Assurance Case Templates: Principles for Their Development and Criteria for Their Evaluation

Assurance Case Templates: Principles for Their Development and Criteria for Their Evaluation

By

Thomas Chowdhury, M.Sc.

A Thesis

Submitted to the School of Graduate Studies
in Partial Fulfillment of the Requirements for the Degree

Doctor of Philosophy

McMaster University

Doctor of Philosophy (August, 2021) McMaster University
(Software Engineering) Hamilton, Ontario

Title:	Assurance Case Templates: Principles for Their Development and Criteria for Their Evaluation
Author:	Thomas Chowdhury, M.Sc. (University of Quebec)
Supervisor:	Dr. Alan Wassyng
Number of Pages:	xiii, 233

Lay Abstract

This thesis contributes specifically to how we build effective Assurance Cases (ACs) for safety-critical systems, and how we can evaluate the quality of an AC. An Assurance Case (AC) captures and presents explicit reasoning associated with assuring critical properties of a software-intensive system, such as safety. Concerning this, we defined principles to develop an Assurance Case Template (ACT) that complies with a standard and applied those principles to *ISO 26262* (functional safety for automotive vehicles) and *SAE J3061* (cybersecurity). An Assurance Case Template (ACT) is a complete assurance case that guides the development of systems within a product line. Later we used the resulting ACT's modification in a case study to guide us to pre-emptively mitigate against potential vulnerabilities in automotive over-the-air update implementation. Furthermore, we defined effective evaluation criteria for an AC and developed a systematic evaluation process to make it less subjective.

Abstract

An Assurance Case (AC) captures and presents explicit reasoning associated with assuring critical properties of a software intensive system, such as safety. This thesis contributes specifically to how we build effective ACs, and how we can evaluate the quality of an AC. Rather than simply add yet another set of patterns to the existing AC literature, we developed ten principles for constructing ACs from existing safety and security standards. This is our first contribution in this thesis. An Assurance Case Template (ACT) is a complete assurance case that guides the development of systems within a product line. In most cases safety critical systems have to comply with existing standards. Thus, an ACT that complies with a relevant standard can be used to guide development of systems that must comply with that standard. We applied our principles to *ISO 26262* (functional safety for automotive vehicles) and *SAE J3061* (cyber-security), and used the resulting ACT's specialization in a case study to guide us to pre-emptively mitigate against potential vulnerabilities in automotive over-the-air update implementations. A vital attribute of an AC is to facilitate the identification of fallacies in the validity of any claim. There is considerable published research related to confidence in ACs, which primarily relates to a measure of the soundness of reasoning. Evaluation of an AC should be more general than measuring confidence and should consider multiple aspects of the quality of an AC. Standard evaluation criteria could play a significant role in making the evaluation process more systematic. Another contribution of this research is the identification of effective evaluation criteria for ACs. Concerning this, we developed five criteria for structure evaluation and seven criteria for content evaluation of an assurance case. A final contribution of the thesis is the development of detailed AC evaluation methods that use the aforementioned evaluation criteria from the perspective of the developer of the AC as well as from the perspective of an external reviewer. The evaluation criteria and methods are applied in a simple case study to demonstrate how they may be used in practice.

Acknowledgments

I would like to express my gratitude towards my supervisor, Dr. Alan Wassyng for his superb advice, ideas, and continued support throughout my graduate study. I enjoyed our discussions and appreciated the way Dr. Wassyng answered all questions with care. I asked him (still I do!) a lot of questions even they are not related to my studies. As a part of his dynamic team, I was involved in several projects that gave me diverse experience and invaluable knowledge. I won't hesitate to take his opinion in future if I come across any technical issue.

Many thanks to Dr. Mark Lawford and Dr. Spencer Smith for serving on my supervisory committee, as well as for providing me constructive feedback and guidelines throughout this journey that improved my knowledge as well as research outcomes.

Many thanks to Dr. Richard Paige for his support. I enjoyed our interesting discussions and I learned a lot from him.

Many thanks to my external examiner, Dr. John Rushby, for reviewing my thesis and for insightful discussions at my defence.

I appreciate the interesting discussions with friends, fellow graduate students and colleagues from McSCert that I enjoyed a lot.

Last but not least, I truly believe that without blessings from my parents, Gopal Krishna Chowdhury and Nilima Chowdhury, it would not be possible to achieve this. I am very grateful to my parents for their support. I would like to thank Pinkey, my life partner, for supporting me throughout this incredible journey. It would not be possible without her constant mental support. It would be incomplete if I forgot the contribution of my little daughter, Audrina! She always makes our life joyous and keeps us smiling.

Contents

List of Figures

List of Tables

List of Acronyms

AC Assurance Case iii

ACT Assurance Case Template iii

ACC Adaptive Cruise Control 37

ASIL Automotive Safety Integrity Level 41

HARA Hazard Analysis and Risk Assessment 14

TARA Threat Analysis and Risk Assessment 58

OTA Over-The-Air xv

FTA Fault Tree Analysis xvi

STPA System Theoretic Process Analysis xvii

GSN Goal Structuring Notation 7

CAE Claims, Arguments and Evidence 8

OEM Original Equipment Manufacturer 3

FDA U.S. Food and Drug Administration 98

ALARP As Low As Reasonably Practicable 101

MITM Man-In-The-Middle attack 76

ECU Electronic Control Unit 17

DST Dempster-Shafer Theory 194

Declaration of Academic Achievement

I, Thomas Chowdhury, declare this thesis to be my work. I am the sole author of this document.

My supervisor, Dr. Alan Wassyng, as well as the members of my supervisory committee, Dr. Mark Lawford and Dr. Spencer Smith, provided me guidance and support at all stages of this research. I completed all of the research work contained within this document.

Chapter 1

Introduction

If a safety-critical system fails in some way, it can endanger human life and/or cause harm to its environment. It is thus crucial that we are able to develop safety-critical systems in ways that we can convince ourselves and others that they are safe to deploy. Concerning this, we need to plan the development and assessment of system functionality proactively. Assurance Cases (ACs) are a generalization of Safety Cases and are gaining momentum as a preferred way of demonstrating assurance of critical properties (e.g. safety, security) in complex software-intensive systems. Currently, the U.S. Food and Drug Administration (FDA) guidelines for some medical device submissions recommend the development of safety 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" [1].

It is a widespread belief that a "good" AC provides adequate confidence to stakeholders regarding the critical properties of their products. However, there is no standardized pattern to document ACs, which results in differently structured assurance cases for different products in a particular product family. This thesis focuses on the development of effective ACs. Concerning this, we develop principles to document an Assurance Case Template (ACT) from existing safety and security standards/guidance (e.g. *ISO 26262*, *SAE J3061*). An ACT is a complete assurance case that guides the development of systems within a particular product line. An ACT has characteristics of optionality and

multiplicity to provide support for the inclusion of likely changes. Furthermore, it also guides us in choosing real evidence based on defined acceptance criteria. In most cases, safety-critical systems have to comply with existing standards. Thus, an ACT that complies with a relevant standard can be used to guide the development of systems that must comply with that standard. We applied our principles to *ISO 26262* (functional safety for automotive vehicles) and *SAE J3061* (cyber-security) and used the resulting ACT's specialization in a case study to guide us to pre-emptively mitigate against potential vulnerabilities in automotive over-the-air update implementations.

Furthermore, this research focuses on a qualitative evaluation of ACs. ACs should be free from fallacies to provide adequate confidence to stakeholders. There is considerable published research related to confidence in ACs, which refers primarily to a measure of the soundness of reasoning. This is not the only way to evaluate ACs. Multiple aspects of an AC should be taken into consideration for an "effective" evaluation. We focus on two issues: structure and content evaluation of an AC. For an "effective" evaluation, we first define criteria for both structure and content analysis and then develop a systematic evaluation process using those criteria. Moreover, this thesis illustrates the evaluation process from two perspectives: developers and external reviewers. In general, the developer of the AC will have a more detailed knowledge of the AC than any future external reviewer. Furthermore, developers are in an advantageous position of having multiple ACs to compare during an evaluation, whereas external reviewers use their expertise and experience for evaluation.

1.1 Motivation

The value of an AC depends on how it presents arguments of assuring a critical property (e.g. safety, security, dependability). Developing an AC is often viewed as something that needs to be done simply to obtain regulatory approval, and in many cases, is prepared after the product has been built. However, an AC should be documented before a product is built to guide the development of a safe system.

In this thesis, we primarily focus on how an effective AC is built and how we effectively perform a qualitative evaluation of ACs. Instead of developing

ACs directly, we have chosen to develop ACTs and then instantiate an AC from the ACT. This is advantageous because an ACT is a complete assurance case for a particular product family and takes into account variations in the product family. Instantiating an ACT produces an AC for a product within that product family. This facilitates incremental assurance which is otherwise much more difficult to achieve. If we develop the ACT such that it is compliant with an existing, relevant standard, such as *ISO 26262*, then documenting compliance with that standard for a product within the product family is substantially complete, as long as the variations in the product were foreseen in the ACT. To this end, we have developed principles for systematically building an ACT from an existing standard. This is demonstrated in Chapter 4.

Relatively recently, automotive Original Equipment Manufacturers (OEMs) have focused on Over-The-Air (OTA) updates to save time and cost for customers. The original motivation for OTA updates to automotive software was that customers view a trip to the dealership to install a software patch an avoidable waste of their time. This is true even when the patch introduces a new exciting feature that they may be willing to install. An OTA can take place without the presence of the owner. Whether the update is installed automatically or needs approval before driving depends on the criticality of the update. For example, if the update is for parts of the infotainment system, perhaps it can be installed automatically. If the update is for a critical component of the vehicle, it may be necessary to have driver approval for an update. Also, OEMs hope that OTA updates can be a lot more cost-effective than paying dealerships for update installation. However, with the implementation of OTA firmware updates come new entry points for hackers to tamper with a vehicle's software. Not only do we introduce the potential for hacking, but we also remove a trained technician from the process. These trained professionals help validate that the new firmware installation is successful and ensure that no safety hazards result from the update. For example, in a real incident, a simple update to an infotainment system caused cycles of rebooting the heads-up display, accompanied by distracting bright purple flashes, resulting in a severe safety concern [2].

A vital attribute of an AC is to facilitate the identification of fallacies in the validity of any claim. There is considerable published research related to

confidence in ACs, which primarily relates to a measure of the soundness of reasoning. Evaluation of an AC should be more general than measuring confidence and should consider multiple aspects of the quality of an AC. The evaluation process typically identifies structural errors and errors in content in an AC and makes it more comprehensible to stakeholders. At least two types of experts are involved in the AC evaluation process: developers and external reviewers. Developers consist of system developers, software developers, assurance case developers, system verifiers, validators, internal reviewers, etc. External reviewers are reviewers from regulatory organizations or third party organizations, etc. Criteria systematically guide the evaluation process to discover incorrect, incomplete or inconsistent argument structures. The motivation for developing a systematic evaluation process is to make it less subjective. However, it is not possible to make a completely objective one due to its qualitative nature. A widely accepted concept is that a quantitative evaluation is not possible because it is not free from subjectivity, and in some cases, the confidence measure is an error-prone process as it depends on expert knowledge.

1.2 Research objectives

Throughout the thesis, we focus on the following research objectives:

R1. We have identified a lack of methods to develop an ACT complying with standards. Our way of dealing with this was to define principles to develop an ACT that complies with a standard such as *ISO 26262* [3].

R2. We applied our principles to *ISO 26262* (functional safety for automotive vehicles) and *SAE J3061* (cybersecurity), and used the resulting ACT's specialization in a case study to guide us to pre-emptively mitigate against potential vulnerabilities in automotive over-the-air update implementations.

R3. Evaluation of an AC should be more general than measuring confidence and should consider multiple aspects of the quality of an AC. Standard evaluation criteria play a significant role in making evaluation more systematic. Concerning this, we define effective evaluation criteria.

R4. To perform an effective evaluation using these criteria, we developed a systematic evaluation process to make it less subjective.

1.3 Challenges

This research faced the following challenges:

C1. In some cases, the standards-writing process may not follow a consistent process flow, which, in turn, obstructs the development of an ACT complying with that standard;

C2. The combined safety and security argument is still in a preliminary stage of research, and no industry best practice is known;

C3. There are no standardized requirements for OTA updates during the maintenance of an automotive vehicle;

C4. An AC of a complex system is typically very large and difficult to evaluate by a single person;

C5. There is no standardized or formal approach to represent an argument in an AC. As a result, arguments are expressed in different styles, using different patterns, and exhibit different characteristics;

C6. The guidelines provided by regulators and current standards are inadequate in their description of acceptable context, assumptions, and structure;

C7. An AC may be subject to confirmation bias [4].

1.4 Papers published on research outcomes

The following papers presented research outcomes in reputable conferences:

1. **Thomas Chowdhury**, Chung-Wei Lin, BaekGyu Kim, Mark Lawford, Shinichi Shiraishi, Alan Wassyng: "Principles for systematic development of an assurance case template from ISO 26262". In: 28^{th} International Symposium on Software Reliability Engineering Workshops (ISSREW). pp. 69-72. IEEE (2017)

2. **Thomas Chowdhury**, Eric Lesiuta, Kerianne Rikley, Chung-Wei Lin, Eunsuk Kang, BaekGyu Kim, Shinichi Shiraishi, Mark Lawford, Alan Wassyng: "Safe and Secure Automotive Over-the-Air Updates". In: 37^{th} International Conference on Computer Safety, Reliability and Security (SAFECOMP). Lecture Notes in Computer Science, Springer, pp. 172-187 (2018)

3. **Thomas Chowdhury**, Alan Wassyng, Richard F. Paige, Mark Lawford: "Criteria to Systematically evaluate (Safety) Assurance Cases". In: 30^{th} International Symposium on Software Reliability Engineering (ISSRE). pp. 380-390. IEEE (2019)

4. **Thomas Chowdhury**, Alan Wassyng, Richard F. Paige, Mark Lawford: "Systematic Evaluation of (Safety) Assurance Cases". In: 39^{th} International Conference on Computer Safety, Reliability and Security (SAFECOMP). Lecture Notes in Computer Science, Springer, pp. 18-33 (2020)

1.5 Outline

In this thesis, three research problems are resolved through rigorous analysis. In Chapter 2, preliminaries are described to provide a glimpse into the topic that is dealt with throughout the thesis. In Chapter 3, previous research related to the research problems is discussed. In Chapter 4, principles to develop an ACT complying with standards (e.g. *ISO 26262*) are developed and presented. In Chapter 5, development of a partial ACT using principles complying with both standard *ISO 26262* and guidebook *SAE J3061* are explained and an extension of the ACT including arguments related to OTA is presented. In Chapter 6, all criteria to evaluate an AC, both from the developer's perspective and the regulator's perspective, are presented. In Chapter 7, the evaluation process of an AC based on Chapter 6 is explained. In Chapter 8, one AC is considered as an example to apply the evaluation process described in Chapter 7. Two other evaluation results are illustrated in Appendix B. Chapter 9 concludes with a recap of the outcome of this research.

Chapter 2

Preliminaries

Before illustrating technical details for defining principles for an ACT and evaluation for an AC, this chapter discusses the useful entities of this research.

2.1 Assurance Case

An AC is a living document that assures a critical property of a system under consideration. An AC is a generalization of a safety case. According to Adelard [1], *"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."*

Furthermore, according to FDA Draft Guidance document [5] *"An assurance case is a formal method for demonstrating the validity of a claim by providing a convincing argument together with supporting evidence. It is a way to structure arguments to help ensure that top-level claims are credible and supported. In an assurance case, many arguments, with their supporting evidence, may be grouped under one top-level claim. For a complex case, there may be a complex web of arguments and sub-claims."*

Moreover, according to the Goal Structuring Notation (GSN) committee, the definition of assurance case is [6], *"A reasoned and compelling argument, supported by a body of evidence, that a system, service or organisation will operate as intended for a defined application in a defined environment."*

From the above definitions of an assurance case, the compelling and explicit

argument is the necessary part of an assurance case. The argument is the link between the claim and the evidence. The structure of claim, argument and evidence is the foundation to assure and certify the safety or other critical features of any system [7]. Assurance cases must be clear and straightforward, characteristics that are enforced by practitioners, evaluators, and regulators [8].

2.1.1 Assurance Case notation

There are two popular graphical notations used in assurance cases. They are: Goal Structuring Notation (GSN) [9], developed by John McDermid and others, and Claims-Argument-Evidence Notation (CAE) by Adelard [10].

Goal Structuring Notation

GSN is a graphical representation for ACs popularized by Tim Kelly [11]. The main concept of GSN is to decompose the claim into subclaims as a form of goal and subgoals. The elements of GSN are 'goal', 'strategy', 'context', 'assumption', 'justification' and 'evidence'. GSN itself is considered as an argument. The goal is decomposed into subgoals using a strategy where the evidence supports the lowest subgoals. Contexts support goals for further clarification. Assumptions are considered true without evidence. Justification, explaining why the decomposition is necessary, supports the strategy. A simple example of a GSN structure is shown in figure 2.1

Claim-Argument-Evidence notation

Adelard, a U.K. company, is extensively involved in safety assurance cases development using Claims, Arguments and Evidence (CAE) notation. It is a straightforward graphical format; An CAE structure is expressed in figure 2.2.

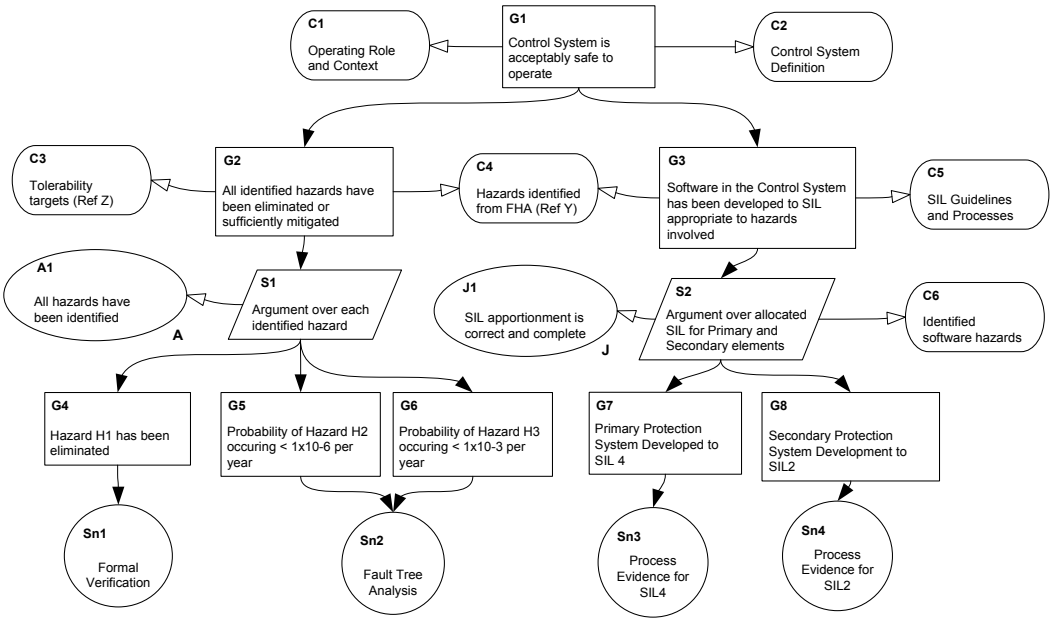


Figure 2.1: GSN example [12]

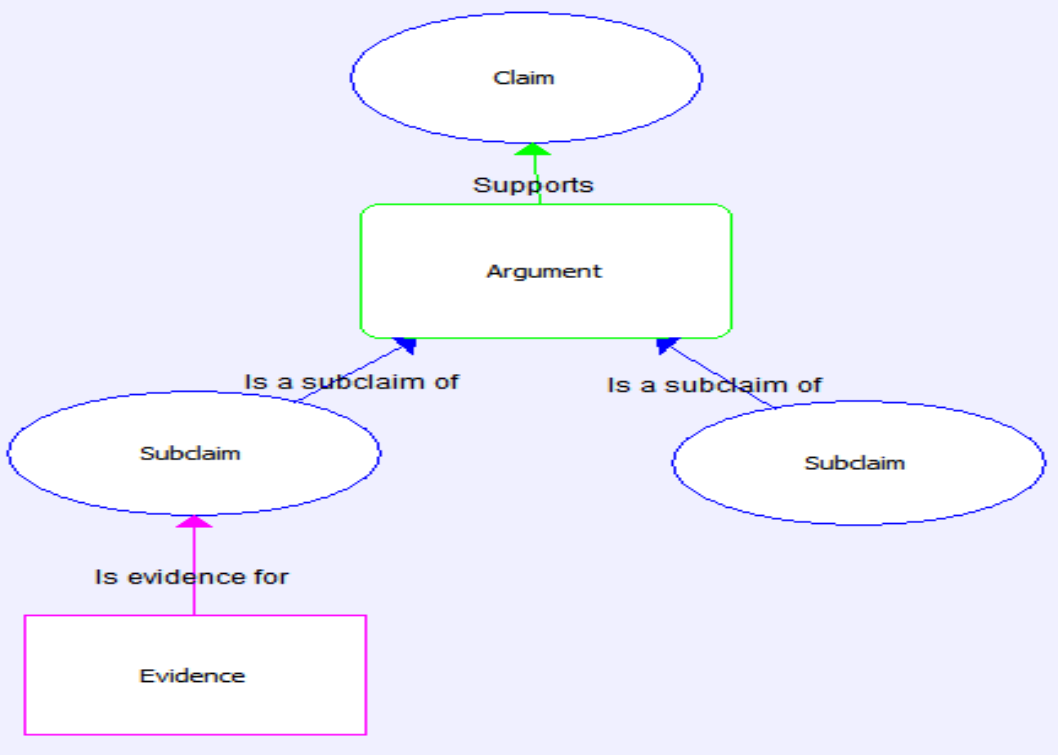


Figure 2.2: CAE Notation [8]

Similar to GSN, figure 2.2 shows each claim is supported by sub-claims or evidence through arguments. The claim may contain additional contextual information, assumption etc. For instance, an item of evidence may be test results or validation results or a list of hazards.

2.1.2 Toulmin's style for argumentation

Toulmin's [13] argumentation is used in situations that are not amenable to typical logical argumentation, for instance, in legal arguments. Kelly's [11] goal structuring notation is also based on Toulmin's notation. Figure 2.3

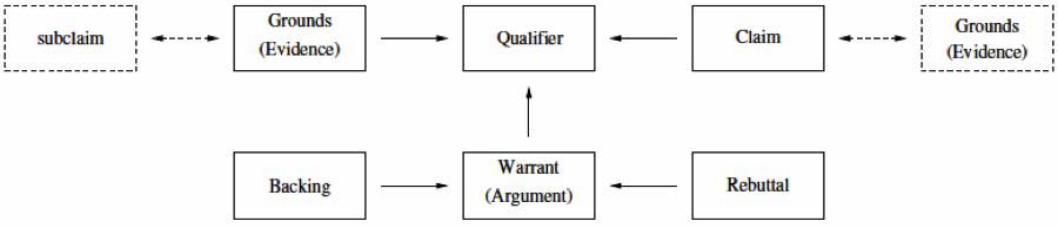


Figure 2.3: Toulmin's Style Notation for Argumentation [7]

shows Toulmin's model of argument that has six elements: claim, grounds, qualifier, warrant, backing, rebuttal.

2.2 Assurance Case Template

The main drawback of documenting an AC is that each system may have a differently structured AC, making it very difficult for a regulator to evaluate it. Recently, an assurance case template (ACT) was proposed, which represents a class of assurance cases for a specified product line [14]. An ACT is thus a complete assurance case that guides the development of a product, and the instantiation of the template for that product becomes the assurance case for that product. Figure 2.4 demonstrates an ACT documented using GSN like notation. The rationale of calling it a GSN like notation is optional paths that exist differently in GSN.

2.2.1 Benefits of Assurance Case Templates

The reasons behind the assurance case template are manifold:

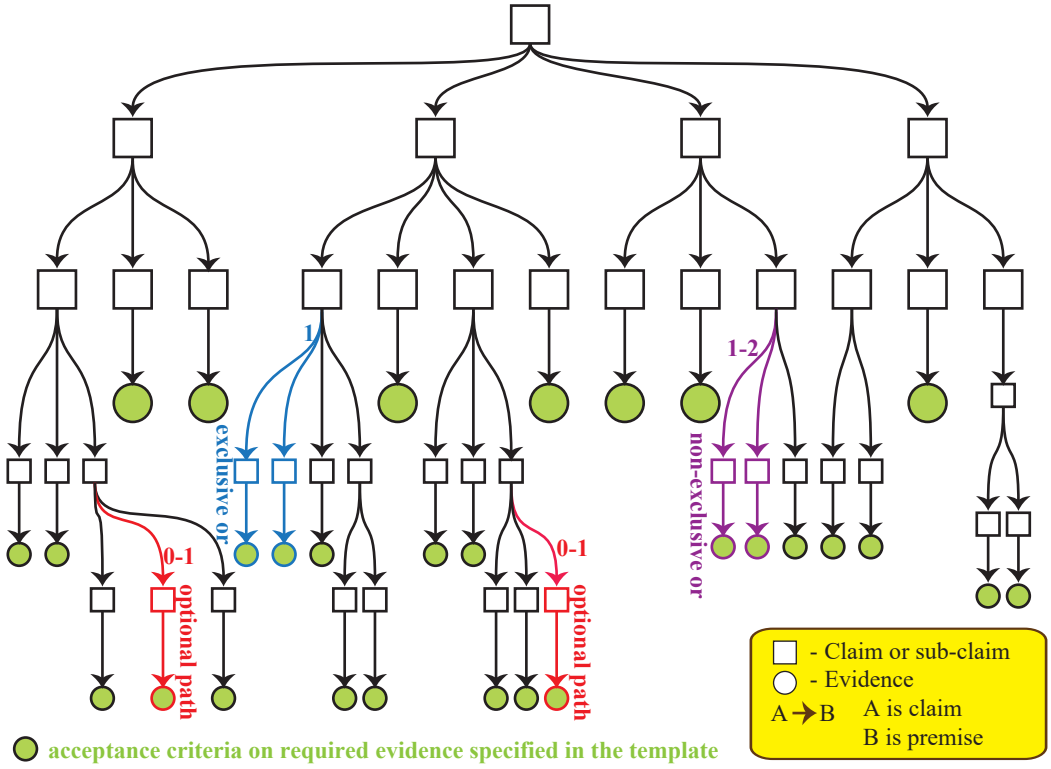


Figure 2.4: Assurance case template (modified from [15]).

1. An assurance case template can be used as an alternative to a standard.
2. It helps regulators and lay people understand the assurance case with ease because of its common structure.
3. We can design assurance case templates for a hierarchy of product lines.
4. It is postulated that an assurance case template can be documented in a way that makes it robust with respect to incremental changes.

2.2.2 Features of Assurance Case Templates

Two specific features of an ACT are optional paths and evidence with acceptance criteria [14].

Optional paths

The paths in blue, purple or red in figure 2.4 are showing all alternative argument paths that apply to products within that product line. The number beside the optional paths represents the multiplicity of the paths.

1. *Optional 0-1* (highlighted in red colour): This is a single path that may or may not be necessary for a specific product.

2. *Exclusive-Or 1* (highlighted in blue colour): One of the paths (there can be more than 2) must be instantiated for a specific product.

3. *Non-exclusive-Or 1-n* (highlighted in purple colour): One or more of the paths can be instantiated for a specific product.

Evidence with acceptance criteria

The evidence in the assurance case template supports the claim. The evidence should be unique for each terminal claim. To define evidence node in the assurance case template, the following are required:

1. the description of the evidence

2. the acceptance criteria of the evidence for a specific claim which denotes what must be true to raise the level of confidence of that claim

2.3 Feature model

A key technical innovation of software product-lines is the use of feature to distinguish product-line members. The unique combination of features [16] defines a specific member of a product line. A feature is a system property that is relevant to some stakeholders and is used to capture commonalities or discriminate among systems in a family [17]. A feature model is a hierarchically arranged set of features. A feature diagram is a graphical representation of a feature model [18]. It is a tree where primitive features are leaves and compound features are interior nodes [16].

2.3.1 Benefits of feature diagram

A feature model helps to develop the assurance case template for a specific product domain. The reasons to develop a feature diagram before ACT are:

1. Identify specific features of a product from a feature diagram to define a specific assurance case for the product domain.
2. Define optional paths of the assurance case template

2.3.2 Notations for feature diagram

Current feature modeling notations may be of three main groups:

1. Basic feature models
2. Cardinality-based feature models
3. Extended feature models

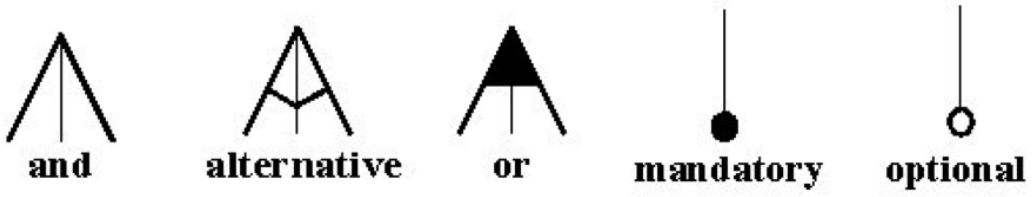


Figure 2.5: Feature Diagram Notation [16]

In this research, only the basic feature models are discussed shown in figure 2.5. Relationships between a parent feature and its child features (or subfeatures) are categorized as (figure 2.6):

1. Mandatory – child feature is required.
2. Optional – child feature is optional.

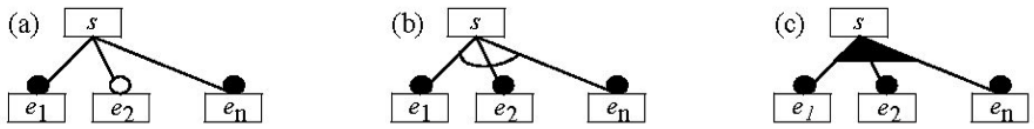


Figure 2.6: Parent-Child Relationships in Feature Diagram [16]

3. Or – at least one of the sub-features must be selected.

4. Alternative (XOR) – one of the sub-features must be selected.

2.4 Relevent standards

Throughout this research, we particularly focus on one ISO standard, *ISO 26262* and one cybersecurity guidebook, *SAE J3061*. There is also an unpublished standard *ISO/SAE 21434* [19] for cybersecurity of automotive vehicles. *ISO/SAE 21434* defines requirements for cybersecurity risk management for road vehicles throughout the development process [20]. This standard is currently under development.

2.4.1 ISO 26262

ISO 26262 [3] has become the de facto functional safety standard for electric and software components in automotive vehicles loosely based on *IEC 61508* [21]. Almost all OEMs and their suppliers voluntarily comply with *ISO 26262* to ensure functional safety. *ISO 26262* provides a strong way of risk analysis called Hazard Analysis and Risk Assessment (HARA). *ISO 26262* defines an item of interest for the functional safety, and calls it an 'item.' The item is typically a vehicle feature. To resolve risks, *ISO 26262* provides guidance by given appropriate requirements and processes. An overview of *ISO 26262* is given in figure 2.7.

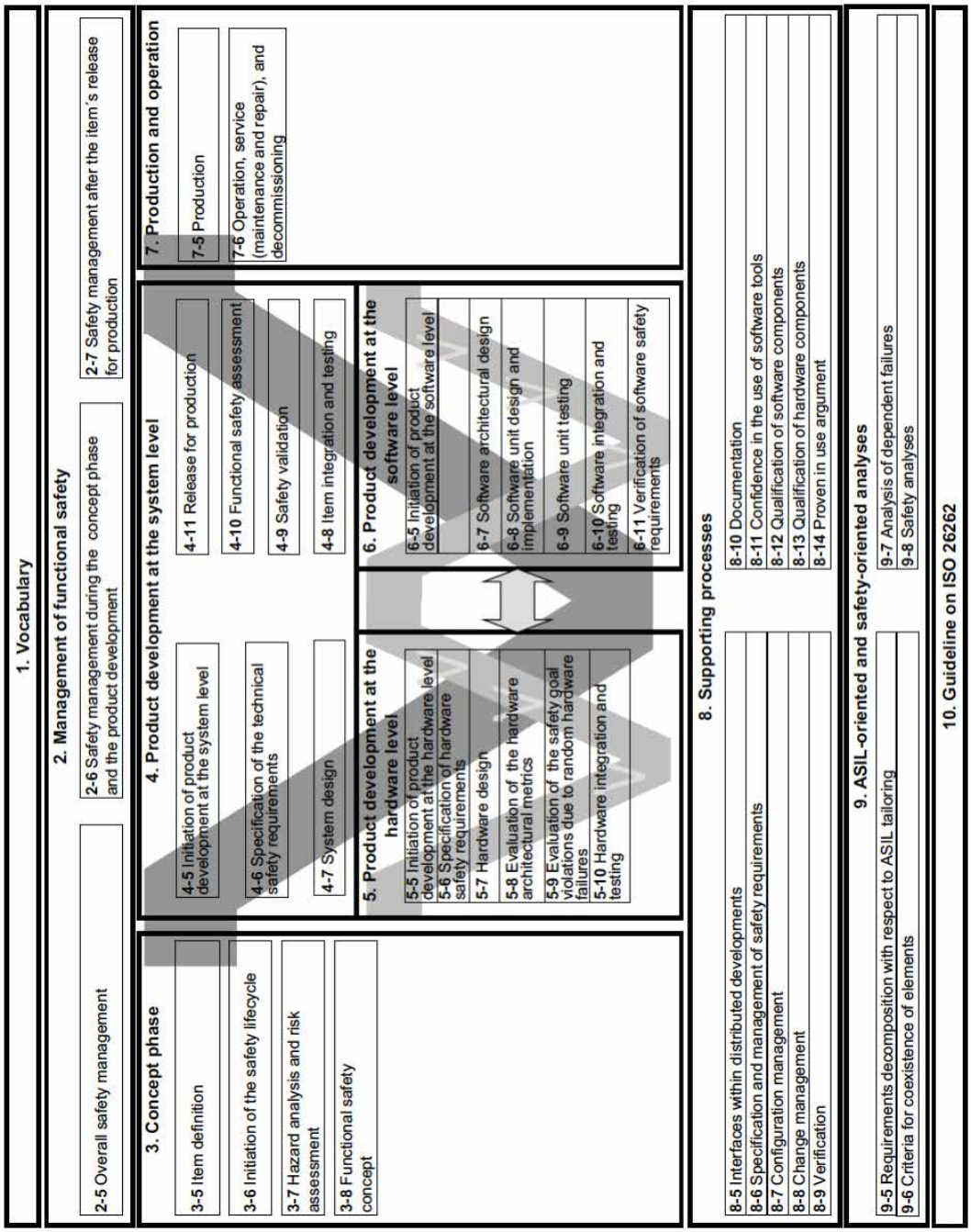


Figure 2.7: Overview of ISO 26262 [3]

2.4.2 SAE J3061

SAE J3061 is a cybersecurity guidebook [22] that provides a similar framework to *ISO 26262* for the security life-cycle for cyber-physical automotive vehicle systems. It is intended as a companion standard to *ISO 26262* and has been organized to mesh well with *ISO 26262*, but its written structure differs significantly from *ISO 26262*. The security lifecycle defined in *SAE J3061* is heavily influenced by *ISO 26262*. In some processes, *SAE J3061* mentions communication between safety and security engineers to perform combined safety and security analysis. An overview of *SAE J3061* is given in figure 2.8. Similar to *ISO 26262*, *SAE J3061* also has similar phases to synchronize

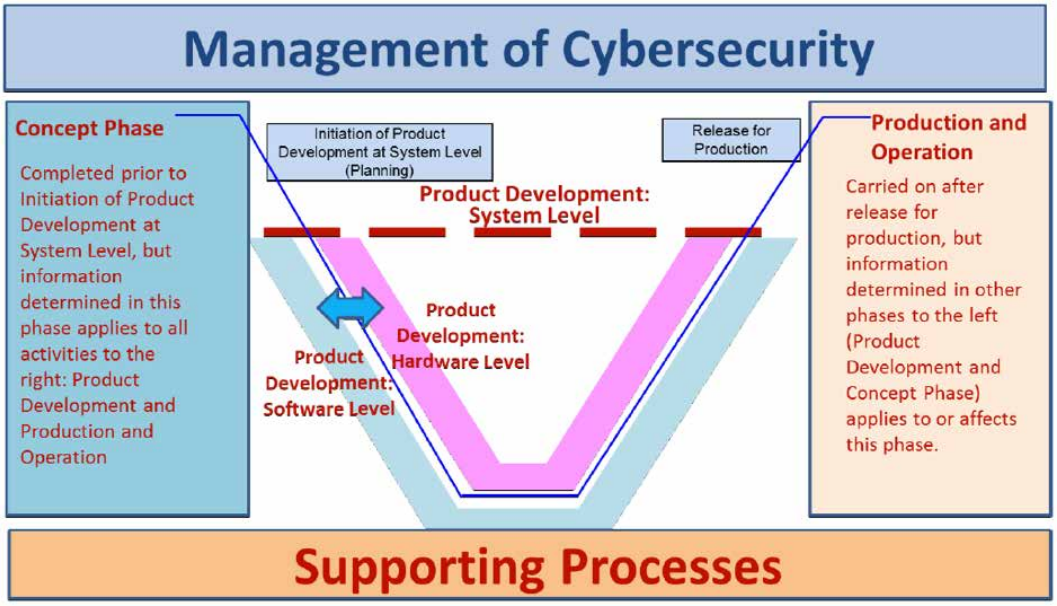


Figure 2.8: Overview of SAE J3061 [22]

appropriately with *ISO 26262*. It also has supporting processes, e.g. change, quality management.

2.5 Over-The-Air update (OTA)

Nowadays, modern vehicles rely on a software-based systems which contain more than 100 computing units with millions of lines of code. These complex

systems sometimes require updates to improve user satisfaction. Sometimes going to the dealership is not feasible to customers due to time and expense. To update software and add new functionalities during its lifetime while saving time and expenses, remote updates popularly known as "Over-The-Air" updates may take place. However, OTA update is not free from vulnerabilities as a trained technician is removed from the process to validate the update. As such, safety is a critical concern with OTA updates.

2.6 Uptane

'Uptane' [23] is the first software update framework that protects software from a comprehensive array of security attacks and is resilient to partial compromise. Uptane performs the countermeasures by adding features to the popular software update framework, 'TUF'. Uptane adds the following features to the traditional software update framework to protect a vehicle from threats:

- Additional storage: Uptane adds additional storage to save Electronic Control Units (ECUs) from attacks where incorrect data may be overwritten in ECUs.
- Broadcasting metadata: Uptane adds broadcasting metadata to prevent attacks where different ECUs use different versions of metadata.
- Vehicle version: Uptane uses a vehicle version to detect appropriate software that is updated in ECUs.
- Time server: Uptane uses time servers to limit attacks and prevent ECUs from holding back from the latest updates.

2.7 STRIDE

"STRIDE" is a threat model developed by Praerit Garg and Loren Kohnfelder at Microsoft [24]. It is an acronym of six types of security threats. They are [25]:

- **S**poofing: Spoofing attacks occur when a person's credentials are used to access a secure location without his knowledge or permission. This attack

usually targets weak authentication. For instance, a simple password or hint can help malicious attackers to get in.

- **T**ampering: Data tampering is to modify the existing data for malicious intent. For instance, changing or modifying data in a database results in wrong execution or a security breach of personal identity. This degrades confidence in security assurance.

- **R**epudiation: Repudiation means the first user denies some action that was performed by him to the second user as the second user does not have any proof to prove that fact. It is an illegal operation to make the security system weak and non-reliable. For instance, if the first user accesses the system and does modification without leaving any traces and denies having performed any action, it degrades the confidence. Non-repudiation is a way to resolve the issue by proving the evidence of malicious action.

- **I**nformation disclosure: Information disclosure attack releases information to unauthorized personnel due to malicious intent. For example, an unauthorized person gains access to confidential information that should not be disclosed to him. This is a violation of the security of that information.

- **D**enial of service: Denial of service attack denies access to authorized users due to malicious intent. This attack should be prevented to increase availability and reliability.

- **E**levation of privilege: In this attack, a user without an appropriate access privilege accesses a system for malicious intent. It decreases the reliability of the system.

2.8 CIA triad

Security attacks consistently target and exploit weaknesses of four main security properties: confidentiality, integrity, availability, and authenticity [26]. By adequately protecting these four main properties, which have been at the

root of all known attacks, it is possible to provide security assurance for the system. The first three of these properties are widely considered to be the most crucial components of information security [27], and are known as the CIA triad. CIA is currently being used to analyze the security requirements of more than one hundred use cases of the connected vehicle proposed by the ARC-IT project funded by the U.S. Department of Transportation [28].

- **Confidentiality**: Confidentiality of communicated information is put at risk by reading attacks. Data encryption is suggested to mitigate these attacks [26].
- **Integrity**: The integrity of data can be protected through the use of hashing, cyclic redundancy checks (CRC) and signatures, preferably used in combination.
- **Availability**: A comprehensive backup strategy, anomaly detection, and timeouts are recommended to mitigate different attacks [26].

2.9 Microsoft threat analysis tool

The Microsoft threat modeling tool [29] performs threat analysis using a data flow diagram of a system. Microsoft accomplishes the analysis using the following three steps in sequence:

1. **Diagramming**: The data flow diagram is the first essential element for the threat analysis. Microsoft has introduced the notion of trust boundaries to show the data flow from one privilege level to another privilege level, such as network sockets, external entities, and processes with different trust levels.
2. **Threat Analysis**: Threats are generated based on the STRIDE method for each interaction.
3. **Mitigation**: Mitigation information should be included for each threat based on priority unless it does not apply to the system.

Furthermore, the Microsoft tool has the capability of adapting a new template to create a data flow diagram of a system. In addition to this, the tool has

options for adding user-defined threats. For the automotive domain, the NCC group [30] created a customized template for the automotive domain to perform threat analysis using the Microsoft threat analysis tool.

Chapter 3

Literature Review

This chapter presents previous research on ACT development complying with standards, and on ACs related to integrated safety and security. Furthermore, we discuss relevant research on the evaluation of assurance cases.

3.1 Focus of our literature review

There is extensive published research on ACs. There is far too much to include a review of all AC research in this thesis. We have thus focused on the research papers that are aligned with our scope of work. In particular, for research in building ACs compliant with standards we focused on previous work regarding safety cases in compliance with any standard, e.g. *ISO 26262*, *DO 178C*, *BS 7799-2*. For safety and security, we focused on publications that integrated assurance of safety and security. For publications on evaluation of ACs we used criteria or keywords to assess an AC, review methods for ACs, and quantitative methods for evaluating ACs. To focus on confidence assessment, we searched for confidence assessment using uncertainty modeling, Bayesian belief networks, subjective logic, Belief networks, Dempster-Shafer theory, and expert review.

