes [CS1, CS2, CS3, CS6]	no	yes	no	yes	yes
Safety Cases for IMS [34]	yes [CS2, CS3]	yes	yes	no	yes	no
Safety Case Synthesis [3]	yes [CS4, CS5]	no	yes	no	yes	yes
SAM [25]	yes [CS1, CS2, CS3]	yes	yes	no	yes	yes
SANESAM [24]	yes [CS5]	yes	no	yes	no	yes
SPIRIT [28]	yes [CS1, CS2 CS3, CS6]	yes	yes	no	no	yes
Weaving Safety Cases [23]	no (safety case regeneration) [CS2, CS6]	-	no	no	no	yes

Table 2. Overview of the identified safety case maintenance approaches.

element (**CS2**). The Assurance and Safety Case Environment (ASCE) [17], [32] reflects the impact of modifications in referenced files on the safety case model. Further, ASCE supports the comparison between two structured safety cases, by specifying each version of a safety case as a Kripke structure.

Change impact propagation in safety cases. There are several approaches supporting change impact propagation, i.e., which identify the safety case elements indirectly impacted by a change. The approach of Nicholson et al. [34] is only presented in an abstract manner, and is only adequate for Integrated Modular Systems (IMS). The Sensitivity ANalysis for Enabling Safety Argument Maintenance (SANESAM) [24] is an accurate safety case maintenance approach specifying any system change as the modification of the failure rates of hardware components. SANESAM is a quantitative CIA and also provides support for impact propagation. AutoFOCUS3 (AF3) [8], SAM [25], SPIRIT [28] and Model Management INTeractive for Assurance cases (MMINT-A) [27] offer support for automated identification of challenged safety case elements and automated change impact propagation given the deletion (**CS1**) or modification of any referenced artifact (**CS3**). However, all four approaches are prone to output false positives. Additionally, MMINT-A checks for correct Automotive Safety Integrity Level (ASIL) decomposition, given changes in the ASIL attribute of safety case goals (**CS7**). Checkable Safety Cases [9] is a novel approach for accurately checking the consistency between safety case and other system models, supporting different change scenarios (**CS1**-**CS5**).

Automated (re)generation of safety cases via formal methods. Several works, such as ETB [12], GAGE [5], the extension of Event-B for safety case modeling [35], SACM/Isabelle [33], Resolute [19], the safety case synthesis approach proposed by Bagheri et al. [3], and the HIP-HOPS extension for modeling safety cases [36] propose the usage of formal verification methods for the specification of entire safety cases, and/or the formal specification of system safety properties referenced in the safety case. Some of these approaches even support the automated generation of the system safety case, by instantiating patterns with information from formal verification engines. The satisfaction of formally specified safety claims can be verified against a certain system specification (model or code) and the obtained verification results can be automatically integrated as evidence in the system safety case. Given a change in a formally specified safety requirement (**CS4**), it is checked if the system architecture or the code (still) implements the respective requirement. However, the impact of that change on the rest of the artifacts (e.g., hazards lists) is not assessed. Further, given a change in the system architecture (**CS5**), some of these approaches identify the impacted safety case claims and suggest for reverification. However, given counter-evidence or additional evidence (**CS8**), which has not been referenced in the safety case before, there is no support for change impact propagation throughout the rest of the argumentation. All existing approaches may output false positives, as not every system change invalidates the verification evidence.

(Re)generation of safety cases via automated pattern instantiation. Approaches such as the weaving safety case models approach proposed

by Hawkins [23], DMILS [11], and AdvoCATE [15] remove the need for change impact analysis altogether by instead regenerating the impacted part of the assurance case model, based on automated pattern instantiation (see Table 2). The automated pattern instantiation is done by the usage of a third model (i.e., an instantiation model) containing the mappings between pattern parameters to be instantiated and the values with which they shall be instantiated. On the one hand, in the approach proposed by Hawkins [23], the parameters may be instantiated with direct traceability links to other safety engineering models. Therefore, given the deletion (**CS1**) or the modification (**CS3**) of a referenced system model, the impact of the change on the safety case model is automatically reflected and the patterns are automatically re-instantiated. However, the approach is too pessimistic, triggering the need for re-instantiation whenever a system change occurs, even when the change does not impact the validity of the safety argumentation. On the other hand, in DMILS and AdvoCATE the instantiation models only contain an ID or name of the referenced model elements, instead of having a direct traceability link. Therefore, the user of AdvoCATE needs to manually assess the impact of a system change on the safety case and decide if the patterns shall be re-instantiated. However, AdvoCATE supports the automated identification of outdated verification evidence and automated integration of regenerated verification results as evidence in the system safety case. None of these two approaches provides guidance for how to update the assurance case given invalidated claims or evidence.

