Two Decades of Assurance Case Tools: A Survey

Mike Maksimov[1](✉), Nick L. S. Fung[1], Sahar Kokaly[2], and Marsha Chechik[1]

[1] University of Toronto, Toronto, Canada
{maksimov,nlsfung,chechik}@cs.toronto.edu
[2] McMaster University, Hamilton, Canada
kokalys@mcmaster.ca

Abstract. In regulated safety-critical domains, such as the aerospace and nuclear domains, certification bodies often require systems to undergo a stringent safety assessment procedure to show their compliance to one or more safety standards. Assurance cases are an emerging way of communicating safety of a safety-critical system in a structured and comprehensive manner. Due to the significant complexity of the required materials, software tools are often used as a practical way of constructing assurance cases. This paper presents the first, to the best of our knowledge, systematic review of assurance case tools. Specifically, we provide a comprehensive list of assurance case tools developed over the past 20 years and an analysis of their functionalities.

Keywords: Assurance case · Tools · Systematic literature review

1 Introduction

Assurance cases (ACs) can be very complex; e.g., an assurance case for an air traffic control system may comprise over 500 pages and 400 referenced documents [34]. Tools to support safety engineers in creating, maintaining and analysing ACs have been developed. For example, Resolute [23] can automatically generate ACs based on a system's architectural models, while AGSN [35] supports the assessment of an AC's validity. The development of these tools has been enabled by the introduction of formal syntaxes for ACs, such as the Goal Structuring Notation (GSN) [28]. In this paper, we aim to perform a systematic review of the progress made in the development of tools for ACs. To the best of our knowledge, this is the first such study. More specifically, the main contributions of this work are (1) a comprehensive list of AC tools developed over the past 20 years; and (2) an analysis of these tools according to their functionality.

The remainder of this paper is organised as follows. Sect. 2 presents our methodology for finding and comparing AC tools. In Sect. 3, we present and summarise our findings and potential threats to validity. We conclude by discussing the implications of our work in Sect. 4.

B. Gallina et al. (Eds.): SAFECOMP 2018 Workshops, LNCS 11094, pp. 49–59, 2018.
https://doi.org/10.1007/978-3-319-99229-7_6

2 Methodology

We carried out a Systematic Literature Review (SLR) in order to establish a complete list of AC tools and provide a comprehensive assessment of their features. Our SLR followed a simplified version of the guidelines proposed by Kitchenham et al. [8], as well as the search strategy proposed by Zhang et al. [46]. It consists of three stages: **(1)** establishing the *quasi-gold standard* (QGS) through a manual search of different publication venues, **(2)** an automated literature search of digital libraries, e.g., Springer Link and IEEE Xplore, and **(3)** a web-based search for commercial tools and tools that may not have been mentioned in publications. We describe these steps below.

Manual Search and Establishing the QGS. A QGS is a set of high quality studies from the related publication venues on a research topic, e.g., domain-specific conferences and journals recognized by the community in the subject, for a given time span [46]. To create a QGS, relevant publication venues are identified and manually searched in order to retrieve studies that serve as a benchmark for the subsequent automated search. Through consultation with domain experts, we identified six major conferences and journals that published research on ACs: **(1)** SAFECOMP (International Conference on Computer Safety, Reliability, & Security), **(2)** HASE (International Symposium on High Assurance Systems Engineering), **(3)** IMBSA (International Symposium on Model-Based Safety and Assessment), **(4)** ISSRE (International Symposium on Software Reliability Engineering), **(5)** Reliability Engineering & System Safety (journal), and **(6)** COMPSAC (International Conference on Computers, Software & Applications). We performed a manual search through the proceedings of these venues including all associated workshops, for 2015-17 inclusive, yielding 10 relevant AC tool papers which established our QGS.

Defining the Search String and Performing the Automated Search. A careful examination of the papers in our QGS constructed the search string to be "*("Safety Assurance" OR GSN OR SACM OR "Safety Case" OR "Safety Cases" OR "Assurance Case" OR "Assurance Cases" OR "Safety Compliance") AND (Editor OR Tool OR Editors OR Tools OR Toolset OR Toolsets)*". We used it to conduct an automated literature search on IEEE Xplore, Engineering Village, ACM Digital Library and Spring Link[1], combined with the criterion that the papers were in English and published after 1998.