The databases we relied on most were *IEEE Xplore*, *ScienceDirect*, and *SpringerLink*.

3.2 Research related to Assurance Case compliant with standards

Here we focus on previous research on safety case development, complying with standards.

The first publication as far as we are aware, that examined the link between ACs and standards, was published in 2005 [31]. In this work the authors examined three standards and developed a framework for 'structured' ACs derived from these standards. The standards they used were: the *Common Criteria* (security), *DO-178B* (digital avionics) and *ISO 14971* (risk management for medical devices).

In [32], the authors propose a model-based approach to develop a generic safety case pattern for a future automotive safety case. However, they did not validate the proposed method for complex scenarios.

In [33], the author develops an AC from *DO 178C*. The author transforms different chapters of *DO 178C* into subclaims of an AC. Concerning this, the author classifies objectives of *DO 178C* into three categories. After several revisions of ACs, the author proposes four fundamental concepts in [34] to develop explicit ACs: transforming safety into correctness, allowing life-cycle flexibility, using confidence arguments, and explicating before evaluating.

In [35], the authors describe an approach to develop an assurance case complying with *ISO 26262*. They consider three types of arguments: standard compliance arguments, requirement-satisfaction arguments, and hazardous mitigation arguments. For software compliance, they use fitness-for-purpose arguments. The authors validate the approach by using an electronic control system of a hypothetical anti-lock braking system.

In [36], The authors propose a generic conceptual model for "chain of evidence" to argue about software safety compliant with *IEC 61508*. This model consists of several concepts and helps to understand IEC 61508, generate safety reports and perform automatic rule checking.

In [37], the authors describe industrial experience (a fuel level estimation and display system of heavy trucks manufactured by Scania) in building a safety case in compliance with *ISO 26262*. They focus on part 3 of *ISO 26262*. The authors create process-based and product-based arguments col-

lected through interviews and observations during the safety case development.

In [38], the authors describe patterns of developing a *ISO 26262* safety case with several reusable safety arguments using GSN patterns and modules. They propose a "product safety" argument as the top-level argument, which is then supported by three separate arguments: crash protection, particular risks, E/E system safety modules. However, the authors did not present their instantiated model due to confidentiality.

In [39], the authors categorize and analyze what they consider to be the main argument structures for a safety case. In this scenario, they mainly focus on product-based safety rationale within these arguments to assess functional safety.

In [40], the authors describe a template complying with *BS 7799-2*. However, the paper does not mention any technique or principles about how the requirements can be converted to claims.

3.3 Safe and secure Assurance Case Template

Various threat analysis methods are described in [41, 29, 42, 43, 44]. OTA Update specific security is discussed in [26, 23, 45]. In particular, the *Uptane Project* [23, 45] defines an open-source software security system with a flexible design, allowing it to be adapted easily to various systems. The Uptane project presents a comprehensive look at common types of attacks that an unsecured vehicle will be vulnerable to, specifically when updated remotely. The attacks described in [23] are: Read Attacks, Replay Attacks, Denial-of-Service (DoS) Attacks (including Drop Attacks, Slow Retrieval Attacks, Flood Attacks, Freeze Attacks), Rollback Attacks, Modify Attacks (including Partial Bundle Attacks, Mixed Bundle Attacks, Mix-and-match Attacks), Spoof Attacks and Control Attacks.

Long-term, we believe that the best way of integrating safety and security is to use an integrated hazard/threat analysis and risk management. Implementing safety and security requirements derived separately from independent hazard analysis and threat analysis may lead to conflicting requirements which result in new hazards and/or vulnerabilities, and also may miss hazards/threats resulting from combined security/safety concerns. There are some early at-

tempts at this in the literature. Unfortunately, this is not yet a common approach, simply because the relevant "standard" *ISO 26262* and "guidebook" *SAE J3061* deal with the two aspects separately to limit their scope during their initial development. Our work is currently based on compliance with *ISO 26262* and *SAE J3061*.

Systems-Theoretic Process Analysis (STPA) [46], developed by Nancy Leveson, is a well-regarded hazard analysis technique that is focused strictly on ensuring safety. *STPA-Sec* [47], developed by Leveson and Young, is a derivative of STPA in the security domain. STPA-Sec is an extension of STPA considering security aspects in a top-down fashion. However, in striving to integrate safety and security analysis, a separate analysis of safety and security does not seem to cover the combined effects of safety and security adequately. Another method, *STPA-SafeSec* [48], based on STPA, proposes a more unified analysis technique for safety and security. To support the unified approach, STPA-SafeSec defines the component layer diagram and extends the causal factors of security domains. This method considers the cyberattacks on integrity and availability at the component layer. The authors do not show a relationship between safety and security, and how conflicts can be resolved is not explicitly defined.

In [49], the authors propose a method called *SAFE (Systematic Analysis of Faults and Errors)*. To combine safety and security, SAFE considers a semantic framework of error "effect" that integrates an adversary model used in security analysis with fault/error categorization used in hazard analysis. Safety and security analysis are also combined in [50]; namely, STPA and NIST SP800-30 [51] are considered to derive the safety constraints and security constraints, respectively. The authors use an automatic scheme to detect conflicts and reinforcement. However, they do not define the automatic scheme precisely, which is the key mechanism in detecting conflicts. In [41], the *SAHARA (Security Aware Hazard Analysis and Risk Assessment)* method derives a measure of the security impact on the *"Automotive Safety Integrity Levels" (ASILs)*. This approach uses STRIDE (**S**poofing identity, **T**ampering with data, **R**epudiation, **I**nformation disclosure, **D**enial of service, **E**levation of privilege) to derive *"Security Levels"* to combine with the ASILs based on *ISO 26262*'s HARA (Hazard analysis and risk assessment). Amorim et al [52]

use patterns to interlink safety and security in the development process. Some of the authors of this paper were also involved in creating SAHARA, described above.

In [53] the authors describe a structured way of creating security informed safety case that shows justification of safety taking security into consideration. This paper also provides an overview of a structured assurance case concept, security-informed safety methodology and a layered approach to create a safety case. A security gateway used to control data flow between security domains in the avionics environment is used as an example.

In [54] the authors emphasize a security-informed safety approach. The paper shows a structured safety case and an impact of security on that safety case and how they can be resolved to make it safe and secure. The authors also mention some challenges to create security-informed safety cases.

In [26], the author proposes a proof of concept implementation to secure a part of the update system of an electronic control unit (ECU) in cars. The proposed system ensures different aspects, e.g. confidentiality, authenticity and integrity of a supplied update. However, the author does not consider vehicle-to-vehicle (V2V) communication.

3.4 Research related to Assurance Case evaluation

Previous research on AC development considered evaluation issues implicitly in the form of methods, processes and guidelines for the construction of ACs and explicitly in specific evaluation methods. Of note is that across all of this research, there has been a noticeable lack of defined criteria for AC evaluation.

3.4.1 Review of Assurance Cases

Kelly [55] presents what he considers to be the primary problems in reviewing an AC. Kelly describes the following problems: implicit arguments and reviewers not having sufficient knowledge of the system under consideration to comprehend the arguments that are presented. He prescribes a staged review process for an assurance case: argument comprehension, well-formedness

check, expressive sufficiency checks and argument criticism and defeat. For argument criticism and defeat, the author considers both deductive and inductive arguments. He claims that supporting inductive reasoning necessitates sufficiency that depends on coverage, dependency, definition, directness, relevance, robustness. Moreover, he postulates factors for evidence evaluation: buggy-ness, level of review, experience and competency, tool qualification and assurance. To review, the author proposes two forms of argument defeat: rebuttal and undercutting. Kelly, in a book chapter [56], points out the importance of identifying weaknesses of an assurance case early to make a valid and sound argument. Thus, he emphasizes reviewing an AC during the initial stage of its development.

In [57] the authors introduce a dialectical model SARM to review arguments. To facilitate reviewing, the model satisfies six requirements: a) support persuasive dialogues, b) support information-seeking dialogues, c) equality between participants, d) sufficient room for opinion expression, e) prevent fallacious argument, and f) high usability, a low cognitive load on the user. The process in the model goes through three distinct phases: initiation, review and revision. The revised version is reviewed by the external reviewer until reviewers accept or reject the argument, or the proposer withdraws the argument. The number of iterations depends on mutual agreement between the proposer and external reviewer.

A system theory-based assurance case review is proposed in [58]. The approach uses a systemigram to show a configuration diagram consisting of artifact models, quality attributes and risk definitions. The review process generates claims, evidence and mitigations from GSN. Claims are divided into two groups: attribute claims and measure claims. Attribute claims are higher-level claims defining system constituency, whereas measure claims are the lowest claims, and are supported by evidence. The review process follows the steps: a) context understanding; b) problem identification; c) cause analysis; and d) revision. Thirteen review rules guide the review process. One drawback is that a simplified assurance case was used to validate the rules. Furthermore, transformation rules for converting an assurance case to a systemigram should be unambiguous and adequately defined, but systemigrams are notoriously subjective.

A preliminary work on safety case review using Verification studio, an industrial tool for system artefact quality analysis, is defined in [59]. The authors include ASCE (Assurance and Safety Case Environment) within Verification Studio to assess safety case quality. The quality analysis is still in the primitive stage and does not take essential aspects into account. The paper highlights requirements for a safety case assessment. Essential requirements are a single framework to collect quality from different domains; quality metrics for assessing a safety case; a quality check that combines syntax, structural and semantic checks; effective ways to review textual safety cases; an effective way to check quality of artefacts of a system that impacts safety case quality; and a quality assessment of safety cases along with safety case development. They use RSHP language, a basis for artefact representation in Verification studio. They utilize OSLC-KM technology to integrate ASCE with Verification studio which converts an ASCE generated safety case into an RSHP formatted artefact in Verification studio. They used default metrics used by Verification studio to assess the quality of a safety case after conversion to RSHP format.

In [60] the author extends GSN to include attributes to quantify architectures based on quality claims such as safety and security. The approach provides a technique of quantitative evaluation of arguments for assuring safety and security architectures.

In [61] the authors develop a method to identify argument fallacies using predicate logic. Instead of counting different fallacies, the authors focus on argument fallacy using DiaSAR with the help of an SARM model. Thus, a complete evaluation does not take place in this research.

In [62], the author prescribes some criteria to evaluate an assurance case. They are quality, correct symbols, correct relationships, and correct argument. The fourth criterion should be used both by the author of an assurance case and an external reviewer.

In [63], the researchers describe an evaluation of an assurance case. For an effective evaluation, the regulator may take a slice or full assurance case. Moreover, the reviewer may compare the assurance case with the model case they have, or they can develop an approval case.

In [64], the authors proposed a safety case assessment process consists of five steps: preparation, logic and structure validation, quality evaluation,

record and feedback and revision. The first four steps are performed by a safety assessor and based on their recommendations; a safety case developer performs the revision step.

In [65], the authors mention some challenges of developing and reviewing safety cases: size and complexity, readability, a variety of evidence, challenges with context and assumption, challenges with arguments, misleading notation, confirmation bias, challenges of the process and product-based approach. Furthermore, the authors argue that regulators should thoroughly analyze the argument for completeness and soundness.

In [66] the group develops some checklists to assess the safety case based on safety case presentation, argument structure, evidence, caveats (e.g. assumption, all outstanding issues, etc.)

In [7] Rushby describes a two-part process to assess arguments of assurance cases. One part uses epistemic methods to assess the credibility of evidential steps. Another part uses deductive logic to assess the truth of reasoning steps.

In [67] the researchers mention some factors responsible for losing assurance case persuasiveness:

1. Incompleteness including incomplete argument, incomplete claim, incomplete evidence, interdependent argument/evidence.

2. Fallacious arguments, including indirect effect and circular reasoning.

Reference [68] presents an assessment process for a safety case evaluation. The author uses checks based on guidewords and proposes brief solutions on how to fix problems related to those guidewords.

3.4.2 Research related to regulatory guidance of evaluation

Regulatory organizations also provide guiding principles to evaluate an assurance case. The motivation is to increase their reviewers' competency as well as developers' competency. Good guidance helps a developer create a valid and sound assurance case.

The FDA provides some recommendations based on their experience in reviewing safety cases for infusion pumps [5]:

1. separate argument structure showing the completeness of the hazard analysis, including techniques, procedures, results etc.;
2. specific argument structures for particular domains such as software, human factors and reliability;
3. arguments should include justification for the selection of acceptability criteria for safety control;
4. a traceability analysis is useful to trace all identified hazardous situations;
5. if a safety case is documented using user-defined notation, an executive summary should be provided to assist the FDA in navigating safety cases;
6. the FDA uses post-market data to verify the safety argument;
7. the FDA also provides feedback through the pre-submission process.

In the U.K., the Offshore Installation Regulations 2005 defines the regulation of submitting safety cases for installation related to oil and gas operations in offshore waters [69]. There is a corresponding document for internal waters. The Health and Safety Executive (HSE) assesses safety cases for validity. They define 36 principles categorized into factual information, management of health and safety, major accident hazard identification, major accident risk evaluation, major accident risk management, emergency response, rescue and recovery, life cycle requirements, combined operations and decommissioning and dismantlement to assess safety cases.

Reference [70] provides guidance on purpose, qualities, structure and content of a safety case. To review a safety case, inspectors should use their experience. Inspectors should analyze evidence and discuss with a licensee how the lessons can be implemented to improve safety. Inspectors should check whether or not the causes of problems are addressed.

Although not specifically related to assurance cases, in [71], the Office of Nuclear Regulatory Research provides guidelines to evaluate an applicant's hazard analysis and corresponding acceptance criteria. It also provides technical knowledge to the US Nuclear Regulatory Commission (NRC's) licensing staff.

3.4.3 Research related to confidence

There is extensive research on the confidence assessment of an assurance case. Publications that seemed most relevant to our work are listed below.

Reference [72] presents a confidence measure technique 'INCIDENCE' that considers both design time and run time evidence . The authors also measure the uncertainty measure of technical debt (requirement debt) for software systems. The proposed method has two disadvantages: their method is not backed by a robust empirical validation that can prove their trustworthiness to deploy the system and they did not consider all the scenarios of requirement debt.

In [73] the authors propose a quantitative approach to assess confidence in assurance cases. The authors follow different models: Toulmin model, Hitchcock's evaluation criteria and Bayesian Belief Network to quantify confidence.

In [74] the authors propose a subjective logic-based approach to assess confidence in an assurance case. The authors define four basic argument types and confidence propagation rules for them. They calculate confidence in a bottom-up fashion.

Reference [75] presents an evidential reasoning approach to assess confidence in safety evidence, which propagates to a top-level claim of a safety case. Concerning this, the authors define a confidence argument pattern to assess confidence in safety evidence. Then they use the evidential reasoning approach to a) explicitly details reason of confidence in safety evidence, b) identifies uncertainties and c) describes confidence at each level of a safety case visually and quantitatively.

In [76] the authors propose an approach to evaluate a defeasible argument and describe how they relate to the confidence modeling for the safety case. The authors propose to treat rebuttals formally, and dialectical interpretation provides a sound foundation for evaluation.

Reference [77] presents a structured approach to measure the sufficiency and insufficiency of an argument node of a safety case. Initially, the degree of belief and degree of belief of evidence is measured. Then an aggregation rule is applied to measure the sufficiency and insufficiency of conclusions.

In [78] the authors present a framework using Dempster-Shafer theory to assess argumentation based on experts' opinion and confidence in the lowest level claim of the arguments.

Reference [79] presents a confidence calculation framework using Dempster-Shafer theory and vector space model to assess confidence for the leaf claims of an assurance case. The framework initially takes an acceptable assurance case for an extensive system and generates assurance cases similar to the input assurance case except for leaf claims, which are supported by real evidence. The confidence of leaf claims is adapted by checking similarity in its supporting evidence node.

In [80] the authors present an extension of Baconian probability with the use of the Beta distribution, opinion triangle to calculate confidence. The authors also derive uncertainty associated with evidence. They also incorporate a weighting scheme to calculate confidence realistically. However, the authors do not mention how to deal with low-threshold confidence value.

Reference [81] presents an approach combining Beta distribution with opinion triangle and subjective logic to assess confidence in an assurance case. The analysts provide an opinion of evidence in terms of the degree of belief, disbelief or uncertainty. Then these values are calculated using Beta distribution and subjective logic.

In [82] the authors propose an approach to compute the uncertainty in safety claims by building Bayesian Belief Networks analogous to the structure of a safety argument. In the Bayesian network, leaf node models, each source of uncertainty and intermediate nodes combine confidence of leaf nodes. The authors also mention if an only subjective judgement is available, for quantitative information, the Goal-Question-Metric (GQM) method [83] may be a suitable candidate.

Chapter 4

Principles for Assurance Case Templates

This chapter presents principles we developed that help to create an assurance case template from existing safety and security standards. This chapter extends work published in [84].

4.1 Principles for developing an Assurance Case Template

Coping with the sheer amount of work we need to perform daily, there is a growing interest in automating tasks. Also, automation often helps to reduce human error that can negatively bias and otherwise impact the assurance. At this stage of our research, full automation is a dream. However, the principles we have developed allow us to start developing structured, manual methods for building assurance case templates from safety and security standards and start planning tools that will help us semi-automate the production of these templates in the future.

4.2 Methodology

Our first attempt to develop principles to build an assurance case template complying with a standard, targeted *ISO 26262* [3]. The objective was to

convert requirements in *ISO 26262* into claims and acceptance criteria for evidence.

4.2.1 Principles for constructing a template complying with *ISO 26262*

We developed/discovered 10 principles to develop an Assurance Case Template that complies with a safety standard e.g. *ISO 26262*. The benefit of these principles is to motivate the semi-automation of the development of ACTs. The principles with examples are now described in detail.

- **Principle 1: Modeling a standard**: To develop an assurance case template complying with *ISO 26262*, it is mandatory to understand the standard properly. Reading the standard (many times) does not give a complete idea about the standard. Thus, in our example, we used three models to understand the standard more completely, especially dependencies in the standard. The first of these is a diagrammatic flow diagram (figure 4.1) of the interrelation among processes and work products in *ISO 26262*. The second model is a conceptual model of *ISO 26262*. This model captures definitions and dependencies in the standard.This conceptual model will help to automate the aspects of the assurance case template (such as checks on completeness, etc) as well as being useful in understanding the standard. Figure 4.2 represents an extract from a conceptual model [85] of *ISO 26262*.

[1]Diagram provided by Neeraj Kumar Singh.

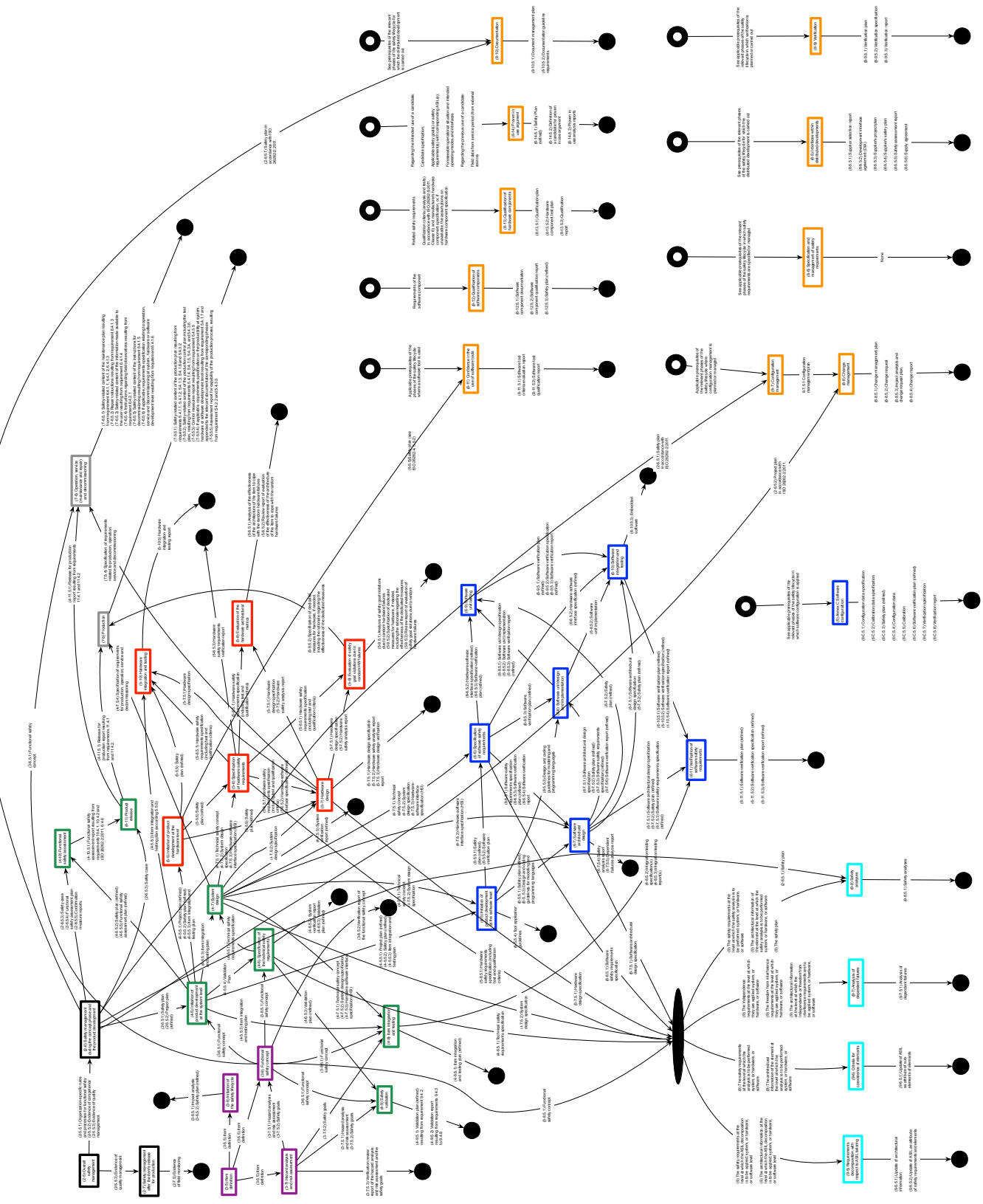


Figure 4.1: Major Process Flow & Work Products in *ISO 26262*[1]

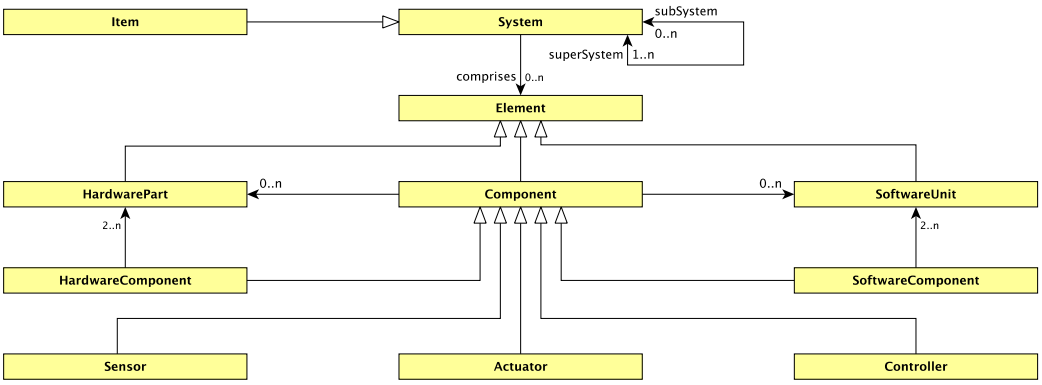


Figure 4.2: Extract from a conceptual model [85] of *ISO 26262*

We also used a list of "consolidated work products". "Consolidated" meaning that some work products are generated in a process and then refined in later processes. The processes are then grouped accordingly in this list. Since work products in *ISO 26262* will typically be used as evidence in ACs, this list also helps to understand the standard by presenting a view that is focussed on the work products within the overall process flow. Figure 4.3 shows an extract from a list of the consolidated work products.

CONSOLIDATED WORK PRODUCTS – ISO 26262

2-5.5.1 Organization-specific rules and processes for functional safety, resulting from 5.4.2 and 5.4.5.

2-5.5.2 Evidence of competence, resulting from 5.4.3.

2- 5.5.3 Evidence of quality management, resulting from 5.4.4.

2-6.5.3 Safety case, resulting from 6.4.6.

2-6.5.5 Confirmation measure reports, resulting from 6.4.7 to 6.4.9.

2-7.5.1 Evidence of field monitoring, resulting from 7.4.2.4.

3-5.5 Item definition resulting from the requirements of 5.4.

3-6.5.1 Impact analysis resulting from the requirements of 6.4.2.1 to 6.4.2.4.

3-7.5.1 Hazard analysis and risk assessment resulting from the requirements of 7.4.1.1 to 7.4.4.2

3-7.5.2 Safety goals resulting from the requirements of 7.4.4.3 to 7.4.4.6

3-7.5.3 Verification review report of the hazard analysis and risk assessment and the safety goals resulting from the requirement of 7.4.5.

3-8.5.1 Functional safety concept resulting from the requirements of 8.4.1 to 8.4.4.

3-8.5.2 Verification report of the functional safety concept resulting from the requirements of 8.4.5.

2-6.5.2 Project plan (refined), resulting from 6.4.3.4. (Original is "external"?)
4-5.5.1 Project plan (refined) resulting from requirement 5.4.4.

2-6.5.4 Functional safety assessment plan, resulting from 6.4.9.
4-5.5.5 Functional safety assessment plan (refined) resulting from requirement 5.4.3.

4-6.5.1 Technical safety requirements specification resulting from requirements 6.4.1 to 6.4.5.

4-7.5.1 Technical safety concept resulting from requirements 7.4.1 and 7.4.5.

4-7.5.2 System design specification resulting from requirements 7.4.1 to 7.4.5.

4-7.5.4 Specification of requirements for production, operation, service and decommissioning resulting from requirements 7.4.7.

4-6.5.2 System verification report resulting from requirement 6.4.6.
4-7.5.5 System verification report (refined) resulting from requirement 7.4.8.

4-7.5.6 Safety analysis reports resulting from requirement 7.4.3.

4-5.5.3 Item integration and testing plan resulting from requirement 5.4.1.
4-8.5.1 Item integration and testing plan (refined) resulting from requirement 8.4.1.

4-8.5.2 Integration testing specification(s) resulting from requirements 8.4.1.

Figure 4.3: Extract from the list of the consolidated work products of *ISO 26262*

- **Principle 2: Modeling system variability**: This principle is also reasonably evident in that we cannot assure safety and dependability of a

system without understanding it completely. However, aspects of the system are crucially important to building an effective assurance case template that we may sometimes not explore adequately. In particular, the assurance case template is designed for assuring products within a product family. It is thus essential that we document variations in the product family. An assurance case template should have all the arguments related to different products' different options in the same product family.

A feature model attempts to show all the optional features of a particular product family. For example, to create an assurance case template for an Adaptive Cruise Control (ACC) we need to consider all variations in an ACC family. As a start, we need to develop and document a Feature Diagram for the product family. Figure 4.4 shows an example of a feature model for an ACC family. This example feature diagram shows mandatory, optional, and alternative features. ACTs include this kind of variability. Optional (0-1), Exclusive-Or (1) and Non-exclusive-Or (1-n) arguments are thus possible. Our example feature diagram for ACC has three components: *Input*, *Controller* and *Output*. *Input* and *Controller* both have Non-exclusive-Or constructs for their components. In other words, they can each have either one of the identified components, or both of them. *Output* has an 'alternative' construct (Exclusive-Or) and so can have one of the other identified component.

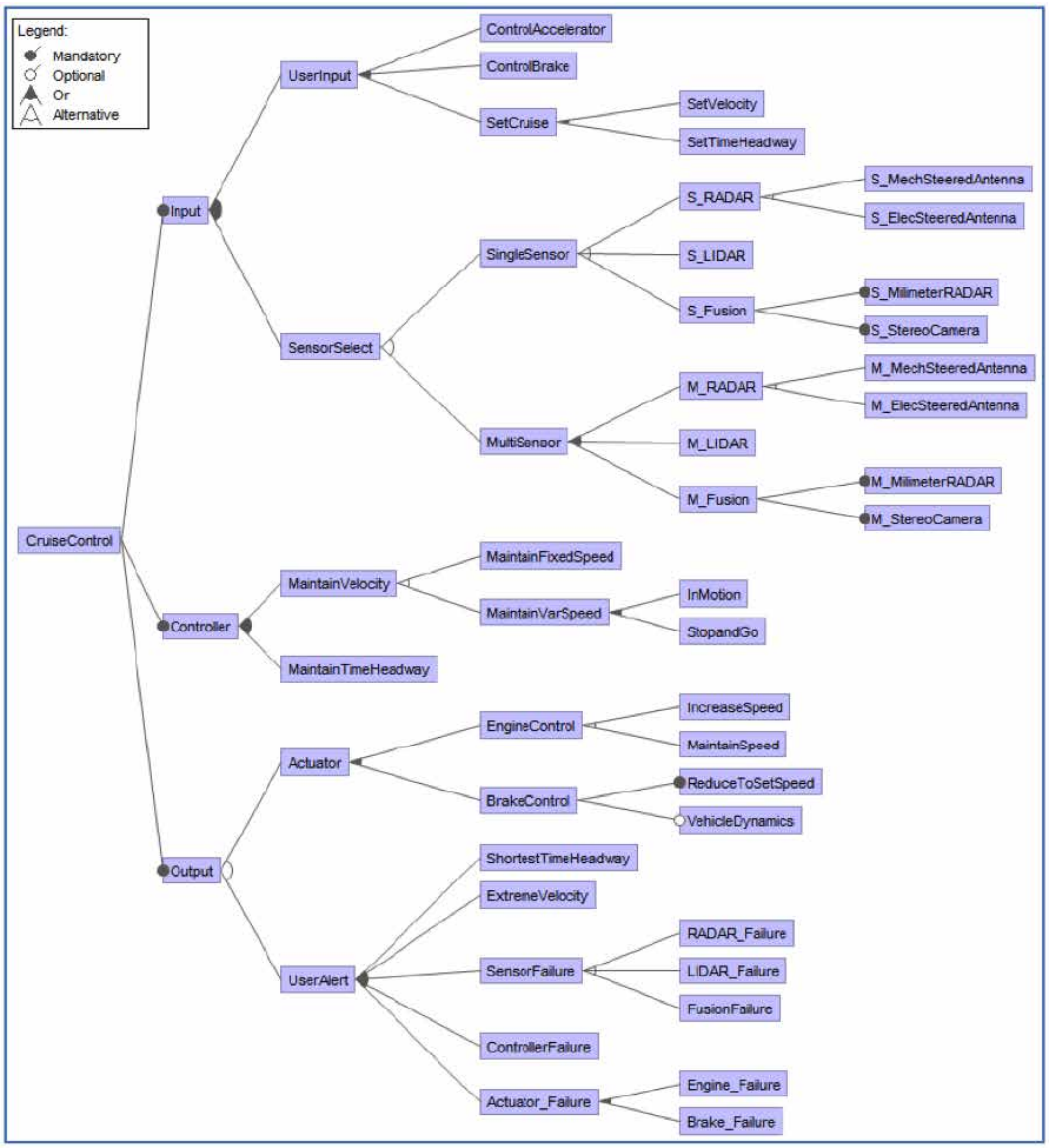


Figure 4.4: An example feature diagram of Adaptive Cruise Control

- **Principle 3: Flip-it**: There is a widespread assumption that a good process will lead to a good product, but it is often not true. *ISO 26262* is mainly a process-oriented standard, but it also deals with product-related requirements. An assurance case expresses the argument using claims and sub-claims supported by evidence. The sequence of steps in a standard translates into an argument branch consisting of claims, subclaims in an ACT. For instance, in part 3 of *ISO 26262*, we find

the clauses shown in Figure 4.5. Note that $< p - s >$ indicates 'Part p, Section s', in ISO 26262.

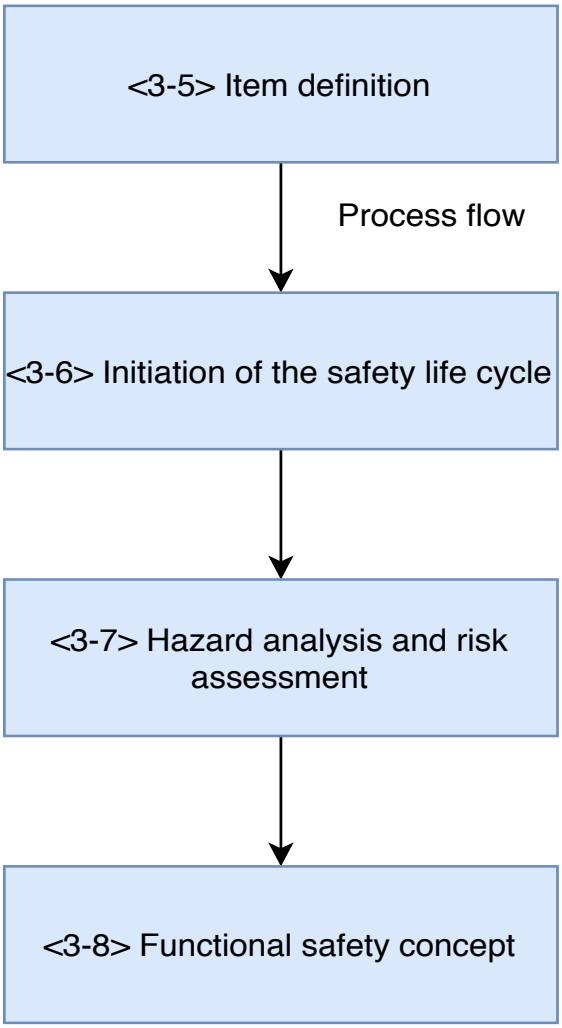


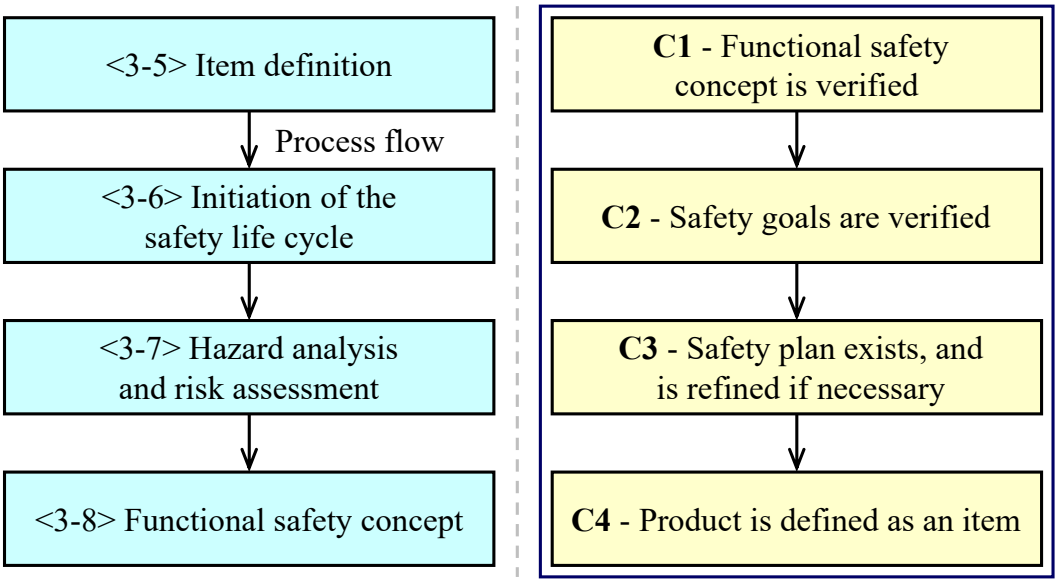
Figure 4.5: Sequence of Process Clauses in *ISO 26262* Part 3.

Figure 4.5 shows a process flow that shows sequential processes in part 3. Each of these processes can be transformed into a necessary claim in the associated argument fragment. The following shows a transformation of clauses into claims or subclaims:

- Item definition → "Product is defined as an item", in compliance with <3-5>.

- Initiation of the safety life cycle → "Safety plan exists, and is refined if necessary" (item is not new), in compliance with <3-6>.
- Hazard analysis & risk assessment → "Safety goals are verified", in compliance with HARA (hazard and risk analysis) <3-7>.
- Functional safety concept → "Functional safety concept is verified", in compliance with <3-8>.

We observe that the last process claim "Functional safety concept is verified" depends on the previous process claim "safety goals are verified" with appropriate reasoning, and process claim "safety goals are verified" relies on the process claim "safety plan exists, and is refined if necessary". So, we flip the order of the claims with respect to the process steps. Figure 4.6 shows how the process claims are flipped with respect to process order.



(a) Process steps in *ISO 26262* (b) Related claims

Figure 4.6: Illustration of the Flip-It principle. (modified from [84])

It is important here to realize that in GSN the arrows were designed to show decomposition of claims (goals). We use GSN notation in our ACTs and so the arrows go downward from C1 to C2 to C3 to C4. However, in terms of reasoning within the argument, what we are saying is that if we

want to show that the *Safety plan exists, and is refined if necessary* then we must first show that the *Product is defined as an item* is demonstrated to be true. Similarly, to show that the *Safety goals are verified* we must first show that the *Safety plan exists, and is refined if necessary* is true. And finally to show that the *Functional safety concept is verified* we must demonstrate that the *Safety goals are verified* is true. Of course, there may be additional claims and evidence required to demonstrate the truth of these claims, and they must be added to the ACT.

- **Principle 4: Conjunctive**: In our previous principle, "Flip-it", we show that a top-level process claim is supported only by process sub-claims. However, not all arguments are quite that simple. For example, three parameters (estimates of severity, the probability of exposure and controllability of hazardous event) determine the Automotive Safety Integrity Level (ASIL) associated with a hazard. All three of these estimates are required, but they do not have to be obtained in any specific order. What is important is that we need all three to determine the ASIL. The clauses in *ISO 26262* for determining the ASIL can be represented by figure 4.7. Similar to figure 4.5, <p-s> indicates "Part p, Section clause s", in *ISO 26262* in this diagram.

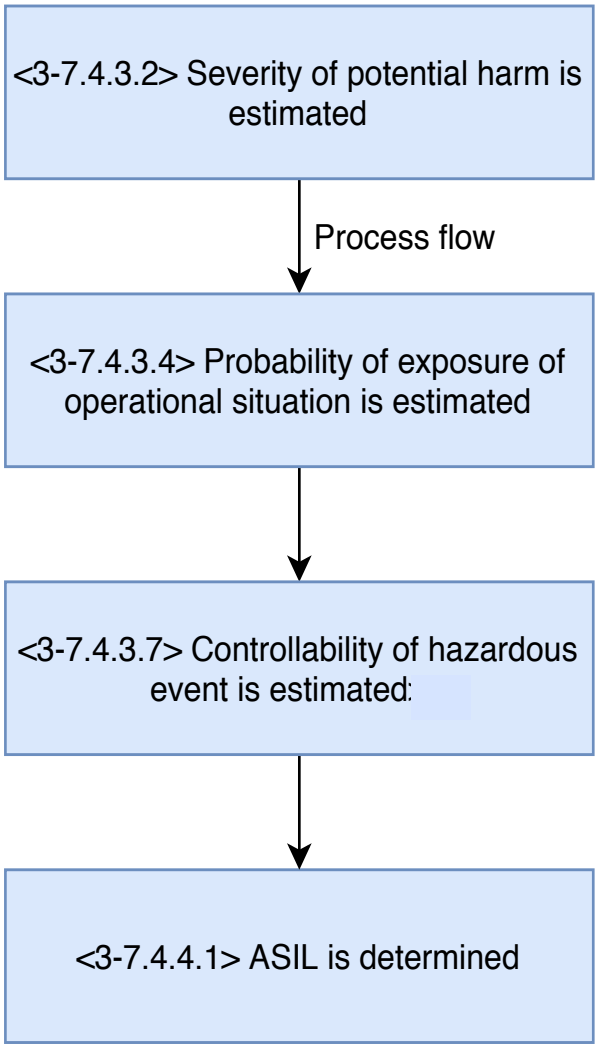


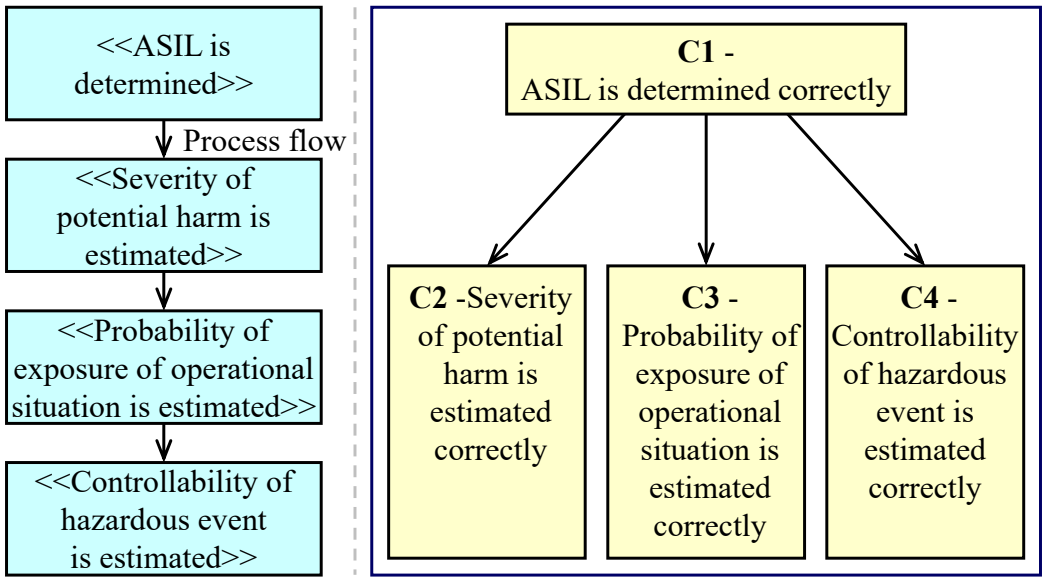
Figure 4.7: Process Sequence in *ISO 26262* Part 3 for determining the ASIL.