Safety cases updated at runtime. Some approaches such dynamic safety cases proposed by Denney et al. [16], interlocking safety cases [42], ENTRUST [7], AC-ROS [10] and D-CASE [30] support the automated update of assurance cases at runtime, based on the feedback received from online monitors. However, these frameworks only address changes of certain system configuration parameters at runtime (specialization of **CS3**) and do not provide any solution for handling changes of other assurance artifacts such as hazards, or requirements. Only the dynamic safety cases approach supports to some extent change impact propagation, either by computing how the confidence level is affected by a parameter change or by propagation based on the relations between GSN elements.

Change impact analysis for safety cases. The Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems (AMASS) platform [14] and the Safety Artifact Forest Analysis (SAFA) [1] approach are unique approaches, which cannot be fit only into one of the categories above. AMASS supports the identification of invalidated verification evidence due to changes in system specification [22] (change scenarios **CS4**, **CS5**) and change impact analysis given changes in the features of the systems [20]. AMASS provides some support for updating the safety case by updating the contracts of a system component, given changes in the contextual assumptions [38]. SAFA automatically generates GSN structures based on a model specifying the traceability links among different safety artifacts. Further, SAFA can compare two different GSN structures in order to support the assessment of the evolution of a safety case by identifying the elements added (**CS2**), deleted (**CS1**) or modified

(**CS3**) in a new version of the same GSN structure. SAFA also provide guidance for updating the safety case, given certain change scenarios, namely **CS1**, **CS2**, **CS3**, **CS4**. The two approaches are bound to output false positives, and do not support automated computation of impact propagation.

Threats to validity. One of the main threats to validity is that our results may be incomplete. To address this threat, we used a hybrid search strategy, combining manual and automated searches with snowballing. However, our results may be unreliable due to our lacunary interpretation of the capabilities of some of the approaches, especially of the ones for which little information was available. *Internal validity.* We conducted our SLR based on a defined review protocol, as recommended by Kitchenham and Charters [26]. Further, the selection of relevant publications was peer reviewed. While the third author of this paper executed all the search phases presented before, after the execution of each search phase, the first author of this paper reviewed all the exclusions and the final set of included papers. Towards the end of the SLR, the second author of this paper checked, agreed upon, and refined the whole set of extracted data. *External validity.* Threats to validity such as bias in data selection, extraction, and classification may impair the generalizability of the results. While we aimed at providing complete and valid results, the SLR protocol presented in Section 3 could be used for further updates and/or replication reviews to reinforce its results. For example, the review could be complemented by searching for approaches for the maintenance of assurance cases addressing other types of requirements, such as security, dependability, or trustworthiness.

5 Discussion

Inaccurate automated CIA. The results of our analysis show that all identified approaches have some support for automated detection of change impact, by exploiting traceability links between safety case elements and other engineering artifacts. Only the approaches for (re)generating safety cases via automated pattern instantiation remove the need for change impact analysis altogether. However, 15 out of 26 identified approaches only support the identification of challenged safety case elements, namely the ones directly impacted by a system change, without also computing the impact propagation throughout the entire safety argumentation. Further, with few exceptions, most of the approaches are inaccurate, namely they are prone to output false positives (see Figure 1-a). The approaches that do not provide false negatives only focus on very specific types of changes or are only adequate to be used for specific types of systems.

Lack of support for quantitative CIA. Only 2 out of 26 approaches provide support for the quantitative assessment of change impact (see Figure 1-b). The dynamic safety cases approach proposes the assessment of the impact a system change has on the confidence in the safety argumentation. However, there are no details provided on the implementation of this assessment. SANESAM computes the impact of system changes on the results of a failure probability analysis. However, such analysis can only be done for hardware components.