IEEE Xplore, Engineering Village, ACM Digital Library, and Springer Link returned 112, 739, 21, and 80 papers respectively, for a total of 952 papers. We checked the resulting papers against our QGS which captured 8/10 papers, achieving the recommended 80% sensitivity [46]. After filtering out duplicate papers, papers not accessible in full text, irrelevant papers (based on a manual review of their abstracts or the full text), we identified 82 papers.

Performing the Web-Based Search. To obtain knowledge about commercial AC tools, tools that were published but were not found by our literature search,

[1] The literature search was conducted in the dates between 02.02.2018 - 19.02.2018.

or tools that simply were not mentioned in publications, we conducted a web-based search[2] using Google as the search engine. We used the same search string as for the literature search and viewed the first 100 results. This step yielded eight additional tools.

Table 1. Tool functionality categories and the corresponding degrees of support.

Feature category	**Level of tool support**			
	D (No support)	**C (Minimal support)**	**B (Moderate support)**	**A (Strong support)**
Support for AC creation (Creation).	None	Basic support for the manual creation of ACs.	Partial automation or re-use in creating ACs is available (e.g. argument patterns and templates).	Automatic creation of complete ACs.
Support for maintaining ACs as they evolve (Maintenance).	None	Manual editing with no guidance on affected parts provided.	Tracking of relevant artefacts (e.g. system models and evidence), notifying user of changes and/or indicating their potential impact on the AC.	Automatic updates of ACs to reflect changes in the relevant artefacts (e.g., evidence, system models, requirements specifications).
Support for assessing ACs (Assessment).	None	Support for manual annotations to indicate potential problems.	Support for syntactical checks (e.g., for well-formedness, completeness and/or consistency).	Syntactic and semantic checking (e.g., validity of overall argument given its supporting arguments and evidence).
Support for collaboration between users (Collaboration).	None	A basic concurrent multi-user environment.	Additional features such as user access/permission management.	A complex multi-user environment (e.g., change requests and change reviews).
Support for creating reports from ACs (e.g. for certification purposes or for different stakeholders) (Reporting).	None	Generic reports with no user configurability, limited range of document formats and/or limited content.	Some user configurability, in multiple document formats and/or containing more content.	High user configurability, extensive document formats and/or detailed/interactive content (e.g., generating different reports).
Support for other design/assurance lifecycle processes (e.g. RE specs, hazop, verification) (Integration).	None	Manual integration.	Some support (e.g., bundling with specific third-party tools).	Extensive support for many other design/assurance lifecycle processes.

Evaluating the Tools. Having read all of the publications and the resources gathered by our searches, we established six distinct recurring tool functionalities, using them as the basis for our evaluation. These functionalities are categorized as AC creation, maintenance, assessment, collaboration, reporting and integration (see Table 1). We then defined four levels of tool support for each of the categories, ranging from D (no support) to A (strong support), thus creating our grading criteria. We then graded each tool's degree of support for each

[2] Carried out on 25.02.2018.

category, using information from the publications and the web resources. Since information in some of the publications can be out of date, we made an effort to use the newest publications so as to arrive at a more accurate evaluation. Please note that our evaluation is based purely on the information found in the above resources rather than on the hands-on testing of the tools.

3 Results

Our systematic literature review discovered a total of 46 AC tools. Eight of these tools (AssureNote [1], PREEVision [3], SMS Pro [4], Artisan GSN modeler [2], Assure-It [45], SEAS [12], TurboAC [5] and eDependabilityCase [33]) were discovered by our web search; two (MMINT-A [22] and Resolute [23]) were identified with the help of domain experts, and the remainder were found by our literature search. Nine tools (AssureNote [1], DECOS Test Bench [11], e-Safety Case [32], GSN CaseMaker ERA [32], ISIS High Integrity Solutions [32], PREEVision [3], SCAPT [10], SEAS [12] and SMS Pro [4]) did not provide sufficient information allowing us to conduct an educated evaluation, and are thus excluded from further discussion[3].

