Systematic Evaluation of (Safety) Assurance Cases

Thomas Chowdhury(✉), Alan Wassyng, Richard F. Paige, and Mark Lawford

McMaster Centre for Software Certification, McMaster University, Hamilton, Canada
{chowdt2,wassyng,paigeri,lawford}@mcmaster.ca

Abstract. An Assurance Case (AC) documents an argument that supports a claim made about a system. An *effective* Assurance Case provides adequate belief to stakeholders that the system under consideration adequately embodies specific critical properties, for example safety and security. Comprehensive evaluation of an AC is a necessary step in building this belief. This involves measuring confidence in the assurance case argument, but also includes an overall quality assessment of the AC. This paper describes essential components of a (safety) AC evaluation process using previously defined evaluation criteria. These criteria were classified as applying to either *structure* or *content* of the (safety) AC. Two example (safety) ACs are used to demonstrate the approach, and for brevity, we illustrate the examples using purely *Goal Structuring Notation (GSN)* and in a second example, a *GSN*-like notation.

Keywords: Assurance case · Safety · GSN · Traceability

1 Introduction

An Assurance Case (AC) is a generalization of a safety case for a particular system. It is a living document that provides arguments that assure critical properties the system is required to embody. For demonstration purposes within the restrictions of this paper, we focus on safety as the critical property to be assured. One of the main objectives of an AC evaluation is to ensure that arguments in the AC are valid and sound. However, sound argumentation is not enough. The AC must be understandable by all stakeholders. It must also be of sufficient quality to engender trust that sufficient care was taken in its construction. Any weakness in claims, arguments or evidence potentially degrades the quality of an AC. The objective of our work is to assess the *overall quality* of an AC. This paper describes essential elements of a systematic and comprehensive evaluation process guided by previously defined evaluation criteria [4].

There are many notations used to document ACs. Goal Structuring Notation (GSN) [7] has gained considerable popularity in academia and is making inroads in industry. Thus, we use GSN examples to illustrate our evaluation process.

A. Casimiro et al. (Eds.): SAFECOMP 2020, LNCS 12234, pp. 18–33, 2020.
https://doi.org/10.1007/978-3-030-54549-9_2

1.1 Contribution

The contribution of this work is the definition of a systematic approach to the comprehensive evaluation of ACs. In this paper:

- we present pertinent aspects of our approach, supplemented by high-level, semi-formal models of the evaluation process and its outcomes;
- we provide sufficient detail to illustrate our approach on three criteria from [4] – two that apply to content, and another one that applies to structure of an AC;
- we evaluate our approach using two example ACs. The one AC is documented in GSN, the other is documented using a GSN-like notation.

2 Preliminaries

This section briefly introduces assurance cases, GSN, and recaps the evaluation criteria we presented in [4].

2.1 Assurance Cases

According to Bloomfield et al., "An Assurance Case is a documented body of evidence that provides a convincing and valid argument that a specified set of critical claims about a system's properties are adequately justified for a given application in a given environment" [3]. In general, an AC starts with a top-level claim regarding critical properties of a system, which is then supported by sub-claims. Terminal claims are grounded in evidence from the development of the system. There are many notations used to document ACs. Some of them are largely textual, and others are largely graphical with extensive references to textual documents produced during system development. The primary reason to develop a (safety) AC is to present explicit, understandable reasoning as to why we should believe that the system of interest is adequately safe.

2.2 Goal Structuring Notation

Goal Structuring Notation (GSN) was developed by Tim Kelly [7], motivated by Toulmin's work on argumentation [11]. The main body of a GSN diagram consists of *goals* representing claims and sub-claims, optional *strategies* that describe how goals are decomposed into sub-goals as representative of an argument, and *solutions* representing evidence that supports terminal goals. In addition, the AC developer can include supplementary information by adding *assumptions*, *context*, and *justifications*.

2.3 Recap of Our Evaluation Criteria in [4]

In [4] we defined criteria for the evaluation of an AC. We categorized the criteria into two groups, one for the structure of an AC and another for the content of an AC. Furthermore, we defined them from two different evaluation perspectives: i) the AC developer's perspective; and ii) the external reviewer's perspective. Table 1 presents all the criteria. We have added a little extra discussion for three of these criteria below the table. These three criteria are the same criteria for which details of the evaluation process are described in Sect. 4.

Table 1. A list of evaluation criteria and their rationale (from [4]).