Limited support for updating the safety case. In Figure 1-c, we see that only for certain system change scenarios some guidance for updating the impacted safety case is provided, whereas 9 approaches do not offer any guidance. 7 out of 26 approaches propose re-verification given changes in either the system requirements or in the implementing system specification (i.e., system architecture, system configuration, or source code). However, these approaches do not give any guidelines on what to do if the newly generated verification results are negative or if additional evidence (i.e., evidence that has not been referenced in the safety case) is generated. Further, these approaches are bound to output false positives, meaning that they cannot determine if the verification evidence actually needs to be re-generated, given a certain change. Another type of safety case update recommendation is provided by SAFE, MMINT-A and NOR-STA. Given the addition of a new element in a referenced set of elements, these approaches can identify that the argumentation is incomplete, and suggest the addition of new claims in the argumentation regarding the newly added elements. Further, all the approaches for maintaining safety cases consistent with the system configuration at runtime propose to switch from one safety case to another, in correspondence to the system re-configuration.

Few addressed change scenarios. Each of the state-of-the-art approaches for safety case maintenance addresses one or more change scenarios (see 1-f). However, not every change scenario we identified as relevant for current practice in Table 1 is addressed by current approaches. Some change scenarios are poorly addressed in the literature. While approaches such as ETB, GAGE, Event-B Extension, Resolute, HIP-HOPS extension, ENTRUST and SAFA can detect the addition of new verification evidence, which they integrate in the safety case (**CS8**), they do not assess the extent of the impact the new evidence has on the safety argumentation. Moreover, to our knowledge, there is no approach addressing the modification of contextual assumptions (**CS9**). Currently, given the modification of a contextual assumption, the entire argumentations depending on that assumption needs to be manually checked by the safety engineer.

Tool support. Most of the identified approaches have some tool support (see Figure 1-d). The approach proposed by Nicholson et al. [34] and the dynamic safety cases proposed by Denney et al. do not provide tool support, and also their usage is not exemplified, leaving certain open questions regarding how to actually apply them. **Used safety case languages.** 19 of the identified approaches can be applied for safety case models compliant with the GSN (see Figure 1-e).

6 Summary and Future Lines of Work

In the recent years, due to the stringent practical needs for automating safety assurance, we have witnessed a boom in state-of-the-art approaches for automated safety case maintenance. In this paper, we reported on the results of a systematic literature review, which we conducted to identify all the existing approaches for safety case maintenance. These results may be extended by also searching for maintenance approaches for any type of assurance cases. Further, another pos-

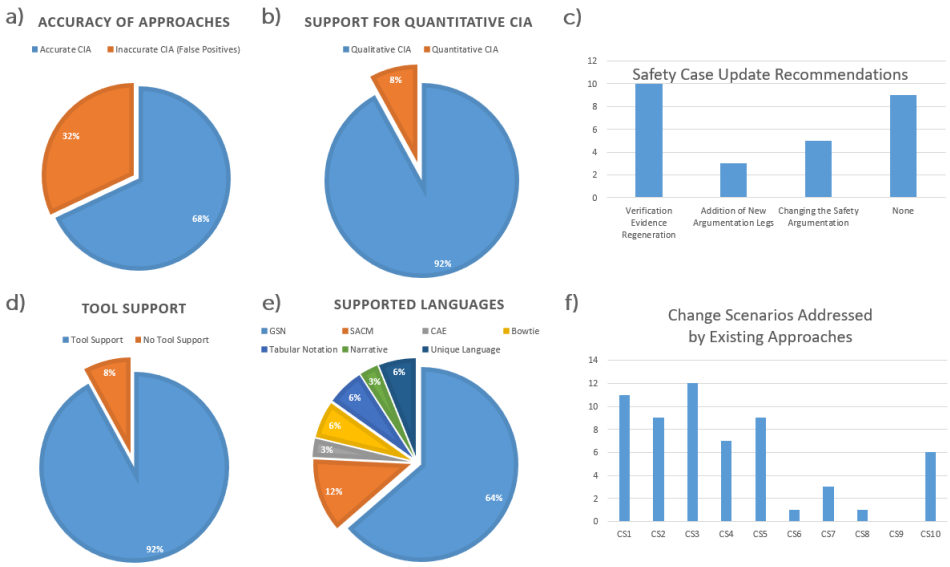

Fig. 1. Statistics on existing safety case maintenance approaches.