If we applied the 'flip-it' principle, the claims transformed from clauses would be in the reverse order. However, analysis of these clauses reveals that the flip-it principle is not applicable in this case. Instead, we have the conjunction of these three claims that will support the top claim of "ASIL is determined correctly". Figure 4.8 shows clause 7.4.4.1 from part 3 of *ISO 26262* that describes how ASIL can be determined. Figure 4.9 shows the combination of the three process claims supporting the top-level claim.

7.4.4 Determination of ASIL and safety goals Principle 4

7.4.4.1 An ASIL shall be determined for each hazardous event using the parameters "severity", "probability of exposure" and "controllability" in accordance with Table 4.

Figure 4.8: Part 3, Clause 7.4.4.1 of *ISO 26262* [3]



(a) Process steps in *ISO 26262* (b) Related claims

Figure 4.9: Illustration of the Conjunctive principle. (modified from [84])

Another example of the principle is illustrated in figure 4.11. According to *ISO 26262* part 3 and clause 5.4.1, functional and non functional requirements of the item as well as the dependencies between the item and its environment have six components. Figure 4.10 shows six components.

5.4.1 The functional and non-functional requirements of the item as well as the dependencies between the item and its environment shall be made available.

Principle 4

This information includes:

a) the functional concept, describing the purpose and functionality, including the operating modes and states of the item;

b) the operational and environmental constraints;

c) legal requirements (especially laws and regulations), national and international standards;

d) behaviour achieved by similar functions, items or elements, if any;

e) assumptions on behaviour expected from the item; and

f) potential consequences of behaviour shortfalls including known failure modes and hazards.

Figure 4.10: Part 3, Clause 5.4.1 of *ISO 26262* [3]

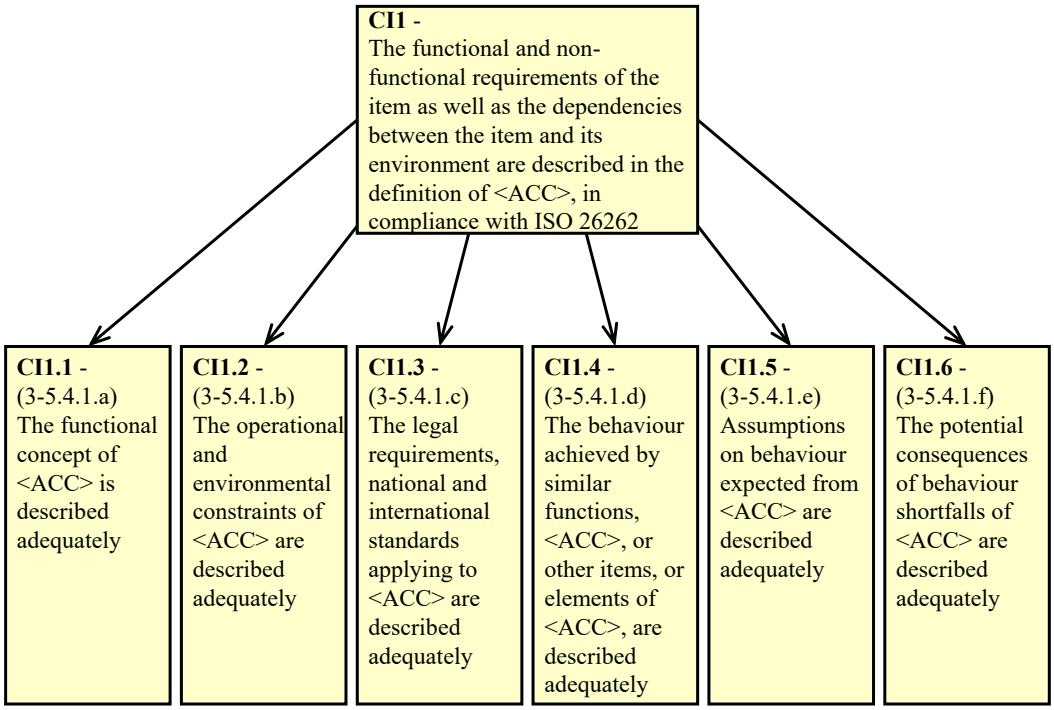


Figure 4.11: Example of developing claims from clause 5.4.1 (*ISO 26262*, part 3) using principle 4. (modified from [84])

Thus, the definition of an item related claim is supported by six sub-

claims (figure 4.11). It is noted that the argument branch consisting of a top-level claim with supporting sub-claims created using the conjunctive principle will work as a unit when using the "Flip-it" principle.

- **Principle 5: Optional pattern**: Sometimes the standard does not have a specific guideline of a process involved in developing the product. A manufacturer may develop a product or outsource or buy off-the-shelf. In this scenario, an assurance case will potentially have an alternative argument branch for different product development approaches. Part 8 of *ISO 26262* describes the safety compliance of the product developed by a third-party, namely a supplier. Figure 4.12 shows an example argument branch related to a supplier of a specific item or element. This is an optional argument in an ACT because an OEM does not need a supplier for all items or elements and the assurance depends on these details related to various options associated with parts and services from third-party suppliers.

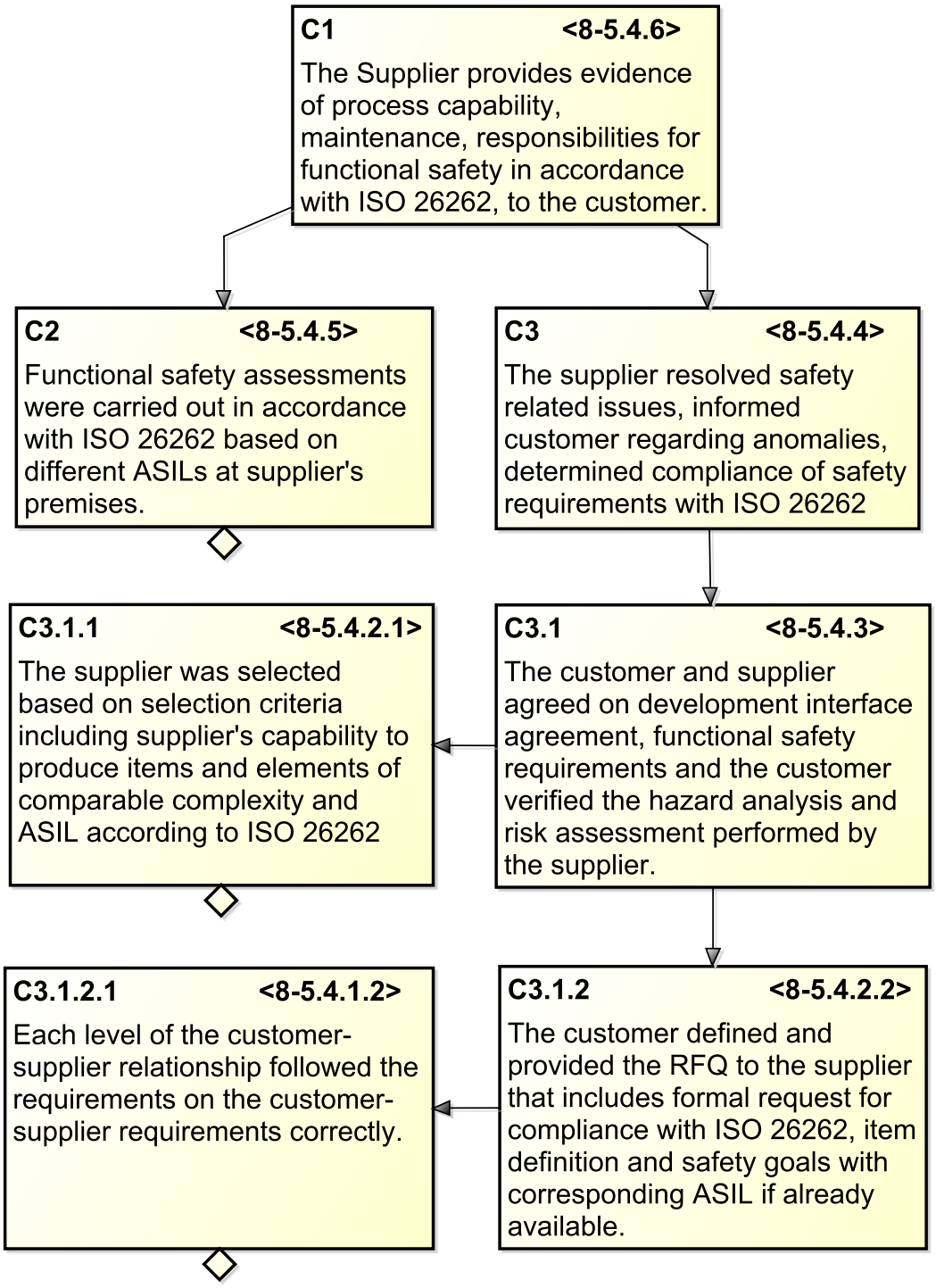


Figure 4.12: Example of optional argument path.

Another example is that verification of the implementation complying

with its requirements can be performed by either/both mathematical analysis and testing. So, an assurance case template will potentially have two branches of argument; one will be optional because testing of verification is mandatory, and the mathematical analysis is a plus. Figure 4.13 shows the optional pattern of the argument.

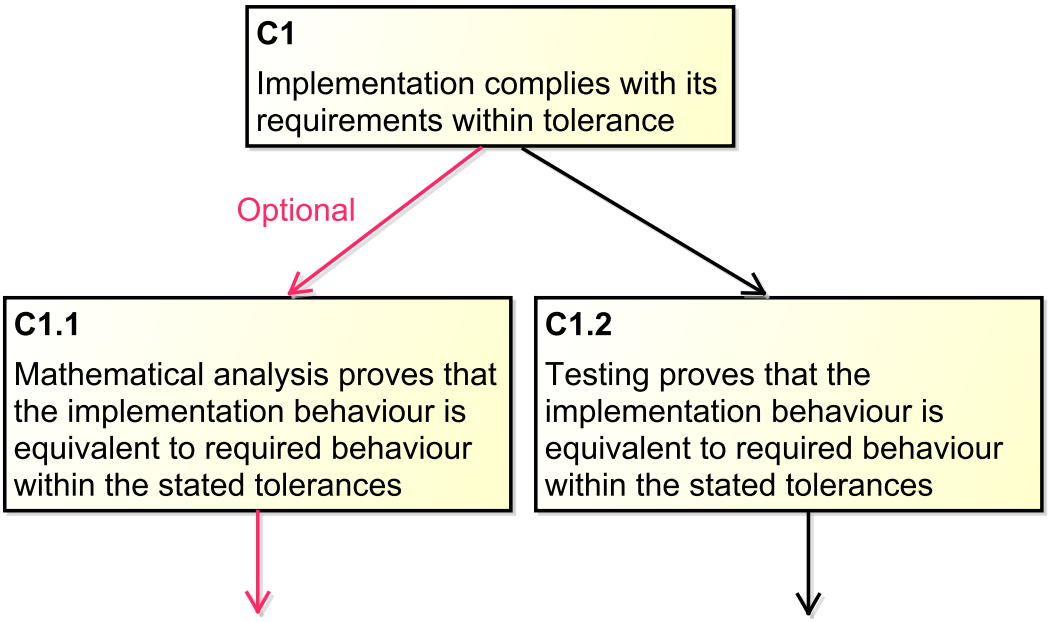


Figure 4.13: Example of optional argument path. (modified from [84])

Similarly, since we are developing a template for a product family, different features in a family, and even different sensors, for example, will require different claims and evidence.

- **Principle 6: Evidence specification**: *ISO 26262* is better than many other standards in that it specifies desired attributes and characteristics for various artefacts. For example, Part 8, clause 6, specifies attributes and characteristics for safety requirements. Evidence plays a vital role in the assurance case argument. It grounds the support for terminal claims using tangible, verifiable artefacts. It is vital that the evidence really does adequately support the claim. To this end, ACTs specify acceptance criteria in evidence nodes so that the instantiation of the ACT can be check to see that the actual evidence satisfies the predetermined

acceptance criteria. The desired attributes and characteristics in the standard can be used as acceptance criteria in the associated evidence nodes. An example from *ISO 26262* is shown in figure 4.17.

6.5 Work products

6.5.1 **Technical safety requirements specification** resulting from requirements 6.4.1 to 6.4.5. Principle 6

Figure 4.14: Part 4, Clause 6.5.1 of *ISO 26262* [3]

6.4.2.4 Safety requirements shall have the following characteristics: Principle 6

a) unambiguous and comprehensible,

b) atomic,

c) internally consistent,

d) feasible, and

e) verifiable.

Figure 4.15: Part 8, Clause 6.2.4 of *ISO 26262* [3]

6.4.2.5 Safety requirements shall have the following attributes: Principle 6

a) a unique identification remaining unchanged throughout the safety lifecycle,

b) a status, and

c) an ASIL.

Figure 4.16: Part 8, Clause 6.4.5 of *ISO 26262* [3]

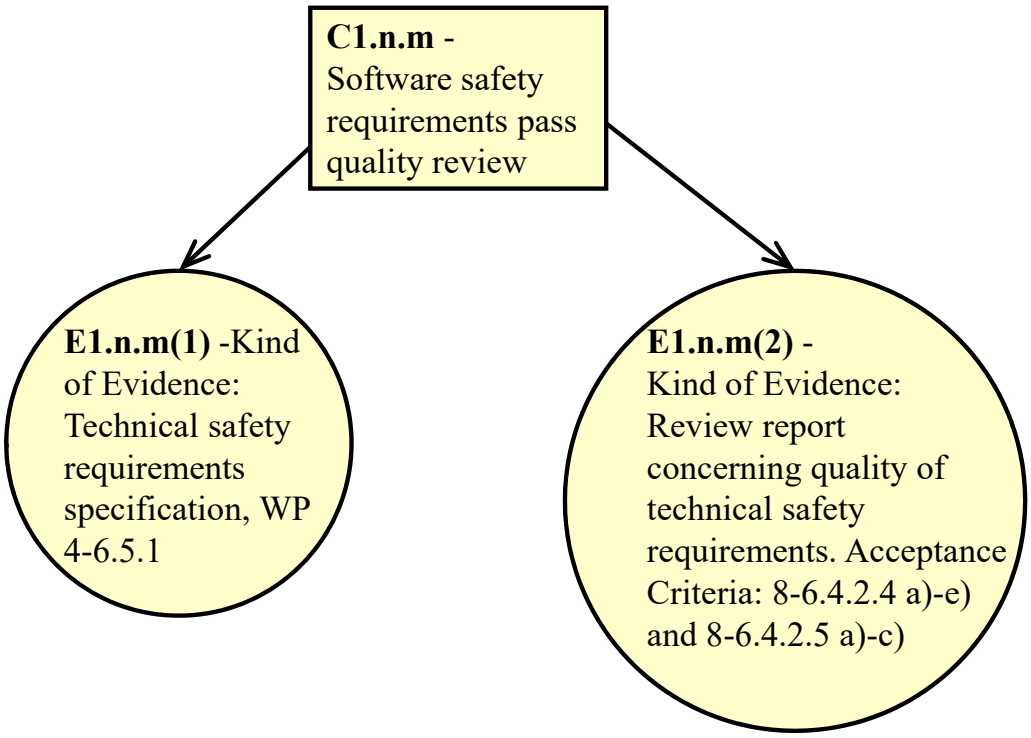


Figure 4.17: Example of specifying evidence from parts 4 (clause 6.5.1) and 8 (clauses 6.2.4 and 6.4.5) of *ISO 26262* using Principle 6. (modified from [84])

This example describes characteristics and attributes of the safety requirements, so we annotate the extract to show that we will use Principle 6. Figure 4.17 shows the acceptance criteria in the evidence nodes, directly extracted from *ISO 26262* (shown in figures 4.15, and 4.16). In this instance, goal C1.n.m is supported by two pieces of evidence: E1.n.m(1) and E1.n.m(2). Evidence E1.n.m(1) is supported by work product from part 4, clause 6.5.1 of *ISO 26262* (shown in figure 4.14) that illustrates technical safety requirements specification. Evidence E1.n.m(2) has acceptance criteria taken from part 8, clauses 6.4.2.4 and 6.4.2.5 of *ISO 26262* (shown in figures 4.15 and 4.16) that show characteristics and attributes of safety requirements.

- **Principle 7: Evidence classification**: We have identified evidence associated with four types of claims. The first three types were identified in *ISO 26262*. The last type is one that we should observe in assurance cases, but is often missing.

1) Evidence for claims related to planning
2) Evidence for claims related to process
3) Evidence for claims related to verification/qualification of tools
4) Evidence for claims related to expertise

Figure 4.18 shows the different types of evidence with their acceptance criteria.

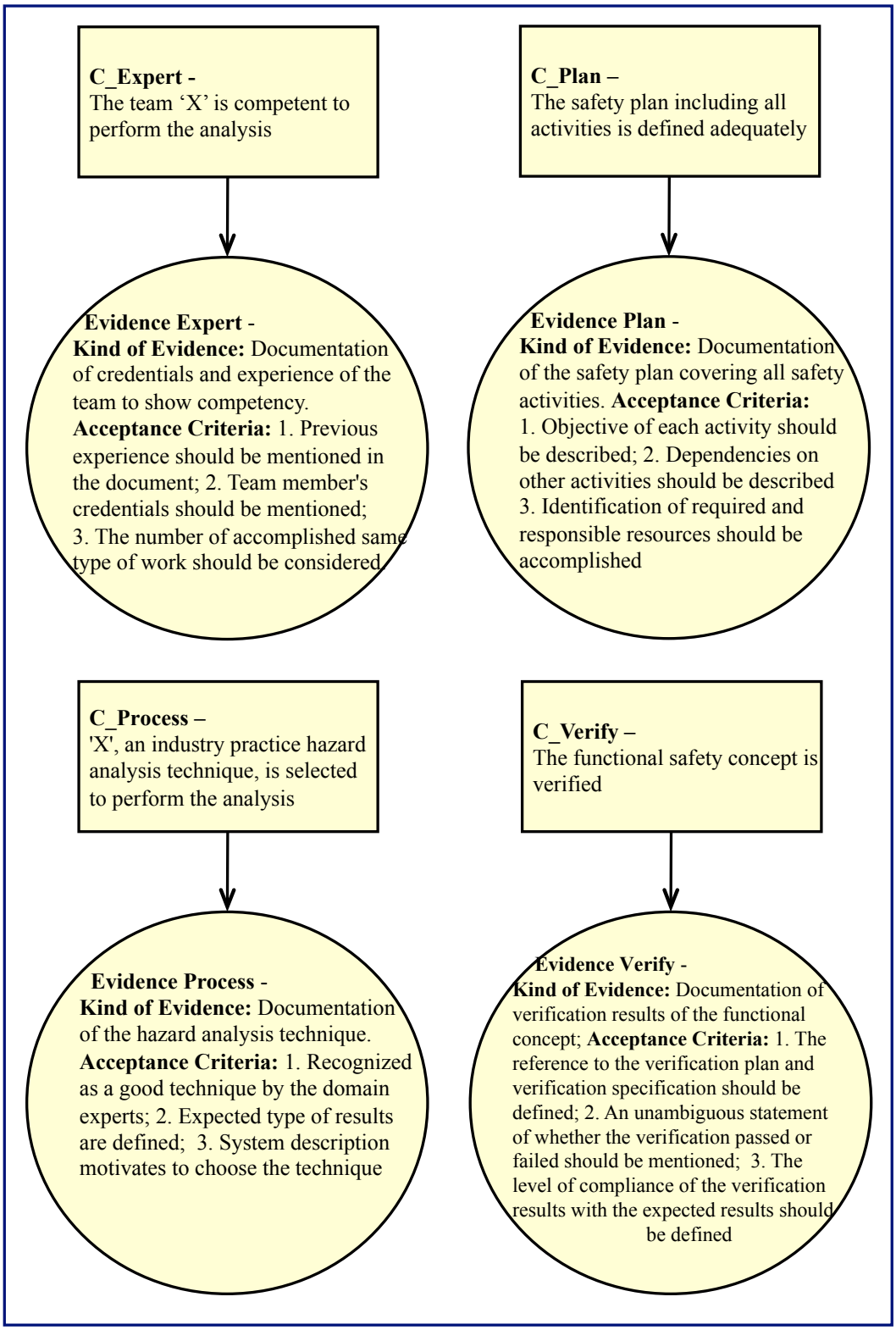


Figure 4.18: Example of different types of evidence (modified from [84])

Claim 'C_Plan' is supported by evidence related to a plan. Evidence provides documentation for a safety plan, and acceptance criteria define the specifications of a safety plan. For instance, work products 2-6.5.2 (project plan-refined), 2-6.5.4 (functional safety assessment plan), 4-5.5.3 (item integration and testing plan), 4-5.5.4 (validation plan), 6-5.5.2 (software verification plan), 8-12.5.3 (safety plan-refined), etc. are evidence related to plan. Note that each work product is represented as 'part x-clause no' of *ISO 26262*. Claim 'C_Process' is supported by evidence related to processes. Evidence mentions a description of a process with associated acceptance criteria that illustrate characteristics of a process description. For instance, work products 6-8.5.2 (software unit implementation), 7-6.5.2 (repair instructions), 5-10.5.1 (hardware integration and testing report), etc. are evidence related to processes. Claim 'C_Verify' is supported by evidence related to verification or qualification. For instance, work products 3-8.5.2 (verification report of the functional safety concept), 4-9.5.2 (validation report), 5-7.5.3 (hardware design verification report), 6-7.5.6 (software verification report), etc. are evidence related to verification. Identifying the type of claim supported by evidence helps us define the type of evidence required and the acceptance criteria for the evidence. In some cases, the standard will help us with attributes and characteristics such as mentioned above. Much of the time, we have to rely on our knowledge of software engineering and safety-critical systems to specify appropriate acceptance criteria. The above classification helps us organize our knowledge so that we can reuse (or slightly modify) appropriate acceptance criteria.

- **Principle 8: Completeness arguments**: One of the most challenging arguments we have to contend with in assurance is one that depends on completeness. For example, the claim that "all hazards are mitigated" is vital in arguing safety – if all hazards were identified. In such a case, we insist on adding a claim that explains why the best effort was expended in determining that "no additional hazards were identified". The sub-claims that support such a claim must include claims as to why the process was thorough enough to have discovered additional hazards; claims that the results of these investigations show conclusively that no additional

hazards are likely to exist; and claims regarding the expertise of the people who conducted this investigation. Figure 4.19 illustrates how an alternative hazard analysis argues that no additional hazard exists.

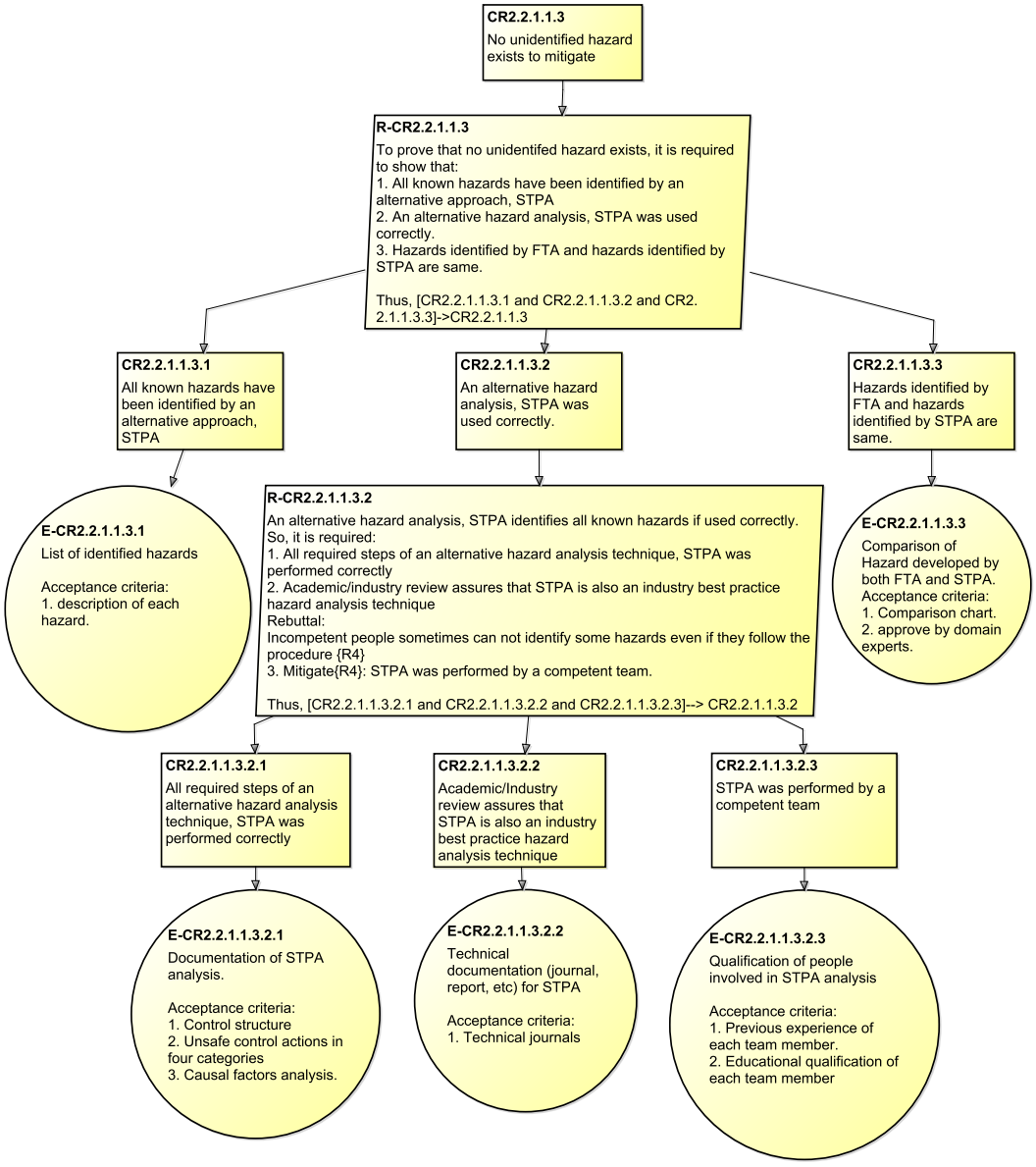


Figure 4.19: Example of completeness argument

Figure 4.19 shows that arguments related to an alternative hazard STPA are used to assure that no unidentified hazard exists. Furthermore, an argument related to competency shows that experts performed an alter-

native approach, STPA. In general, the completeness principle calls on the developer of the ACT to supplement the straight forward argument that supports the completeness claim by an additional claim related to negating a due diligence effort devoted to refuting the claim.

- **Principle 9: Argument options**: Since an assurance case template is developed prior to developing a system, it must take into account that there may be multiple ways of achieving a convincing argument. For instance, a claim that the implementation complies with its requirements may be supported by mathematical analysis and/or testing. This is an example of an argument options, as shown in figure 4.20. In this figure, a red arrow represents an optional argument path and a black arrow represents a mandatory argument path. Testing will always be used, but mathematical analysis may not be. In general, we may need optional paths, exclusive-or paths, or non-exclusive-or paths. This principle is really just a specialization of Principle 5.

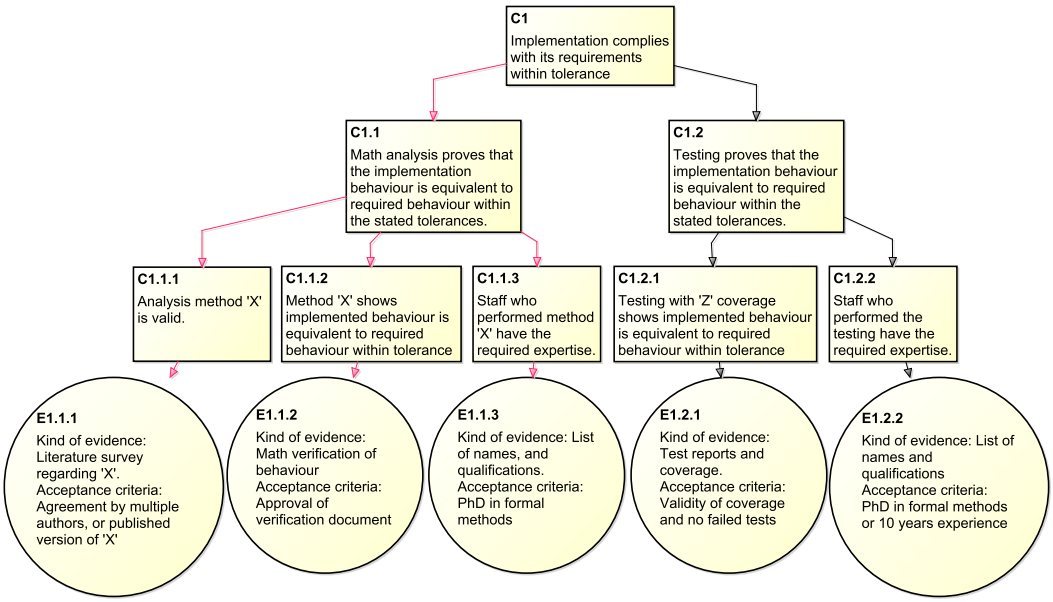


Figure 4.20: Example of argument options

- **Principle 10: Feature options**: Analogous to the argument motivated options discussed above, we may need optional, exclusive-or, or non-exclusive-or argument paths because of differences in features between

products within a product family. For example, we already saw in section 4.2.1 that figure 4.4 shows an example feature diagram for ACC. We have different possibilities within this product family concerning hardware (typically sensors and actuators) and features (behaviour). These differences dictate different claims and evidence to support safe, dependable and secure vehicles. Again, this principle is a specialization of Principle 5. Figure 4.21 shows an example of feature options. In this argument branch, a black arrow represents a mandatory argument, and a red arrow represents an optional argument branch. Here, claim 'CI1.1.2.1.2' is supported by three mandatory sub-claims 'CI1.1.2.1.2.1', 'CI1.1.2.1.2.2' and 'CI1.1.2.1.2.3' and zero or more optional sub-claims denoted by 'CI1.1.2.1.2.4+' that denotes different features of a product in a particular product family. In this scenario, it deals with mandatory three modes: cruise, follow and critical. Optional modes exist as optional features in an ACC of a specific product family. Moreover, all these terminal claims are supported by evidence. Similar to optional terminal sub-claims, evidence is also optional that supports an optional terminal claim. This specialization of Principle 5 is useful because feature variability is at the heart of product lines, and the development of ACs that can cope well with product lines is extremely important in many industries. In particular this is essential in the automotive industry.

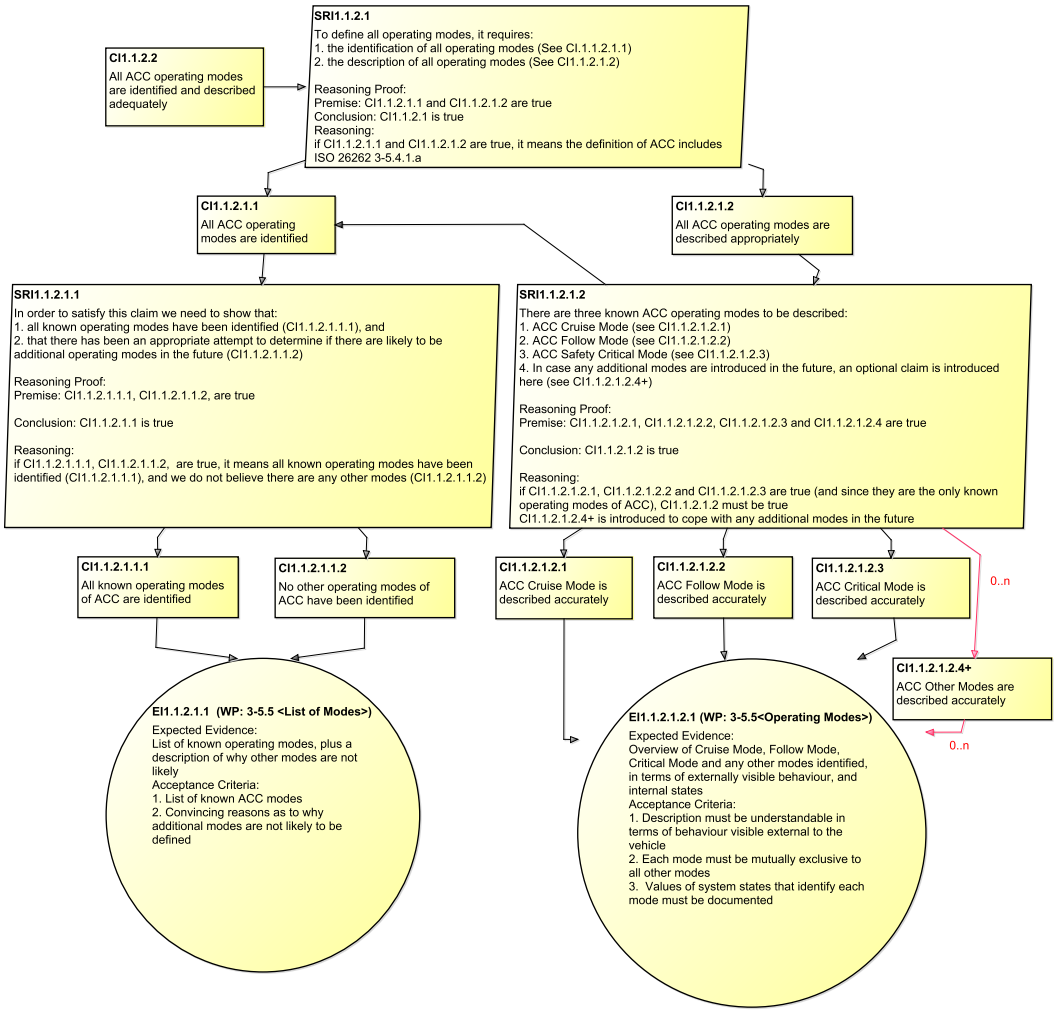


Figure 4.21: Example of feature options

4.3 Principle coverage in a safety standard, *ISO 26262*

We applied our principles to develop an ACT that complies with *ISO 26262*. We can now demonstrate coverage of our principles as applied to *ISO 26262*. Principle 1 is obvious and will not be demonstrated in this section. We start by annotating major sections of the standard with the relevant principles for each section. We can take Part 3, Section 5.4.1 as an example. The top part of figure 4.11 shows the specific sections on the relevant page of the standard with our annotations. Table 4.1 shows the percentage of *ISO 26262* covered

by each principle. Our calculations are based on 349 core requirements (Parts 2–7), 519 total requirements (Parts 2–9, including core requirements), 118 work products, and 23 HARA requirements (Part 3 only). We calculated the percent coverage of each principle based on total requirements, except for principles 3, 7 and 8. For example, principle 4 is applicable to 192 out of 519 total requirements. So, the percent coverage of principle 4 is 37%. Principle 3 is applicable to 218 out of 349 core requirements. So the percent coverage of principle 3 is 63%. We need to point out that more than one principle can apply to a set of clauses in the standard, so the coverage of the various principles will sum to more than 100%. Note that we have not yet come across requirements in the standard that we cannot include in the assurance case template.

Table 4.1: Coverage of *ISO 26262* clauses. [84]

Principle	Target	%
1	Total Requirements	100
2	Total Requirements	100
3	Core Requirements	63
4	Total Requirements	37
5	Total Requirements	9
6	Total Requirements	5
7	Work Products	75
8	HARA	1
9	Total Requirements	7
10	Total Requirements	7

4.4 Application to cybersecurity guidelines

Generally, security and safety are considered separate disciplines because of their own regulations, standards and methodologies [54]. It is clear that we cannot assume that a cyber-physical system is immune to cyber threats, and it is not feasible to assure the safety of the cyber-physical system independent of security. In this regard, a safety case is incomplete and unconvincing without consideration of the impact of security. In [54], the authors emphasize that the impact of security on the safety case should be explicitly mentioned

to make the system safe and secure. In [86], the authors describe a layered assurance approach that combines safety and security. Once the SAE cybersecurity guidelines *SAE J3061* [22] were published, we decided to apply our principles to *SAE J3061* and thus develop an ACT that assures both safety and security. We selected *SAE J3061* specifically because it is intended as a companion standard to *ISO 26262* and has been organized to mesh well with *ISO 26262*.

We now show coverage metrics of using our principles to develop an ACT complying with *SAE J3061* [22].

4.4.1 Coverage of our principles in the cybersecurity guidelines, *SAE J3061*

Table 4.2 shows the percentage of *SAE J3061* covered by each principle. The significant difference between *ISO 26262* and *SAE J3061* is the written style. Our calculations are based on 47 core requirements. This includes the overall management of cybersecurity to the production phase. The total requirements are 53, that includes the previous requirements along with supporting processes. *SAE J3061* does not have any defined work products, and 1 Threat Analysis and Risk Assessment (TARA) requirement (concept phase only). We calculated the percent coverage of each principle based on total requirements except principle 3. For example, principle 4 applies to 23 out of 53 total requirements. So, the percent coverage of principle 4 is 43%. Principle 3 applies to 31 out of 47 core requirements. So the percent coverage of principle 3 is 66%. We can not calculate coverage for Principle 7 because there is no defined work product in *SAE J3061*. Note that we have not yet come across requirements in the guidelines that we cannot include in the assurance case template.

Table 4.2: Coverage of *SAE J3061* clauses.

Principle	Target	%
1	Total Requirements	100
2	Total Requirements	100
3	Core Requirements	66
4	Total Requirements	43
5	Total Requirements	19
6	Total Requirements	6
7	Work Products	-
8	TARA	2
9	Total Requirements	4
10	Total Requirements	2

Chapter 5

Case Studies: Principles and Safe and Secure Over-the-Air Updates

Chapter 4 presented our proposed development principles along with coverage metrics of both *ISO 26262* and *SAE J3061*. This chapter shows an excerpt of an ACT for safety and security complying with *ISO 26262* and *SAE J3061* by applying these principles. Furthermore, we discuss the extension of our template to comply with both *ISO 26262* and *SAE J3061* in building an ACT that assures safety and security for automotive Over-the-air updates, which we will represent simply by (OTA). We will also highlight how an ACT developed for OTA can guide us in identifying potential threats along with suggesting a mitigation strategy. This chapter extends work published in [87].

5.1 Assurance Case Template complying with *ISO 26262*

We started this aspect of the research by developing an ACT for an Advanced Driver Assistance System (ADAS) compliant with *ISO 26262*. We will later add compliance with *SAE J3061*. Concerning this, figure 5.1 shows the top-level of a safety ACT. The top-level claim is: *"<ADAS> considered as an ISO 26262 item, delivers the behaviour required and does not adversely affect the*

safety in the vehicle, over its expected lifetime in its intended environment.". Six sub-claims support the top-level claim. All six sub-claims deal with safety. The tabs on the top left of a claim node indicate that this is a *module*, and the remainder of that argument path can be seen by "opening" that module (in the tool we use, achieved by double-clicking the tab). The relevant ISO part-clauses are indicated inside a smaller text box within the claim. In terms of

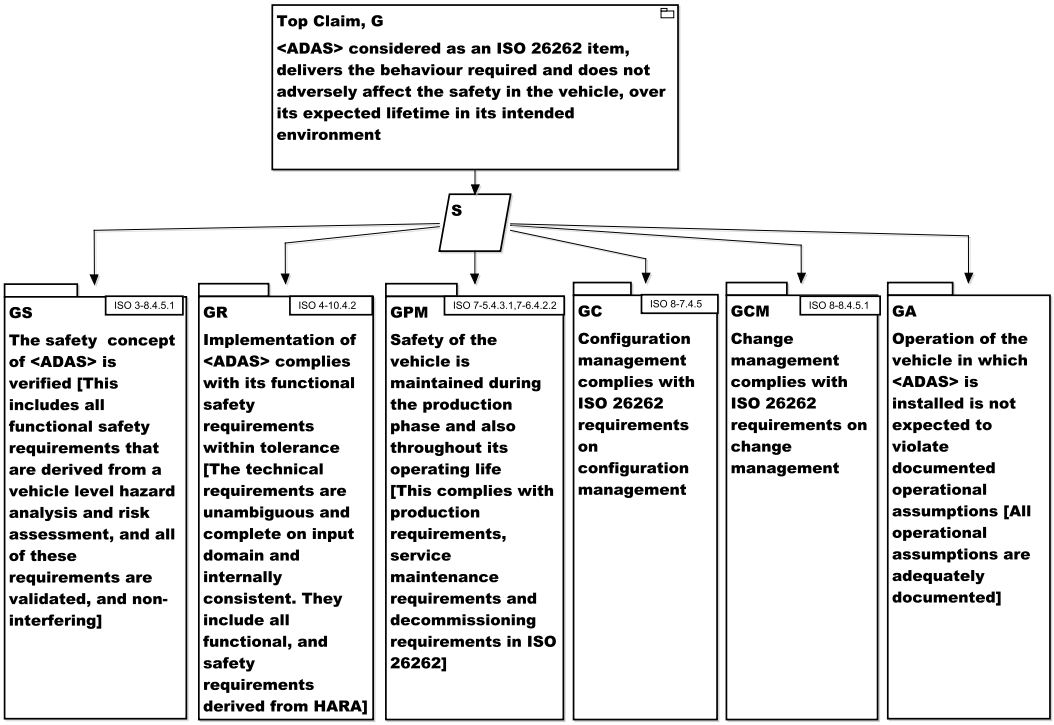


Figure 5.1: Top-Level of a Safety ACT

software engineering, four argument paths can be shown to adequately support this top claim of a specific system's safety. An informal description of the four sub-claims supported by these arguments is:

1. The system's requirements are validated. [**GS** in Figure 5.1.]
2. The system is implemented to meet its requirements. [**GR** in Figure 5.1.]
3. The system is safe even when maintenance is performed. [**GPM** in Figure 5.1.]
4. The system is operated within its operational assumptions. [**GA** in Figure 5.1.]

ISO 26262 takes a similar approach, and add two more claims:

1. Compliance with configuration management requirements. [**GC** in Figure 5.1.]

2. Compliance with change management requirements. [**GCM** in Figure 5.1.]

Claim 'GS' shows the generation of functional safety requirements and validation of requirements related arguments. Note that "all" requirements include functional safety requirements derived from hazard analysis and risk assessment. This claim is supported by sub-claims transformed from clauses of part 3, *ISO 26262*. Claim 'GR' shows arguments of compliance of the implementation with requirements. It assures that the implementation meets its requirements (within tolerance). This claim is supported by sub-claims transformed from part 4 of *ISO 26262*. Claim 'GPM' deals with production, maintenance and decommission related arguments in compliance with *ISO 26262*. This claim is supported by sub-claims transformed from clauses of part 7, *ISO 26262*. Claim 'GC' shows configuration management related arguments in compliance with *ISO 26262*. This claim is supported by sub-claims transformed from clauses of part 8, *ISO 26262*. Claim 'GCM' shows change management related arguments in compliance with *ISO 26262*. This claim is supported by sub-claims transformed from clauses of part 8, *ISO 26262*. Claim 'GA' is not a part of *ISO 26262*. However, it is necessary to assure that <ADAS> does not violate any assumption that may make it unsafe during operation. Each of the claims ('GS', 'GR', 'GPM', 'GC', 'GCM' and 'GA') is true and together imply that top claim 'G' is true. The benefit of the top-level structure in the ACT is that it makes it easier to understand the reasoning built into *ISO 26262*.