Evaluation criteria for structure		Evaluation criteria for content	
Criterion	Rationale	Criterion	Rationale
Syntax check	Difficult to navigate and understand if syntax is not well-defined	Convincing basis	A feasible top-level claim is essential. Reasoning needs to be explicit so that it can be reviewed. Confirmation bias can adversely affect reasoning and the acceptance of evidence
Traceability	Necessary for understanding and maintenance	Rigour of argument	Rigour is important in making the reasoning less subjective and more repeatable
Robustness	Essential to achieving incremental assurance	Quality of hazard analysis	Hazard identification and mitigation is a critical aspect in assuring safety
Understandability	Need to facilitate understanding through structure	Arguing complete-ness	Deficiencies in completeness are a common source of error
Efficiency	Need to facilitate the ease with which ACs can be evaluated through structure and notation	Repeated arguments	A source of error if they are used where not completely appropriate
		ALARP	ALARP and associated principles are essential in demonstrating cost-benefit considerations and due diligence
		Confidence	An essential measure of trust in the reasoning and associated evidence. Not dealt with in detail in this paper because of the abundance of publications on this topic

- ***Structure Evaluation-Syntax Check:*** In addition to affecting understandability, syntax errors in the AC may indicate a lack of care taken in its construction and thus adversely affect the perceived quality of the AC.
- ***Content Evaluation-Convincing Basis:*** As with any complex document, the overview presented to readers is crucially important. The overview in an AC is represented by the top-level of the argument – the top claim and its immediate supporting sub-claims and their associated assumptions and context. Throughout the AC, the argument that supports upper-level claims should be explicit. It may be described in natural language, a logic of some kind, or a combination of these. The important point is that for every (sub)claim, there needs to be some reasoning that shows why, if its premises are true, then the parent claim is true. "Confirmation bias" [8] is another challenge in ACs. A simple example is when people look for specific evidence that supports a claim without considering counter-evidence. Apparent confirmation bias degrades confidence in an AC.
- ***Content Evaluation-Rigour of the argument:*** Explicit argumentation is an important characteristic of an AC. Evaluation of rigour of the argument is complementary to evaluating the convincing basis. Presentation of the argument in natural language is not as convincing as semi-formal notations or rigorous application of reasoning patterns.

3 Related Work

Reference [4] included a thorough literature review on the evaluation of an AC in the context of defining evaluation criteria. We have added some additional publications that specifically deal with the process of evaluating ACs. In [10], the author uses problem focused guide words (incorrectly phrased, relevance, directness, deductively invalid, undercutting evidence, rebutting evidence, low inductive strength, high inductive strength, coverage, replicability) to structure the evaluation process, followed by suggestions on how these often can be fixed. In [9], the assessment process consists of four steps (preparation, logic and structure validation, quality evaluation, record and feedback) performed by a safety assessor, who makes recommendations that are then implemented by the safety case developer. In [6], the Health and Safety Executive (HSE) defines 36 principles categorized into 10 groups that are used to assess safety cases.

A key part of evaluating ACs is assessing confidence; there has been much work published on this (much of which is reviewed in [4]). A relatively recent publication [2] has been added to the literature on AC confidence. It introduces a confidence measure technique 'INCIDENCE', which considers both design time and run time evidence and uses GSN as an example.

4 Evaluation of an Assurance Case

This section presents details of our systematic evaluation of ACs. To put this on a well-structured footing, we started by modelling the evaluation process and its

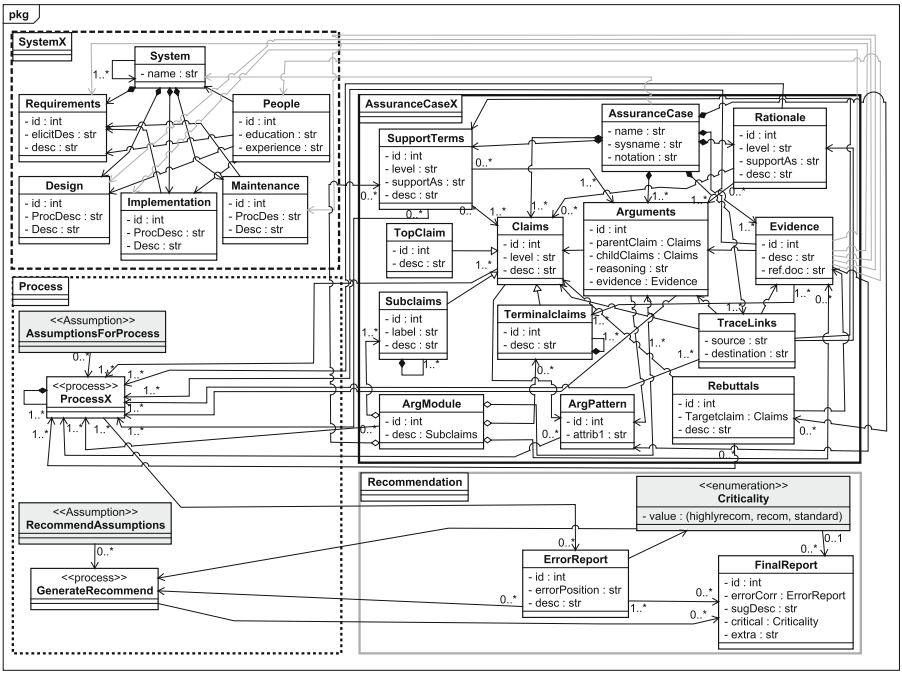

Fig. 1. The generic model of the AC evaluation process.