sible extension of the SLR is to also consider product-line oriented approaches, which integrate maintenance of safety cases with variability approaches. The SRL resulted in the selection of 65 papers, presenting 26 different approaches for safety case maintenance, within the interval 2000-2020. While analyzing the strengths and weaknesses of the identified approaches, we identified a set of literature gaps, from which we drew some future lines of work. **More accurate CIAs.** With few exceptions, the existing safety case maintenance approaches are inaccurate. More accurate CIA, requiring less involvement of safety engineers, would be beneficial as it would decrease the time and effort needed for the execution of the whole safety case maintenance process. Quite recently, it was estimated that, the change of one line of code in an avionics system costs around 1 million dollars, and that it takes approximately one year to be implemented[4].**Increased support for safety case update.** Our SLR results showed that guidance for how to update the safety case is only available for few change scenarios, which may have serious consequences[5]. **Quantitative CIAs.** There are few approaches assessing the impact of a change in a quantitative manner. Quantitative analyses could provide the safety engineers with a better understanding of the implications of a certain change on the system safety, especially in the context of systems dominated by uncertainty [2]. **Addressing more change scenarios.** According to our analysis results, there is a lack of support in handling change scenarios **CS5-CS10** specified in Table 1. Similar to Kokaly et al. [27], we believe that these gaps could be covered if safety case

[4] `https://insights.securecodewarrior.com/one-line-of-code-1-million/`
[5] `https://libertyvillepersonalinjurylawyer.com/software-fault-liability/`

maintenance approaches would focus on more concrete change scenarios, and by enhancing safety case models with metadata specifying the sensitivity of the safety case to specific system changes.

References

1. A. Agrawal, S. Khoshmanesh, M. Vierhauser, M. Rahimi, J. Cleland-Huang, and R. R. Lutz. Leveraging artifact trees to evolve and reuse safety cases. In *Proceedings of the 41st International Conference on Software Engineering*, pages 1222–1233. IEEE / ACM, 2019.
2. E. Asaadi, E. Denney, J. Menzies, G. J. Pai, and D. Petroff. Dynamic assurance cases: A pathway to trusted autonomy. *Computer*, 53(12):35–46, 2020.
3. H. Bagheri, E. Kang, and N. Mansoor. Synthesis of assurance cases for software certification. In *Proceedings of the 42nd International Conference on Software Engineering, New Ideas and Emerging Results*, pages 61–64. ACM, 2020.
4. J. Betz, A. Heilmeier, A. Wischnewski, T. Stahl, and M. Lienkamp. Autonomous drivinga crash explained in detail. *Applied Sciences*, 9(23), 2019.
5. S. Björnander, R. Land, P. Graydon, K. Lundqvist, and P. Conmy. A method to formally evaluate safety case arguments against a system architecture model. In *Proceedings of the 2nd edition of the Workshop on Software Certification*. IEEE Computer Society, 2012.
6. R. E. Bloomfield and P. G. Bishop. Safety and assurance cases: Past, present and possible future - an adelard perspective. In *Making Systems Safer - Proceedings of the 18th Safety-Critical Systems Symposium*, pages 51–67. Springer, 2010.
7. R. Calinescu, D. Weyns, S. Gerasimou, M. U. Iftikhar, I. Habli, and T. Kelly. Engineering trustworthy self-adaptive software with dynamic assurance cases. *IEEE Transactions on Software Engineering*, 44(11):1039–1069, 2018.
8. C. Cârlan, V. Nigam, S. Voss, and A. Tsalidis. Explicitcase: Tool-support for creating and maintaining assurance arguments integrated with system models. In *Proceedings of the 38th International Symposium on Software Reliability Engineering Workshops*, pages 330–337. IEEE, 2019.
9. C. Cârlan, D. Petrisor, B. Gallina, and H. Schoenhaar. Checkable safety cases: Enabling automated consistency checks between safety work products. In *Proceedings of the 31st International Symposium on Software Reliability Engineering - ISSRE Workshops*, pages 295–302. IEEE, 2020.
10. B. H. C. Cheng, R. J. Clark, J. E. Fleck, M. A. Langford, and P. K. McKinley. AC-ROS: assurance case driven adaptation for the robot operating system. In *Proceedings of the 23rd International Conference on Model Driven Engineering Languages and Systems*, pages 102–113. ACM, 2020.
11. A. Cimatti, R. DeLong, D. Marcantonio, and S. Tonetta. Combining MILS with contract-based design for safety and security requirements. In *Proceedings of the 34th Int. Conference on Computer Safety, Reliability, and Security Workshops*, volume 9338 of *Lecture Notes in Computer Science*, pages 264–276. Springer, 2015.
12. S. Cruanes, G. Hamon, S. Owre, and N. Shankar. Tool integration with the evidential tool bus. In *Proceedings of the 14th International Conference on Verification, Model Checking, and Abstract Interpretation*, volume 7737 of *Lecture Notes in Computer Science*, pages 275–294. Springer, 2013.
13. J. L. de la Vara, M. Borg, K. Wnuk, and L. Moonen. An industrial survey of safety evidence change impact analysis practice. *IEEE Trans. Software Eng.*, 42(12):1095–1117, 2016.