We now focus on the argument structure supporting claim 'GS.' Figure 5.3 illustrates the first few levels of the argument supporting claim 'GS.'

8.4.5 Verification of the functional safety concept

8.4.5.1 The functional safety concept shall be verified in accordance with ISO 26262-8:2011, Clause 9, to show

a) its consistency and compliance with the safety goals; and

b) its ability to mitigate or avoid the hazardous events.

Principle 4

Figure 5.2: Part 3, Clause 8.4.5.1 of *ISO 26262* [3]

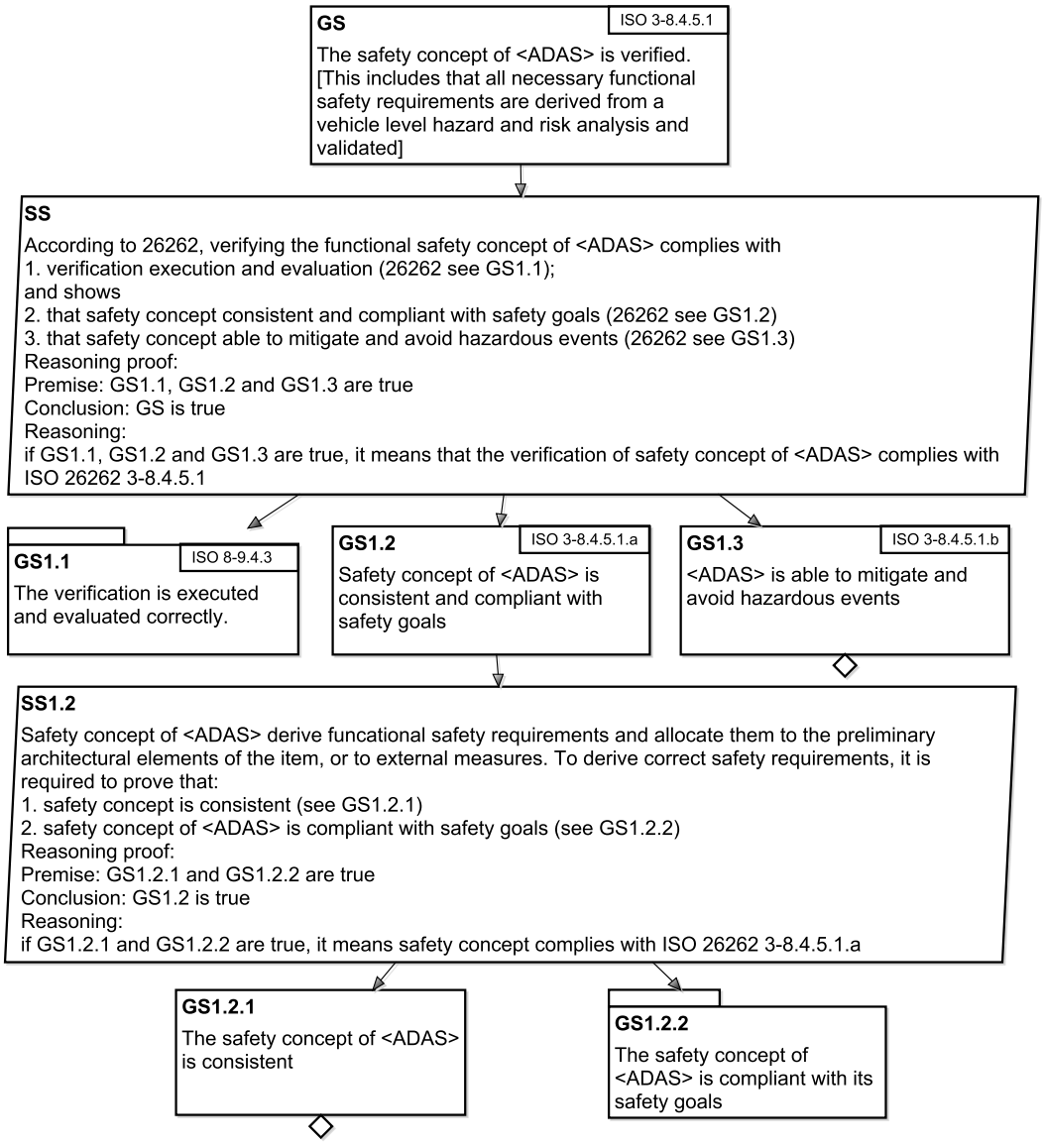


Figure 5.3: An excerpt of the argument supporting claim 'GS'

9.4.3 Verification execution and evaluation

9.4.3.1 The verification shall be executed as planned in accordance with 9.4.1 and specified in accordance with 9.4.2.

9.4.3.2 The evaluation of the verification results shall contain the following information: Principle 4

a) the unique identification of the verified work product,

b) the reference to the verification plan and verification specification,

c) the configuration of the verification environment and verification tools used, and the calibration data used during the evaluation, if applicable,

d) the level of compliance of the verification results with the expected results,

e) an unambiguous statement of whether the verification passed or failed; if the verification failed the statement shall include the rationale for failure and suggestions for changes in the verified work product, and

NOTE The verification is evaluated according to the criteria for completion and termination of the verification [see 9.4.1.1 c)] and to the expected verification results.

f) the reasons for any verification steps not executed.

Figure 5.4: Part 8, Clause 9.4.3 of *ISO 26262* [3]

Figure 5.3 shows a claim 'GS' that mentions "The safety concept of <ADAS> is verified. [This includes that all necessary functional safety requirements are derived from a vehicle level hazard and risk analysis and validated.]" This claim is converted from a clause in *ISO 26262* (part 3, clause 8.4.5.1). Figure 5.2 shows a clause '8.4.5.1' of part 3, *ISO 26262*. The clause mentions that functional safety concept shall be verified according to clause 9 of part 8 of *ISO 26262* and show consistency, compliance with safety goals and mitigate or avoid hazardous events. Claims 'GS', 'GS1.1', 'GS1.2' and 'GS1.3' are converted from clauses 8.4.5.1 of part 3 (ISO 26262), 9.4.3 of part 8 (ISO 26262), 8.4.5.1.a of part 3 (ISO 26262) and 8.4.5.1.b of part 3 (ISO 26262) respectively. Based on "Principle 4 (Conjunctive)", argument 'SS' argues sub-claims 'GS1.1', 'GS1.2' and 'GS1.3' support upper-claim 'GS' if all sub-claims are valid. 'SS' argues that safety concept verification execution and evaluation comply with *ISO 26262* clauses from part 8. Furthermore, 'SS' argues that safety concept is verified to show that safety concept is consistent, compliant and able to mitigate or avoid hazardous events complying with *ISO 26262*. Similarly, claim 'G1.1' is supported by an argument branch converted from a clause '9.4.3' of part 8, *ISO 26262*, shown in figure 5.4. Figure 5.5 shows a claim 'GS1.1' that mentions "The verification is executed and evaluated correctly". Based on "Principle 4 (Conjunctive)", argument 'SS1.1' argues that

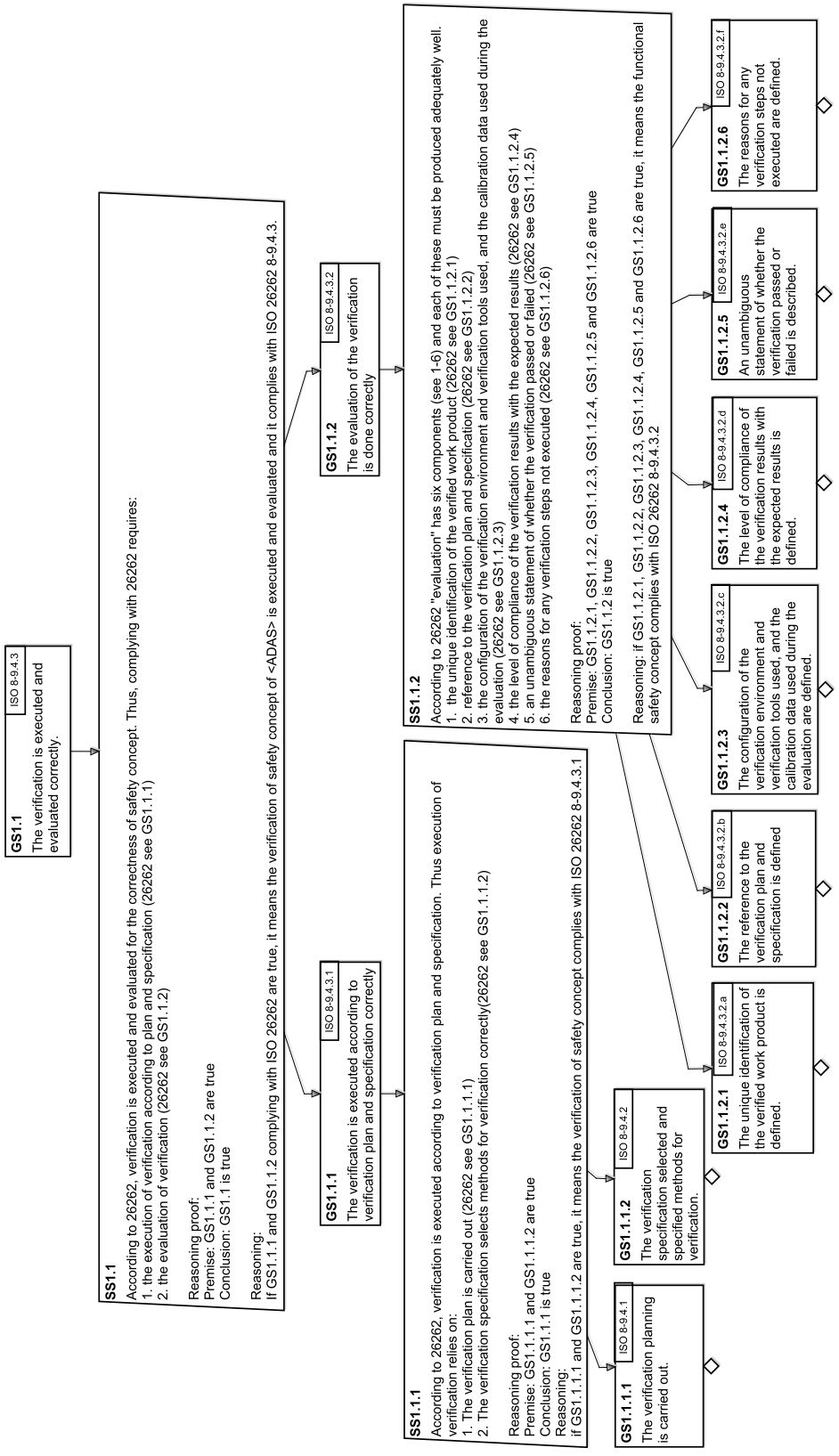


Figure 5.5: An excerpt of the argument supporting claim 'GS1.1'

verification of safety concept is executed and evaluated complying with *ISO 26262*. Concerning this, 'SS1.1' argues that verification execution is done in accordance with *ISO 26262* and the verification is evaluated also in accordance with *ISO 26262*. When both claims 'GS1.1.1' and 'GS1.1.2' are valid, then can support the upper level claim based on "Principle 4 (Conjunctive)". Two subclaims 'GS1.1.1' and 'GS1.1.2' are converted from clauses '9.4.3.1' and '9.4.3.2' of part 8, *ISO 26262*. Similarly, claim 'GS1.1.1' is supported by two sub-claims 'GS1.1.1.1' and 'GS1.1.1.2' that are converted from clauses '9.4.1.1' and '9.4.2.1'. Argument 'SS1.1.1' argues that the verification is executed according to plan and specification when the verification plan is carried out correctly and specification selected methods for verification according to *ISO 26262*. Argument 'SS1.1.1' follows "Principle 3 (flip-it)" that guides how two subclaims support top-claim. Claim 'GS1.1.2' is supported by six sub-claims 'GS1.1.2.1', 'GS1.1.2.2', 'GS1.1.2.3', 'GS1.1.2.4', 'GS1.1.2.5' and 'GS1.1.2.6' that are converted from clauses '9.4.3.2.a', '9.4.3.2.b', '9.4.3.2.c', '9.4.3.2.d', '9.4.3.2.e' and '9.4.3.2.f' of part 8, *ISO 26262* (shown in figure 5.4) respectively. Argument 'SS1.1.2' argues that evaluation is performed by adequately completing six components in accordance with *ISO 26262*. Argument 'SS1.1.2' follows "Principle 4 (Conjunctive)". The rest of the argument supporting claim 'GS' can be found in Appendix A.

5.2 Assurance Case Template complying with both *ISO 26262* and *SAE J3061*

Nowadays, it is widely accepted that a cyber-physical system must be immune to cyber threats. Thus, it is not feasible to assure the safety of a cyber-physical system independent of security. In the previous section we applied our development principles to *ISO 26262* in order to construct an ACT that assures the safety of <ADAS>. We now apply these principles to *SAE J3061* to modify that ACT to build a version that assures both safety and security of <ADAS>, in compliance with both *ISO 26262* and *SAE J3061*. It is important to note that the writing style of *SAE J3061* is a bit different in comparison with *ISO 26262*. For instance, *ISO 26262* illustrates each requirement in a

separate clause whereas *SAE J3061* describes more than one requirements in a single clause.

Figure 5.6 shows the top-level of a "security informed safety ACT". The top-level claim is *"<ADAS> considered as an ISO 26262 item/SAE J3061 feature, delivers the behaviour required and does not adversely affect the safety or create security vulnerabilities in the vehicle, over its expected lifetime in its intended environment"*. Six sub-claims support the top-level claim. All six sub-claims deal with safety and security issues together with consistent interaction. Similar to figure 5.1, the relevant *ISO* and *SAE* clauses/sections are indicated inside a smaller text box within the claim. In terms of software

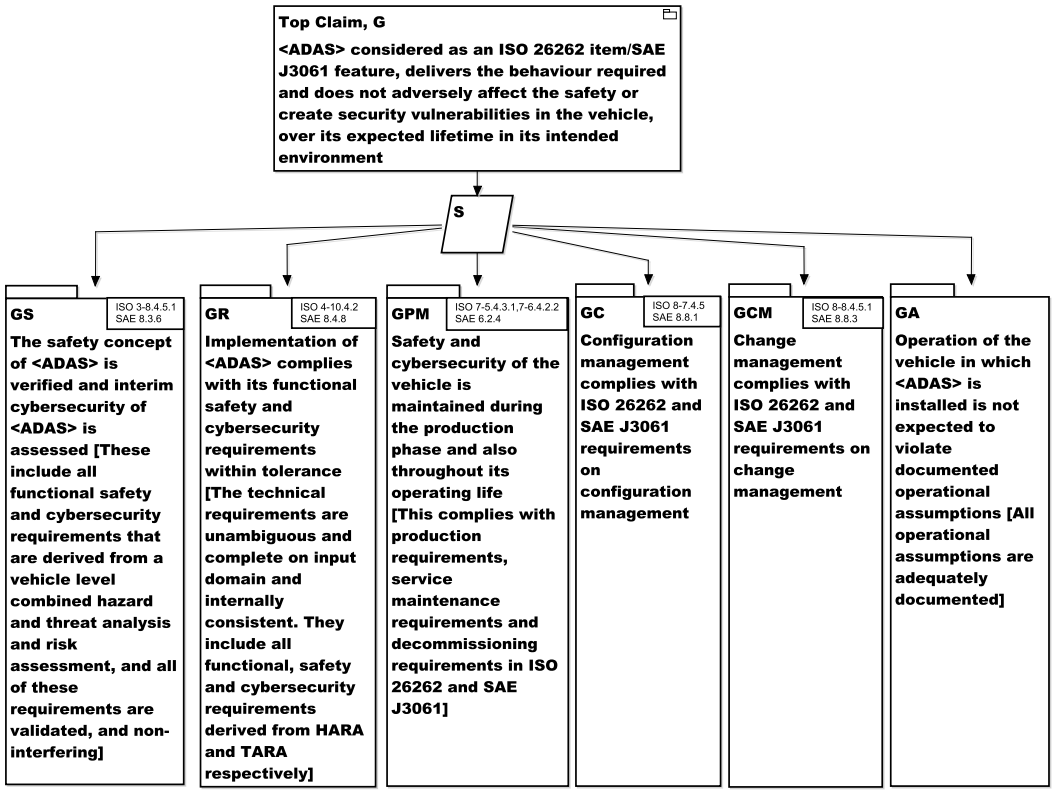


Figure 5.6: Top-Level of a Safety and Security ACT (modified from [87])

engineering, four argument paths can be shown to adequately support this top claim of a specific system's safety and security. An informal description of the four sub-claims supported by these arguments is:

1. The system's requirements are validated. [**GS** in Figure 5.6.]

2. The system is implemented to meet its requirements. [**GR** in Figure 5.6.]

3. The system is safe and secure even when maintenance is performed. [**GPM** in Figure 5.6.]

4. The system is operated within its operational assumptions. [**GA** in Figure 5.6.]

ISO 26262 and *SAE J3061* take a similar approach, and add two more claims:

1. Compliance with configuration management requirements. [**GC** in Figure 5.6.]

2. Compliance with change management requirements. [**GCM** in Figure 5.6.]

Functional safety and cybersecurity requirements are derived from a vehicle level hazard and threat analysis and risk assessment. This claim 'GS' is supported by sub-claims transformed from a clause 8.4.5.1 of part 3, *ISO 26262* and a clause 8.3.6 of *SAE J3061*. Claim 'GR' shows arguments of compliance of the implementation with requirements. It assures that implementation meets the requirements. This claim is supported by sub-claims transformed from part 4 of *ISO 26262* and clauses (8.4.8) of *SAE J3061*. Claim 'GPM' deals with production, maintenance and decommission related arguments in compliance with both *ISO 26262* and *SAE J3061*. This claim is supported by sub-claims transformed from clauses of part 7, *ISO 26262* and clauses (6.2.4) of *SAE J3061*. Claim 'GC' shows configuration management related arguments in compliance with both *ISO 26262* and *SAE J3061*. This claim is supported by sub-claims transformed from clauses of part 8, *ISO 26262* and clauses (8.8.1) of *SAE J3061*. Claim 'GCM' shows change management related arguments in compliance with both *ISO 26262* and *SAE J3061*. This claim is supported by sub-claims transformed from clauses of part 8, *ISO 26262* and clauses (8.8.3) of *SAE J3061*. Claim 'GA' is not a part of *ISO 26262* and *SAE J3061*. However, it is necessary to assure that <ADAS> does not violate any assumption that may make it unsafe and insecure during operation. Each of the claims ('GS', 'GR', 'GPM', 'GC', 'GCM' and 'GA') is true and together imply that top claim 'G' is true.

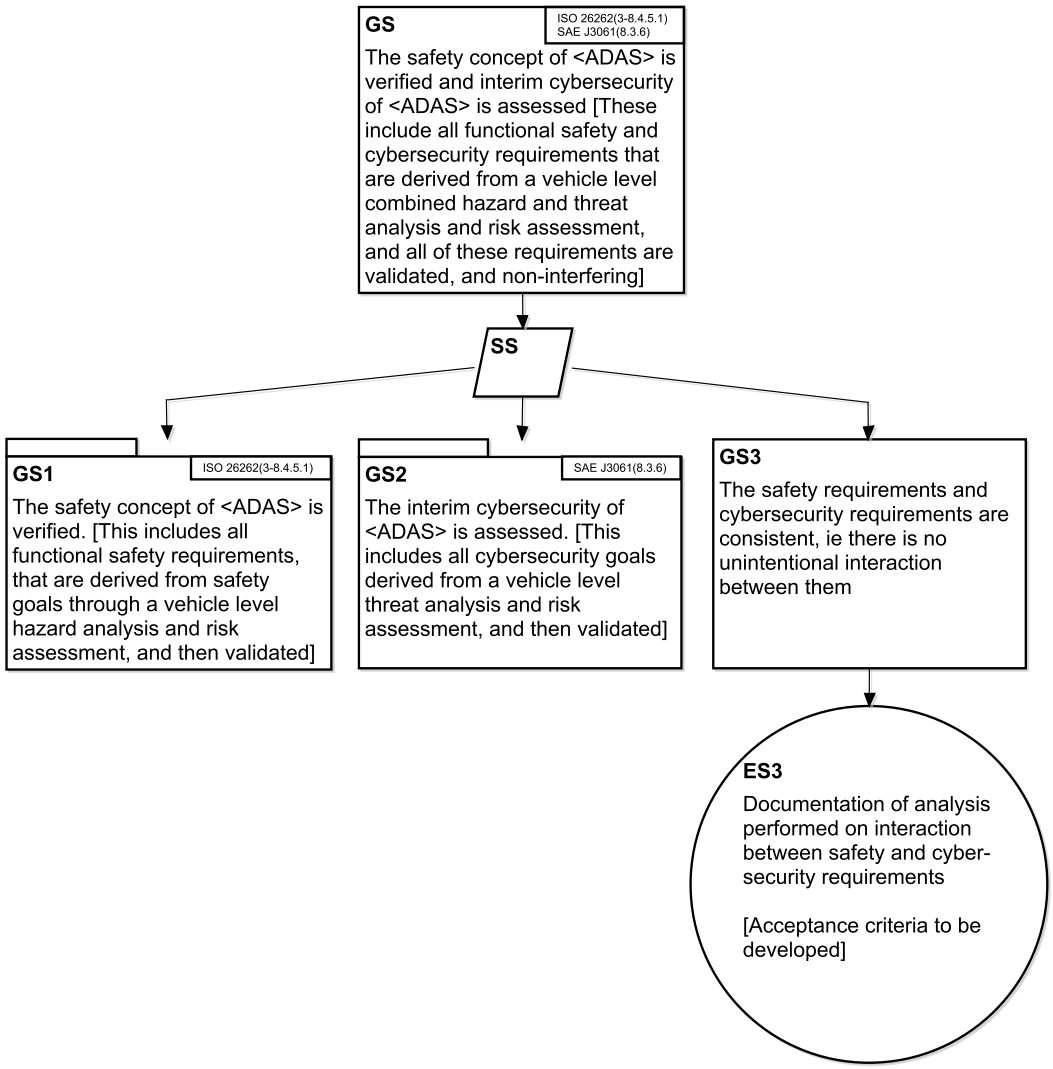


Figure 5.7: An excerpt of arguments supporting claim 'GS' for safety and security

In figure 5.7, claim 'GS' mentions that "The safety concept of <ADAS> is verified, and interim cybersecurity of <ADAS> is assessed. [These include all functional safety and cybersecurity requirements that are derived from a vehicle level combined hazard and threat analysis and risk assessment, and all of these requirements are validated and non-interfering]" and is supported by three sub-claims, 'GS1', 'GS2' and 'GS3'. Claim 'GS1' deals with safety argument complying with a clause 8.4.5.1 of part 3, *ISO 26262*. The upper right corner of claim 'GS1' represents a clause of *ISO 26262*. Claim 'GS2' deals with security argument complying with a clause 8.3.6 of *SAE J3061*. Claim 'GS3' assures consistency of safety requirements and cybersecurity requirements. Claim 'GS1' follows the same argument shown in figure 5.3. Claim 'GS2' shows an argument branch that deals with the interim cybersecurity assessment. Figure 5.8 shows a clause 8.3.6 of *SAE J3061*. It demonstrates

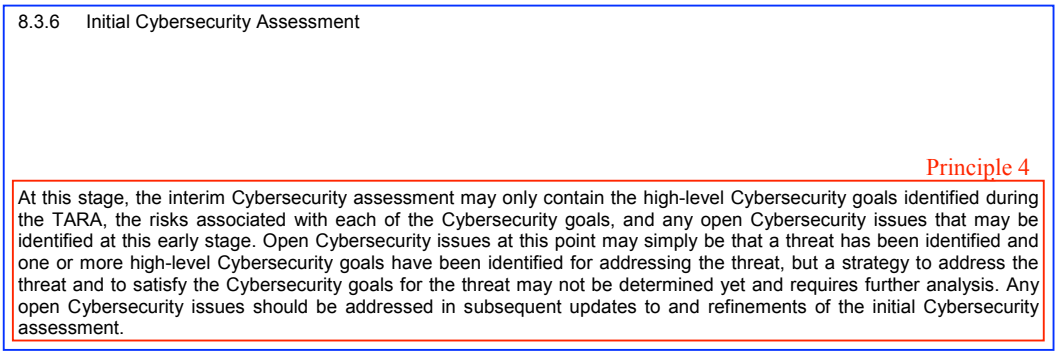


Figure 5.8: Clause 8.3.6 from *SAE J3061* [22]

that assessment of interim cybersecurity, which includes high-level cybersecurity goals identified during the TARA and risks associated with each of the cybersecurity goals and any open cybersecurity issues identified at this stage. In figure 5.9, claim 'GS2' is converted from a clause 8.3.6 of *SAE J3061*. Based on "Principle 4 (Conjunctive)", argument 'SS2' argues that interim cybersecurity assessment is performed when high-level cybersecurity goals along with risks are assessed, and any open cybersecurity issues are determined. Thus, claim 'GS2' can be supported by two sub-claims when both 'GS2.1' and 'GS2.2' are valid. It is important to note that claims 'GS2.1' and 'GS2.2' are also converted from clause 8.3.6 of *SAE J3061*. This argument is considered a single unit in argument branches based on "Principle 4 (Conjunctive)". Figure

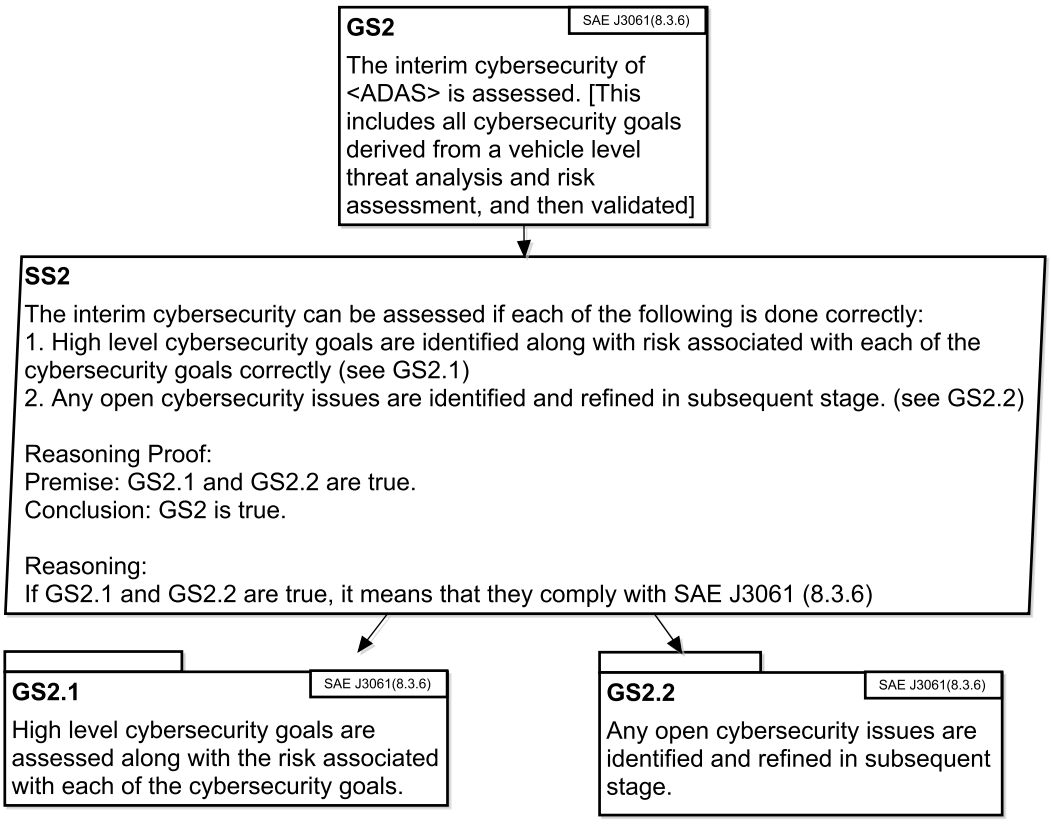


Figure 5.9: An excerpt of arguments supporting claim 'GS2' for cybersecurity

5.10, shows two clauses 8.3.4 and 8.3.5 of *SAE J3061*. Clause 8.3.4 illustrates the cybersecurity concept, and clause 8.3.5 illustrates functional cybersecurity requirements. Claim 'GS2.1' and supporting two sub-claims 'GS2.1.1' and 'GS2.1.2' are converted from three clauses 8.3.6, 8.3.4 and 8.3.5 of *SAE J3061*. Based on "Principle 3 (flip-it)", argument 'SS2.1' argues that cybersecurity goals and risks associated with each of the cybersecurity goals are assessed when the cybersecurity concept is defined, and functional cybersecurity requirements are determined correctly. Thus, claim 'GS2.1' is supported by two sub-claims, 'GS2.1.1' and 'GS2.1.2', when both sub-claims are valid. Similarly, claims 'GS2.1.1', 'GS2.1.1.1', 'GS2.1.1.2', and 'GS2.1.1.3' are converted from clause 8.3.4 of *SAE J3061*. Based on "Principle 4 (Conjunctive)", argument 'SS2.1.1' argues that the cybersecurity concept is defined if cybersecurity goals, risks associated with cybersecurity goals and potential high-level strategy for satisfying cybersecurity goals are determined adequately. Thus,

8.3.4 Cybersecurity Concept

Principle 4

The Cybersecurity concept is a description of the high-level strategy for obtaining Cybersecurity for the feature. At this stage, the Cybersecurity concept may contain the high-level Cybersecurity goals identified during the TARA, the risks associated with each of the Cybersecurity goals, and a potential high-level strategy for satisfying the Cybersecurity goals. The strategy for addressing the Cybersecurity goals may be dependent on the potential risk level of the threat associated with the Cybersecurity goals. An organization may be able to create a template of high-level strategies for the different classifications of potential threats that are identified. Creating a template based on threat risk level would simplify and streamline creation of a Cybersecurity concept. During the next phase of development, product development at the system level, the Cybersecurity concept will be updated and refined to a technical level. That is, the high-level Cybersecurity strategy will be refined from a functional level strategy to a technical strategy.

8.3.5 Identify Functional Cybersecurity Requirements

Once the high-level strategy is determined for satisfying the Cybersecurity goals for the identified threats, the functional Cybersecurity requirements can be determined. Essentially, the Cybersecurity goals identified during the TARA are the highest-level Cybersecurity requirements. These functional Cybersecurity requirements are derived from the Cybersecurity strategy and derived and refined from the Cybersecurity goals. Figure 18 provides a graphical depiction of the flow from the Cybersecurity goals to the functional Cybersecurity requirements.

Figure 5.10: Clauses 8.3.4 and 8.3.5 from *SAE J3061* [22]

claim ‘GS2.1.1’ is supported by three sub-claims ‘GS2.1.1.1’, ‘GS2.1.1.2’, and ‘GS2.1.1.3’ are valid. The remaining argument branches are shown in the Appendix A.

5.3 Extensions for Over-the-air (OTA) updates

As an example of applying this assurance to a specific example, we chose OTA. OTA is being used by OEMs to maintain software in vehicles. The original motivation for over-the-air (OTA) updates to automotive software seems to have been a realization that customers view a trip to the dealership to install a software patch as an avoidable waste of their time. This is true even when the patch introduces a new feature that they are pleased to install. An update can take place without the presence of the owner. Whether the update is installed automatically or needs approval before driving depends on the criticality of the update. For example, if the update is for parts of the infotainment system, perhaps it can be installed automatically. If the update is for a critical component of the vehicle, it may be necessary to have driver approval. In all cases, the update will be installed when the car is stopped or in park mode. In addition, OEMs hope that OTA Updates will be a lot more cost-effective than paying dealerships to install the updates. However, with the implementation of OTA firmware updates come new entry points for hackers to tamper with a vehicle’s software. Not only do we introduce the

potential for hacking, but we also remove a trained technician from the process. These trained professionals help validate that the new firmware installation is successful and ensure that there are no safety hazards resulting from the update. For example, even a simple update to an infotainment system caused cycles of rebooting the heads-up display, accompanied by distracting bright purple flashes, thus resulting in a severe safety concern [2]. We now extend our template for safety and security to cover specific arguments related to OTA. It is important to note that we are primarily interested in the final safety of the vehicle. To this end, we have to consider safety aspects of OTA independent of security concerns, as well as the effect of security issues on vehicle safety - and even the adverse effect of safety mitigation on security.

5.3.1 An ACT for safety & security of OTA updates

Our approach here is to modify the previously developed ACT to include assurance when maintenance is performed using OTA Updates. This approach is used since both of the relevant standards do not include specific guidance for OTA updates, and we believe that there is general guidance we can provide that covers both maintenances implemented at a dealership and through OTA updates.

Part 7 of *ISO 26262* and Section 6 of *SAE J3061* define maintenance requirements on production, and operation. We highlight this path because it is of central importance in arguing the safety and security of OTA Updates. The rationale of highlighting this path is this path deals with the safety of maintenance-related arguments and OTA is a process for updating a vehicle's features. Figure 5.12 shows a slice of "GPM" developed from *ISO 26262* and *SAE J3061*. The assurance argument is primarily structured on the structure visible in *ISO 26262* and *SAE J3061*. For example, the safety argument is in the 'GPM1' branch and the security argument in the 'GPM2' branch. *ISO 26262* describes requirements on production, maintenance, and decommissioning. One option would have been to split these at the sub-claim level shown in Figure 5.6. We chose to combine them in a single claim, and so the premises for 'GPM1' are 'GPM1.1' (production), 'GPM1.2' (maintenance) and 'GPM1.3' (decommissioning). Similarly, the premises for 'GPM2' are 'GPM2.1' (production), 'GPM2.2' (maintenance).

Figure 5.12 shows compliance with the ISO and SAE standards before any specialization for OTA Updates. It is reasonably obvious that OTA Updates will affect claim 'GPM1.2' and its argument path (safety) and claim 'GPM2.2' and its argument path (cyber-security) – primarily the 'GPM2.2.1' argument path. In order to include OTA Updates explicitly in the ACT, we have to analyze exactly what is different between traditional at the dealership maintenance and OTA Update maintenance. This involves both hazard and threat analyses. OTA updates introduce both safety hazards and security vulnerabilities. The manufacturer will have thoroughly tested the update, but there are significant issues of *completeness* that complicate this task. An obvious example is the malfunctioning heads-up display discussed in chapter 1:1.1. In terms of safety, OTA updates are performed remotely, without the aid of a knowledgeable technician responsible for testing the update. It is important to assure two things concerning safety: OTA updates are performed correctly, and there is no unidentified hazard after updates. Figure 5.13 shows an argument branch that supports claim 'GPM1.2'. In that argument branch, claim 'GPM1.2.8' mentions that the installed update does not create any hazard and claim 'GPM1.2.1.2' mentions that OTA activities follow the procedure correctly. Proving that updates do not create any hazard (claim 'GPM1.2.8'), all identified hazards have been mitigated, and no unidentified hazard exists are assured. Similarly, proving that installation follows all procedures, tests are done correctly, and an expert's review is assured.

In terms of security, *SAE J3061* describes in general how to protect the vehicle from cyber-security attacks. The guidebook does not explicitly consider what is necessary when maintenance is performed through OTA. We want to include the option of OTA updates in our ACT. To do this, we use the work reported in the design of Uptane [23, 45] as the basis of the OTA-specific arguments in the ACT, as far as security is concerned. Once we have a design in mind (and Uptane is sufficiently generic in terms of identification of communication channels), we are in a position to generate threats and mitigations that can be used as a base for the assurance case argument. Figure 5.14 shows arguments related to OTA complying with cybersecurity related requirements for update procedures. Claim 'GPM2.2.1.5' mentions that OTA updates comply with cybersecurity-related requirements, further supported by

two sub-claims (claims 'GPM2.2.1.5.1', and 'GPM2.2.1.5.2') that argue no security vulnerability in OEM update server and primary and secondary ECUs based on Uptane.

When analyzing OTA Updates, not only must the security of data be considered, but the protocols that handle this data must also be considered. We note that relevant attacks consistently target and exploit weaknesses of four main security properties: *confidentiality*, *integrity*, *availability*, and *authenticity* [26]. By adequately protecting these four main properties, which have been the root targets of all known attacks, it is possible to provide security assurance for the system.

Although the 'CIA' triad are considered the most crucial components of information security, they are not enough to completely secure the system. The 'STRIDE' Threat Model from Microsoft [42] recommends protection of Authenticity, Authorization, and Non-repudiation as well.

Authenticity: Authenticity ensures that the data received comes from a trustworthy source. This protects against man in the middle (MITM) and spoofing attacks [26].

Authorization: Authorization prevents unprivileged parties gaining access [44].

Non-repudiation: Maintaining secure logs of activities and the entities to which they are attributed protects non-repudiation scenarios [44].

We also need to consider two generic security measures – **private key protection** and **version control**. Private key protection can help prevent "key extraction" [26], and version control is essential in general, but can also help protect against installation of an older version of software.

There exist several tools and methodologies for classifying and managing security-related threats. Many of these are outlined in *SAE J3061*. We chose to use Microsoft's threat modelling tool which performs threat analysis using 'STRIDE' [42] and a data flow diagram of the system [29]. 'STRIDE' classifies attacks (threats) into six categories – *Spoofing identity*, *Tampering with data*, *Repudiation*, *Information disclosure*, *Denial of service*, and *Elevation of privilege*.

For each type of threat presented by 'STRIDE', Microsoft suggests a security property countermeasure.

We created a data flow diagram using the template developed by the 'NCC' group [30] that describes relevant communication flow in the connected car as input to 'STRIDE.' We modelled our data flow diagram on a Uptane design (Figure 1 in [23]). Our data flow diagram of a partial vehicle network illustrating OTA Updates is shown in Figure 5.15. We used the 'NCC' template together with the data flow diagram in Figure 5.15 to analyze OTA Updates for the connected car, to generate threats and corresponding mitigations to include in our ACT.

A slice of 'GPM' specialized for OTA Updates using the results from the 'STRIDE' analysis, is shown in figure 5.16. We discuss this in more detail in the following section.

5.4 Identification of potential vulnerability using ACT for OTA updates

We use this slice (see figure 5.16) of the ACT to explore what we need to do to develop a safe and secure OTA update design. Thanks to the extensive work in the Uptane project, we could use their design and a python implementation as an example. We found that part of the implementation does not satisfy one of the threat mitigation requirements in the acceptance criteria of the ACT (see figure 5.16).

In particular, Threat 2 in 'EPM 2.2.1.1.1.a.2.1' refers to a Man-In-The-Middle attack (MITM) threat. This may lead to a vulnerability in the Uptane implementation. The suggested mitigation strategy is that communication must be secured (using 'TLS' or cryptographically signed). In the sample implementation, requests from the primary ECU to the OEM's time server for an updated timestamp are sent as unsigned plain text. (The OEM time server is included in the OEM Update Server in Figure 5.15). Although the communication is just a pseudorandom nonce from each secondary ECU, this allows MITM agents to alter the communication as they see fit, and force the system into an unexpected state. Depending on a vendor's implementation, attacks such as a buffer overflow could be possible. In this case, editing the

packet to contain no nonces, then allowing it to go through, causes the primary ECU to ignore the updated time. However, it will then make its next request to the time server without sending any nonces, at which point the MITM can inject a subset of the previously blocked nonces, and the primary ECU will accept the reply from the time server. The primary ECU will then pass the message from the time server along with all the secondary ECUs, but since the MITM manipulated the exchange to only contain a subset of nonces, only secondary ECUs in this selected subset will accept the updated time. If a vendor decides to implement a check for a recent timestamp from the time server on each secondary ECU before installing an update, a Mixed Bundle Attack could be possible. The ACT suggests mitigating this vulnerability by signing the packet of nonces from the primary ECU to the time server. If the developer does this, this specific MITM attack can be mitigated. This example demonstrates one of the benefits of pre-determining acceptance criteria for evidence.