relevant data, including all primary components of an AC as well as development artefacts from the system of interest. This generic model is shown in Fig. 1. The generic model shows explicitly process, recommendation, AC data and system development data with associations and data flows required for the systematic evaluation. The model serves as a guide to the evaluation as well as a check on it consistency. The main components are (colours are not shown in this paper):

- The Process for evaluating the AC (represented by Green rectangles);
- The Recommendation arising from the evaluation (Blue rectangles);
- AC data that is the subject of the evaluation (Yellow rectangles);
- System development data that is referred to in the AC (Orange rectangles).

In addition,

- Black arrows/lines are used for input and output and associations;
- Red arrows/lines are used to highlight links between the AC and system development artefacts.

Figure 1 contains all essential links for refinement of all 12 criteria. In this paper, we had space to focus on only 3 criteria. The generic model must be refined and instantiated for specific evaluation criteria. The generic model systematizes the process of defining an evaluation process for arbitrary AC criteria, making AC

evaluation more repeatable and less error prone, as we can rely on the model to guide us as to how to carry out the instantiation. Refinement will involve precisely modeling inputs and outputs of individual steps in an evaluation process. Instantiation will involve adding textual descriptions for process stages, which can be checked for conformance with the components of the model.

To illustrate refinement and instantiation of our model we chose one content criterion and one structural criterion as examples for instantiation. The content criterion we selected was "Convincing basis for the AC" and we have included the refined version of the model for *Convincing basis* in Fig. 2, and the instantiated process in Sect. 4.1:1. We also include a much briefer discussion on another content criterion, "Rigour of the argument" in Sect. 4.1:2. The structural criterion we selected to include in this paper was "Syntax check". In this case we show the instantiated process in Sect. 4.2, but there is no space to include another figure showing the refined model. (Actually, we wanted to include "Traceability" as the structural criterion, but it is too complex to show in the space available).

We conducted a self-validation of our AC evaluation processes as a first step in evaluating our approach. A full-scale evaluation of the process is very difficult to arrange at this stage of development. The result of our self-validation of the process for *Convincing basis* is documented in Sect. 5.1; for *Rigour of the argument* in Sect. 5.2; and for *Syntax check* in Sect. 5.3.

4.1 Evaluating Content of an Assurance Case

We can now describe how we refined the high-level model for each of the evaluation criteria. We start with criteria related to content of the AC, and will show the major steps in evaluating *the convincing basis for the AC*.

1. ***Convincing basis for the AC:*** Figure 2 shows the relevant aspects of a refinement of the model in Fig. 1. We did not include the documentation resulting from the development of the system, since that part of the model does not change depending on the specific criterion being evaluated, and the links to that data are obvious.

One of the main intentions of *convincing basis* is to check explicitness of claims, arguments, supporting terms and evidence. In addition to this, a convincing basis looks for a complete top-level claim description, and compliance of evidence with acceptance criteria to avoid confirmation bias highlighted by Leveson [8].

The refinement shows that "ProcessX" now consists of 4 main steps (reading bottom to top):

- TopLevelClaimCheck – a review of the top level claim. Inputs to this process are the AC data items of the TopClaim itself, and TopClaimSupp.Terms. Output is simply to the ErrorReport. These links make it reasonably clear that the focus of this check is the wording of the top-level claim. Assumptions and criteria for this check are found in TopLevelAssumptions.