14. J. L. De La Vara, E. Parra, A. Ruiz, and B. Gallina. The AMASS Tool Platform: An innovative solution for assurance and certification of cyber-physical systems. In *CEUR Workshop Proceedings*, volume 2584. CEUR-WS, 2020.
15. E. Denney and G. Pai. Tool support for assurance case development. *Automated Software Engineering*, 25(3):435–499, 2018.
16. E. Denney, G. J. Pai, and I. Habli. Dynamic safety cases for through-life safety assurance. In *Proceedings of the 37th International Conference on Software Engineering*, pages 587–590. IEEE Computer Society, 2015.
17. M. Felici. Modeling safety case evolution - examples from the air traffic management domain. In *Proceedings of the 2nd International Workshop on Rapid Integration of Software Engineering Techniques*, volume 3943 of *Lecture Notes in Computer Science*, pages 81–96. Springer, 2006.
18. H. Fujita, Y. Matsuno, T. Hanawa, M. Sato, S. Kato, and Y. Ishikawa. Ds-bench toolset: Tools for dependability benchmarking with simulation and assurance. In *Proceedings of the 42nd International Conference on Dependable Systems and Networks*, pages 1–8. IEEE Computer Society, 2012.
19. A. Gacek, J. Backes, D. D. Cofer, K. Slind, and M. Whalen. Resolute: An assurance case language for architecture models. *Computing Research Repository (CoRR)*, abs/1409.4629, 2014.
20. B. Gallina. AMASS Deliverable: Design of the AMASS tools and methods for cross/intra-domain reuse. Technical Report D6.3, AMASS Consortium, 2018.
21. S. B. Govardhanrao. A comparative analysis of argumentation languages in the context of safety case development. Master's thesis, Mälardalen University, School of Innovation, Design and Engineering, 2019.
22. T. Grüber. AMASS Deliverable: Prototype for multi-concern assurance. Technical Report D4.6, AMASS Consortium, 2018.
23. R. Hawkins, I. Habli, D. Kolovos, R. Paige, and T. Kelly. Weaving an assurance case from design: A model-based approach. In *Proceedings of the 16th International Symposium on High Assurance Systems Engineering*, pages 110–117. IEEE, 2015.
24. O. T. S. Jaradat and I. Bate. Using safety contracts to guide the maintenance of systems and safety cases. In *Proceedings of the 13th European Dependable Computing Conference*, pages 95–102. IEEE Computer Society, 2017.
25. T. P. Kelly and J. A. McDermid. A systematic approach to safety case maintenance. *Reliability Engineering System Safety*, 71(3):271–284, 2001.
26. B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. (EBSE 2007-001), 2007.
27. S. Kokaly, R. Salay, M. Chechik, M. Lawford, and T. Maibaum. Safety case impact assessment in automotive software systems: An improved model-based approach. In *Proceedings of the 36th International Conference on Computer Safety, Reliability, and Security*, volume 10488 of *Lecture Notes in Computer Science*, pages 69–85. Springer, 2017.
28. C. Lin, W. Shen, T. Yue, and G. Li. Automatic support of the generation and maintenance of assurance cases. In *Proceedings of the 4th International Symposium on Dependable Software Engineering. Theories, Tools, and Applications*, volume 10998 of *Lecture Notes in Computer Science*, pages 11–28. Springer, 2018.
29. M. Maksimov, N. L. S. Fung, S. Kokaly, and M. Chechik. Two decades of assurance case tools: A survey. In *Proceedings of the 37th International Conference on Computer Safety, Reliability, and Security Workshops*, volume 11094 of *Lecture Notes in Computer Science*, pages 49–59. Springer, 2018.