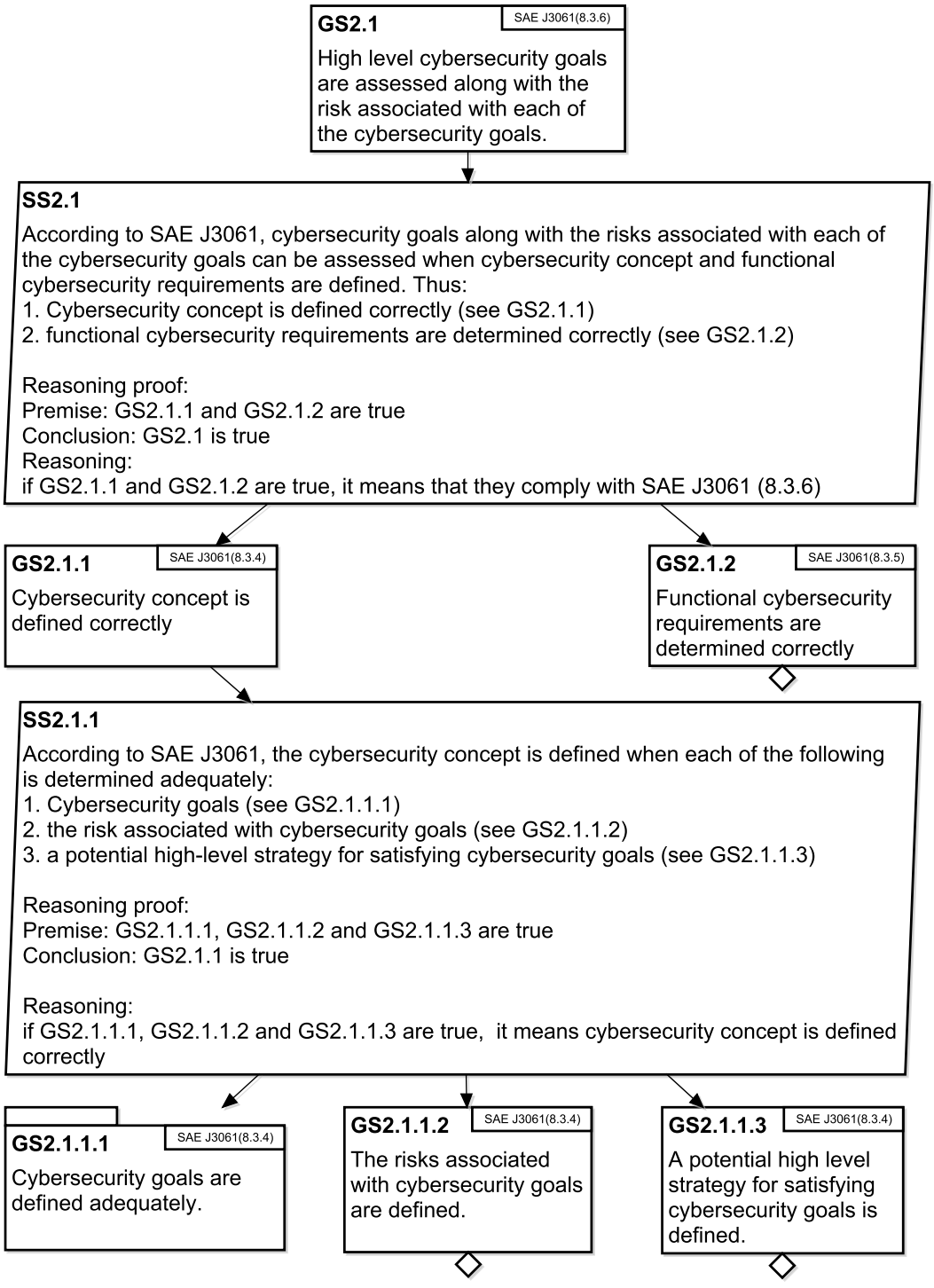


Figure 5.11: An excerpt of arguments supporting claim 'GS2.1' for cybersecurity

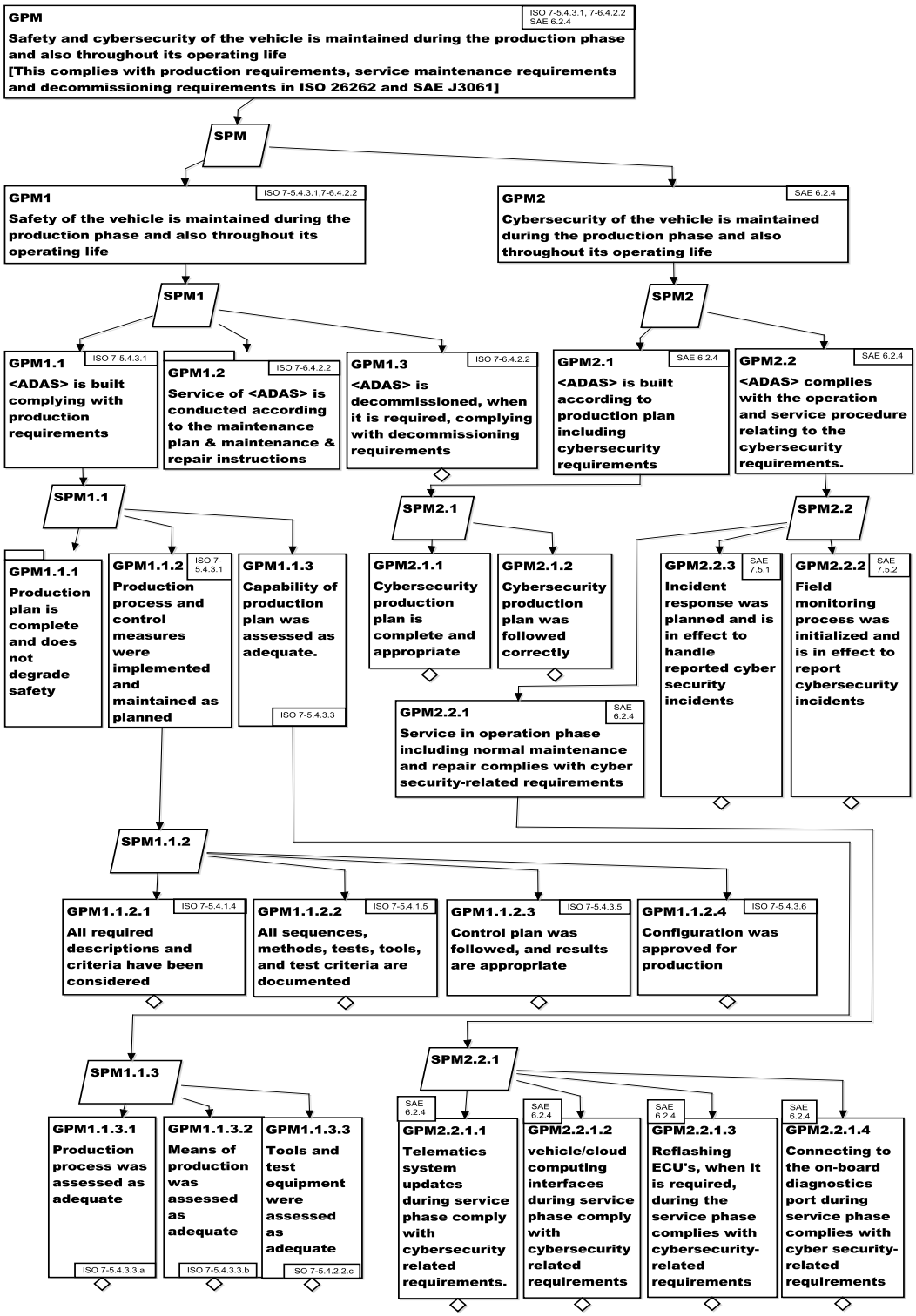


Figure 5.12: Extract from Assurance Case for Maintenance of Automotive Vehicles (GPM) [87].

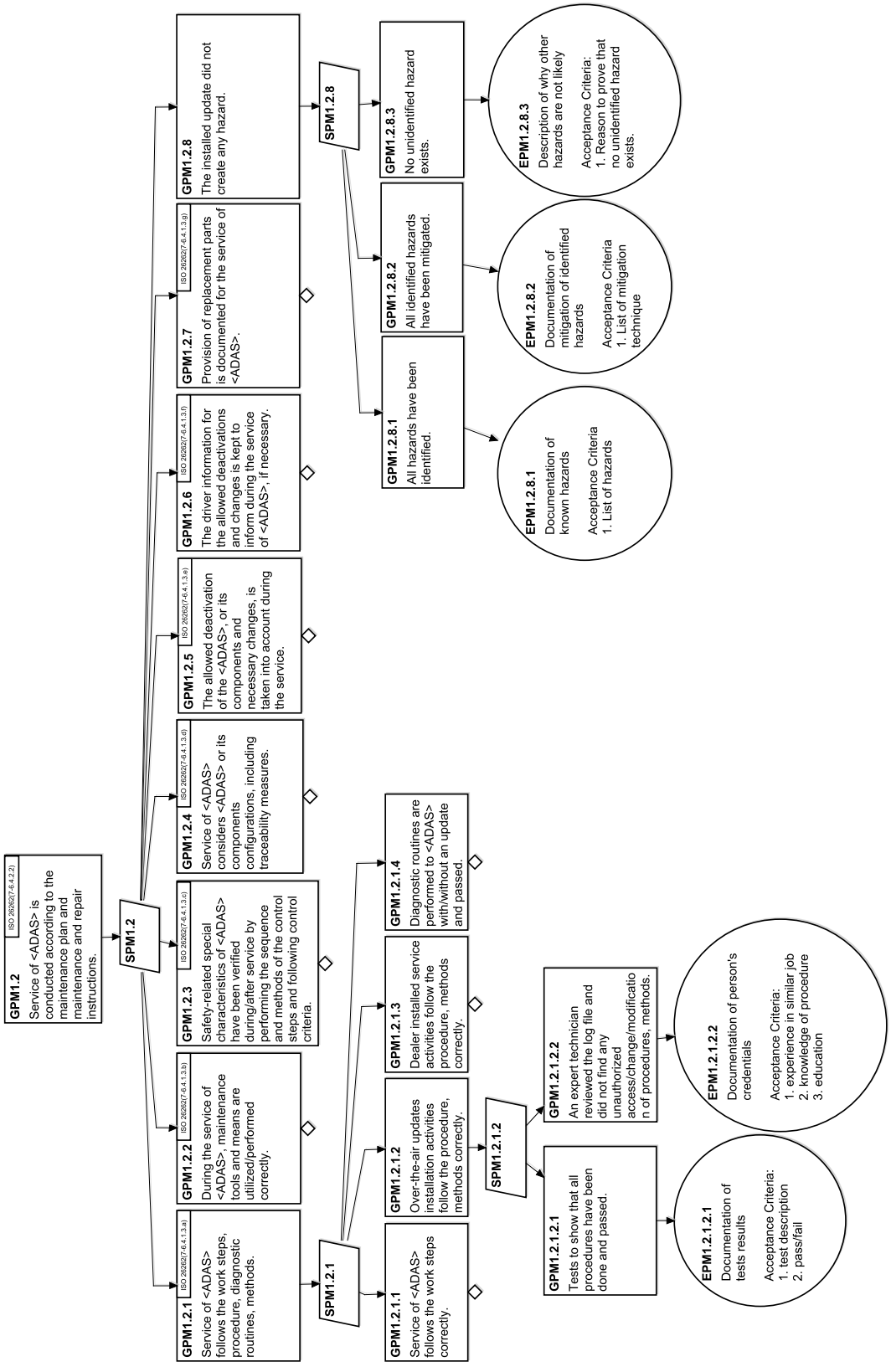


Figure 5.13: Excerpt of an ACT for assuring safety of OTA updated

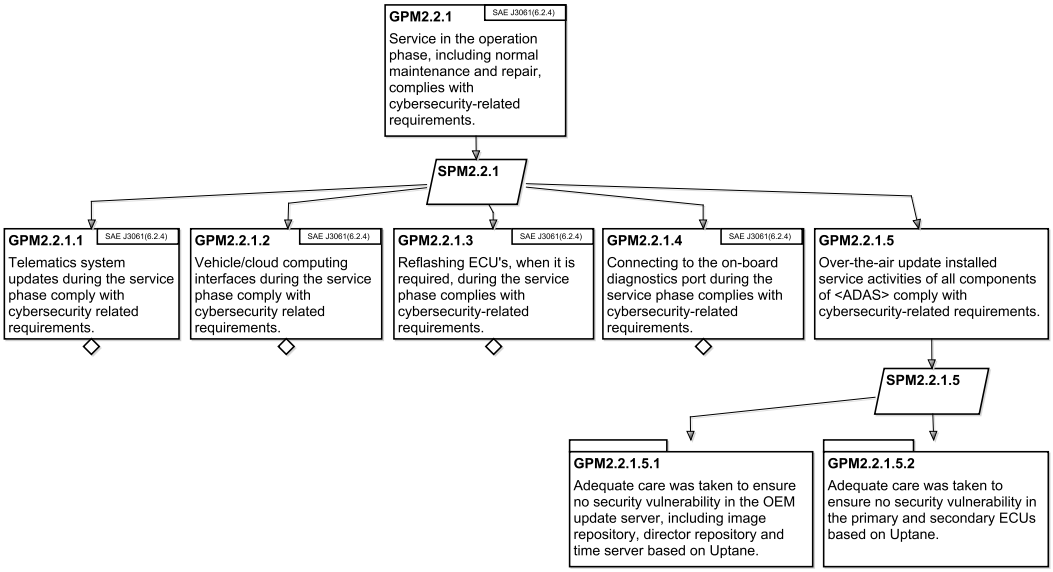


Figure 5.14: Excerpt of an ACT for assuring security of OTA updated

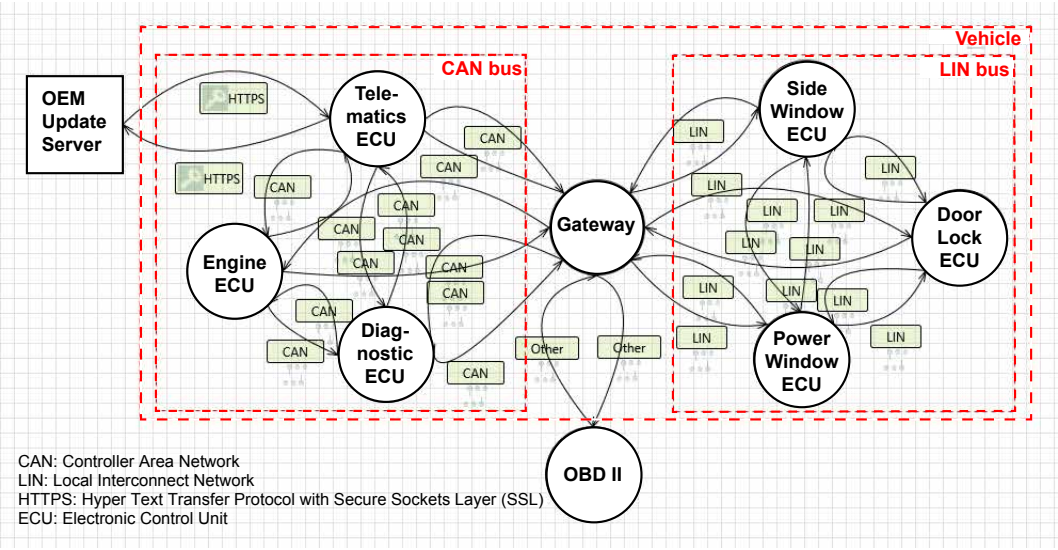


Figure 5.15: Data Flow Diagram of a partial vehicle network based on Uptane [23] [87]

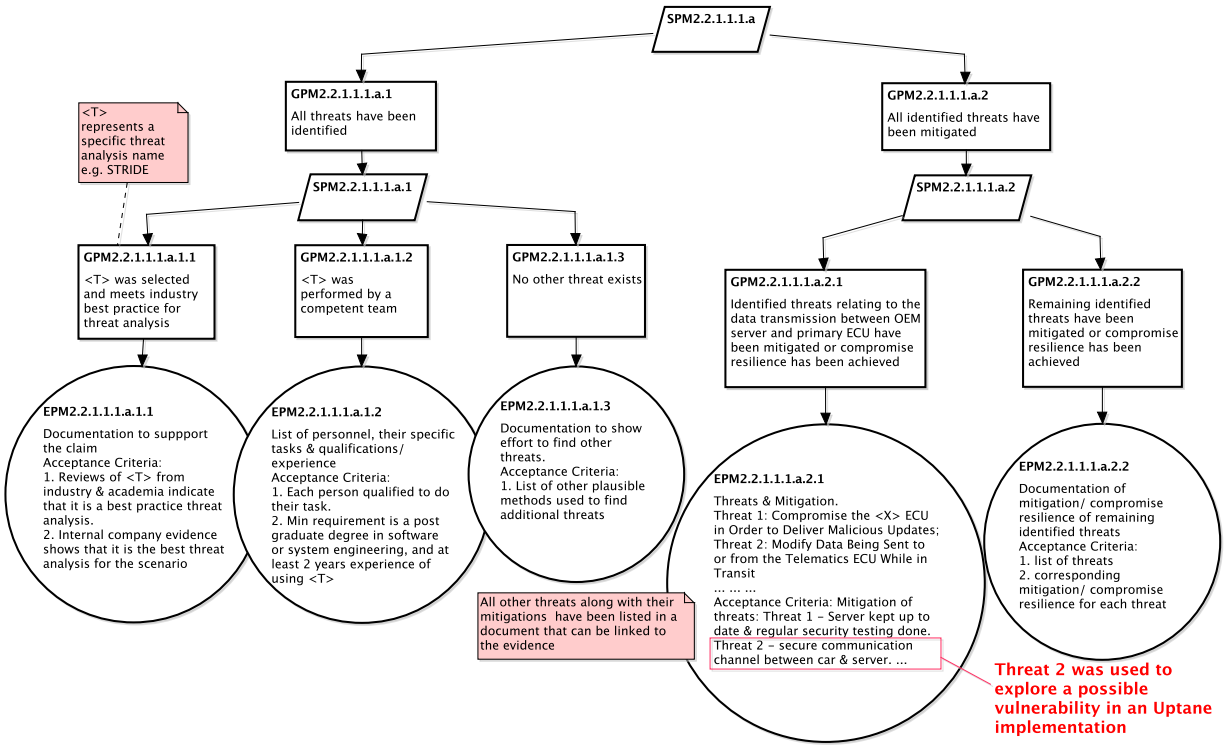


Figure 5.16: Slice of safe & secure ACT for OTA Updates [87].

Chapter 6

Criteria for Evaluation

This chapter demonstrates the criteria to evaluate ACs. The criteria are categorized into two groups: the structure and content of an AC. Furthermore, we highlight two perspectives of performing evaluation of ACs: developers and external reviewers. This chapter extends work published in [88].

6.1 Criteria for evaluation of ACs

Evaluation of an AC is used to identify weaknesses in reasoning or evidence. For example, criteria to evaluate the structure and content of an AC will improve understanding of weaknesses or considerations left implicit in an argument, and provide guidance to make them explicit and more compelling. Criteria systematically guide the evaluation process to discover incorrect, incomplete or inconsistent argument structures. Criteria play a vital role in guiding the evaluation process. An AC has a defined structure and content. For a complete evaluation, content and structure criteria should be defined separately to perform systematic evaluation. There are two perspectives from which to view AC evaluation:

- the Developer perspective: which includes system developers, software engineers, AC developers, etc.
- an External perspective: which includes external reviewers from regulatory and other third party organizations.

As mentioned before, we have divided criteria into two sets: Criteria in the first set are designed to be used in evaluating the structure/notation of an AC. Criteria in the second set are to be used in evaluating the content of an AC. In both cases we discuss each criterion under the headings of "Overview", "rationale" and then describe additional detail regarding the "Developer's perspective" and "External Reviewer's perspective". The developer of an AC has several tasks:

1. to document honest and explicit reasoning as to why the system is safe (or not);

2. to provide guidance to the developers of the system that demonstrates what they need to do in order to build a safe system;

3. and to continuously question the validity of the AC as it is developed.

In general, the developer of the AC will have a more detailed knowledge of the AC than any future external reviewer. Developers have an additional advantage as far as evaluation is concerned. They are able to apply "pairwise comparison" for specific evaluation criteria. This occurs quite naturally in developing the product, and it seems obvious to apply it during evaluation. In some cases, developers create more than one AC for a product. They may also have an assurance case template for their product line, which can be used as a basis for argument comparison. This allows the development of a "good" argument in an AC. A quantitative assessment may not be a right candidate, for several reasons:

1. To perform a quantitative assessment, in some cases, the probability of a basic element must be assigned which is subjective;

2. Similar to software development, the selection of metrics in an AC is not purely mathematical, which can result in a lower confidence quantitative assessment.

However, the external reviewer has the advantage of not being influenced by beliefs and views that the AC developer may share with the system development team. The external reviewer may also benefit from experience gained by reviewing ACs created by different development teams. In some cases, the

external reviewer will benefit from information provided by the AC developer specifically included aiding external review. Figure 6.1 shows a high-level view of the evaluation process. However, this high-level view is useful in understanding the big picture.

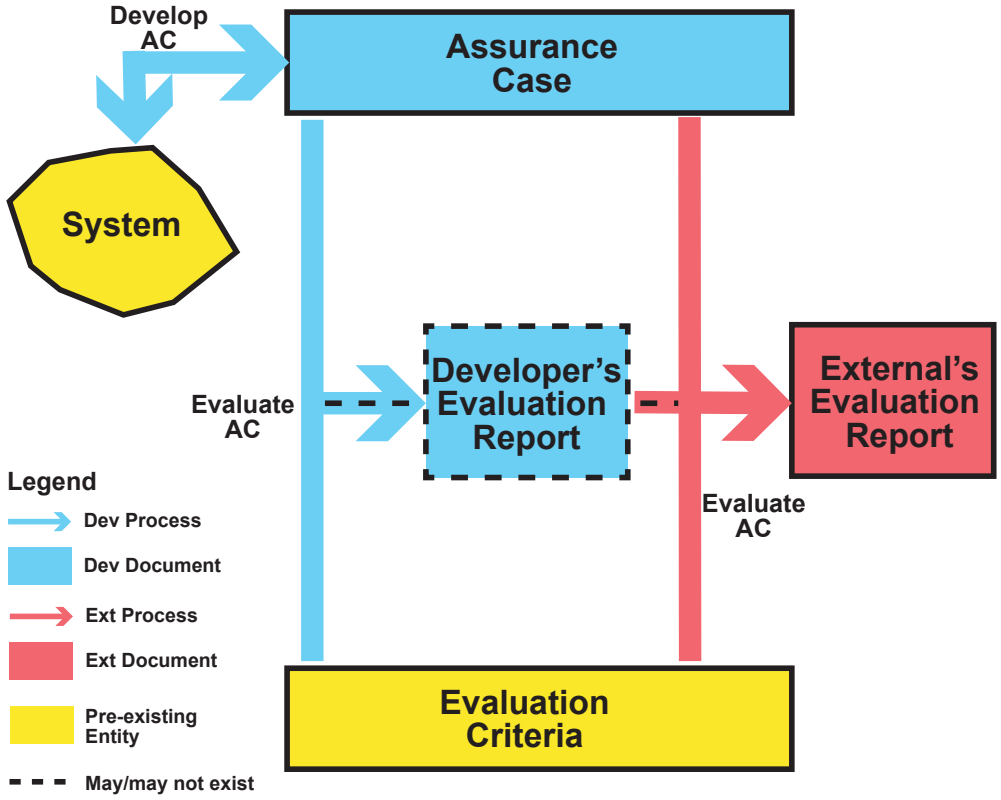


Figure 6.1: High Level View of the Evaluation Process

6.2 Criteria for evaluating structure/notation

AC can be structured in many ways and use different notations; the structure and notation used may have a profound impact on the effectiveness of the AC, particularly when tools that do not provide syntax-aware support for constructing the AC. The following criteria guide the evaluation of the structure and notation of an AC.

6.2.1 Syntax check

6.2.1.1 Overview

A syntax check can provide an indication of the quality of the AC. Aside from its effect on understandability, if there are syntax errors in the AC, it may be an indication of the care the developer took in its construction. For example, if the AC is presented using GSN, then there are expectations that specific shapes and connectors have been used in specific ways. This is independent of the content of those shapes. A GSN-specific tool will generally either only allow syntactically valid GSN to be produced, or will provide ways to check that syntactic rules have not been violated. Syntax errors are likely to indicate a degraded quality of care or competence of the developer. In general, tools are invaluable (even necessary) for performing syntax checks. However, they typically do produce both false-positives and false-negatives. Apart from bugs in tools, incorrect or incomplete requirements are responsible for producing unexpected results. False-positives are annoying. False-negatives result in not detecting syntax errors. Thus the AC still needs to be checked by humans. Syntax-aware tools (e.g. a GSN editor) need not be used; in some cases an AC may be presented in a textual format, or via a general-purpose tool such as Powerpoint. In such cases, it is still vital that the presentation be checked for consistent formatting, notation and structure.

6.2.1.2 Rationale

We realize that many (useful) ACs may not have a well-defined semantics. However, they really must have a well-defined syntax. Without that, developers of the AC and readers/users of the AC will have great difficulty in navigating and understanding the "case" presented – especially since industrial level ACs are likely to be huge.

6.2.1.3 Developer's perspective

The developer's task is eased considerably if the AC is developed using an AC tool. In this scenario, it should be trivial for the AC developer to perform a syntax check to ensure that the AC's syntax is correct. The developer can then also document this information for any external reviewers. Knowing that

the syntax is correct enables the developer to focus on the content of the AC. Conversely, lack of a tool will result in the AC developer having to check consistency and structure, etc. and thus not focus as much on content – or not worry about syntax and possibly deliver a document of much lower quality.

6.2.1.4 External reviewer's perspective

Explicitly checking the syntax of an AC is time-consuming at best, and perhaps too time-consuming if the external reviewer does not have suitable tools for that specific AC. If the developer of the AC has not documented checks done to protect against syntax errors, the external reviewer may rather just note syntax errors when coming across them. Too many errors may be a cause to doubt the quality of the AC. Syntax errors may also cause difficulties in understanding the AC well enough to evaluate the argument regarding the system's safety.

6.2.2 Traceability

6.2.2.1 Overview

Links between assurance and system development artefacts are of crucial importance in dealing with evidence that supports a claim [89]. No matter what notation we use, we need to use a cross-reference to an item in the system development to demonstrate convincingly that a claim is valid. The finer the granularity of this evidence, the more useful it is in understanding why it supports the claim. This is true when we create the initial AC. It is even a more significant advantage in dealing with the system's future maintenance and assurance. A significant concern with ACs is that after a design change, for instance, updating the AC to take into account that change without redoing the entire AC is extremely challenging. Incremental assurance after a change in the system, or in its environment, or the AC itself, requires that we perform an effective change impact analysis, even in the case of assuring emergent properties such as safety. Performing an adequate change impact analysis requires extensive and thorough traceability links between all the relevant artefacts.

6.2.2.2 Rationale

Thorough traceability between all assurance, system and environmental artefacts is absolutely essential for effective maintenance of the system and its AC.

6.2.2.3 Developer's perspective

Maintenance of an AC differs from the maintenance of the system itself. ACs are much more effective when developed (or widely developed) before the system development is started [90]. During system development, it may become necessary to make changes to the AC. So, we can see that AC maintenance takes place even during the initial development of the system. Thus, the AC developer will create, maintain and use the traceability links throughout the development of a system.

6.2.2.4 External reviewer's perspective

An external reviewer should evaluate how well the traceability of the various artefacts was achieved. The mechanisms the AC developer used for creating, maintaining and using the traceability links should be obvious to an external reviewer. The external reviewer's motivation for evaluating traceability is to gauge how easy it was for the AC developer to understand the complex interactions between artefacts.

6.2.3 Robustness

6.2.3.1 Overview

Kelly defines robustness of an AC as follows: "how fragile is the argument to possible changes in the evidence and consequent claims?" [55]. We assume that changes will be made, during the development of the initial AC, and after future changes in the associated product itself. We want the AC to be robust in the sense that changes should be as contained as possible. In other words, the effect of a change should be constrained to be as local as possible. The principle of information hiding from software design [91] could serve as an excellent model for creating and evaluating the robustness of an AC. For

example, it was suggested in [15] that GSN ACs will be more robust if the sub-claims likely to change are located "lower" in the GSN graph. A robust AC should facilitate incremental assurance. However, incremental assurance will require much more than robustness – extensive traceability, for instance.

6.2.3.2 Rationale

In general, an AC should be robust with respect to likely changes. Incremental assurance highly depends on the robustness of an AC. More robust AC leads to more feasible AC due to less effort for changeability. Incremental assurance guides the future development of a product in a particular product line. Robust AC can accelerate an incremental assurance process. However, it is not feasible to construct ACs so that they are robust with respect to all changes. However, robust with respect to likely changes are more tractable and may prove to be very effective.

6.2.3.3 Developer's perspective

The developer of the AC should have knowledge of the likely variability in the AC (and associated system). The developer should also take into account likely rebuttals of the current argument and alternatives in the evidence that could support terminal sub-claims in GSN-like ACs. The AC developer could then gauge the robustness of the AC by simulating the likely changes. In this way, the developer will be able to use robustness as an effective evaluation criterion for that AC.

6.2.3.4 External reviewer's perspective

If the AC developer's evaluation of robustness is well documented, the external reviewer should be able to conduct an audit and arrive at an independent evaluation of robustness. If the AC developer's documentation is lacking, the external reviewer has a much more difficult task. However, based on experience, the reviewer could simulate pertinent likely changes and ascertain how constrained the impact of those changes would be.

6.2.4 Understandability

6.2.4.1 Overview

"Understandability" may look like it is more appropriate to include in criteria for evaluation of content rather than in criteria for evaluation of structure/notation. Understandability with respect to content is dealt with in discussion related to arguments. This instance of understandability pertains to how the structure of the AC, its appearance and notation help or hinder understandability of the AC. Structure related to navigation through the AC may be especially important.

6.2.4.2 Rationale

Understanding an AC to serve the purpose is one of the critical elements in evaluation. Ambiguous AC does not provide adequate confidence to stakeholders about their product. "Understandability" criterion plays an active role in facilitating the comprehensibility of the AC for evaluation.

6.2.4.3 Developer's perspective

This is one criterion that is used entirely differently by developers of ACs compared with external reviewers. The developer creates the AC, and so bears the responsibility for ensuring that the AC is understandable by all stakeholders. This can involve:

- Structuring the AC so it is easy to navigate. For instance, if the AC uses GSN, *modules* should be used to encapsulate argument threads.
- Choosing font colours and sizes to enhance the AC's readability.
- Laying out graphical ACs (like GSN) in a consistent way so as not to mislead readers in understanding the structure of the argument. For instance, keeping subclaims of a parent claim on the same decomposition level in the graph.
- Using consistent assurance case patterns for similar arguments, to reduce cognitive load on reviewers or the assurance case, who may benefit from recurring structures in the ACs they must analyze.

6.2.4.4 External reviewer's perspective

A simple checklist like the one described above can be used by an external reviewer to evaluate how well the developer has managed to facilitate a deep understanding of the AC. This is so important that we support the idea of reviewers *rejecting* to evaluate an AC if it is not understandable because of the structure/notation/appearance.

6.2.5 Efficiency

6.2.5.1 Overview

Efficiency in this context refers to how the structure/notation of an AC affects the throughput and accuracy of people developing an AC, developing a system, and reviewing an AC.

6.2.5.2 Rationale

An AC provides valid reasoning for the critical properties of a system. Evaluation of an AC is time-consuming and requires extensive expertise. It is thus vital to ensure that the AC's structure/notation facilitates the ease with which experts can perform this evaluation. Efficiency guides the development and review of an AC through easily comprehensible structure and notation, which escalates a 'good' evaluation process.

6.2.5.3 Developer's perspective

There is a stark difference between developers of the AC compared with external reviewers of the AC. The developer of the AC needs to consider how the structure of the AC will impact the creation of the AC itself. For instance, GSN ACs may require extensive duplication because of cross-cutting concerns. In such a situation, the developer may decide to use patterns that can be instantiated automatically with appropriate tooling. If the AC is used to guide the development of a system, the AC developer may structure the AC to include items that will facilitate the collection of evidence during the development of a system.

6.2.5.4 External reviewer's perspective

The earlier criterion of *understandability* assumes that there is an effort to be made in understanding the AC. This criterion is really about whether the structure/notation of the AC impacts that effort in a positive or negative way. For example, in GSN ACs, there is leeway in where to include specific arguments. The argument that each hazard has been mitigated adequately may be described in a graphical GSN decomposition, or it could be implemented by having a claim that *all identified hazards have been mitigated*, and then evidence in the form of a work product that describes each hazard and how it has been mitigated, and why that mitigation is adequate. The latter option is more efficient in many ways, even for the external reviewer, as demonstrated in figure 6.2.

6.3 Criteria for evaluating content

Claims made in the argument of an AC must be understood by stakeholders within the context of the developed product and environment. As such, evaluating the content of an AC is of critical importance, as this involves making judgments about the argument in a particular context. We now present criteria to assist in such an evaluation.

6.3.1 Convincing basis for the AC

6.3.1.1 Overview

To determine whether or not there is a convincing basis for an AC, the reviewer should understand and review the explicit argument presented in an AC; understand and review the explicit relevant assumptions about the product, its environment and the efficacy and quality of its development process; and understand and review the context in which the product will be deployed. There are three significant steps in this process: understand and review the "top-level" claim of the AC; understand and review the detailed argument, including how sub-claims support parent claims, and how evidence supports terminal claims; and an explicit check that the review is not affected by "confirmation bias" [46].

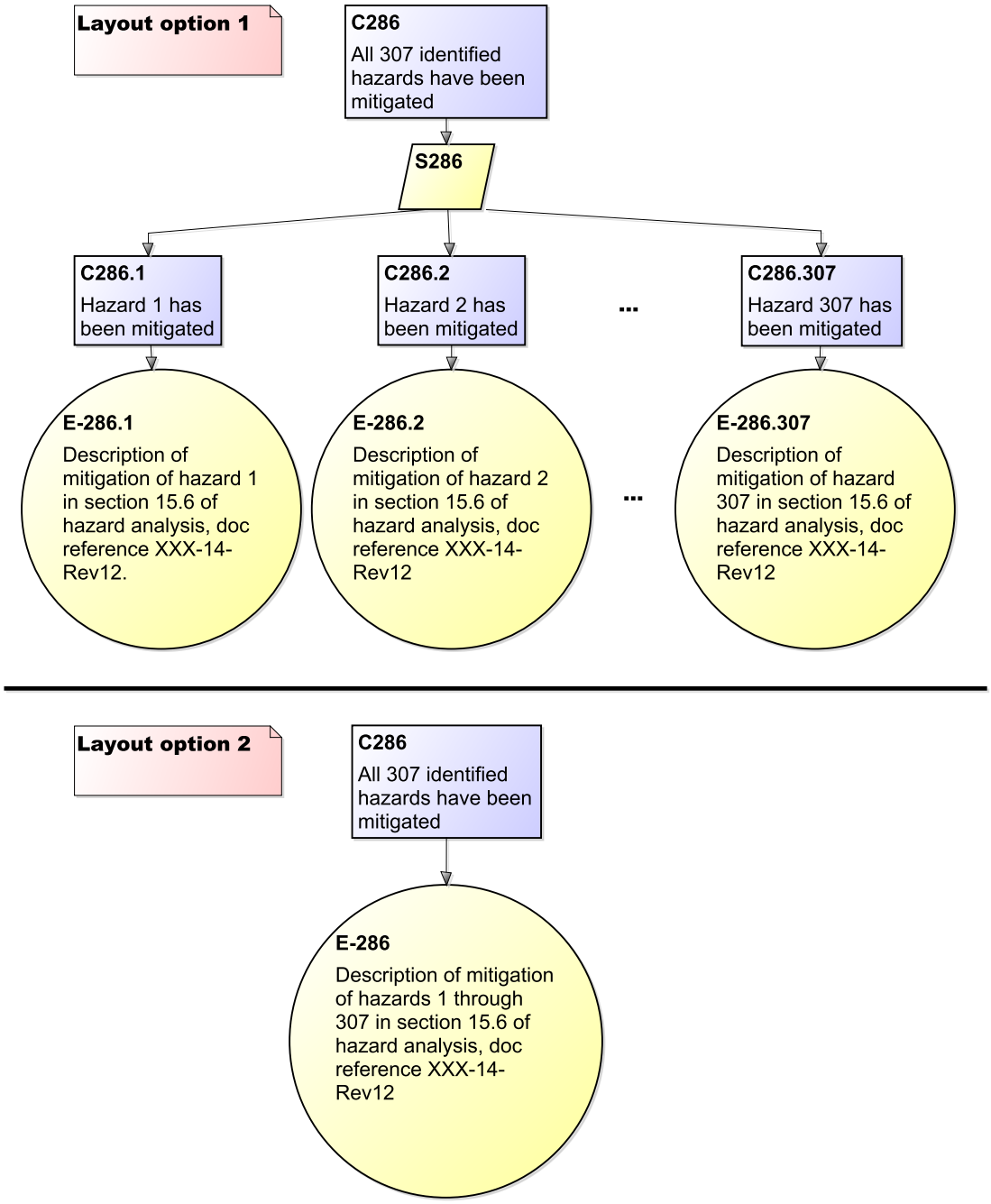


Figure 6.2: Example in GSN Illustrating Differences in Efficiency

- *Top-level claim:* In general, the top-level claim of an AC describes a claim about critical properties of a system under consideration. The top-level claim must be supported by sufficient context for readers to understand how the system fits within its environment and necessary assumptions for the reader to understand the boundaries and limitations of the argument. The top-level claim must also be achievable, given the stated context and assumptions. For example, a top-level claim of "System X is safe" is not feasible.

- *Explicit reasoning for the arguments:* The argument that supports the top-level claim should be explicit. This criterion is not about the validity or the rigour of the argument. Those are dealt with in other criteria within this section. This criterion is simply about whether or not the reasoning for the argument is explicit. It may be described in natural language, a logic of some kind, or a combination of these. The important point is that there needs to be some reasoning for every (sub)claim that shows why, if its premises are true, then the parent claim is true. The premises may be a combination of sub-claims and evidence. Sadly this is often not the case in GSN-like ACs. Therein, the decomposition of claims is explicit, but the reasoning is often left implicit.

- *Avoiding "confirmation bias":* This is another challenge in ACs highlighted by Leveson [46]. A simple example is when people look for specific evidence that supports a claim without considering counter-evidence. Discovery of any confirmation bias clearly degrades confidence in an AC.

6.3.1.2 Rationale

A feasible top-level claim is essential as it is the starting point of assuring critical properties of a system, and it also initiates the assurance building process. Moreover, reasoning needs to be explicit so that it can be reviewed. Without an explicit argument, it always depends on the expertise of reviewers, which is subjective in nature. Confirmation bias needs to be guarded against explicitly. It can manifest itself in various ways in the reasoning steps. When developers provide explicit reasoning, confirmation bias affects reasoning involving claims as premises as well as evidence as support for a terminal claim. Even when

explicit reasoning is not provided, confirmation bias may affect the validity of evidence.

6.3.1.3 Developer's perspective

Developers should provide explicit argumentation for review, and they need to review it themselves. This means that: i) they need to challenge their reasoning by explicitly documenting rebuttals and why the rebuttals are defeated, and ii) they need to list acceptance criteria for evidence before the evidence is generated (see discussion related to assurance case templates in [15]) and document why each item of evidence meets its acceptance criteria.

6.3.1.4 External reviewer's perspective

This is a difficult criterion for external reviewers to evaluate. If the developer does not document the items suggested in i) and ii) above, an external reviewer has to consider each step in the argument from the point of view of confirmation bias. This can be onerous. If the AC developer does document i) and ii), then the external reviewer is in a position to evaluate confirmation bias by reviewing that documentation. This should prove to be both easier and potentially more accurate since the developer is in a position to describe any steps taken to avoid confirmation bias. Otherwise, the external reviewer has to guess at the developer's approach and frame of mind.

6.3.2 Rigour of the argument

6.3.2.1 Overview

One of the main characteristics of an AC is explicit argumentation. An additional evaluation criterion is the rigour of the argument. We should expect that the reasoning steps are documented in a semi-formal way, or at least through rigorous application of an argument pattern. Rushby [92], [7] argues that the reasoning involving sub-claims as premises can be performed deductively, while the reasoning that evidence supports a sub-claim is done inductively, and that it then may be able to check many AC arguments mechanically. It may be controversial to consider documenting a formal argument in current

ACs, but the point is that utterly informal reasoning in these complex systems is subject to errors in which we miss ambiguities and logical fallacies. Even if we cannot achieve formality, making these arguments more rigorous invites a greater focus on the reasoning. Rigorous application of reasoning patterns in natural language may also effectively improve the quality and thoroughness of the reasoning.

6.3.2.2 Rationale

It is crucial that we trust the reasoning in the AC. Ad hoc, natural language reasoning, while better than no reasoning at all, is not adequate to protect against errors introduced merely through lack of appropriate precision and rigour. Rigour is vital in making the reasoning less subjective and more repeatable. Less subjectivity in reasoning reduces the possibility of having fewer fallacies. On the other hand, repeatability of arguments escalates possibilities of automation in instantiation using tool support.

6.3.2.3 Developer's perspective

The developer bears the responsibility of including rigorous reasoning in each step of the argument. For example, in GSN if a goal (claim) A is decomposed into subgoals (sub-claims) B and C, then the Strategy documents how the goal A was decomposed into subgoals B and C. In addition, as discussed above, there needs to be reasonably rigorous reasoning that demonstrates that if B and C are both valid, then A will be valid. The existence of this reasoning and the extent to which it can be considered to be *rigorous* should be evident.

6.3.2.4 External reviewer's perspective

This is one case in which the external reviewer's task is relatively easy. The reviewer should not have to provide an argument. Evaluation of this criterion involves only a judgement of what was achieved by the AC developer. We note that it is not necessarily true that an utterly formal approach should be evaluated to be better than a semi-formal approach, or one that uses natural language aided by argument patterns. For instance, a completely formal approach may need to make unrealistic assumptions.

6.3.3 Quality of the hazard analysis

6.3.3.1 Overview

This is one of the criteria that applies to safety case evaluation in particular. Adequate mitigation of all known hazards is a prerequisite for system safety. In addition, we need to know with reasonable certainty that there are not likely to be additional hazards that we have not considered. This criterion is so important that many people have made it the primary focus of a safety AC. While we disagree with that as a way of structuring ACs, we do agree that it is of vital importance. There are a number of ways in which we can gain confidence that "all" hazards have been identified, which is a necessary precursor to all hazards have been mitigated. Details are included below.

6.3.3.2 Rationale

In the case of hazard identification and mitigation, we believe that there is no alternative but that the AC developer must document the effort that went into identifying hazards, as well as sufficient detail regarding mitigation of those hazards. Again, we believe that the basic plan for hazard identification and mitigation should start with the development of the AC, and that specific detail has to be provided from system development documents.

6.3.3.3 Developer's perspective

The AC developer needs to be able to include the following, before development of the system starts:

- Specify claims and evidence acceptance criteria related to what hazard analysis method(s) are acceptable. For instance, applicability of the hazard analysis technique in that domain; published reports on how effective the method has proven to be in practice; technical publications on the soundness of the method, etc.

- Specify claims and evidence acceptance criteria related to the minimum experience/educational requirements of people tasked with the hazard analysis.

- Specify claims and evidence acceptance criteria related to a comparison with known hazards. For instance, in many domains, a list of hazards is already know, especially if the domain is regulated. For example, the U.S. Food and Drug Administration (FDA) has identified infusion pump system hazards [5] and recommended mitigation strategies. Moreover, the FDA provides examples of hazards and their causes under headings such as: operational, environmental, electrical, hardware, software, etc.

- Specify claims and evidence acceptance criteria related to what effort must be expended in exploring whether there are additional hazards.

- Specify claims and evidence acceptance criteria related to what checks must be performed on mitigation of hazards. Evidence derived from the development of the system must satisfy the relevant acceptance criteria and then be included in the AC documentation.

6.3.3.4 External reviewer's perspective

The external reviewer's task concerning this criterion is to review/audit the specific material in the AC. In this case, it is mandatory that such material is presented and argued thoroughly. The external reviewer may use the experience to supplement items in the AC. For instance, a knowledgeable reviewer in a regulated domain with years of experience may be aware of hazards not listed in the AC. This would naturally reduce the reviewer's trust in the AC.