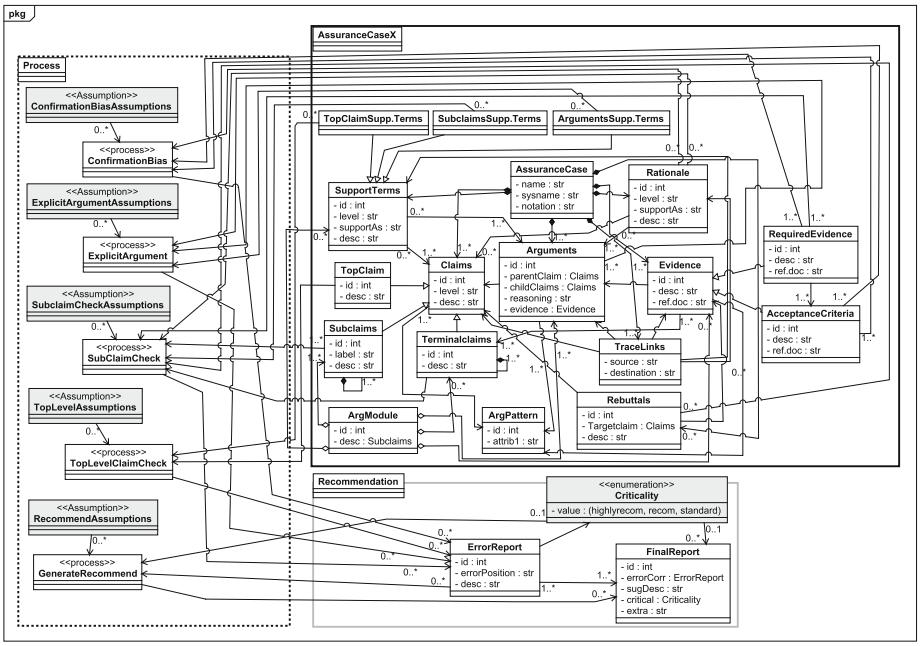

Fig. 2. The evaluation process for *Convincing basis for the AC*.

- SubClaimCheck – a review of all subclaims. Inputs to this process are Subclaims, SubclaimsSupp.Terms, Rationale, TerminalClaims, AcceptanceCriteria and the RequiredEvidence. Output is again to the ErrorReport. The focus of this check is on the wording and rationale for the decomposition of the argument, and also on whether or not the evidence required to support terminal claims makes sense. Assumptions and criteria for this check are to be found in SubclaimCheckAssumptions.
- ExplicitArgument – a review that evaluates how explicit the argument is, in general. Inputs to this process are ArgumentsSupp.Terms, Arguments and Rationale. Indirect inputs are Claims, Evidence, Rebuttals, ArgPatterns and ArgModules. Output is again to the ErrorReport. The focus of this check is on whether the argument, i.e., reasoning, is made visible explicitly in the AC.
- ConfirmationBias – a review that evaluates how susceptible the argument is to confirmation bias. Inputs to this process are Rebuttals, RequiredEvidence and AcceptanceCriteria. Output is again to the ErrorReport. The focus of this check is to ensure that the AC has specific safeguards against confirmation bias.

Instantiated Evaluation Process: We can now instantiate the model. We do this by describing the major steps in each of the 4 sub-processes. We can then check these steps to see that they conform to the model.

- ***TopLevelClaimCheck:***

(1) Top-level claim should consist of two parts: subject and predicate. The subject should represent a system or a component or subsystem of a system and the predicate should represent critical properties of that system to assure, contextual, environmental and operational information.
(2) The meaning of a top-level claim shall be clear and not create any ambiguity.
(3) All critical terms mentioned in a top-level claim shall be clarified.
(4) Necessary assumptions shall be stated explicitly.

- ***SubClaimCheck:***

(1) The meaning of a claim shall be clear and not create any ambiguity.
(2) All critical terms mentioned in a claim shall be clarified.
(3) Claims related to process or product or people shall be clarified to support upper-level claims.
(4) Necessary assumptions to support claims related to process or product or people shall be stated explicitly.
(5) Terminal claims shall be supported by proper evidence and acceptance criteria for evidence shall be clarified.

- ***(Review)ExplicitArgument:***

(1) The reasoning of how an upper-level claim is decomposed into supporting claims and/or evidence and how lower-level claims and/or evidence together support an upper-level claim shall be documented explicitly. The latter is more important than the former one.
(2) The rationale for reasoning shall be documented if it is necessary.
(3) All key terms mentioned in reasoning shall be clarified.
(4) Necessary assumptions in reasoning shall be clarified.

- ***(Review)ConfirmationBias:***

(1) Rebuttals shall be documented and resulting violation of a claim shall be documented.
(2) Evidence to support rebuttals shall be clarified.
(3) Evidence description shall comply with acceptance criteria for that specific evidence.

- ***GenerateRecommend:***

(1) For any error found in an AC, a recommendation shall be made with appropriate criticality (e.g. highly recommended, recommended, standard).

This process guides AC developers and external reviewers as follows:

- *AC Developer:* AC developers use the evaluation process from the beginning of an AC development. For example, AC developers may provide guidelines to system developers for defining boundary values for system functionalities, etc. It also guides AC developers to use rebuttals and thus avoid "confirmation bias", and to check that evidence complies with its acceptance criteria.

- *External Reviewer:* External reviewers are guided as to how to check claims, arguments and evidence using proposed procedures, and especially to examine claims for ambiguity. Furthermore, external reviewers are reminded to check for rebuttals, and judge whether or not they are adequately resolved.