Out of the 37 AC tools (see Table 2), 32 offer support for GSN [6]. Some exceptions to this are Modus [44] (a plug-in for Enterprise Architect), ACBuilder [27], NOR-STA [24], etc., which have their own notations. Multiple tools (e.g., CertWare [13] and ASCE [40]) also offer support for a variety of different notations, such as the Structured Assurance Case Metamodel (SACM) [7] and the Claims-Arguments-Evidence (CAE) [15] notations, in addition to others. Our findings also show that most of the tools are not domain specific, meaning that they can be used to construct ACs for military, automotive, medical, and nuclear systems, among others. Exceptions to this are tools such as ACBuilder [27] (hardware security analysis) and TurboAC [5] (medical devices). Non domain specific tools (e.g., D-Case Editor [37]) have been marked with a hyphen under the domain column in Table 2.

3.1 Evaluation of the Tools and Discussion

Each tool has been manually evaluated for its support in the previously established categories, with the results shown in Table 3. Figure 1 represents the overall grade distribution for each category. To simplify visualization, all split grades have been rounded up and represented as the higher grade.

Creation. Support for creation of ACs primarily ranges between minimal (43%) and moderate (49%) (see Fig. 1(a)). The notable exceptions, ENTRUST [16] and Resolute [23], offer strong support by providing the automatic generation of

[3] A table listing more information about each evaluated tool, such as where it was produced, how it was discovered, a link to the tool, its availability, its supported notations and domain, can be accessed at goo.gl/A4yWs9.

Table 2. General tool information.

Tool name	Supported notations	Domain
ACBuilder [27]	Textual	Hardware security analysis
ACCESS [32]	GSN	-
ACEdit [32] https://github.com/arapost/acedit	GSN, ARM	-
AdvoCATE [20] https://ti.arc.nasa.gov/tech/rse/research/advocate/	GSN, SACM, Bowtie	-
AGSN [35] https://github.com/AGSNeditor/development	GSN	-
ASCE [40] https://www.adelard.com/asce/choosing-asce/index/	CAE, SACM, GSN, Bowtie	
Assure-It [45]	GSN	-
Astah GSN [32] http://astah.net/download	GSN, ARM, SACM	-
Artisan GSN modeler [2]	GSN	-
AutoFOCUS3 [17] https://af3.fortiss.org/download/	GSN	Distributed, reactive, embedded software systems
CertWare [13] https://nasa.github.io/CertWare/	ARM, CAE, GSN, EUROCONTROL	-
D-Case Communicator [38] https://mlab.ce.cst.nihon-u.ac.jp/dcase/login.html	GSN	-
D-Case Editor [37] http://www.jst.go.jp/crest/crest-os/osddeos/en/tech.html	GSN, SACM	-
D-Case Weaver [21] http://www.jst.go.jp/crest/crest-os/osddeos/en/tech.html	GSN	-
D-MILS [18] https://github.com/phy3rdh/DmilsMBAC	GSN	-
Eclipse & Papyrus extension [26]	GSN	-
eDependabilityCase [33]	GSN	-
ENTRUST [16] https://github.com/gerasimou/ENTRUST	GSN	Self-adaptive software
eSafetyCase Toolkit [41]	GSN	-
ETB (Evidential Tool Bus) [19] https://github.com/SRI-CSL/ETB	Claims table	-
Event-B extension [30]	GSN	-
EviCA [39] Can be acquired by emailing the authors	GSN	-
GAGE [14]	GSN	-
HiP-HOPS extension [43] http://www.hip-hops.eu/	GSN	-
ISCaDE [32] http://www.iscade.co.uk/	GSN, ASCAD, WeFA	-
MMINT-A [22] https://github.com/adisandro/MMINT	GSN	-
Modus [44] http://modelme.simula.no/Modus	KAOS	-
NOR-STA [24] https://www.argevide.com/purchase/assurance-case/	TRUST-IT Argument Representation	-
OpenCERT [31] https://www.polarsys.org/proposals/opencert	GSN	-
Resolute [23] https://github.com/smaccm/smaccm	Unique notation	Distributed real-time embedded systems
SafeEd [25] http://cs-gw.utcluj.ro/~adrian/tools/safed/gsn/gsn.html	GSN	-
Safety.Lab [42]	GSN	-
SAM [29]	GSN	-
SCT: Safety Case Toolkit [9] http://www.dependablecomputing.com/	GSN, MDD (MultiMarkdown doc.)	-
SBVR/GSN Editor [36]	GSN	-
TurboAC [5] http://www.gessnet.com/	Subset of GSN, Tabular, Narrative	Medical devices
Visio add-on [32] http://www-users.cs.york.ac.uk/~tpk/gsn/	GSN	-

Table 3. Evaluation of capabilities of individual tools.