6.3.4 Arguing completeness

6.3.4.1 Overview

We already discussed the quality of the hazard analysis as an evaluation criterion. One of the aspects of this criterion that we had to consider is the claim that "all" hazards have been identified. This is an example of a completeness claim that cannot be "proved". There are typically many such claims in an AC. Examples extend from modelling the system (did we include all relevant system inputs and outputs) to implementing behaviour that copes with all possible combinations of inputs. These same problems arise again when we document the AC for that system.

6.3.4.2 Rationale

"Completeness" is an elusive property in system and software engineering. We have numerous opportunities to err during system development simply because we did not try hard enough to cope with a completeness requirement. Deficiencies in completeness are a common source of error. We suggest that there are consistent approaches to dealing with completeness.

6.3.4.3 Developer's perspective

There is no single way of dealing with all the different completeness arguments. The developer of the AC is in an excellent position to decide on an approach depending on the specific situation at hand, and then guiding system development by including that approach in the AC. We need to emphasize that we know of no way of guaranteeing completeness in general. We see three different kinds of completeness, captured as follows:

- A claim related to a property for which the subdomain that includes all items with that property is unknown. This type of completeness, such as identification of all hazards, is extremely difficult to deal with. The important thing to note is that when we make such claims, we have to explicitly show how we have justified using the word "all". We illustrated one way of justifying this type of claim in our discussion of hazard analyses. Another way of justifying such a claim is to list *rebuttals* of the claim and then document how these rebuttals can be overcome.

- A claim related to a property for which the subdomain that includes all items with that property is known, but so large as to make dealing with each item infeasible. An instance of this is well-known. Verification of large, complex systems through testing acknowledges that we cannot test every combination of inputs over all time steps. In this specific case years of research have resulted in methods by which we may substitute a less onerous claim that can adequately substitute for the original claim.

- A claim related to a property for which the subdomain that includes all items with that property is known, and dealing with each item is feasible. This is a claim for which we have developed many interesting

and successful arguments. For example, if we want to claim that we have a requirement described by a mathematical function, we can claim that we have specified complete and unambiguous behaviour of that function by using a tabular expression [93]. Well-formed tabular expressions have to be complete on the input domain, and also disjoint (unambiguous). Mathematical expressions for completeness and disjointness are known and can be used to check that the tabular expression is well-formed.

6.3.4.4 External reviewer's perspective

The external reviewer can use this criterion in the same way but is likely restricted to whether or not the AC developer has included sufficient information to reach a conclusion. If the AC developer has not included documentation to help with evaluating this criterion for any specific claim/argument, the external reviewer's safest option is to call into question the completeness of the specific claim/argument.

6.3.5 Repeated arguments

6.3.5.1 Overview

AC notations differ in how often we see similar arguments being used in a single AC. For example, in GSN it is quite common to see very similar arguments used in many parts of the AC. There are typically two different ways of achieving this. The first is through the equivalent of "cut-and-paste". The second way, especially predominant in GSN, is through the use of so-called "argument patterns" [94, 95]. A slightly different problem, but essentially similar, is the use of similar arguments in many different ACs. The methods of implementing them are the same, and argument patterns are much more preferable.

6.3.5.2 Rationale

When AC patterns or argument patterns are used, it is important to check that the patterns are used appropriately, and that the context and assumptions for the pattern are in complete agreement with the context and assumptions for

the position in which they are placed in the AC. It is a source of error if they are used where not completely appropriate.

6.3.5.3 Developer's perspective

The AC developer is clearly responsible for deciding how to cope with similar arguments in the AC. As usual, there is an onus on the AC developer to document how patterns are used, for example.

6.3.5.4 External reviewer's perspective

It may not always be evident to an external reviewer on how and where similar arguments have been implemented in an AC. The external reviewer needs to try and find these "repetitive" arguments. The reason is that the external reviewer is unlikely to check absolutely everything in a specific AC, especially if that reviewer is a regulator. It is too much to expect that external reviewers, in general, spend as much or more time on the AC as compared with the developer. This means that the external reviewer may not find every occurrence of the use of a pattern, and may then miss where a pattern has been used inappropriately. This is slightly different for missing where an AC developer has made an error in an argument, since the AC developer has likely not even reviewed the local argument after instantiating a pattern.

6.3.6 ALARP

6.3.6.1 Overview

This is the other criterion specific to safety. As Low As Reasonably Practicable (ALARP) has a number of counterparts depending on domain and country. They are all quite similar, and while often viewed as a cost-benefit analysis, it requires that we provide adequate confidence about safety based risk assessment in a software-intensive system. This principle demonstrates that the impact of risks associated with unmitigated hazards is low. To provide confidence to stakeholders the ALARP principle should be incorporated into safety ACs.

6.3.6.2 Rationale

ALARP and associated principles are essential in demonstrating cost-benefit considerations and due diligence.

6.3.6.3 Developer's perspective

The AC developer must explicitly deal with ALARP. One excellent way of doing that is to use an argument pattern expressly designed and widely reviewed that demonstrates ALARP for the specific AC. One such pattern was developed by Kelly [94], but more current patterns may exist.

6.3.6.4 External reviewer's perspective

One of the key intentions of manufacturing safety critical systems is to build a safe system. Any unmitigated risk may jeopardize this purpose. Nobody can guarantee complete safety. So, demonstrating that the risk is as low as reasonably practicable (ALARP) is necessary. The external reviewer should look expressly for demonstration of ALARP and review the presented argument.

6.3.7 Confidence

6.3.7.1 Overview

Confidence in AC terminology refers to the trust we have in the overall argument presented by the AC. There is considerable published literature on confidence in ACs. Different researchers have developed different confidence assessment frameworks. Most of the confidence assessment frameworks use Bayesian Belief Network, Dempster-Shafer Theory, Beta distribution, or weighted average. Confidence related to the claims, reasoning and evidence must be verified.

6.3.7.2 Rationale

We do not attempt to reproduce or critique the confidence literature and results. Probabilistic nature and subjective assumption lead to implausible confidence measurements in some cases [96]. We simply reinforce that confidence in the argument is critical, and it is thus an important evaluation criterion.

6.3.7.3 Developer's perspective

In some cases, developers provide confidence assessment results. It is important that if they presented, the basic methodology used is explained.

6.3.7.4 External reviewer's perspective

In some cases, external reviewers review confidence assessment result. In other cases, external reviewers may try to ascertain confidence in the AC themselves. This is not feasible without adequate tool support built for this express purpose.

Chapter 7

Evaluation of Assurance Cases

This chapter discusses a systematic evaluation process to review ACs using the criteria described in Chapter 6. We illustrate a generic evaluation model and later refine and instantiate that model for each criterion (criteria for both structure and content of an AC evaluation). This chapter is based on work published in [97].

7.1 Evaluation process

This section presents details of our systematic evaluation of ACs. To put this on a well-structured footing, we start by modelling the evaluation process and its relevant data, including all primary components of an AC, as well as development artefacts from the system of interest in figure 7.1. The rationale of having a class 'ProcessX', data classes of assurance case artefacts, system artefacts and recommendation classes in a generic evaluation model is to show an explicit data flow between processes and assurance case and system artefacts. 'ProcessX' represents the execution of different processes in a refined and instantiated evaluation model of a criterion. The generic model contains 4 main components:

- The Process for evaluating the AC (represented by Green rectangles);
- The Recommendation arising from the evaluation (represented by Blue rectangles);

- AC data that is the subject of the evaluation (represented by Yellow rectangles);
- System development data that is referred to in the AC (represented by Orange rectangles);

Moreover, there is a component for assumptions, e.g. assumptions for process, assumptions for recommendations, criticality (represented by Grey rectangles). In addition to these, the generic evaluation model shows associations (represented by arrows/lines):

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

We illustrate four different components, along with associations among them. The definition of the proposed generic model starts with system development data. In this system development data, each artefact consists of a process description and product description. For instance, an attribute 'elicitDes:str' in the 'Requirements' artefact describes the elicitation process, and 'desc:str' describes requirements. Thus, system development data includes requirements with the elicitation process documentation (represented by class 'Requirements'), detailed design with processes documentation (represented by 'Design'), implementation process with work products documentation (represented by 'Implementation'), and maintenance work with change information documentation (represented by 'Maintenance'). Furthermore, a class 'People' represents a documentation of the qualification of competent people involved in system development and system operation. The principal input to an evaluation process is AC data. AC data represents different artefacts of an AC. An AC primarily consists of claims (represented by a class 'Claims'), arguments (represented by a class 'Arguments') and evidence (represented by a class 'Evidence'). However, claims can be of three types: top-level claim (represented by a class 'TopClaim'), sub-claims (represented by a class 'Subclaims') and terminal claims (represented by a class 'TerminalClaims'). Furthermore, an AC includes other artefacts. They are supporting terms (represented by a

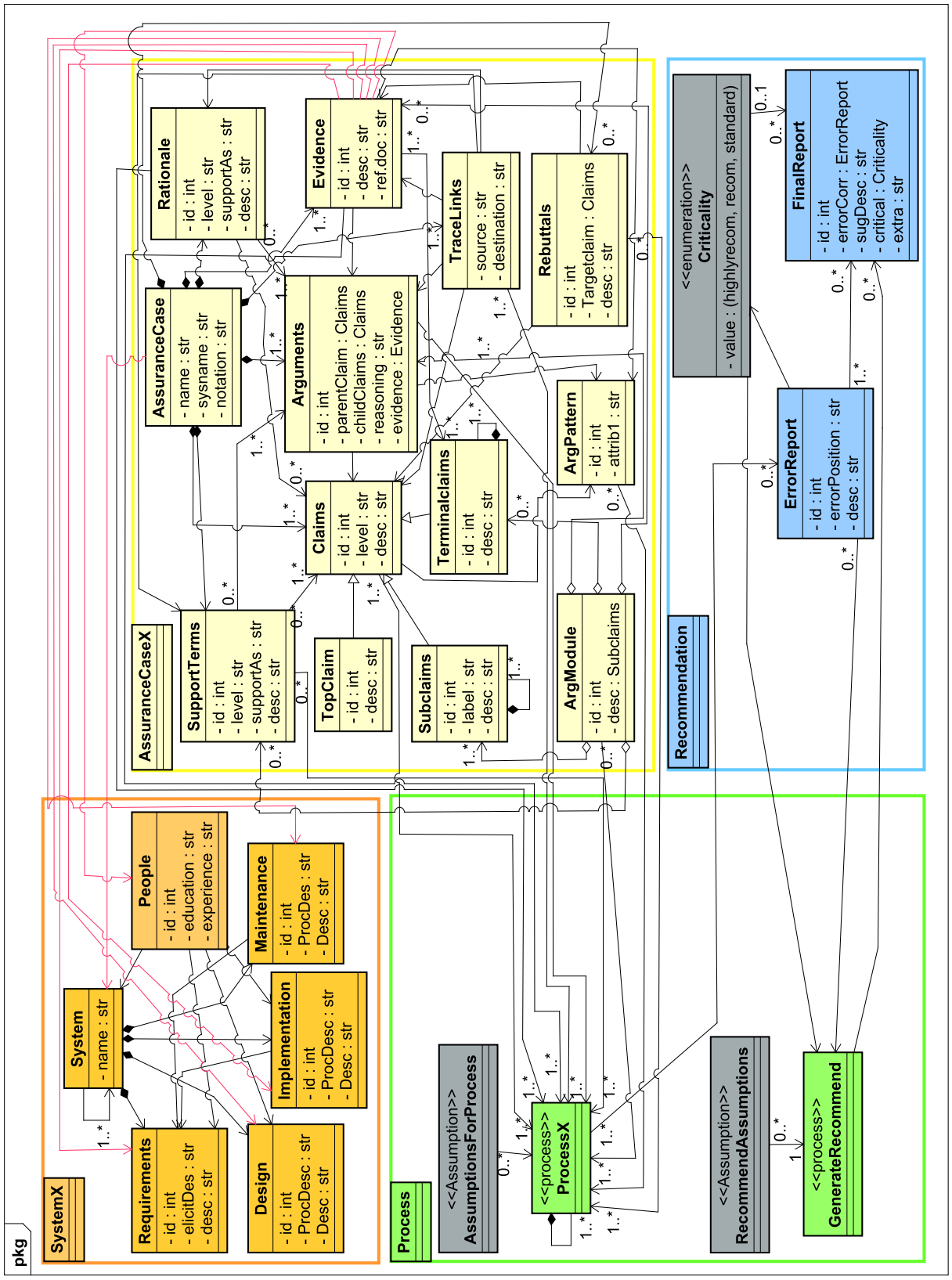


Figure 7.1: Generic Model of the AC Evaluation Process [97].

class 'SupportTerms'), e.g. context, assumptions in *GSN*, rationale, (represented by a class 'Rationale') e.g. justification in *GSN*, rebuttals (represented by a class 'Rebuttals'), trace links (represented by a class 'TraceLinks') illustrate associations between different artefacts of an AC, e.g. 'supportedBy', 'InContextOf' in *GSN*. Moreover, an AC may contain module (represented by a class 'ArgModule') to make it easier to comprehend. Long branches of arguments fit in one page due to modules. Besides, any argument branch of an AC may comply with a pattern (represented by a class 'ArgPattern'). 'ProcessX' represents different processes for the evaluation of an AC; each process takes different artefacts of an AC and checks them based on rules defined for that process and produces error output (represented by a class 'ErrorReport'). 'ProcessX' can be one process or more based on an evaluation criterion. For instance, in case of *GSN* syntax check, 'ProcessX' can be refined to 'CheckGraphSyntax' only, that checks *GSN* syntax of an AC. On the other hand, for traceability check, 'ProcessX' can be refined to three processes 'EvidenceToSystemTrace', 'RecentToPastVersionTrace' and 'ArgumentPatternTrace'. Another process for recommendations (represented by 'GenerateRecommend') takes error output (represented by a class 'ErrorReport') and generates recommendations (represented by a class 'Recommend') with criticality (represented by a class 'Criticality'). Necessary assumptions support each process, recommendations and the criticality (represented by classes 'AssumptionsForProcess', 'RecommendAssumptions' and 'Criticality' respectively). The 'criticality' is an enumerated class consisting of one or more criticality levels. Domain experts can define the criticality levels - highly recommended ('highlyrecom'), recommended ('recom') and no recommendation ('standard'). Experts may determine the criticality levels based on workshops, discussion, experience etc., thus it is subjective. Besides, qualitative assessment motivates us to define qualitative criticality levels. This criticality level is defined based on 'ErrorReport' and supports the recommendation process. For instance, in the case of *GSN* syntax check, fixing wrong shapes or wrong associations is highly recommended, formatting labels of nodes is recommended, and no further action is required (defined by 'standard') if there is no error. The attribute 'extra' in the 'FinalReport' class represents what developers may provide to external reviewers for facilitating evaluation.

The association represents an input dataflow between an assurance case artefact (source-highlighted in yellow colour) and 'ProcessX', 'ErrorReport' and 'GenerateRecommend', 'Criticality' and 'GenerateRecommend'. The association represents an output dataflow between 'ProcessX' and 'ErrorReport', 'GenerateRecommend' and 'FinalReport'. The association represents a compliance between 'Claims' - 'Arguments' - 'Evidence' and 'ArgPattern'. The rest of them represent support relations, e.g. evidence support terminal claims.

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. Refinement involves precisely modeling inputs and outputs of individual steps in an evaluation process. Instantiation involves adding textual descriptions for process stages, which can be checked for conformance with the model's components.

Section 7.2 shows refinement and instantiation for criteria to evaluate structure of an AC. Section 7.3 shows refinement and instantiation for criteria to evaluate content of an AC.

7.2 Evaluation of structure of an assurance case

This section describes how we refined the generic evaluation model in figure 7.1 for each of the evaluation criteria. We start with criteria related to the structure of an AC and illustrate the major steps in evaluating each structure criterion. We present a detailed evaluation process of five structure criteria: syntax check, traceability, robustness, understandability and efficiency.

7.2.1 Syntax check

The "Syntax check" is an early but essential stage of the evaluation process. Without valid syntax, an AC is unusable in more sophisticated evaluation stages. A syntax check can be performed with or without tool support. If a tool is used for syntax checking, experts should still briefly review the syntax of an AC to guard against tool failures.

7.2.1.1 *Refinement of the evaluation model*

Figure 7.2 shows relevant aspects of a refinement of the model in figure 7.1. 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.

The refinement shows that 'ProcessX' now consists of 1 step depending on a notation for an AC development. If a graphical notation is used, the process 'CheckGraphSyntax' will take place, and if a textual notation is used, the process 'CheckTextSyntax' will take place. It reviews the syntax of different nodes of an AC. Inputs to this process are all AC data items, e.g. 'AssuranceCase', 'Claims', 'Arguments', 'Evidence' etc. Output of this process merely is to the 'ErrorReport.' These links make it clear that the focus of this check is the syntax of an AC. Assumptions and criteria for this check are found in 'AssumptionForProcess.'

7.2.1.2 *Instantiated evaluation process*

We can now instantiate the model. We do this by describing rules for both graphical and textual syntax. Reviewers can select a graphical syntax check or textual syntax check process based on their specific AC notation. For instance, in our example we consider graphical syntax checking for ACs only since our example uses *GSN*.

- ***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;

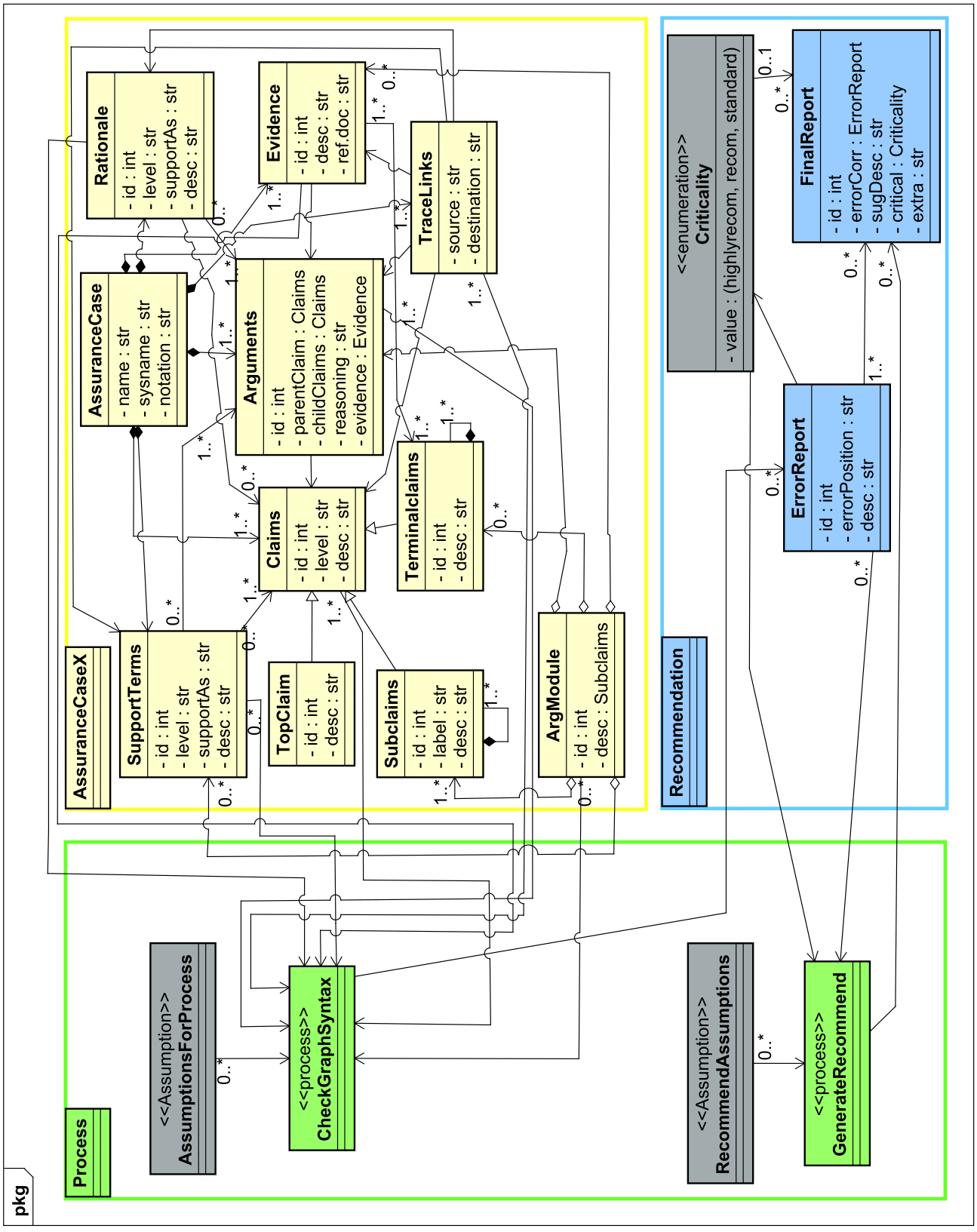


Figure 7.2: Evaluation Process for "Syntax check of an AC"

- ***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).

7.2.2 Traceability

Explicit and legitimate links between different artefacts in an assurance case play a vital role. These links, often in the form of cross-references, are in addition to the notational links used in the documentation of AC. For example, *GSN* links, claims, sub-claims, evidence as indirected arcs. However, cross-references can be added to evidence nodes to link to specific items of evidence. The rationale of having explicit and valid links is to facilitate the maintenance and robustness of ACs for future changes.

7.2.2.1 *Refinement of the evaluation model*

Figure 7.3 shows relevant aspects of a refinement of the model in figure 7.1. In this refined model, we took into account the documentation resulting from the development of the system, since that part of the model is of significant importance because of traceability among assurance case artefacts and system artefacts.

The refinement shows that 'ProcessX' now consists of 3 main steps (reading top to bottom) in figure 7.3:

- (Review) EvidenceToSystemTrace – a review of the traceability between evidence and system artefacts. Inputs to this process are the AC data

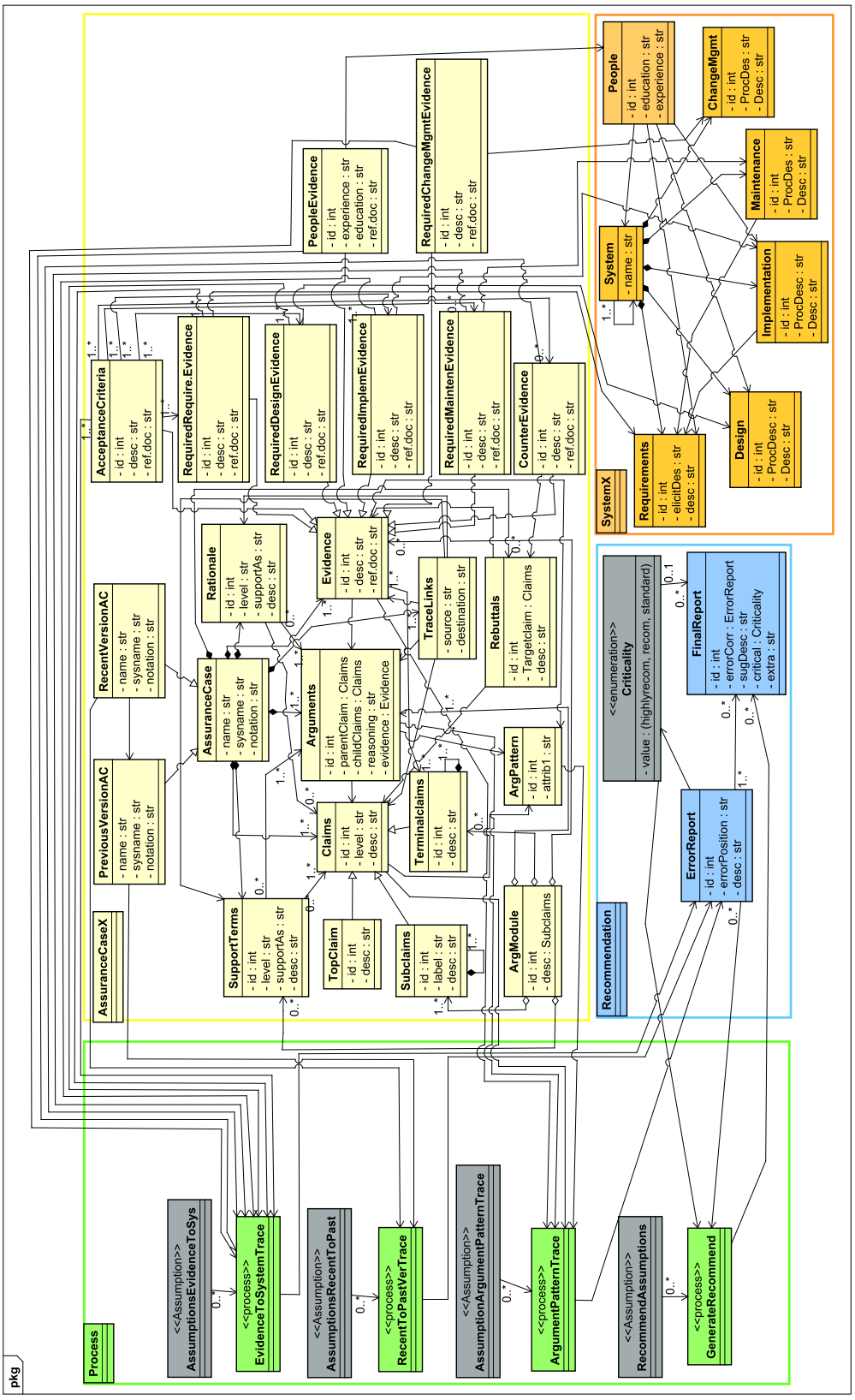


Figure 7.3: Evaluation Process for "Traceability"

items of the 'RequiredChangeMgmtEvidence', 'PeopleEvidence', 'CounterEvidence', 'RequiredMaintenEvidence', 'RequiredImplemEvidence', 'RequiredDesignEvidence', 'RequiredRequire.Evidence' and 'AcceptanceCriteria'. Output is simply to the 'ErrorReport'. These links make it reasonably clear that the explicit traceability exists among evidence and system artefacts. Assumptions and criteria for this check are found in 'AssumptionsEvidenceToSys'.

- (Review) RecentToPastVerTrace – a review of all traceability between artefacts of recent AC and artefacts of previous version of AC. Inputs to this process are 'RecentVersionAC' and 'PreviousVersionAC'. Output is again to the 'ErrorReport'. The focus of this check is on the explicit traceability between previous version of AC and recent version of AC. Assumptions and criteria for this check are to be found in 'AssumptionsRecentToPast'.

- (Review) ArgumentPatternTrace – a review that evaluates how claims, arguments and evidence trace to claims, arguments and evidence of an argument pattern. Inputs to this process are 'Claims', 'Arguments', 'Evidence' and 'ArgPattern'. Output is again to the 'ErrorReport'. Assumptions and criteria for this check are to be found in 'AssumptionArgumentPatternTrace'.

7.2.2.2 *Instantiated evaluation process*

We can now instantiate the model. We do this by describing rules for all processes to review traceability.

- ***(Review) EvidenceToSystemTrace:***

  (1) An explicit link or reference shall exist between evidence supporting a process related claim and a specific section of a document related to that process.

  (2) An explicit link or reference shall exist between evidence representing credentials of people for a process and a specific section of a document related to credentials of people involved in that process.

(3) An explicit link or reference shall exist between evidence supporting a product-related claim and a specific section of a document related to that product.

(4) An explicit link or reference shall exist between evidence supporting a claim related to the validation of a product/process and the description of validation of that product/process in a document.

(5) Evidence shall comply with acceptance criteria defined for that evidence.

(6) An explicit link or reference shall exist between counter-evidence (if it exists) and valid proof (deductive or inductive) defined in a document.

(7) An explicit link or reference shall exist between evidence supporting claims related to change management and a specific section illustrating change management of that system in a document.

- ***(Review) RecentToPastVerTrace:***

  (1) An explicit link or reference shall exist between the previous version of claims / arguments / evidence and the current version of claims/arguments/evidence.

  (2) An explicit link or reference shall exist between the previous version of terms/rationale supporting claims/arguments and the current version of terms/rationale.

- ***(Review) ArgumentPatternTrace:***

  (1) An instantiated claims/arguments/evidence shall comply with an argument pattern consisting of claims/arguments/evidence.

- ***GenerateRecommend:***

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

7.2.3 Robustness

Kelly defines robustness: "how fragile is the argument to possible changes in the evidence and consequent claims?" [55]. The possible changes should be as contained as possible to reduce their effect. During robustness evaluation, we take into account the principle of "information hiding" from software design [91]. We look for an argument that deals with changes and their effects on other argument branches. We also look for the position of the argument in an AC. For example possible changes in an argument branch at a lower level in a *GSN* tree are more robust than an argument at a higher level in the tree [15].

7.2.3.1 *Refinement of the evaluation model*

Figure 7.4 shows the relevant aspects of a refinement of the model in figure 7.1. We included the documentation resulting from the development of the system, since that part of the model is of significant importance as traceability among assurance case artefacts and system artefacts play a vital role in checking robustness.

The refinement shows that 'ProcessX' now consists of 3 main steps (reading top to bottom) in figure 7.4:

- (Review) SystemVariabilityInAC – a review of the robustness of an AC for likely system variations. Inputs to this process are 'Claims', 'Arguments', 'DifferentAC', 'RequiredChangeMgmtEvidence', 'PeopleEvidence', 'RequiredMaintenEvidence', 'RequiredImplemEvidence', 'RequiredDesignEvidence', 'RequiredRequire.Evidence', 'ArgModule', 'ACinConsideration'. Output is simply to the 'ErrorReport'. Assumptions and criteria for this check are found in 'AssumptionSysVariaInAC'. Assumptions for pairwise comparison are found in 'Assumpt.PairWiseComp.'.

- (Review) RebuttalsInAC – a review of the robustness of an AC for likely rebuttals. Inputs to this process are 'ACinConsideration', 'DifferentAC', 'Arguments', 'Claims','Rebuttals', 'CounterEvidence', 'ArgModule'. Output is again to the 'ErrorReport'. Assumptions and criteria for this check are to be found in 'AssumptionRebuttalsInAC'. There are assumptions for pairwise comparison are found in 'Assumpt.PairWiseComp.'.

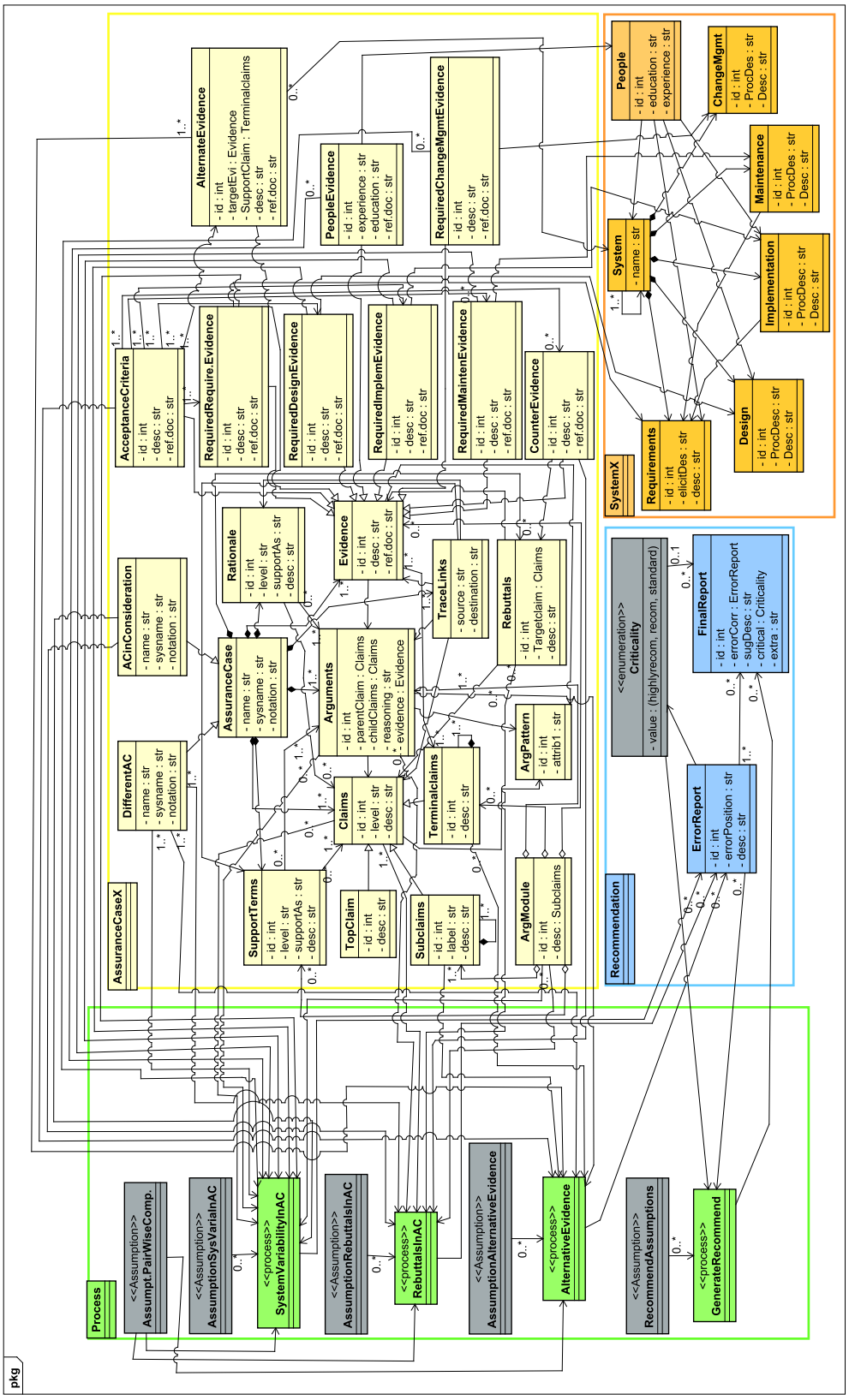


Figure 7.4: Evaluation Process for "Robustness"

- (Review) AlternativeEvidence – a review of the robustness of an AC for an alternative evidence. Inputs to this process are 'AcceptanceCriteria', 'Subclaims', 'AlternateEvidence', 'DifferentAC', 'Terminalclaims', 'Arguments'. Output is again to the 'ErrorReport'. Assumptions and criteria for this check are to be found in 'AssumptionAlternativeEvidence'. There are assumptions for pairwise comparison are found in 'Assumpt.PairWiseComp.'.

7.2.3.2 *Instantiated evaluation process*

We can now instantiate the model. We do this by describing rules for all processes to review robustness. Reviewers may use pairwise comparison to evaluate different schemes for robustness. The examples to compare may be created so that they are tangible, but this can prove to be too time-consuming. It may be sufficient to use "mental models" to effect such pairwise comparisons. It may also be possible to simply rely on experience in evaluating some of these robustness concerns.

- ***(Review) SystemVariabilityInAC:***

  (1) Pairwise comparisons are performed to identify lower and fewer artefact changes or changes in an independent argument branch due to a likely variability in a system.

- ***(Review) RebuttalsInAC:***

  (1) Pairwise comparisons are performed to identify lower and fewer artefact changes or changes in an independent argument branch due to a likely rebuttal.

- ***(Review) AlternativeEvidence:***

  (1) Alternative evidence shall comply with acceptance criteria.

  (2) Pairwise comparisons are performed to identify fewer artifacts changes in terminal claims due to alternative evidence.

- ***GenerateRecommend:***

(1) Based on identifying lower level with a few numbers of changes in an independent branch, a recommendation should be made with criticality (e.g. highly recommended, recommended, standard).

7.2.4 Understandability

The AC has to be understandable by its target stakeholders.

7.2.4.1 *Refinement of the evaluation model*

Figure 7.5 shows the relevant aspects of a refinement of the model in figure 7.1. 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.

The refinement shows that 'ProcessX' now consists of 4 main steps (reading top to bottom) in figure 7.5:

- (Review) FontAttributes – a review of all fonts used in an AC. The rationale of reviewing font attributes is to check the readability of AC. Inputs to this process are 'SupportTerms', 'ArgModule', 'Claims', 'Arguments', 'Evidence' and 'Rationale'. Output is simply to the 'ErrorReport'. These links make it reasonably clear that the focus of this check is the font attributes of an AC. Assumptions and criteria for this check are found in 'AssumptionsFontAttributes'.

- (Review) ModuleArgument – a review of modules used in an AC. The rationale of reviewing module argument is to check how modules are used to encapsulate argument threads for easy viewing. Inputs to this process are 'ModuleInterface', 'ArgModule'. Output is again to the 'ErrorReport'. The focus of this check is the creation of modules without making it less understandable. Assumptions and criteria for this check are found in 'AssumptionsModArgu'.

- (Review) IntersectAssociation – a review that evaluates whether any intersection among association arcs makes it less understandable. Thus, this process identifies whether any association arc intersects with another one and makes it less understandable. This process only applies to

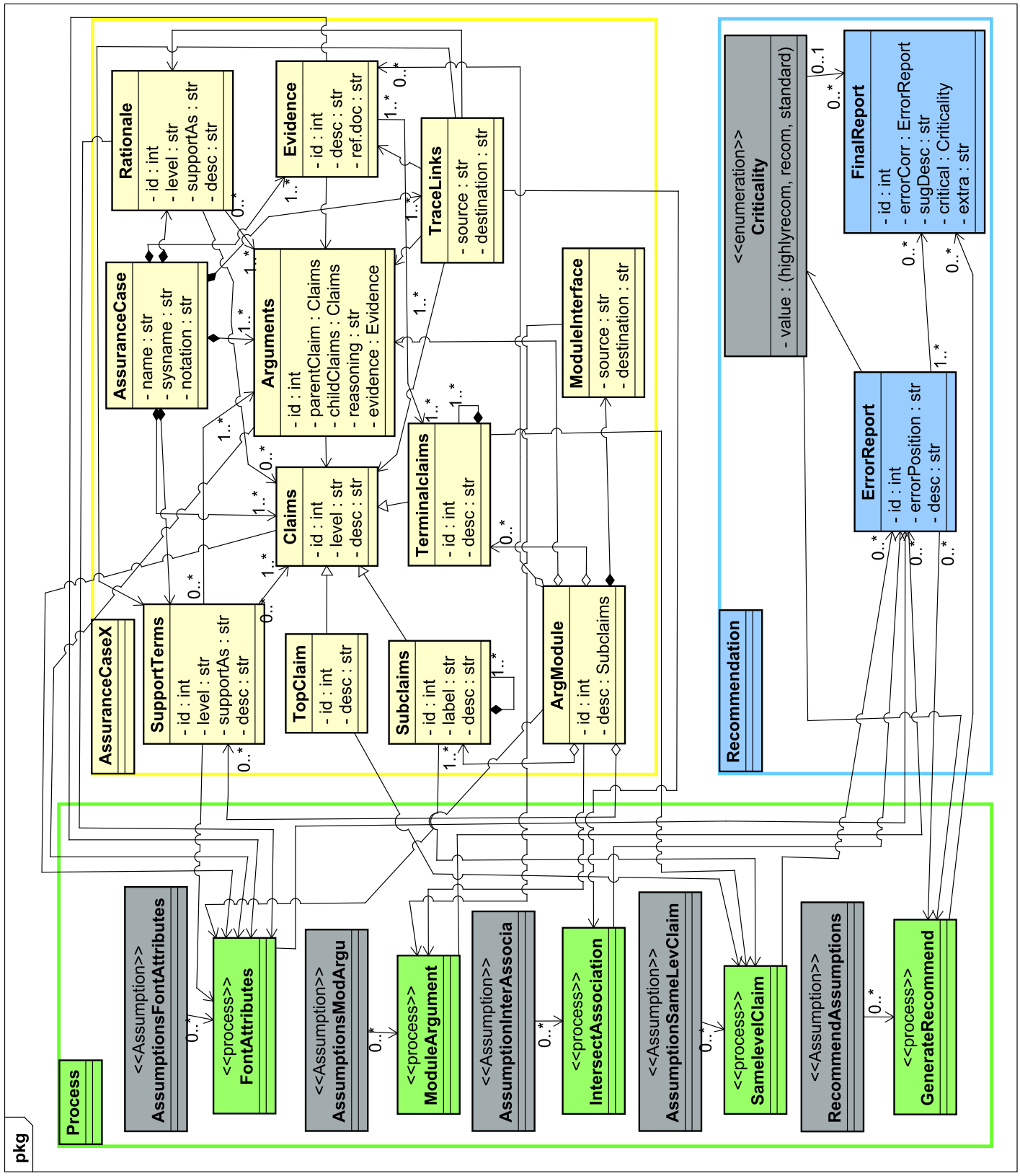


Figure 7.5: Evaluation Process for "Understandability check of an AC"

graphical notation. Inputs to this process is only 'TraceLinks'. Output is again to the 'ErrorReport'. Assumptions and criteria for this check are found in 'AssumptionInterAssocia'.

- (Review) SameLevelClaim – a review that evaluates whether claims stay in the same horizontal line, in general. The rationale of reviewing the same level claim is to check the layout of graphical ACs whether they mislead readers in understanding the structure of the arguments or not. Inputs to this process are 'TopClaim', 'Terminalclaims' and 'Subclaims'. Output is again to the 'ErrorReport'. Assumptions and criteria are found in 'AssumptionSameLevClaim'.

7.2.4.2 *Instantiated evaluation process*

We can now instantiate the model. We do this by describing rules for all processes to review understandability.

- ***(Review) FontAttributes:***

  (1) An acceptable font shall be used in an AC documentation.

  (2) Concerning graphical notation: Font size and colour of all nodes shall be appropriate for ease of viewing, both on-screen and in print.

  (3) Concerning textual notation: Font size and colour of description shall be appropriate for ease of viewing both on screen and in print.

- ***(Review) ModuleArgument:***

  (1) Creation of a module in an AC makes it easier to comprehend, and a module or a cohesive block of the AC shall fit in a single page both on screen and in print and shall be comprehensible.

- ***(Review) IntersectAssociation:***

  (1) Concerning graphical notation: the number of intersections of associations connecting different nodes should be reasonably low, e.g. that do not make an AC illegible. Fewer intersections improve understandability of the AC.

- ***(Review) SameLevelClaim:***

  (1) Concerning graphical notation: Claims of an AC shall be in the same horizontal position based on the level of decomposition.

  (2) Concerning textual notation: Description of any artefacts in an AC shall be in the same horizontal position applying indentation based on the level.

- ***GenerateRecommend:***

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

7.2.5 Efficiency

Efficiency in this context refers to how the structure/notation of the AC affects the throughput and accuracy of people developing the AC, developing the system, and reviewing the AC.