2. ***Rigour of the Argument:*** We briefly describe another instantiated process for the content criterion – *rigour of the argument*. In this case, due to space limitations, we have not shown the refined model for this process.

Instantiated Evaluation Process: This criterion focuses on rigorous argument structure. Pattern instantiation may guide in achieving this, or a thorough description of argument may help in acquiring a rigorous argument. Such a description may be a deductive or inductive proof in an argument. The evaluation process "ProcessX" is refined in four checks: "CheckFormalArgument," "CheckInformalArgument," "CheckClaimForValidity" and "CheckRationaleForValidity". The rules for each check are as follows:

- ***CheckFormalArgument:***

(1) A formal argument shall be valid with necessary assumptions.
(2) Rationale to support the formal argument shall be clarified.
(3) All terms supporting the formal argument shall be valid.
(4) Rebuttals in a formal argument shall be clarified, and they shall be complete and consistent. (if it is found)
(5) Mitigation of rebuttals in a formal argument shall be clarified, and they shall be complete and consistent.(if rebuttals exist)
(6) An argument branch in an AC complying with an argument pattern shall thoroughly follow the pattern.

- ***CheckInformalArgument:***

(1) An informal argument shall be defined inductively, and the steps shall be complete and consistent.
(2) Rationale to support the informal argument shall be clarified.
(3) All terms supporting the informal argument shall be complete and consistent.
(4) Rebuttals in an informal argument shall be clarified, and they shall be consistent. (if it is found)
(5) Mitigation of rebuttals in an informal argument shall be clarified, and they shall be consistent. (if rebuttals exist)
(6) An argument branch in an AC complying with an argument pattern shall thoroughly follow the pattern.

- ***CheckClaimForValidity:***

(1) Claim shall be valid (by reviewing proofs-deductive or inductive), complete and consistent

(2) Rebuttals shall be valid (by reviewing proofs-deductive or inductive) and complete (if it is found)

- ***CheckRationaleForValidity:***

(1) Rationale shall be supported by deductive or inductive proofs.(if it is necessary).

- ***GenerateRecommend:***

(1) For any error found in an AC, a recommendation should be made with criticality (e.g. highly recommended, recommended, standard).

This process guides AC developers and external reviewers as follows:

- *AC Developer:* AC developers are guided to use more rigorous approaches to their arguments. Based on these checks, they are more likely to find and fix gaps/fallacies in arguments. They are reminded that there should be explicit reasoning to show how child claims support a parent claim. The AC developers may also find it worthwhile to provide documentation to external reviewers that aid them in understanding the arguments.
- *External Reviewer:* External reviewers have a basic check list that guides them in evaluating the rigour of the argument. It provides context for them in deciding whether or not the argumentation is defined with adequate rigour – and it does not have to be formal.

4.2 Evaluating the Structure of an Assurance Case

An AC must be evaluated in terms of its structure and content. In this section, we illustrate the instantiation of the evaluation model with one criterion for structure, more specifically for the "Syntax Check" criterion. We use a GSN example to show the approach and also describe how developers and external reviewers utilize the instantiated process.

Syntax check:

- ***Instantiated evaluation process:*** The "Syntax check" is an early but important stage of the evaluation process: without valid syntax, an AC is unusable in more sophisticated stages of evaluation. A syntax check can be performed with or without tool support. If a tool is used for syntax checking, experts should still review the syntax of an AC to avoid tool failures. In this illustration, we consider syntax checking of a graphical notation for ACs only for syntax checking as our example is documented using GSN; nevertheless we have defined rules for both graphical and textual syntax.
- ***CheckGraphSyntax:***

(1) Check what type of notation is defined. If it is a user-defined notation, obtain the documentation. Otherwise, a standard for a particular notation should be followed;

(2) Shapes of nodes shall be compliant with recommended shapes;
(3) There shall be one and only one association between any two nodes;
(4) Only valid associations shall exist between any two nodes;
(5) The only terminal nodes in the AC are those that in the defined syntax have no outgoing associated nodes;
(6) Label/identifier of a claim/argument/evidence should be defined in an acceptable format;

- ***CheckTextSyntax:***

(1) Check what type of notation is defined. If it is a user-defined notation, then one should look for the documentation;
(2) All artefacts of an AC shall comply with notation mentioned in the documentation.
(3) Label/identifier of a claim/argument/evidence should be defined in an acceptable format;

- ***GenerateRecommend:***

(1) For any error found in an AC, a recommendation should be made with criticality (e.g. highly recommended, recommended, standard).