Tool name	Creation	Maintenance	Assessment	Collaboration	Reporting	Integration
ACBuilder [27]	B	D	D	D	D	D
ACCESS [32]	B	C	C	D	C	D
ACEdit [32]	C	C	B	D	D	D
AdvoCATE [20]	B	B	A	D	A/B	B
AGSN [35]	C	C	B	D	C	D
ASCE [40]	C	B	B	B	A/B	C
Assure-It [45]	C	C/D	D	D	D	D
Astah GSN [32]	B	C	B	D	C	D
Artisan GSN modeler [2]	B	C	B	A	D	D
AutoFOCUS3 [17]	B	B	B	D	D	B
CertWare [13]	B	B	A	C	D	B
D-Case Communicator [38]	C	C	D	C	D	D
D-Case Editor [37]	B	B	B	D	D	B
D-Case Weaver [21]	C	C	C	C	C	B
D-MILS [18]	B	B	B	D	D	B
Eclipse & Papyrus Ext. [26]	C	C	A	D	D	D
eDependabilityCase [33]	C	C	B	D	D	D
ENTRUST [16]	A	A	C	D	D	B
eSafetyCase Toolkit [41]	B	C	B	B	B	D
ETB (Evidential Tool Bus) [19]	C	A	D	C	D	B
Event-B extension [30]	B	C	B	D	D	B
EviCA [39]	C	C	B	D	D	D
GAGE [14]	D	B	B	D	D	D
HiP-HOPS extension [43]	B	B	D	D	D	B
ISCaDE [32]	B	C	C	B	A/B	B
MMINT-A [22]	C	B	C	D	D	C
Modus [44]	C	B	A	B	C	C
NOR-STA [24]	B	B	B	A	A/B	C
OpenCERT [31]	B	B	C	B	B	C
Resolute [23]	A	B	A	D	C	B
SafeEd [25]	C	C	A	D	B	C
Safety.Lab [42]	C	B	A/B	D	D	B
SAM [29]	B	B	C	D	B	B
SCT: Safety Case Toolkit [9]	B	C	C	A/B	A	C
SBVR/GSN Editor [36]	C	C	D	D	D	C
TurboAC [5]	B	C	C	D	A/B	B
Visio add-on [32]	C	C	D	D	D	D

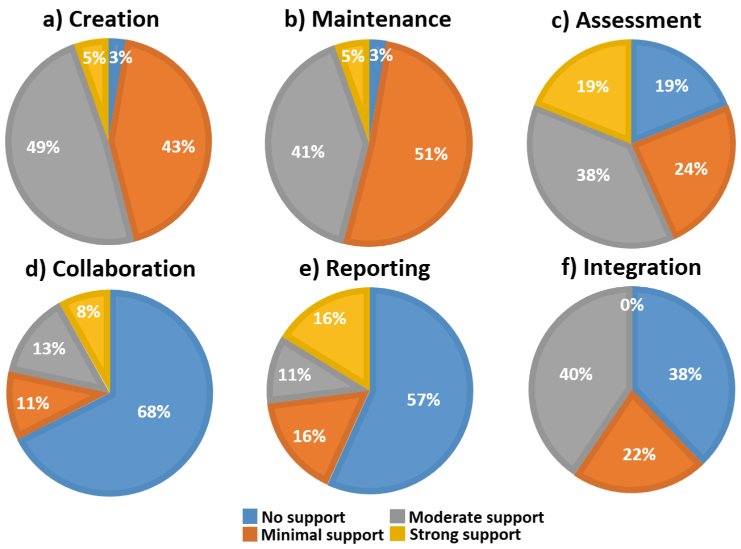

Fig. 1. Overall AC tool support for: (a) creation, (b) maintenance, (c) assessment, (d) collaboration, (e) reporting and (f) integration.

ACs, based on various underlying system and/or behavioral models. As previously mentioned however, these tools are domain specific. Unless modified, their use is confined to the specific underlying architectural languages, models, etc., that they support. To our knowledge, a tool that can automatically generate complete ACs for a broad range of domains is yet to be developed. Based on these observations, it would seem that the benefits obtained by creating a strong dependency between ACs and system models come at the cost of flexibility and generalized usability.