7.2.5.1 *Refinement of the evaluation model*

Figure 7.6 shows the relevant aspects of a refinement of the model in figure 7.1. We also included the documentation resulting from the development of the system, since that part of the model is of significant importance because of efficiency in system development.

The refinement shows that "ProcessX" now consists of 3 main steps (reading top to bottom) in figure 7.6:

- (Review) FacilitateSystemDevelopment – a review of facilitating a system development. The focus of this check is how an AC can provide the required information to develop a system. Inputs to this process are 'PeopleEvidence', 'V&VResultEvidence', 'RequiredMaintenEvidence', 'RequiredImplemEvidence', 'RequiredDesignEvidence', 'RequiredRequir.Evidence' and 'AcceptanceCriteria'. Assumptions and criteria for this check are found in 'AssumptionsFaciliSysDevelop'.

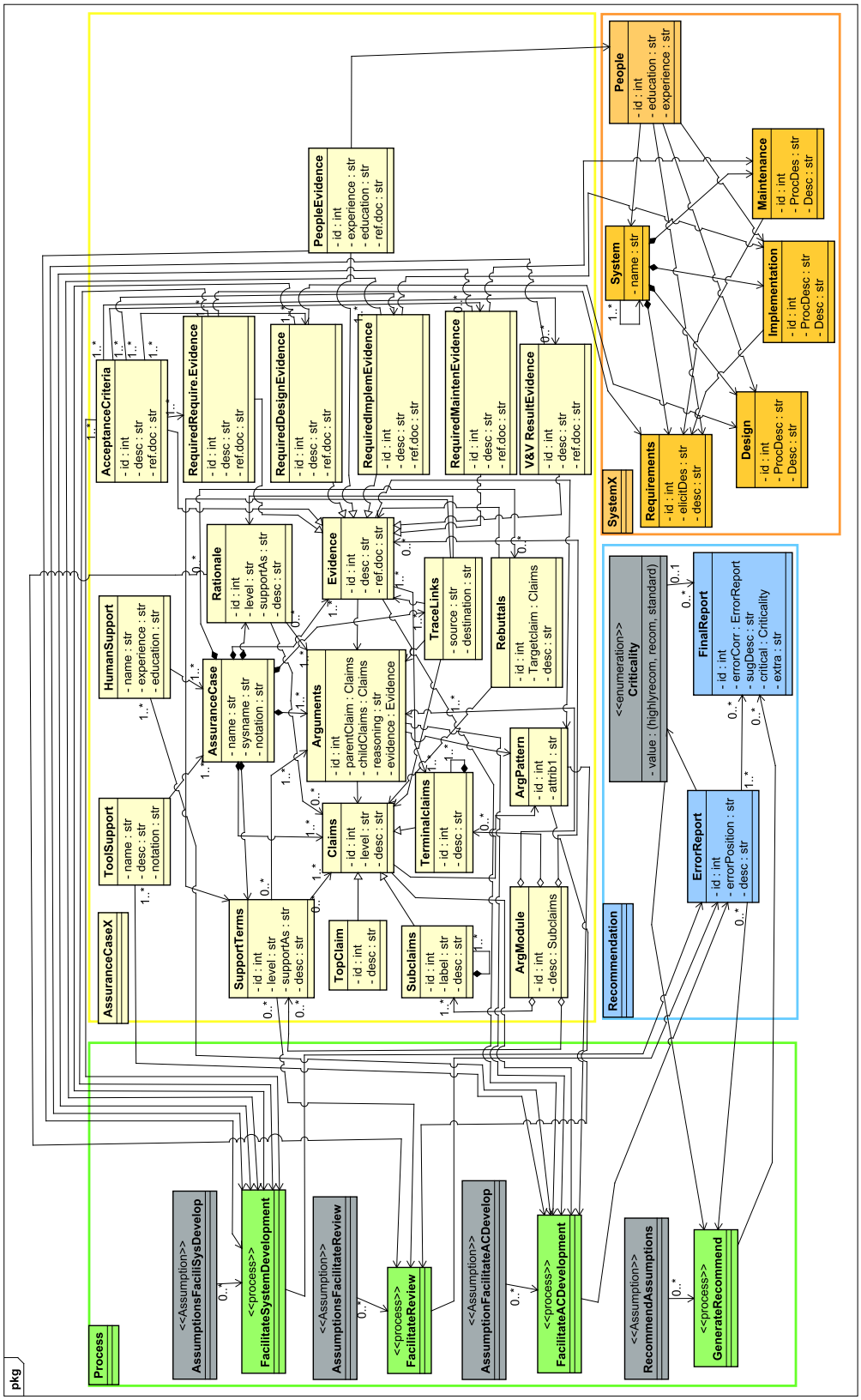


Figure 7.6: Evaluation Process for "Efficiency check of an AC"

- (Check) FacilitateReview – a review that evaluates how an AC can facilitate its review. This process only applies to graphical notation. Inputs to this process are 'Rationale', 'SupportTerms', and 'ArgPattern'. Output is again to the 'ErrorReport'. Assumptions and criteria for this check are found in 'AssumtionsFacilitateReview'.

- (Review) FacilitateACDevelopment – a review of facilitating an AC development. Inputs to this process are 'ToolSupport', 'HumanSupport', 'Evidence', 'Arguments', 'Claims' and 'ArgPattern'. Output is simply to the 'ErrorReport'. These links make it reasonably clear that the focus of this check is how it can facilitate an AC development. Assumptions and criteria for this check are found in 'AssumptionFacilitateACDevelop'.

7.2.5.2 *Instantiated evaluation process*

We can now instantiate the model. We do this by describing rules for all processes to review efficiency.

- ***(Review) FacilitateSystemDevelopment:***

  (1) Evidence complying with acceptance criteria guides system development (e.g. design decision, credentials of people to be involved in process development, validation and verification test) by indicating measures to enact.

  (2) Acceptance criteria help to define system development artefacts.

- ***(Check) FacilitateReview:***

  (1) An argument pattern in an assurance case shall be demonstrated with all nodes.

  (2) Rationale and context of using the recommended notation for developing an AC shall be demonstrated explicitly.

- ***(Review) FacilitateACDevelopment:***

  (1) An argument pattern in an AC shall be instantiated with tool support or require less human intervention.

(2) The same type of claims/argument/evidence shall be represented by a single claim/argument/evidence. In general, the same type of claims assure the same quality of a product.

- ***GenerateRecommend:***

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

7.3 Evaluation of content of an assurance case

Section 7.2 discusses the major steps in evaluating each structure criterion. This section presents a detailed evaluation process of seven content criteria by refining the generic evaluation model in figure 7.1: convincing basis, rigour of the argument, quality of the hazard analysis, arguing completeness, repeated arguments, ALARP, and confidence.

7.3.1 Convincing basis

To fully understand claims, arguments and evidence, they have to be explicit. Claims are explicitly shown as well as evidence, but sometimes the evidence is not precise enough. The arguments is often not explicitly presented in sufficient detail. One of the main intentions of *convincing basis* is to check the 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[4].

7.3.1.1 *Refinement of the evaluation model*

Figure 7.7 shows relevant aspects of a refinement of the model in figure 7.1. 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.

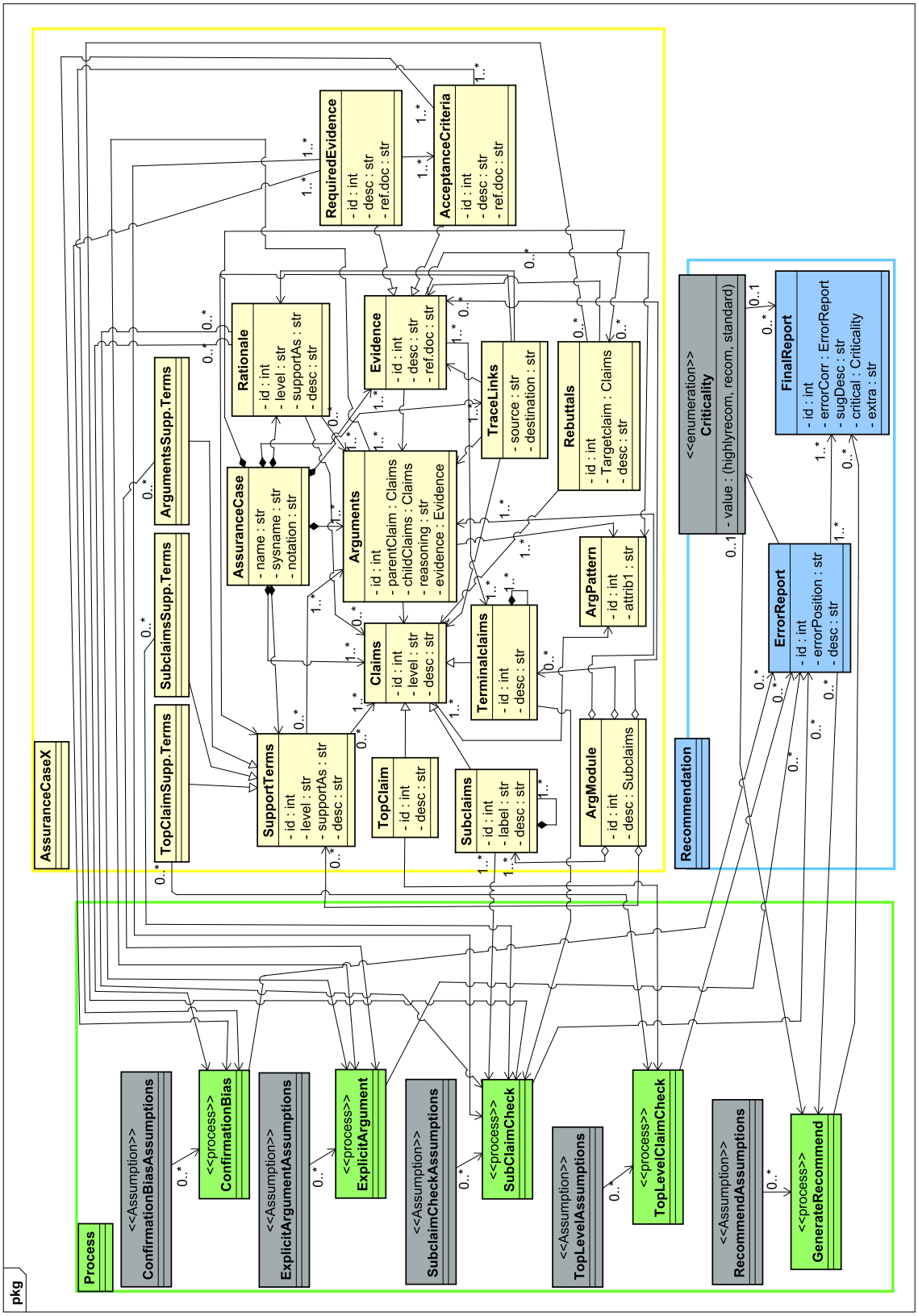


Figure 7.7: Evaluation Process for "Convincing basis for the AC" [97].

The refinement shows that 'ProcessX' now consists of 4 main steps (reading bottom to top) in figure 7.7:

- TopLevelClaimCheck – a review of the top level claim. Inputs to this process are the AC data items of the 'TopClaim' itself, and 'TopClaim-Supp.Terms'. Output is simply to the 'ErrorReport'. These links make it 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'.

- 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.

7.3.1.2 *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 provided.

- ***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 provided.

- ***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 deficiencies identified in an AC, a recommendation shall be made with appropriate criticality (e.g. highly recommended, recommended, standard).

7.3.2 Rigour of the argument

This criterion focuses on rigorous argument structure. Pattern instantiation may guide in achieving this, or a thorough description of an argument may help achieve a rigorous argument. Such descriptions may include a deductive or an inductive proof in an argument.

7.3.2.1 *Refinement of the evaluation model*

Similar to convincing basis of the AC, figure 7.8 shows relevant aspects of a refinement of the model in figure 7.1. 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.

The refinement shows that 'ProcessX' now consists of 3 main steps (reading bottom to top) in figure 7.8.

- CheckInformalArgument – a review of the informal argument. Concerning the formal argument, 'CheckFormalArgument' replaces the 'CheckInformalArgument' in case of formal arguments. Inputs to this process are the AC data items: 'Arguments', 'Rationale', 'ArgumentsSupp.Terms', 'Rebuttals', 'MitigatedRebuttals', 'Evidence', 'ArgPattern'. The output is only to the 'ErrorReport'. These links make it reasonably clear that

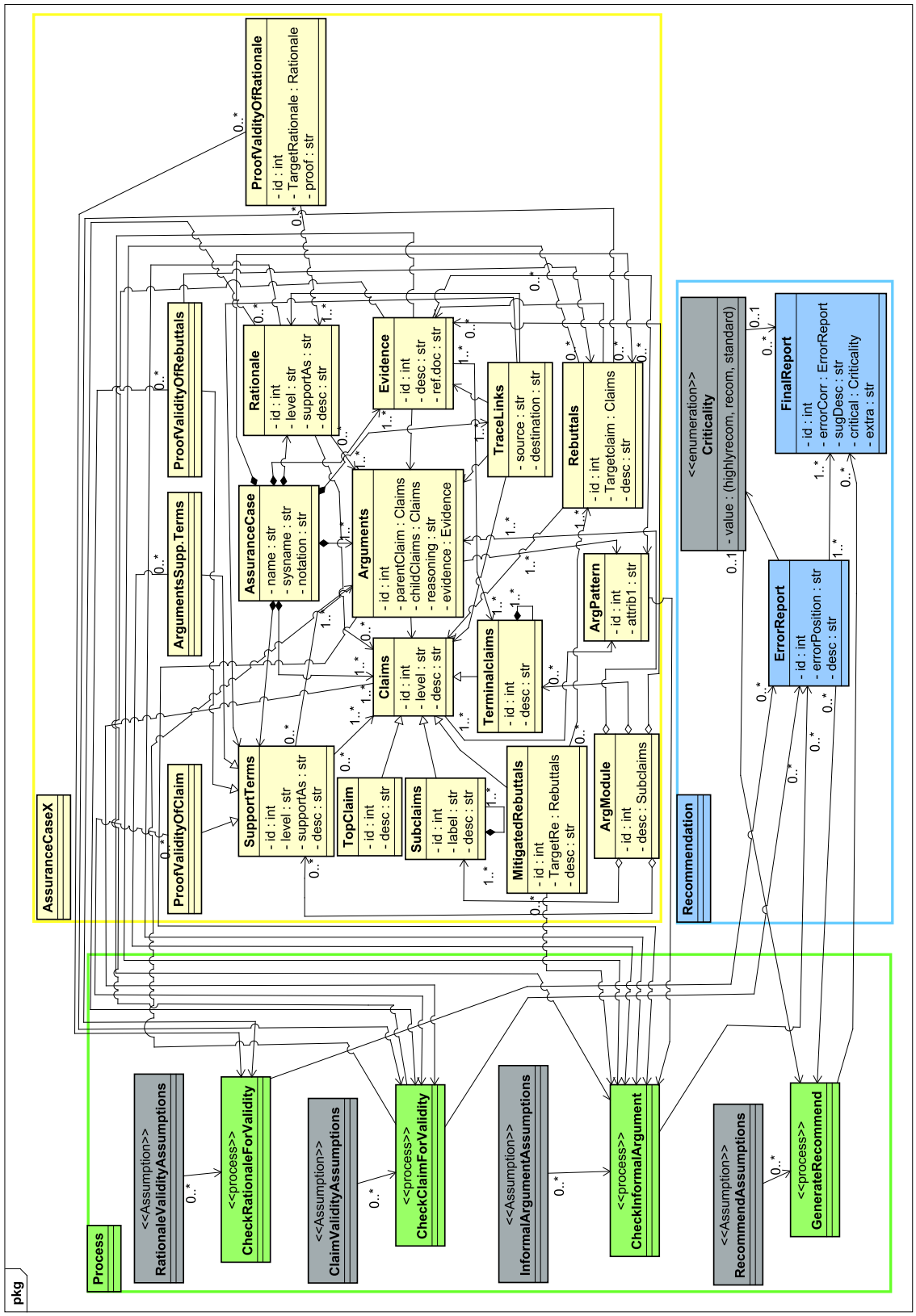


Figure 7.8: Evaluation Process for "Rigour of the arguments"

the focus of this check is rigour or thorough application of the pattern. Assumptions and criteria for this check are found in 'InformalArgument Assumptions'.

- CheckClaimForValidity – a review of the validity of claims. Inputs to this process are 'Claims', 'ProofValidityOfClaims', 'Rebuttals' and 'ProofValidityOfRebuttals', 'Arguments' and 'Evidence'.The output is again to the 'ErrorReport'. The focus of this check is to review proofs of the validity of claims and rebuttals. Assumptions and criteria for this check are to be found in 'ClaimValidity Assumptions'.

- CheckRationaleForValidity – a review that evaluates the validity of rationale. Inputs to this process are 'Rationale', 'ProofValidityOfRationale'. Indirect inputs are 'Claims' and 'Arguments'. The output is again to the 'ErrorReport'. The focus of this check is on whether the rationale supporting reasoning is valid or not.

7.3.2.2 *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 also include major steps for formal arguments check. We can then check these steps to see that they conform to the model.

- ***CheckFormalArgument:***

  (1) A formal argument shall be valid with necessary assumptions.

  (2) Rationale to support the formal argument shall be provided.

  (3) All terms supporting the formal argument shall be valid.

  (4) Rebuttals in a formal argument shall be included, and they shall be complete and consistent. (if they are present)

  (5) Mitigation of rebuttals in a formal argument shall be included, 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 described to prove that if premise is true then conclusion is true, and the steps shall be complete and consistent.

(2) Rationale to support the informal argument shall be included.

(3) All terms supporting the informal argument shall be complete and consistent.

(4) Rebuttals in an informal argument shall be included, and they shall be consistent. (if they are present)

(5) Mitigation of rebuttals in an informal argument shall be included, 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 they are present)

- ***CheckRationaleForValidity:***

(1) Rationale shall be supported by deductive or inductive proofs.(if they are necessary).

- ***GenerateRecommend:***

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

7.3.3 Quality of the hazard analysis

This is one of the criteria that applies to safety case evaluation in particular. Adequate mitigation of all known hazards is a prerequisite for system safety. In addition, we need to know with reasonable certainty that there are not likely to

be additional hazards that we have not considered. There are a number of ways in which we can gain confidence that 'all' hazards have been identified, which is a necessary precursor to all hazards have been mitigated. For a security case evaluation, threat analysis will take place instead of hazards.

7.3.3.1 *Refinement of the evaluation model*

Figure 7.9 shows relevant aspects of a refinement of the model in figure 7.1. 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.

The refinement shows that 'ProcessX' now consists of 4 main steps in figure 7.9.

- ClaimEvidenceHAMethod – a review of the hazard analysis method. Inputs to this process are 'SoundnessHAMethodEvi.', 'HAMethodEvidence', 'ExpertAppraisalEvidence' and 'Claims'. The output is only to the 'ErrorReport'. These links make it clear that the focus of this check is the soundness of the hazard analysis method. Assumptions and criteria for this check are found in 'AssumptionsClEvHAMethod'.

- ClaimEvidencePeopleCredentials – a review of the competency of people. Inputs to this process are 'Claims' and 'PeopleEvidence'.The output is again to the 'ErrorReport'. The focus of this check is competency of people involved in hazard analysis. Assumptions and criteria for this check are to be found in 'AssumptionsClEvPeopleCreden'.

- ClaimEvidenceComparison – a review that evaluates the validity of known hazards and coverage of identified hazards. Inputs to this process are 'ValidationKnownHazards', 'Claims', 'KnownHazards', 'IdentifiedHazards'. Indirect input is 'Arguments'. The output is again to the 'ErrorReport'. The focus of this check is performing comparison whether identified hazards cover known hazards. Assumptions and criteria for this check are to be found in 'AssumptionClEvidenceComparison'.

- ClaimEvidenceMitigation – a review that evaluates the implementation of mitigation. Inputs to this process are 'Claims', 'AcceptanceCrite-

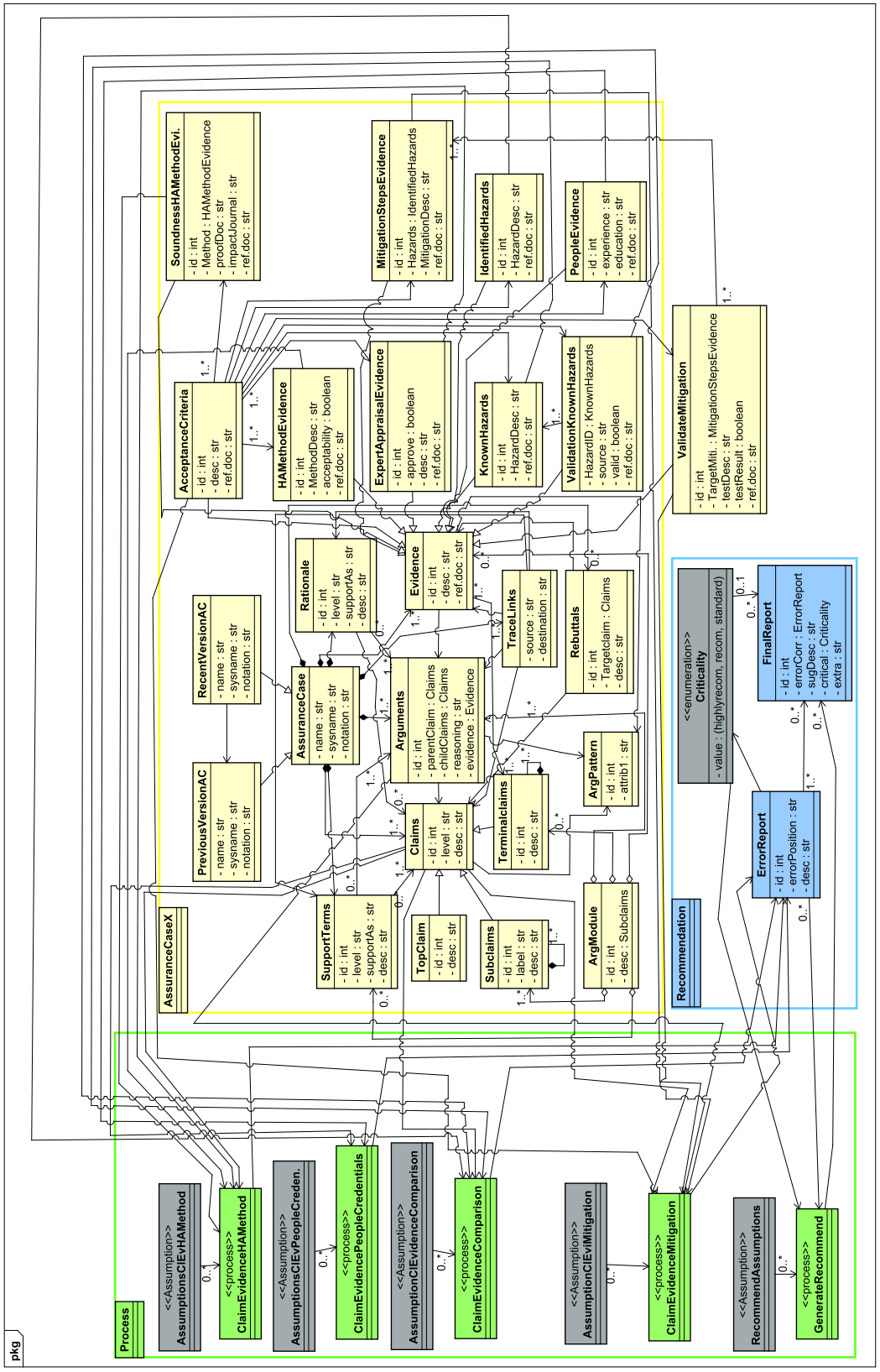


Figure 7.9: Evaluation Process for "Quality of the hazard analysis"

ria', 'MitigationStepsEvidence', 'ValidationMitigation' and 'Arguments'. The output is again to the 'ErrorReport'. The focus of this check is on whether the mitigations of hazards are implemented correctly or not.

7.3.3.2 *Instantiated evaluation process*

We can now instantiate the model. We do this by describing rules for all processes to review quality of the hazard analysis.

- ***ClaimEvidenceMitigation:***

  (1) At least one claim shall mention mitigation of all identified hazards.

  (2) Evidence supporting claims shall document mitigation steps along with hazards and comply with acceptance criteria.

  (3) Claims shall mention that each safety requirement must mitigate atleast one hazard.

  (4) Evidence supporting claims shall document safety requirements for mitigation complying with acceptance criteria.

- ***ClaimEvidenceHAMethod:***

  (1) At least one claim supporting the hazard analysis used shall include why it is considered best practice.

  (2) Evidence supporting the claim related to industry best practice hazard analysis should include an expert appraisal to prove the soundness of the hazard analysis.

  (3) Evidence supporting claims related to hazard analysis shall document the execution of the hazard analysis and comply with acceptance criteria.

- ***ClaimEvidencePeopleCredentials:***

  (1) At least one claim shall mention that people with the necessary competence performed the hazard analysis process.

  (2) Evidence supporting the claims related to credentials shall document the experience and education to perform the hazard analysis and comply with acceptance criteria.

- ***ClaimEvidenceComparison:***

  (1) At least one claim shall mention that the identified hazard list includes all known prior to the hazard analysis performed-if such a list is available.

  (2) At least one claim shall assure the authenticity of a known hazards list.

- ***GenerateRecommend:***

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

7.3.4 Arguing completeness

We already discussed the quality of the hazard analysis as an evaluation criterion. One of the aspects of this criterion that we had to consider is the claim that 'all' hazards have been identified. This is an example of a completeness claim that cannot be 'proved'. There are typically many such claims in an AC.

7.3.4.1 *Refinement of the evaluation model*

Figure 7.10 shows relevant aspects of a refinement of the model in figure 7.1. 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.

The refinement shows that 'ProcessX' now consists of 3 main steps in figure 7.10.

- HazardIdentificationComplete – a review of the identification and mitigation of hazards. Inputs to this process are 'MitigationStepsEvidence', 'AcceptanceCriteria', 'Arguments', 'SupportTerms' and 'Claims'. The output is only to the 'ErrorReport'. These links make it reasonably clear that the focus of this check is the identification and mitigation of rebuttals. Assumptions and criteria for this check are found in 'AssumptionHazardIdenComplete'.

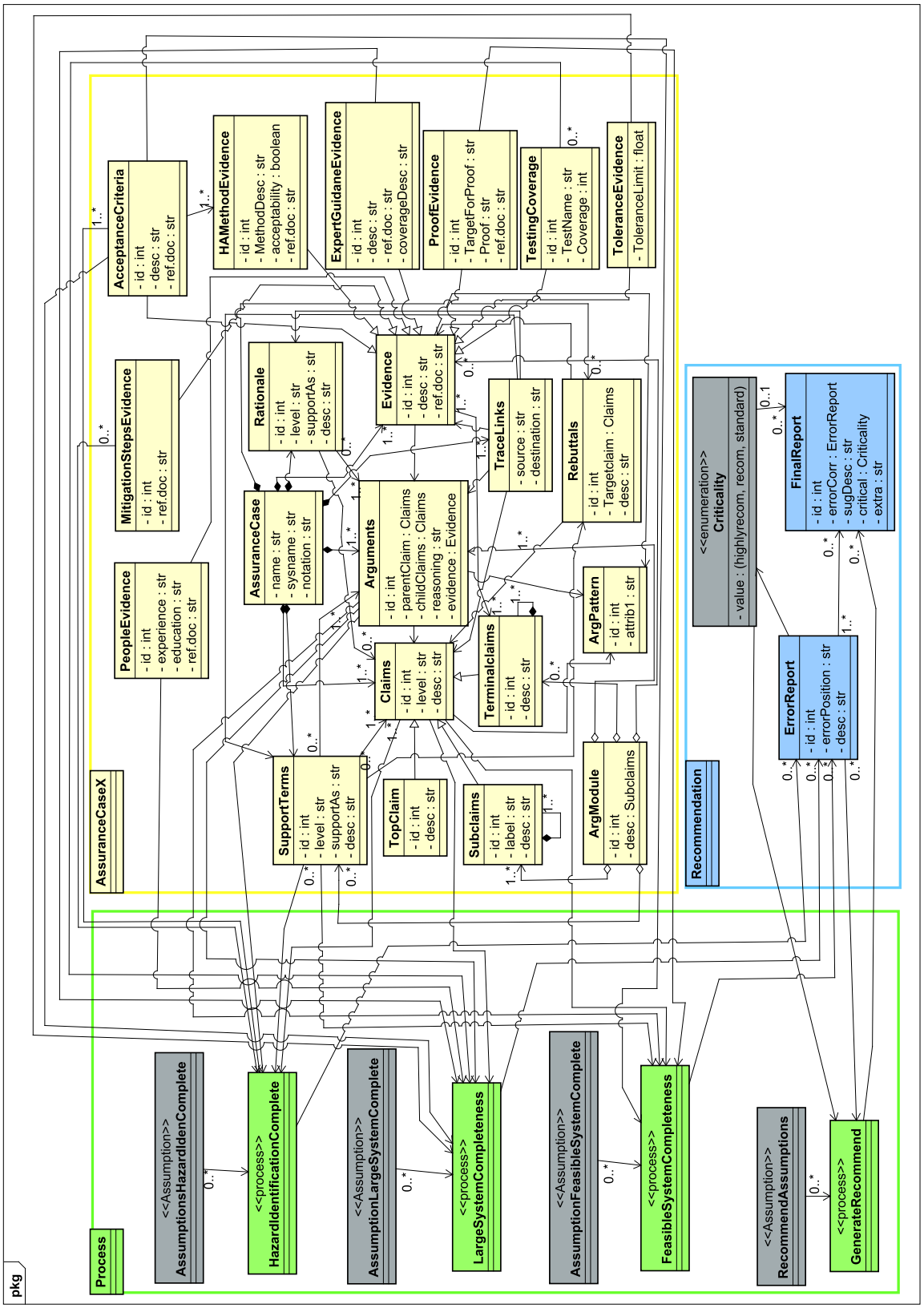


Figure 7.10: Evaluation Process for "Arguing Completeness"

- LargeSystemCompleteness – a review of the completeness of a large system which can not be fully analyzed. Inputs to this process are 'ExpertGuidanceEvidence', 'PeopleEvidence', 'TestingCoverage', 'Arguments', 'Claims', 'AcceptanceCriteria' and 'ToleranceEvidence'. The output is again to the 'ErrorReport'. The focus of this check is completeness of large system argument. Assumptions and criteria for this check are to be found in 'AssumptionLargeSystemComplete'.

- FeasibleSystemCompleteness – a review that evaluates the completeness of feasible system argument. Inputs to this process are 'AcceptanceCriteria', 'SupportTerms', 'Arguments', 'Claims', 'ProofEvidence'. The output is again to the 'ErrorReport'. The focus of this check is completeness of a feasible system argument. Assumptions and criteria for this check are to be found in 'AssumptionFeasibleSystemComplete'.

7.3.4.2 *Instantiated evaluation process*

We can now instantiate the model. We do this by describing rules for all processes to review arguing completeness.

- ***HazardIdentificationComplete:***

  (1) Arguments shall mention known hazards that may violate claims.

  (2) All terms supporting arguments shall provide clarification and/or proofs.

  (3) Claims shall mention the implementation of mitigation to resolve hazards.

  (4) Evidence supporting claims related to mitigation of hazards shall document mitigation steps complying with acceptance criteria.

- ***LargeSystemCompleteness:***

  (1) Claims related to testing coverage shall include an explicit tolerance limit.

  (2) Arguments supporting claims of testing shall provide reasoning on how lower claims support the upper claim.

(3) Evidence shall document the testing coverage metric complying with acceptance criteria.

(4) Claims related to people involved in the testing process shall include that the people involved are competent to perform and plan the relevant tests.

(5) Evidence supporting claims related to credentials of people shall document education, experience complying with acceptance criteria.

(6) Evidence related to a coverage metric mentioned in claims shall comply with acceptance criteria.

- ***FeasibleSystemCompleteness:***

(1) Arguments with formal/semiformal/informal reasoning shall be complete, i.e. shall fulfil necessary assumptions.

(2) All key terms supporting formal/semiformal/informal arguments shall be clarified and valid.

(3) Claims shall mention formal/semiformal/informal proof.

(4) All key terms supporting formal/semiformal/informal claims shall be clarified and valid.

(5) Evidence supporting formal/semiformal/informal claims shall document proof complying with acceptance criteria.

- ***GenerateRecommend:***

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

7.3.5 Repeated arguments

During AC development, developers can use part of the AC using an instantiation of argument patterns for automated development or an equivalent of "cut-and-paste" to ease the development. The evaluation process checks whether a pattern is instantiated in a proper order correctly or not. Sometimes, it is hard

to identify the instantiated argument pattern due to a lack of documentation or even improper instantiation in some cases.

7.3.5.1 *Refinement of the evaluation model*

Figure 7.11 shows relevant aspects of a refinement of the model in figure 7.1. 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.

The refinement shows that 'ProcessX' now consists of 2 main steps in figure 7.11.

- SimilarArgument – a review of similar arguments in an AC. Inputs to this process are 'ExternalNote', 'CriteriaForArgPattern', 'Claims', 'Arguments', 'ArgPattern'. The output is only to the 'ErrorReport'. These links make it clear that the focus of this check is the identification of similar arguments. Assumptions and criteria for this check are found in 'AssumptionSimilarArgument'.

- ClaimArgumentEvidencePattern – a review of the compliance of claims, arguments and evidence with pattern. Inputs to this process are 'Context Arg Pattern', 'Context', 'Assumptions', 'Justifications', 'Claims', 'Arguments', 'Evidence', 'ArgPattern', 'AssumptionArgPattern', 'JustificationArgPattern'.The output is again to the 'ErrorReport'. The focus of this check is compliance with argument pattern. Assumptions and criteria for this check are to be found in 'AssumptionClArEvidencePattern'.

7.3.5.2 *Instantiated evaluation process*

We can now instantiate the model. We do this by describing rules for all processes to review repeated arguments.

- ***SimilarArgument:***

  (1) Argument pattern with a similar context shall be instantiated to a specific argument in an AC.

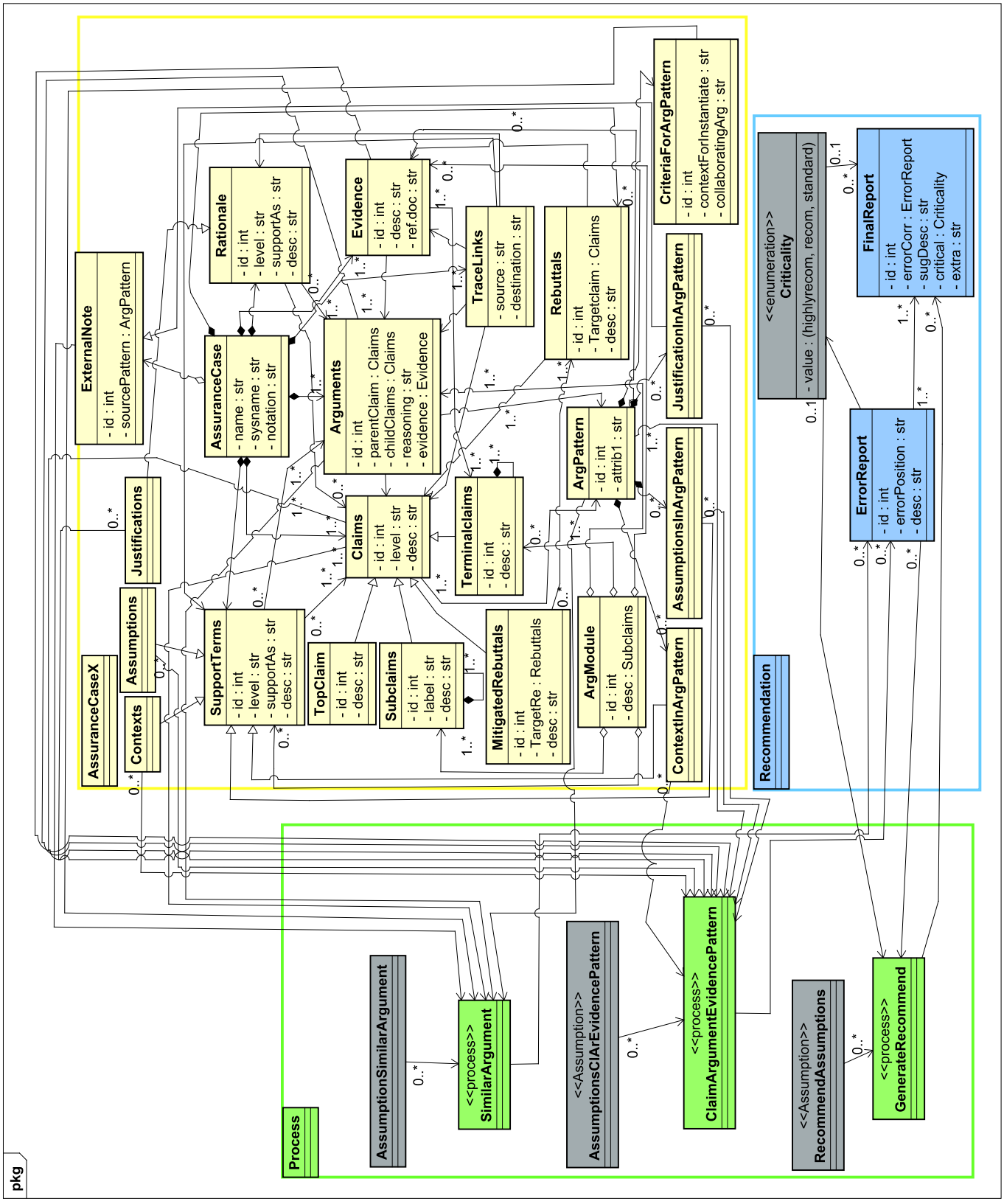


Figure 7.11: Evaluation Process for "Repeated Arguments"

(2) If any "cut-and-paste" occurs, then it shall be mentioned in an external note/documentation to identify the source.

(3) If any "cut-and-paste" occurs, then a target argument branch shall comply with a source argument branch.

- ***ClaimArgumentEvidencePattern:***

(1) Claims, arguments, evidence in an instantiated argument branch shall comply with claims, arguments and evidence of an argument pattern.

(2) All terms supporting claims, arguments, and evidence shall comply with all terms supporting those of an argument pattern.

(3) An argument developed using instantiation or "cut-and-paste" shall be consistent with neighbouring arguments.

- ***GenerateRecommend:***

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

7.3.6 ALARP

This is another criterion that applies to the safety case only. The main intention of this criterion is to ensure adequate confidence in risk assessment in a software-intensive system.

7.3.6.1 *Refinement of the evaluation model*

Figure 7.12 shows relevant aspects of a refinement of the model in figure 7.1. 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.

The refinement shows that 'ProcessX' now consists of 1 main step in figure 7.12.

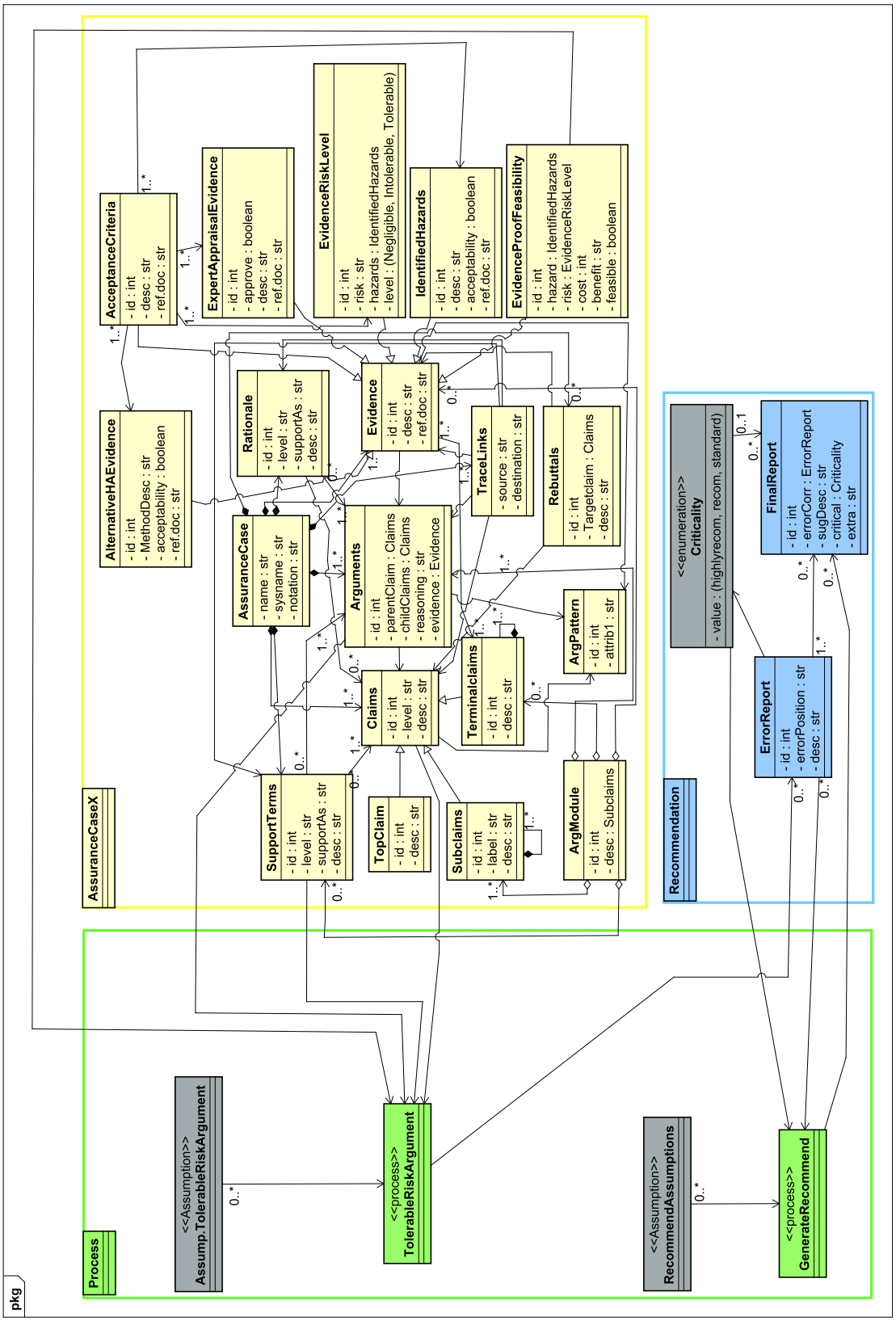


Figure 7.12: Evaluation Process for "ALARP"

- TolerableRiskArgument – a review of the tolerable risk argument. Inputs to this process are 'SupportTerms', 'EvidenceProofFeasibility', 'Arguments', 'Claims'. The output is again to the 'ErrorReport'. The focus of this check is identification of tolerable risk arguments. Assumptions and criteria for this check are to be found in 'Assump.TolerableRiskArgument'.