This process guides AC developers and external reviewers as follows:

- *AC Developer:* AC Developers can evaluate the syntax of an AC with or without tool support. If developers evaluate manually, then they use rules to evaluate the AC. For instance, developers may generate a report if they encounter an error, e.g. shapes not complying with GSN community standard 2.0 [5] for an AC documented by GSN. This report can help developers to fix an AC before final submission to external reviewers. They should start the process as soon as the development of an AC starts. Developers should validate the tool in use. This evaluation should produce qualitative results instead of only boolean values (e.g. 'yes' or 'no'). AC developers can perform their evaluation without tool support as well. This is clearly more time consuming and probably more error prone. It is also more likely for "home grown" AC notations as compared with using a standard/commercially available technique. A simple notation that the AC has been checked for appropriate syntax may be welcomed by the external reviewer.

- *External Reviewer:* The procedure provides specific rules for syntax checking. External reviewers should also use tools when available. Syntax checks are relatively easy to define when the AC developers have provided adequate guidance as to what notation has been used.

5 Validation of the Evaluation Processes

This section presents a self-validation: applying the Evaluation Processes to example ACs.

5.1 Validation of "Convincing Basis for the AC" (A Content Criterion)

We use an excerpt of a GSN-like example of a coffee cup to illustrate the *Convincing basis* check. The top-level claim (represented by a rectangle) of the AC is labelled as 'TopClaim, C', contexts (represented by rounded rectangles) are labelled as 'K1', 'K2', 'K3' and 'K4', assumptions (represented by ovals) are labelled as 'A1' and 'A2', an argument (represented by a parallelogram) is labelled as 'R' and sub-claims (represented by rectangles) supporting the top-level claim are labelled as 'CR', 'CI', 'CPM' and 'CA'.

We use the AC for a coffee cup shown in Fig. 3. Four checks have defined rules that we use to perform an evaluation.

'TopLevelClaimCheck': evaluates the top-level claim in Fig. 3. Concerning rule (1), we find that top-level claim, "TopClaim, C" consists of two parts: the subject "The coffee cup $< X >$" specifies the system and the predicate "is safe in its intended environment, and its intended uses" specifies the critical

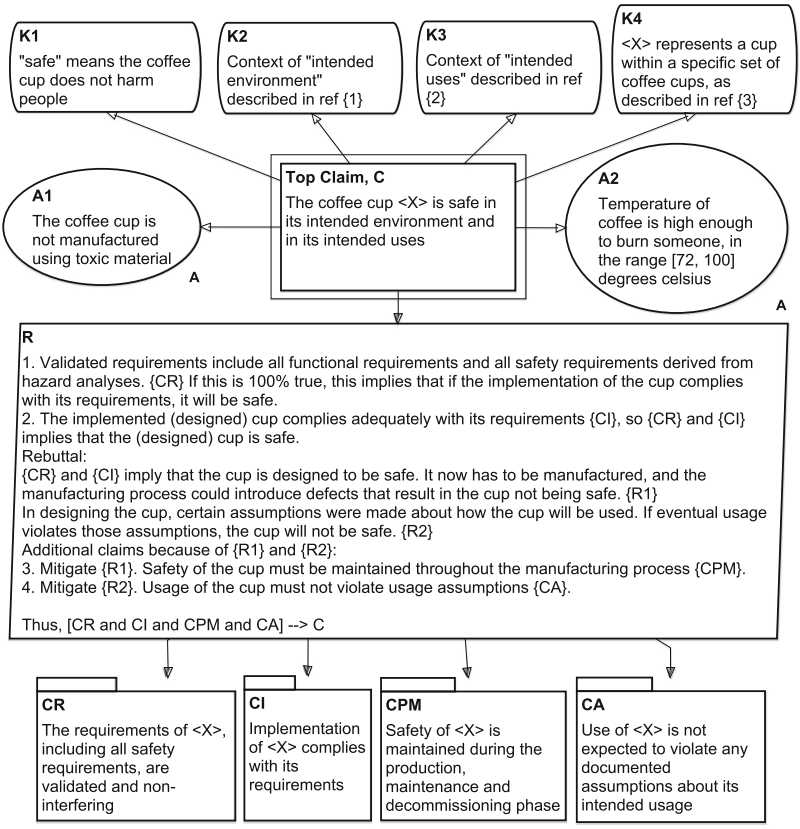

Fig. 3. The top two level claims of an AC for a coffee cup.

property 'safe', with environmental and operational conditions in the description. Concerning rule (2), we find that the meaning of the top-level claim is clear and does not create any ambiguity. Concerning rule (3), we find clarification of all terms (e.g. 'safe,' "intended environment," "intended uses," "coffee cup specification"). Concerning rule (4), we find that necessary assumptions (e.g. non-toxic material for a coffee cup and tolerable temperature) are clarified.