Maintenance. Again, the absolute majority of tools provide either minimal (51%) or moderate (41%) support for maintenance (see Fig. 1(b)). Tools with moderate support for maintenance often allow the linking of evidence, models and other artefacts to the corresponding AC elements, making it easy to notify the user of the impacts of the change. In turn, ENTRUST [16] and ETB [19] offer strong support by automatically reflecting artefact changes on the AC. ETB [19] allows the incorporation of 3rd party tools for the purpose of generating evidence and logs timestamps of their invocations in order to determine which analyses are out of date with respect to the current development artefacts, re-running those that are not synchronized. ENTRUST [16] is tightly coupled with the design-time and runtime models of a system. It has the ability to dynamically verify self-adaptive systems at runtime and update their ACs as necessary.

Assessment. Figure 1(c) shows that the results for AC assessment are fairly distributed among all levels of support as compared to the other functional categories, with the majority offering moderate support (38%). The highest percentage of strong support (19%) is seen in this category. Unlike creation and maintenance however, 19% of tools offer no support for assessment. Furthermore, no correlation is seen between support for assessment and any other category, implying that assessment is a fairly standalone tool functionality, the support for which is not largely dependent on the other categories.

Collaboration and Reporting. Most of the tools we surveyed offer no support for collaboration (68%) or reporting (57%). A pronounced trend (see Table 3) is that tools with support in these categories are usually industrial, such as ASCE [40], ISCaDE [32], NOR-STA [24], OpenCERT [31] and SCT: Safety Case Toolkit [9]. Perhaps such capabilities are not receiving adequate interest among researchers, and thus are being developed only after tools reach significant maturity, if at all.

Integration. Support for integration is split between moderate (40%) and none (38%). Not a single tool among the ones we evaluated offered strong support, indicating that some manual integration between other assurance lifecycle activities and the ACs is always required. Table 3 shows a strong correlation between high support for integration, and high support for AC creation and maintenance. It would appear that a more integrated environment allows tighter coupling between various artefacts, such as system models and evidence, subsequently enabling automation through dependencies. As previously discussed however, the creation of these dependencies might introduce limitations in other aspects.

3.2 Threats to Validity

The main threat to validity in our work is the completeness of our list of tools and tool information. Even though our search methodology is thorough, it is possible that it did not capture all existing AC tools. As discussed in Sect. 2, our evaluation was based only on information found in the corresponding tool's documentation, publications, website and other publically available resources. It is possible that the description of some functionality received a lower grade because it was not adequately described or the relevant resource was unavailable.

4 Summary and Conclusion

In this paper, we reported on a comprehensive identification and a preliminary evaluation of AC tools, comparing them w.r.t. several categories using the available documentation. In the future, we intend to refine our results using deeper analysis, through a systematic evaluation of the tools themselves.

Our experience shows that there is significant room for improvement of the tools in all of the discussed categories. Furthermore, it appears that several categories are interdependent, i.e., high support in one is strongly correlated with high support in another. For example, we expect that improvements in the integration category will significantly benefit other categories such as creation and maintenance. Yet, to the best of our knowledge, there is currently no tool that supports the seamless linking of the various assurance lifecycle processes.

Acknowledgements. The work reported in this paper has been funded by General Motors and NSERC Canada. The authors thank Mark Lawford, Alan Wassyng and Tom Maibaum for many useful discussions about assurance cases.