30. Y. Matsuno. A design and implementation of an assurance case language. In *Proceedings of the 44th Annual International Conference on Dependable Systems and Networks*, pages 630–641. IEEE Computer Society, 2014.
31. Y. Matsuno and S. Yamamoto. A framework for dependability consensus building and in-operation assurance. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, 4(1):118–134, 2013.
32. M. Mistry and M. Felici. Implementation of change management in safety cases. *Formal Aspects of Safety-Critical Systems*, 2008.
33. Y. Nemouchi, S. Foster, M. Gleirscher, and T. Kelly. Mechanised assurance cases with integrated formal methods in isabelle. *Computing Research Repository (CoRR)*, abs/1905.06192, 2019.
34. M. Nicholson, P. Conmy, I. Bate, and J. McDermid. Generating and maintaining a safety argument for integrated modular systems. In *Proceedings of the 5th Australian Workshop on Industrial Experience with Safety Critical Systems and Software*, pages 31–41, 2000.
35. Y. Prokhorova, L. Laibinis, and E. Troubitsyna. Facilitating construction of safety cases from formal models in Event-B. *Inf. and Soft. Technology*, 60:51–76, 2015.
36. A. Retouniotis, Y. Papadopoulos, I. Sorokos, D. Parker, N. Matragkas, and S. Sharvia. Model-connected safety cases. In *Proceedings of the 5th International Symposium on Model-Based Safety and Assessment*, volume 10437 of *Lecture Notes in Computer Science*, pages 50–63. Springer, 2017.
37. A. D. Sandro, G. M. K. Selim, R. Salay, T. Viger, M. Chechik, and S. Kokaly. MMINT-A 2.0: tool support for the lifecycle of model-based safety artifacts. In *Proceedings of the 23rd International Conference on Model Driven Engineering Languages and Systems*, pages 15:1–15:5. ACM, 2020.
38. I. Sljivo, B. Gallina, J. Carlson, and H. Hansson. Using safety contracts to guide the integration of reusable safety elements within ISO 26262. In *Proceedings of the 21st Pacific Rim International Symposium on Dependable Computing - PRDC*, pages 129–138. IEEE Computer Society, 2015.
39. I. Standard. 26262: Road vehicles – functional safety. ISO, 2018.
40. F. Törner and P. Öhman. Automotive safety case A qualitative case study of drivers, usages, and issues. In *Proceedings of the 11th High Assurance Systems Engineering Symposium*, pages 313–322. IEEE Computer Society, 2008.
41. UNDERWRITERS LABORATORIES INC. ANSI/UL-4600 Standard for Evaluation of Autonomous Products. 2020.
42. M. Vierhauser, S. Bayley, J. Wyngaard, W. Xiong, J. Cheng, J. Huseman, R. R. Lutz, and J. Cleland-Huang. Interlocking safety cases for unmanned autonomous systems in shared airspaces. *Transactions on Software Engineering*, 2019.
43. A. Wardziński and P. Jones. Uniform model interface for assurance case integration with system models. In *Proceedings of the 38th International Symposium on Software Reliability Engineering*, pages 39–51, Cham, 2017. Springer.
44. C. Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, pages 1–10. ACM, 2014.
45. H. Zhang, M. A. Babar, and P. Tell. Identifying relevant studies in software engineering. *Information and Software Technology*, 53(6):625–637, 2011.