'SubClaimCheck': checks the second level claims of Fig. 3. The sub-claims 'CR,' 'CI,' 'CPM' and 'CA' are represented by modules as they contain implicit argument branches. Page restrictions prevent us from including them. Concerning rule (1), the meaning of all sub-claims is clear and does not create any ambiguity. Claim 'CR2.2.1.1.1.2.2' assures the competency of people in performing 'FTA' in Fig. 4.

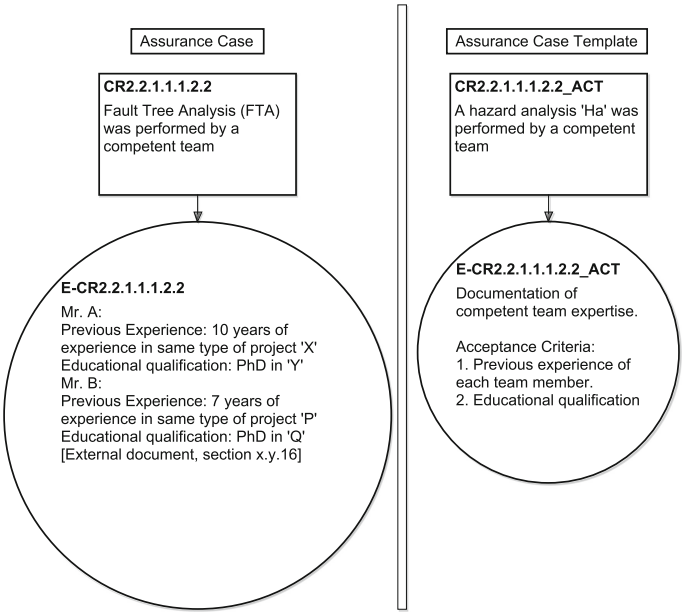

Fig. 4. An excerpt of evidence complying with acceptance criteria for a coffee cup example.

Concerning rule (2), we do not find any clarification for any term mentioned in the claims. Concerning rule (3), we find that claims 'CI' clarifies assuring implementation complies with requirements. Other claims ('CR', 'CPM' and 'CA') explain assuring valid and non-interfering requirements, ensuring safety during production, maintenance and decommissioning and operational assumptions. Concerning rule (4), we find no assumptions. Concerning rule (5), we find that terminal claim 'CR2.2.1.1.1.2.2' is supported by evidence 'E-CR2.2.1.1.1.2.2' and claim 'CR2.2.1.1.1.2.2_ACT' and acceptance criteria

'E-CR2.2.1.1.1.2.2_ACT' from an assurance case template are clarified. We use '(Review)ExplicitArgument' to evaluate the explicitness of an argument. Concerning rule (1), we find that argument 'R' describes explicit reasoning of how subclaims ('CR,' 'CI', 'CPM' and 'CA') support top-level claim 'C.' Argument 'R' demonstrates reasoning adequately along with rebuttals and mitigation of these rebuttals. Concerning rules (2), (3) and (4), we do not find any justification, context or assumption.

'(Review)ConfirmationBias': reviews confirmation bias. Concerning rule (1), we find that argument 'R' demonstrates rebuttals with possible counters explicitly. Concerning rules (2), we do not find any evidence to support those rebuttals. Concerning rule (3), Fig. 4 shows terminal claim 'CR2.2.1.1.1.2.2' is supported by evidence 'E-CR2.2.1.1.1.2.2' and evidence complies with acceptance criteria 'E-CR2.2.1.1.1.2.2'.

'GenerateRecommend' generates the following: a) It is recommended to clarify key terms, e.g. context of "documented assumption" mentioned in claim 'CA.' should be clarified. b) It is highly recommended to state necessary assumptions. c) It is recommended to state assumptions, justifications in reasoning 'R' e.g. justification for 'R' should be clarified. d) It is recommended to clarify evidence to support rebuttals.

5.2 Validation of "Rigour of the Argument" (A Content Criterion)

We use the same example in Fig. 3 to illustrate the rigour of the argument evaluation. The argument is informal so we use 'CheckInformalArgument.' We also perform 'CheckClaimForValidity' and 'CheckRationaleForValidity'.

'CheckInformalArgument': concerning rule (1), we find that the arguments are defined inductively with adequate steps, including rebuttals to support the upper-level claims, and they are complete and consistent. For instance, argument 'R' has four steps of reasoning. Concerning rule (2), we do not find any rationale for the argument, since 'R' is not supported by a justification. Concerning rule (3), we do not find any context or assumption to support the argument. Concerning rule (4), we find that some arguments do use rebuttals, e.g., 'R' uses two rebuttals, 'R1' and 'R2'. Rebuttals mentioned in the argument 'R' are consistent. Concerning rule (5), we find that the mitigation of each rebuttal is demonstrated. For instance, mitigations 'CPM' and 'CA' in argument 'R' resolve the rebuttals, 'R1' and 'R2'. Concerning rule (6), we find that an argument branch considers different phases of the development process.