References

1. AssureNote. https://github.com/AssureNote/AssureNote
2. Impact case study - University of York. https://impact.ref.ac.uk/CaseStudies/CaseStudy.aspx?Id=43445
3. PREEVision. https://vector.com/vi_preevision-iso26262_en.html
4. SMS Pro. https://www.asms-pro.com/Modules/SafetyAssurance/SafetyCaseStudy.aspx
5. TurboAC. http://www.gessnet.com/products
6. Goal Structuring Notation Working Group: GSN Community Standard Version 1, November 2011. http://www.goalstructuringnotation.info/
7. Object Management Group: Structured Assurance Case Metamodel (SACM) version 1.0. Formal, 01 February 2013 (2013)
8. Kitchenham, B.: Guidelines for Performing Systematic Literature Reviews in Software Engineering. Technical rep. EBSE-2007-01, EBSE (2007)
9. Aiello, M.A., Hocking, A.B., Knight, J.C., Rowanhill, J.C.: SCT: a safety case toolkit. In: Proceedings ISSRE 2014 Workshops, pp. 216–219 (2014)
10. Allan, J., Williams, J., Gander-Miller, G., Turner, M., Ballantyne, T., Harvey, J.: Safety case production. WIT Trans. Built Environ. **37** (1998)
11. Althammer, E., Schoitsch, E., Eriksson, H., Vinter, J.: The DECOS concept of generic safety cases - a step towards modular certification. In: Proceedings of SEAA 2009, pp. 537–545 (2009)
12. Ankrum, T.S., Kromholz, A.H.: Structured assurance cases: three common standards (presentation). In: Proceedings of HASE 2005, pp. 99–108 (2005)
13. Barry, M.R.: CertWare: a workbench for safety case production and analysis. In: Proceedings of Aerospace Conference 2011, pp. 1–10 (2011)
14. Bjornander, S., Land, R., Graydon, P., Lundqvist, K., Conmy, P.: A method to formally evaluate safety case arguments against a system architecture model. In: Proceedings of ISSREW 2012, pp. 337–342 (2012)
15. Bloomfield, R., Bishop, P.: Safety and assurance cases: past, present and possible future – an adelard perspective. In: Dale, C., Anderson, T. (eds.) Making Systems Safer, pp. 51–67. Springer, London (2010). https://doi.org/10.1007/978-1-84996-086-1_4
16. Calinescu, R., Weyns, D., Gerasimou, S., Iftikhar, M.U., Habli, I., Kelly, T.: Engineering trustworthy self-adaptive software with dynamic assurance cases. IEEE TSE **PP**(99), 1–30 (2017)
17. Cârlan, C., Barner, S., Diewald, A., Tsalidis, A., Voss, S.: ExplicitCase: integrated model-based development of system and safety cases. In: Tonetta, S., Schoitsch, E., Bitsch, F. (eds.) SAFECOMP 2017. LNCS, vol. 10489, pp. 52–63. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66284-8_5
18. Cimatti, A., DeLong, R., Marcantonio, D., Tonetta, S.: Combining MILS with contract-based design for safety and security requirements. In: Koornneef, F., van Gulijk, C. (eds.) SAFECOMP 2015. LNCS, vol. 9338, pp. 264–276. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24249-1_23

19. Cruanes, S., Hamon, G., Owre, S., Shankar, N.: Tool integration with the evidential tool bus. In: Giacobazzi, R., Berdine, J., Mastroeni, I. (eds.) VMCAI 2013. LNCS, vol. 7737, pp. 275–294. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35873-9_18
20. Denney, E., Pai, G.: Tool support for assurance case development. J. Autom. Softw. Eng. **25**(3), 435–499 (2018)
21. Fujita, H., Matsuno, Y., Hanawa, T., Sato, M., Kato, S., Ishikawa, Y.: DS-bench toolset: tools for dependability benchmarking with simulation and assurance. In: Proceedings of DSN 2012, pp. 1–8 (2012)
22. Fung, N.L.S., Kokaly, S., Di Sandro, A., Salay, R., Chechik, M.: MMINT-A: a tool for automated change impact assessment of assurance cases. In: Proceedings of SAFECOMP 2018 Workshops. Springer (2018, accepted for publication)
23. Gacek, A., Backes, J., Cofer, D., Slind, K., Whalen, M.: Resolute: an assurance case language for architecture models. In: Proceedings HILT 2014, pp. 19–28 (2014)
24. Górski, J., Jarzębowicz, A., Miler, J., Witkowicz, M., Czyżnikiewicz, J., Jar, P.: Supporting assurance by evidence-based argument services. In: Ortmeier, F., Daniel, P. (eds.) SAFECOMP 2012. LNCS, vol. 7613, pp. 417–426. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33675-1_39
25. Groza, A., Marc, N.: Consistency checking of safety arguments in the goal structuring notation standard. In: Proceedings of ICCP 2014, pp. 59–66 (2014)
26. Huhn, M., Zechner, A.: Analysing dependability case arguments using quality models. In: Buth, B., Rabe, G., Seyfarth, T. (eds.) SAFECOMP 2009. LNCS, vol. 5775, pp. 118–131. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04468-7_11
27. Kawakami, H., Ott, D., Wong, H.C., Dahab, R., Gallo, R.: ACBuilder: a tool for hardware architecture security evaluation. In: Proceedings of HOST 2016, pp. 97–102 (2016)
28. Kelly, T.P.: Arguing Safety: A Systematic Approach to Managing Safety Cases. Ph.D. thesis, Univ. of York, UK (1998)
29. Kelly, T., McDermid, J.: A systematic approach to safety case maintenance. J. Reliab. Eng. Syst. Saf. **1**(3), 271–284 (2001)
30. Laibinis, L., Troubitsyna, E., Prokhorova, Y., Iliasov, A., Romanovsky, A.: From requirements engineering to safety assurance: refinement approach. In: Li, X., Liu, Z., Yi, W. (eds.) SETTA 2015. LNCS, vol. 9409, pp. 201–216. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25942-0_13
31. Larrucea, X.: Modelling and certifying safety for cyber-physical systems: an educational experiment. In: Proceedings of SEAA 2016, pp. 198–205 (2016)
32. Larrucea, X., Walker, A., Colomo-Palacios, R.: Supporting the management of reusable automotive software. IEEE Softw. J. **34**(3), 40–47 (2017)
33. Lautieri, S., Cooper, D., Jackson, D., Cockram, T.: Assurance cases: how assured are you? In: Proceedings of DSN 2004 Supplemental Volume (2004)
34. Lewis, R.: Safety case development as an information modelling problem. In: Dale, C., Anderson, T. (eds.) Safety-Critical Systems: Problems, Process and Practice, pp. 183–193. Springer, London (2009). https://doi.org/10.1007/978-1-84882-349-5_12
35. Luo, Y., van den Brand, M., Li, Z., Saberi, A.: A systematic approach and tool support for GSN-based safety case assessment. J. Syst. Archit. **76**(pp), 1–16 (2017)
36. Luo, Y., van den Brand, M., Kiburse, A.: Safety case development with SBVR-based controlled language. In: Desfray, P., Filipe, J., Hammoudi, S., Pires, L.F. (eds.) MODELSWARD 2015. CCIS, vol. 580, pp. 3–17. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27869-8_1

37. Matsuno, Y., Takamura, H., Ishikawa, Y.: A dependability case editor with pattern library. In: Proceedings of HASE 2010, pp. 170–171 (2010)
38. Matsuno, Y.: D-case communicator: a web based GSN editor for multiple stakeholders. In: Tonetta, S., Schoitsch, E., Bitsch, F. (eds.) SAFECOMP 2017. LNCS, vol. 10489, pp. 64–69. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66284-8_6
39. Nair, S., Walkinshaw, N., Kelly, T., de la Vara, J.L.: An evidential reasoning approach for assessing confidence in safety evidence. In: Proceedings of ISSRE 2015, pp. 541–552 (2015)
40. Netkachova, K., Netkachov, O., Bloomfield, R.: Tool Support for assurance case building blocks. In: Koornneef, F., van Gulijk, C. (eds.) SAFECOMP 2015. LNCS, vol. 9338, pp. 62–71. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24249-1_6
41. Newton, A., Vickers, A.: The benefits of electronic safety cases. In: Redmill, F., Anderson, T. (eds.) The Safety of Systems, pp. 69–82. Springer, London (2007). https://doi.org/10.1007/978-1-84628-806-7_5
42. Ratiu, D., Zeller, M., Killian, L.: Safety.Lab: model-based domain specific tooling for safety argumentation. In: Koornneef, F., van Gulijk, C. (eds.) SAFECOMP 2015. LNCS, vol. 9338, pp. 72–82. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24249-1_7
43. Retouniotis, A., Papadopoulos, Y., Sorokos, I., Parker, D., Matragkas, N., Sharvia, S.: Model-connected safety cases. In: Bozzano, M., Papadopoulos, Y. (eds.) IMBSA 2017. LNCS, vol. 10437, pp. 50–63. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64119-5_4
44. Sabetzadeh, M., Falessi, D., Briand, L., Di Alesio, S.: A goal-based approach for qualification of new technologies: foundations, tool support, and industrial validation. J. Reliab. Eng. Syst. Saf. **119**(C), 52–66 (2013)
45. Shida, S., Uchida, A., Ishii, M., Ide, M., Kuramitsu, K.: Assure-It: a runtime synchronization tool of assurance cases. In: SAFECOMP 2013 FastAbstract (2013)
46. Zhang, H., Babar, M.A., Tell, P.: Identifying relevant studies in software engineering. J. Inf. Soft. Technol. **53**(6), 625–637 (2011)