'CheckClaimForValidity': concerning rule (1), we find that the sub-claims are valid, complete and consistent because arguments are valid and evidence complying with acceptance criteria support terminal claims. Sub-claims ('CR', 'CI,' 'CPM' and 'CA' are represented by modules, details of which are not included in this paper) are complete and consistent and valid supported by arguments. Concerning rule (2), we find that arguments have defined rebuttals and mitigations. Rebuttals are complete (shown earlier), but there is no proof to check the validity of those rebuttals.

'CheckRationaleForValidity': concerning rule (1) we find that no justification exists to support the argument.

'GenerateRecommend' produces the following: a) It is recommended that rationale should exist to support argument 'R.' b) It is recommended that for environmental or operational conditions during production, details of the maintenance stage should be clarified.

5.3 Validation of "Syntax Check" (A Structure Criterion)

To illustrate our syntax check process, we use AFI RVSM Pre-Implementation Safety Case [1] as an example. It uses GSN for documentation. The safety case shows safety arguments of RVSM (Reduced Vertical Separation Minimum) implementation and maintenance to reduce the vertical separation between Flight Levels 290 and 410 (inclusive) from 600 m to 300 m in AFI airspace. We apply the rules for syntax check to an assurance case of type GSN.

'CheckGraphSyntax': concerning rule (1), we consider the GSN community Standard 2.0 [5] as a reference. Concerning rule (2), by review we note that shapes of goal and strategy comply with the standard. However, the example refers to a solution as *evidence*, and they used a rounded rectangle for evidence instead of a circle. They used one context and did not use any assumption or justification in their safety case, though mentioned those terms in their example safety case, and they otherwise comply with the standard. However, the shape of the context used in the safety case does not comply with the standard. Concerning rule (3) and (4), there is one and only one valid association ('SupportedBy') that exists between any two nodes. For rule (5), the terminal nodes (in some pages, terminal nodes are goals, and in some pages, terminal nodes are evidence) have no outgoing association with other goals. With rule (6), the label of goals, strategies and evidence follows a hierarchy. Thus, with rule (2), one shape (context) does not comply with the standard.

'GenerateRecommend' generates that it is highly recommended to fix the shape to comply with the standard, or to explicitly document how and why it deviates from the standard.

6 Conclusion

Our proposed approach incorporates rules to identify known weaknesses in an AC. These weaknesses can be associated with specific, previously published evaluation criteria, and the evaluation process made more structured and systematic by using these criteria to drive the evaluation process. We illustrated the application of these evaluation rules, via refinement and instantiation of a generic evaluation process, for two criteria related to content of the AC, and one criterion related to structure of the AC. These three processes were then self-validated using one *GSN* example and another *GSN*-like example. We have thus shown that systematic and comprehensive evaluation of ACs is feasible.

References

1. ALTRAN-ATM Division, National Aerospace Laboratory NLR and AFI RVSM Project Management Team: AFI RVMS Pre-Implementation Safety Case, final edn., February 2008
2. Belle, A.B., Lethbridge, T.C., Kpodjedo, S., Adesina, O.O., Garzón, M.A.: A novel approach to measure confidence and uncertainty in assurance cases. In: 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW), pp. 24–33. IEEE (2019)
3. Bloomfield, R., Bishop, P., Jones, C., Froome, P.: ASCAD. Adelard Safety Case Development Manual. Adelard **5** (1998)
4. Chowdhury, T., Wassyng, A., Paige, R., Lawford, M.: Criteria to systematically evaluate (safety) assurance cases. In: 30th International Symposium on Software Reliability Engineering (ISSRE), pp. 380–390. IEEE (2019)
5. Group, A.C.W., et al.: Goal structuring notation community standard (version 2) (2018)
6. Hse, M.: Assessment Principles for Offshore Safety Cases (APOSC) (2006)
7. Kelly, T.P.: Arguing Safety–A Systematic Approach to Safety Case Management. The University of York, Department of Computer Science (1998)
8. Leveson, N.: Cost-effective safety certification of software-intensive systems. Seventh Software Certification Consortium (SCC), Annapolis, May 2011
9. Luo, Y., van den Brand, M., Li, Z., Saberi, A.K.: A systematic approach and tool support for GSN-based safety case assessment. J. Syst. Archit. **76**, 1–16 (2017)
10. Mayo, P.: Structured safety case evaluation: a systematic approach to safety case review. In: Proceedings of the First IET International Conference on System Safety, pp. 164–173 (2006)
11. Toulmin, S.E.: The Uses of Argument. Cambridge University Press, Cambridge (2003)