7.3.6.2 *Instantiated evaluation process*

We can now instantiate the model. We do this by describing rules for all processes to review ALARP.

- ***TolerableRiskArgument:***

  (1) Claims shall mention any benefit of having a specific hazard (if it exists).

  (2) Evidence shall document the benefit to support the claim mentioning benefit related to a hazard.

  (3) Claims shall mention consideration of measures to reduce risks

  (4) Evidence shall document measures to reduce risks associated with each hazard.

  (5) Claims shall mention the feasibility of the reduction of risk.

  (6) Evidence supporting claims shall document proof of feasibility.

  (7) All necessary terms supporting claims, arguments shall be clarified.

- ***GenerateRecommend:***

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

7.3.7 Confidence

Confidence refers to trust or belief in claims, arguments or evidence. Many researchers quantitatively define confidence assessments. We also consider confidence as one of the evaluation criteria to be taken care of. There are several confidence assessment methods. In our evaluation process, we do not criticize

those methods. Instead, we verify the confidence generated using any of the methods.

7.3.7.1 *Refinement of the evaluation model*

Figure 7.13 shows relevant aspects of a refinement of the model in figure 7.1. 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.

The refinement shows that 'ProcessX' now consists of 4 main steps in figure 7.13.

- EvidenceConfidence – a review of the probability of evidence in an AC. Inputs to this process are 'Evidence', 'ExpertAppraisalEvidence', 'AssessmentProbabilityEvidence'. The output is only to the 'ErrorReport'. These links make it clear that the focus of this check is probability of evidence. Assumptions and criteria for this check are found in 'AssumptionEvidenceConfidence'.

- ArgumentConfidence – a review of the probability/trust of an argument. Inputs to this process are 'Arguments', 'ConfidenceAssessment' and 'SupportTerms'. The output is again to the 'ErrorReport'. The focus of this check is trust of an argument. Assumptions and criteria for this check are to be found in 'AssumptionArgumentConfidence'.

- ClaimConfidence – a review of the trust of a claim. Inputs to this process are 'Claims', 'Evidence', 'ConfidenceAssessment' and 'Terminalclaims'. The output is again to the 'ErrorReport'. The focus of this check is trust of a claim. Assumptions and criteria for this check are to be found in 'AssumptionClaimConfidence'.

- AssociationWeight – a review of the weight of the association. A disjoint weight means the degree that each sub claim can contribute independently to the trustworthiness of a parent claim [78]. This applies to graphical notation only. Inputs to this process are 'SupportTerms', 'TraceLinks'. Indirect input is 'ConfidenceAssessment'. The output is again to the 'ErrorReport'. The focus of this check is identification of

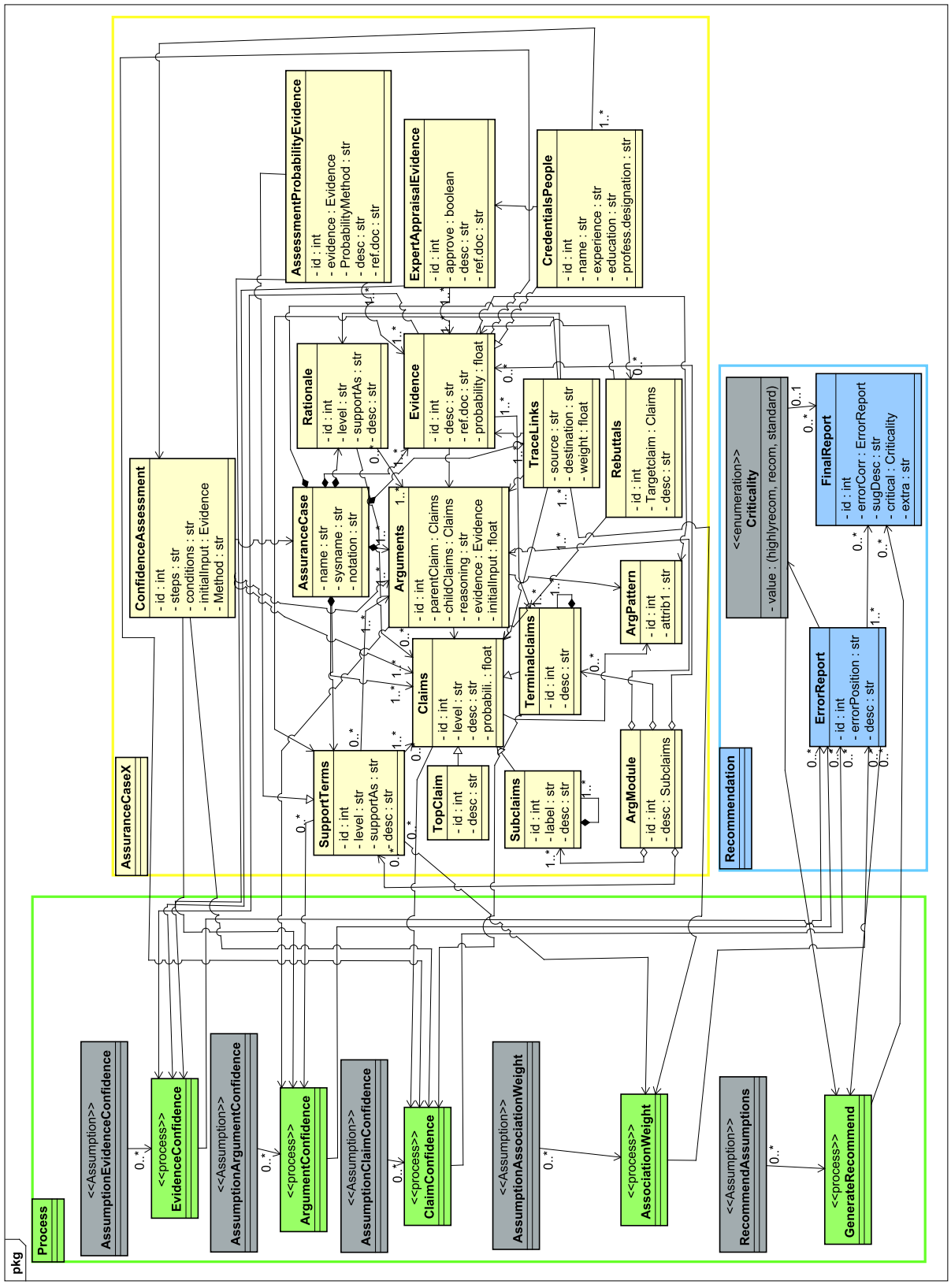


Figure 7.13: Evaluation Process for "Confidence"

weight of association. Assumptions and criteria for this check are to be found in 'AssumptionAssociationWeight'.

7.3.7.2 *Instantiated evaluation process*

We can now instantiate the model. We do this by describing rules for all processes to review confidence.

- ***EvidenceConfidence:***

  (1) Domain experts shall approve the probability of evidence.

- ***ClaimConfidence:***

  (1) The probability/rank/belief of claims shall be derived using a recommended approach.

  (2) All terms supporting the claim shall be clarified.

  (3) The probability/rank/belief of terminal claims shall inherit the probability/belief/disbelief of supporting evidence.

- ***ArgumentConfidence:***

  (1) The probability/rank/belief of arguments shall be derived using a recommended approach.

  (2) All necessary terms supporting the argument shall be clarified.

- ***AssociationWeight (Only applies to Graphical notation):***

  (1) The weight of an association shall be defined based on expert review.

- ***GenerateRecommend:***

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

Chapter 8

Case Study: Evaluation of Assurance Cases

In this chapter we apply the evaluation processes (presented in Chapter 7) to a GSN-like example (an AC of a coffee cup) for validation. One obvious difference in the notation used in this AC compared with GSN is the content of the 'strategy' nodes. According to the GSN community Standard 2.0 [12], a strategy describes how an upper-level claim is decomposed into lower-level claims. It does not represent an 'argument' with reasoning as to why the lower-level claims support the parent claim. We illustrate the validation of our proposed evaluation approach for the structure and content of an example AC from a developer's perspective. Later, we illustrate the validation of the same approach from an external reviewer's perspective. We perform evaluation methodically by following a mechanical way of evaluating them based on rules for each process and conclude each review with a list of recommendations on how to improve the AC. This chapter extends work published in [97].

8.1 Evaluation performed by a developer

This section illustrates the validation of our evaluation process from the developer's perspective. The developer may mandate corrections/modifications and afterwards provide an optional note as an 'extra' to an external reviewer for facilitating the external review process.

8.1.1 Structure evaluation

We start by evaluating the structure of the AC from a developer's perspective.

8.1.1.1 Syntax Check

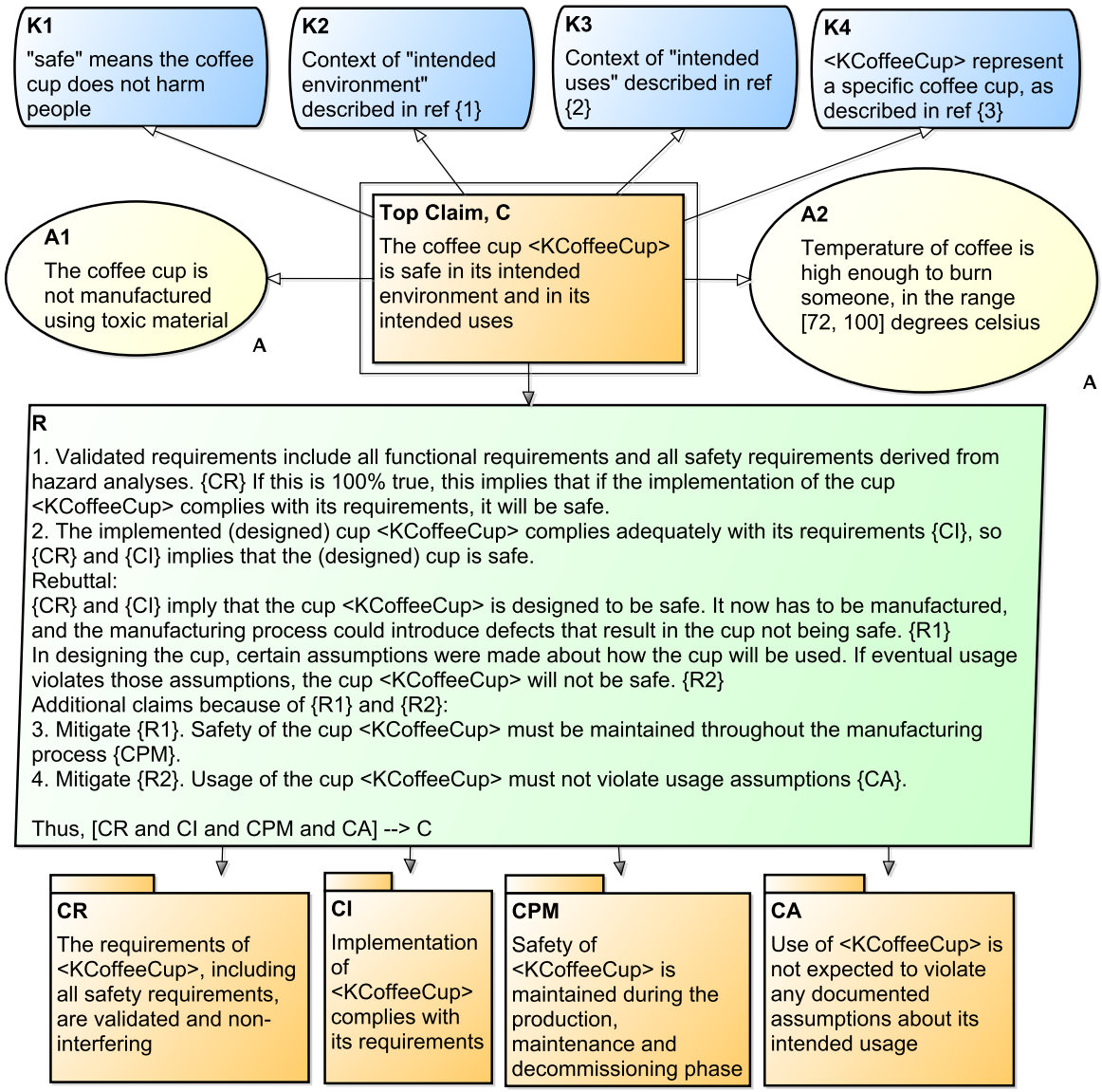


Figure 8.1: Top two level claims of an AC for a coffee cup [97]

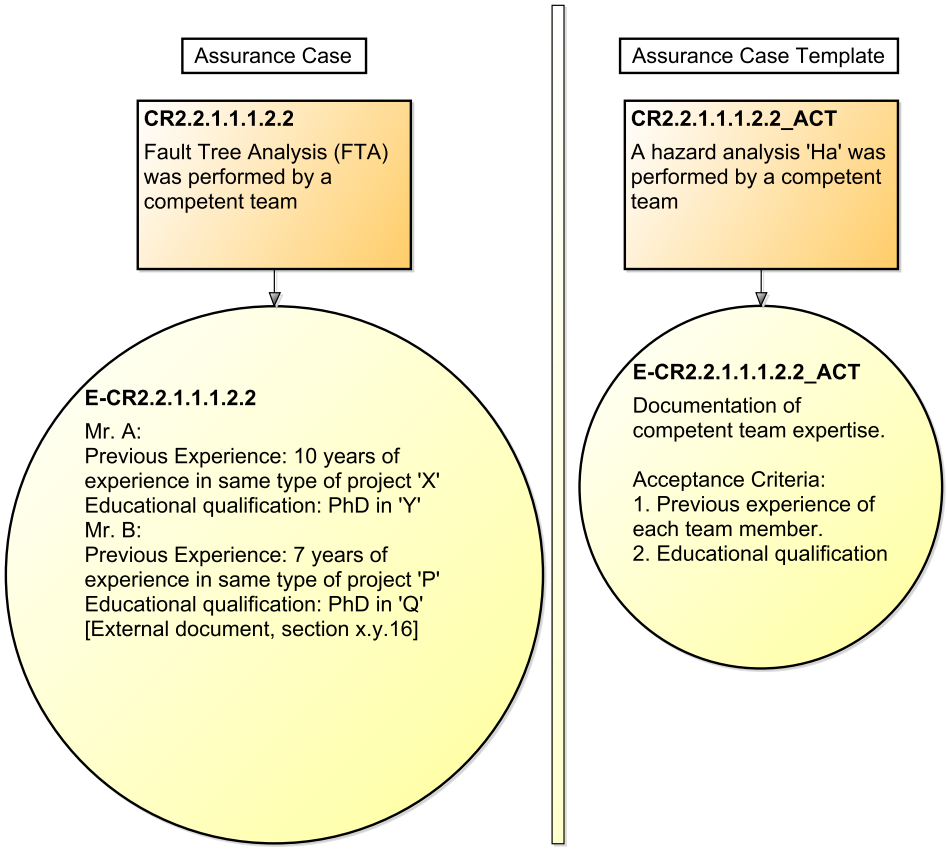


Figure 8.2: An excerpt of evidence complying with acceptance criteria for a coffee cup example [97].

The rules for syntax check show what to check for when identifying syntax errors. The notation attribute in the class "AssuranceCase" indicates 'GSN,' and thus, rules for 'CheckGraphSyntax' apply.

Checking rules for *CheckGraphSyntax*

Concerning Rule (1): We consider the GSN community Standard 2.0 [12] as a reference.

Concerning Rule (2): By review, we note that shapes of goal, strategy, evidence, context, assumption and justification comply with the standard.

Concerning Rules (3) and (4): There is one and only one valid association ('SupportedBy' and 'InContextOf') that exists between any two nodes.

Concerning Rule (5): The terminal nodes (terminal nodes are evidence)

have no outgoing associated nodes.
Concerning Rule (6): The label of goals, strategies and evidence follows a hierarchy.

***GenerateRecommend* produces the following recommendations:**

- Suggestion ('sugDesc'): As the example does not deviate from the GSN-compliant shape, there is no further action required.
- Note delivered ('extra'):
    1. Acceptance criteria ('E-CR2.2.1.1.1.2.2_ACT') are shown separately in figure 8.2 to show compliance with evidence ('E-CR2.2.1.1.1.2.2').
    2. Argument node borrows the shape of 'strategy' for ease of development by a 'GSN' tool. For instance, "Astah GSN" [98] tool is used to document the AC.

8.1.1.2 Traceability

We illustrate validation results of our *traceability* evaluation process. Figure 8.16 shows a hazard analysis argument branch of an AC for a coffee cup. Furthermore, figure 8.3 shows a change management argument branch. The rest of the AC for a coffee cup is in appendix C.

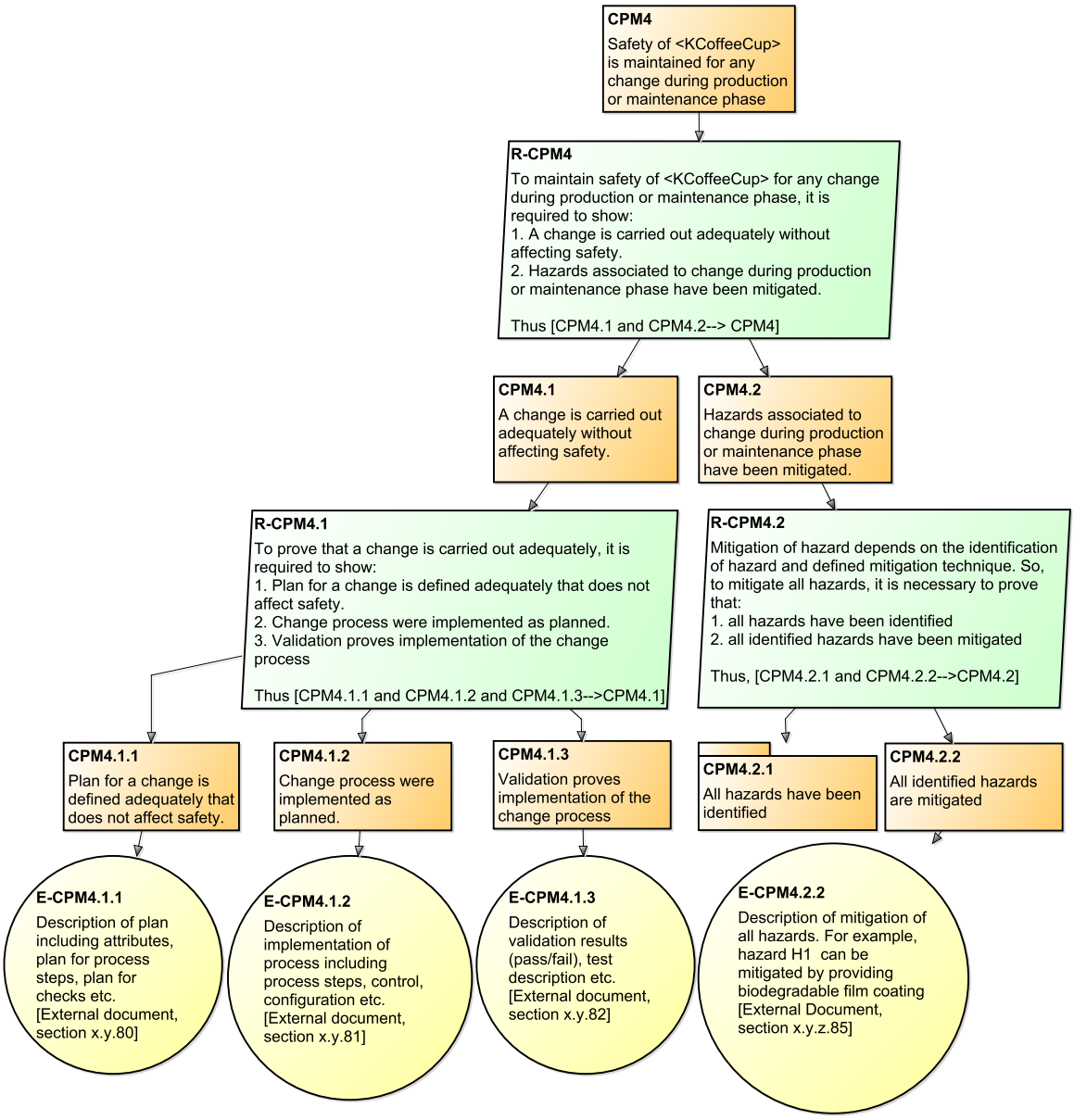


Figure 8.3: Change management argument branch of an AC for a coffee cup

We use three checks consisting of rules to perform traceability evaluation.

We use '(Review) EvidenceToSystemTrace' to evaluate evidence to system artefacts traceability.

Checking rules for *(Review) EvidenceToSystemTrace*

Concerning Rule (1): We find that an explicit link or a reference mentioned

between evidence supporting a process related claim and a specific section of a document describing that specific process. Each evidence refers to a specific section of a document. For instance, evidence 'E-CR2.2.1.1.1.2.1' supports a terminal claim 'CR2.2.1.1.1.2.1' related to FTA execution. It refers to a specific section 'x.y.15' of the requirement chapter in a document describing FTA execution. It is important to note that, a reference to an external document is labelled as a 'x.y.z.p...' format where 'x' represents a chapter of a document, 'y' represents a section of that chapter ('x'), 'z' represents a subsection of that section ('y'), 'p' represents a sub-subsection of that subsection ('z') and so on. Similarly, evidence 'E-CR2.1.2'(prototype generation) refers to 'x.y.4', 'E-CR2.2.1.1.3.2.1'(STPA execution) refers 'x.y.q.18', 'E-CR3.1'(development from requirements with current technology) refers to 'x.y.19', 'E-CR3.2'(built feasibly) refers to 'x.y.18', 'E-CR6.2'(interaction) refers to 'x.y.20', 'E-CI1.1.1.1' (Design process of container) refers to 'x.y.25', 'E-CI1.2.1.1' (design process of lid) refers to 'x.y.28', 'E-CI4.1'(safety assessment process during implementation) refers to 'x.y.51', 'E-CPM1.2' (production process) refers to 'x.y.z.51', 'E-CPM1.3'(safety assessment process) refers to 'x.y.z.52', 'E-CPM2.1' (service process) refers to 'x.y.z.53', 'E-CA2.1.2' (survey analysis) refers to 'x.y.62'.

Concerning Rule (2): We find that explicit links or references mentioned between evidence describing credentials of people for the process and a specific section of a document mentioning credentials of people involved in that process. Evidence 'E-CR2.1.3.3' supports a terminal claim 'CR2.1.3.3' related to competency of people involved in review process. Similarly, evidence 'E-CR2.2.1.1.1.2.3' (people execute FTA) refers to 'x.y.17', 'E-CR2.2.1.1.3.2.3' (people execute STPA) refers to 'x.y.q.20', 'E-CR1.1.2.2' (competent people) refers to 'x.y.c.2', 'E-CI1.1.1.2' (people perform design of container) refers to 'x.y.26', 'E-CI1.2.1.2'(people perform design of lid) refers to 'x.y.29', 'E-CI2.3' (people perform mathematical validation of 'Z') refers to 'x.y.33', 'E-CI3.2' (people perform testing 'X') refers to 'x.y.43'. 'E-CI4.2' (people perform safety assessment) refers to 'x.y.52', 'E-CA2.1.3'(competency of people to perform survey) refers to 'x.y.64'.

Concerning Rule (3): We find that explicit links or references mentioned between evidence supporting a product-related claim and a specific section of a document related to that product. Evidence 'E-CR2.1.3.1.1'

supports a terminal claim 'CR2.1.3.1.1' related to requirements of a coffee cup container. Similarly, evidence 'E-CR1.1.2.1' (signature proof) refers to 'x.y.c.1', 'E-CR2.1.1' (test case definition) refers to 'x.y.3', 'E-CR2.1.3.1.2' (requirements of sleeve) refers to 'x.y.6', 'E-CR2.1.3.1.3' (requirement of lid) refers to 'x.y.7', 'E-CR2.1.3.1.4' (requirement of interaction between all components) refers to 'x.y.8', 'E-CR2.2.1.1.1.1' (all hazards description by FTA) refers to 'x.y.z.1', 'E-CR2.2.1.1.3.1' (all hazards description by STPA) refers to 'x.y.p.1', 'E-CR2.2.1.1.3.3' (comparison list of hazards) refers to 'x.y.21', 'E-CR2.2.1.1.1.2.2' (review report for FTA) refers to 'x.y.16', 'E-CR2.2.1.1.3.2.2' (review report for STPA) refers to 'x.y.q.19', 'E-CR4' (testable criteria document) refers to 'x.y.a.19', 'E-CR6.1' (consistent criteria proof for requirement) refers to 'x.y.b.19', 'E-CR6.2' (dependency graph) refers to 'x.y.b.20', 'E-CR6.3' (traceability matrix for stakeholders) refers to 'x.y.b.21', 'E-CI1.1.2.2.12' (qualification test result of biodegradable film use) refers to 'x.y.b.27', 'E-CI1.1.2.2.3' (bottom part satisfy safety requirement) refers to 'x.y.b.28', 'E-CI1.2.2' (traceability matrix for lid) refers to 'x.y.b.30', 'E-CI1.2.3' (specification of lid) refers to 'x.y.b.31', 'E-CPM1.1' (production plan) refers to 'x.y.z.50', 'E-CA1' (historical data) refers to 'x.y.60', 'E-CA2.1.1'(survey questionnaire) refers to 'x.y.61', 'E-CA3' (assumption descriptions) refers to 'x.y.62', 'E-CR1.1.1' (use case) refers to 'x.y.1'.

Concerning Rule (4): We find that explicit links or references mentioned between evidence supporting a claim related to the validation of a product/process and the description of validation of that product/process in a document. Evidence 'E-CI2.2' supports a terminal claim 'CI2.2' related to verification of behaviour. Similarly, evidence 'E-CI2.1' (validation of mathematical method 'Z') refers to x.y.31, E-CI3.1 (validation of test method 'X') refers to 'x.y.40', 'E-CPM2.2' (verification steps) refers to 'x.y.z.54', 'E-CR1.2.1' refers to 'x.y.p.1', 'E-CR1.2.2' refers to 'x.y.p.2', 'E-CR1.2.3' refers to 'x.y.p.3', 'E-CR1.2.4' refers to 'x.y.p.4'.

Concerning Rule (5): We find that evidence description comply with acceptance criteria. For instance, evidence 'E-CR2.2.1.1.1.2.2' comply with acceptance criteria 'E-CR2.2.1.1.1.2.2_ACT' in figure 8.2.

Concerning Rule (6): We do not find any explicit link or reference mentioned between counter-evidence and valid proof (deductive or inductive) de-

fined in a document.

Concerning Rule (7): We find that explicit links or references mentioned between evidence supporting claims related to change management and change management of that system in a document. Evidence 'E-CPM4.1.1' supports a terminal claim 'CPM4.1.1' related to plan for change management. Similarly, evidence 'E-CPM4.1.2' refers to 'x.y.81', 'E-CPM4.1.3' refers to 'x.y.82'.

We use '(Review) RecentToPastVerTrace' to review the trace between a previous version and a recent version of an AC.

Checking rules for *(Review) RecentToPastVerTrace*

Concerning Rules (1) and (2): We do not find any trace link between artefacts of a previous version and a recent version of an AC.

We use '(Review) ArgumentPatternTrace' to review the compliance of an instantiated argument with a pattern.

Checking rules for *(Review) ArgumentPatternTrace*

Concerning Rule (1): We find that an instantiated claims/arguments/evidence shall comply with an argument pattern consisting of claims/arguments/evidence. For instance, claim 'CR2.2.1.1.4' complies with claim 'G1' from 'ALARP' pattern [99]. Other claims comply with claims from the pattern.

***GenerateRecommend* produces the following recommendations:**

- Suggestion ('sugDesc'):
  1. All evidence refers to specific sections of a document; Thus, no action is required.
  2. There are undeveloped claims: 'CPM3', 'CPM2.3', 'CPM2.4', 'CR5', 'CI1.3', 'CL1.1.4', 'CI1.1.2.1', and 'CI1.1.2.3'. It is recommended to develop argument branches for those claims for complete traceability.
- Note delivered ('extra'):
  1. It is important to note that, a reference to an external document is labelled as a 'x.y.z.p...' format where 'x' represents a chapter

of a document, 'y' represents a section of that chapter ('x'), 'z' represents a subsection of that section ('y'), 'p' represents a sub-subsection of that subsection ('z') and so on.

2. External document contains a previous version of AC and provides a list to show release time of previous versions along with traceability between each version.
3. External document describes system information, and each section/subsection describes specific information referred by evidence.

8.1.1.3 Robustness

We illustrate validation of our *robustness* evaluation process by using pairwise comparison with the help of "mental models" and/or experience. For any likely change, we highlight the effect on the AC by a rectangle (highlighted in red colour) relying on "mental models" and/or experience from previous examples. We take into account three types of likely changes:

1. A likely change/variability in a system introduces changes in an AC artifacts. We consider the following anticipated change: the dimension of a container of coffee cup changes. This change may be the height or diameter or both of a container of a coffee cup.
2. A likely rebuttal in an AC may violate claims, arguments or evidence; We consider the following anticipated change: a new operational assumption may violate the previous assumption and initiate changes in an AC.
3. An alternative evidence in an AC may violate the existing terminal claims. We consider the following anticipated change: new testing may affect terminal claims and/or upper claims and/or arguments.

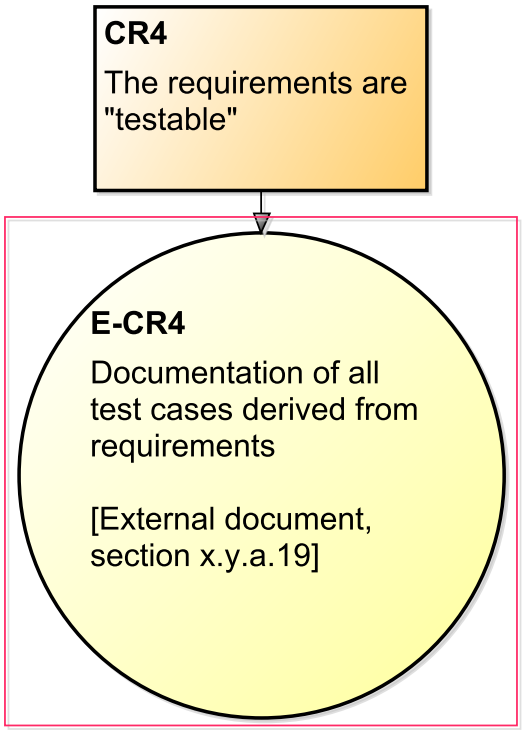


Figure 8.4: Testability branch of an AC for a coffee cup

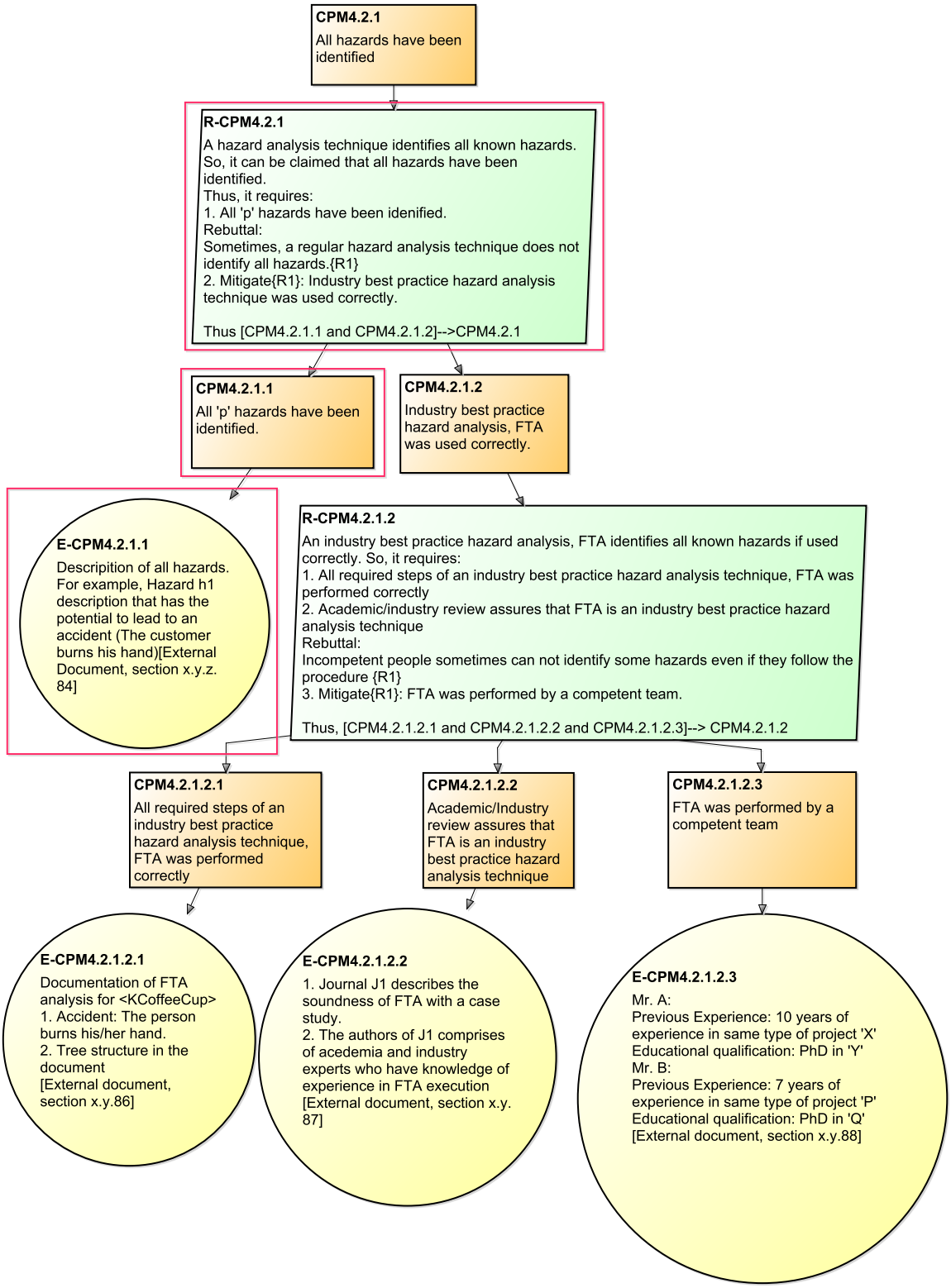


Figure 8.5: Hazard identification branch of an AC for a coffee cup

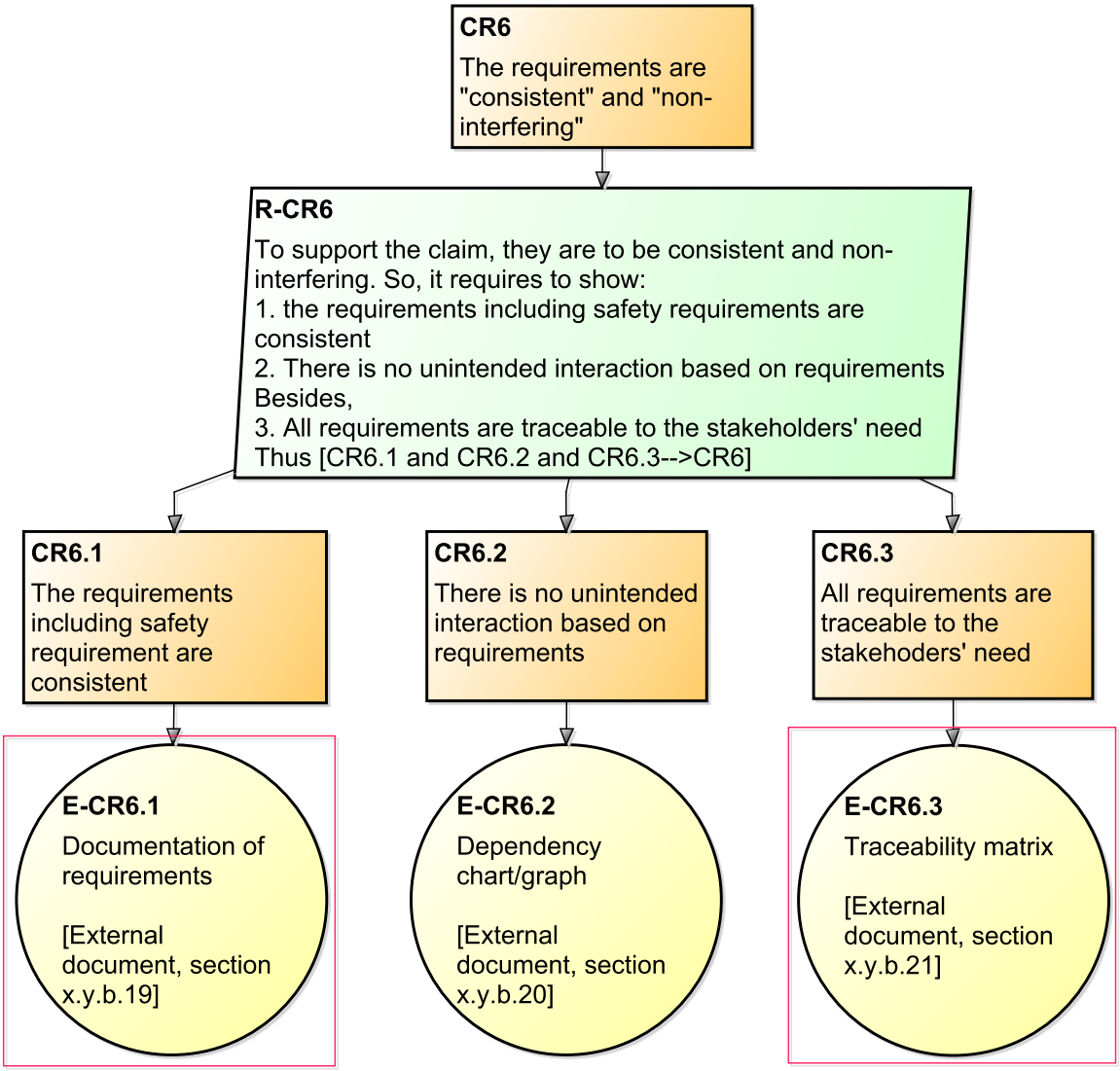


Figure 8.6: Requirement consistency branch of an AC for a coffee cup

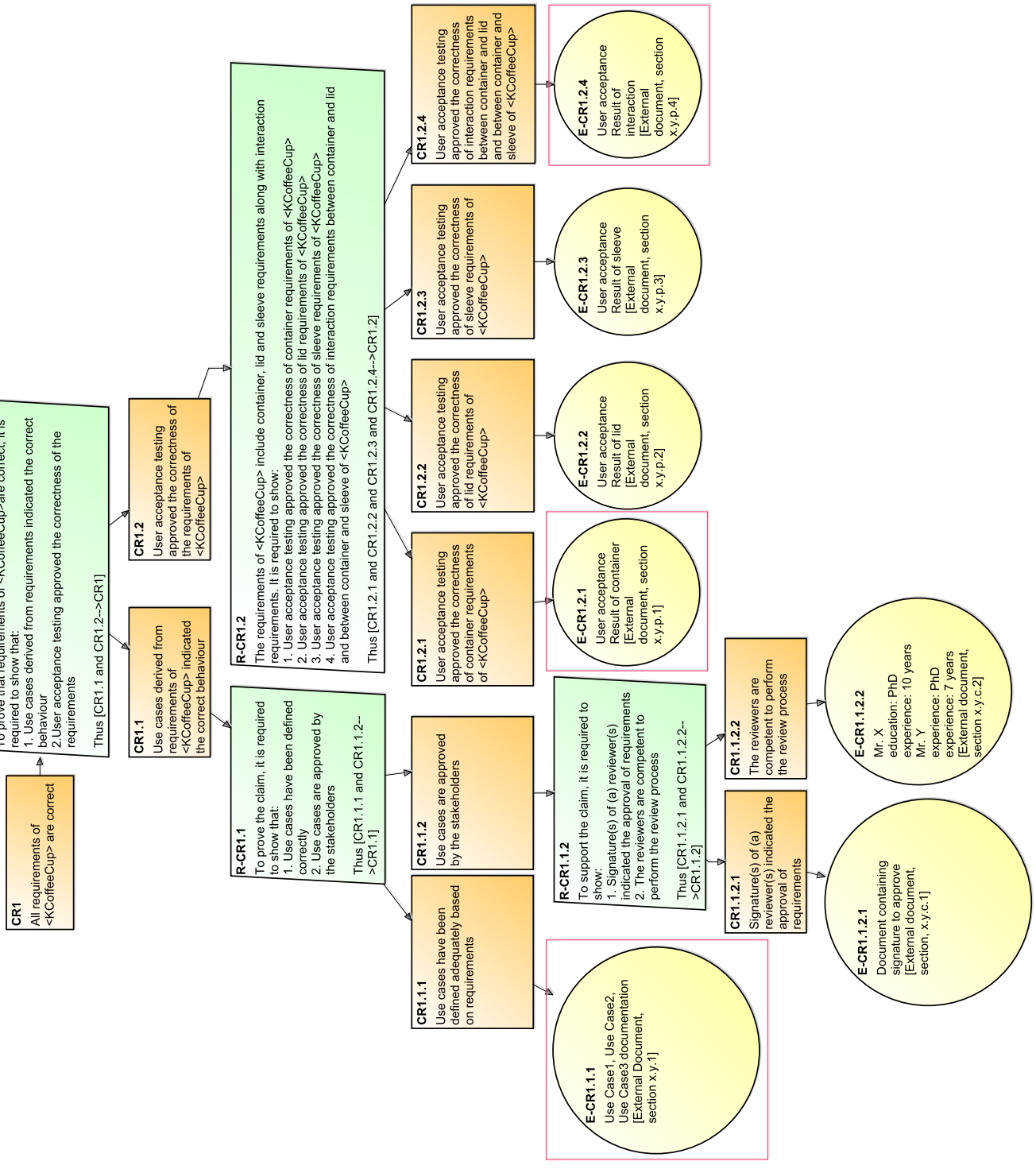


Figure 8.7: Requirement correctness branch of an AC for a coffee cup

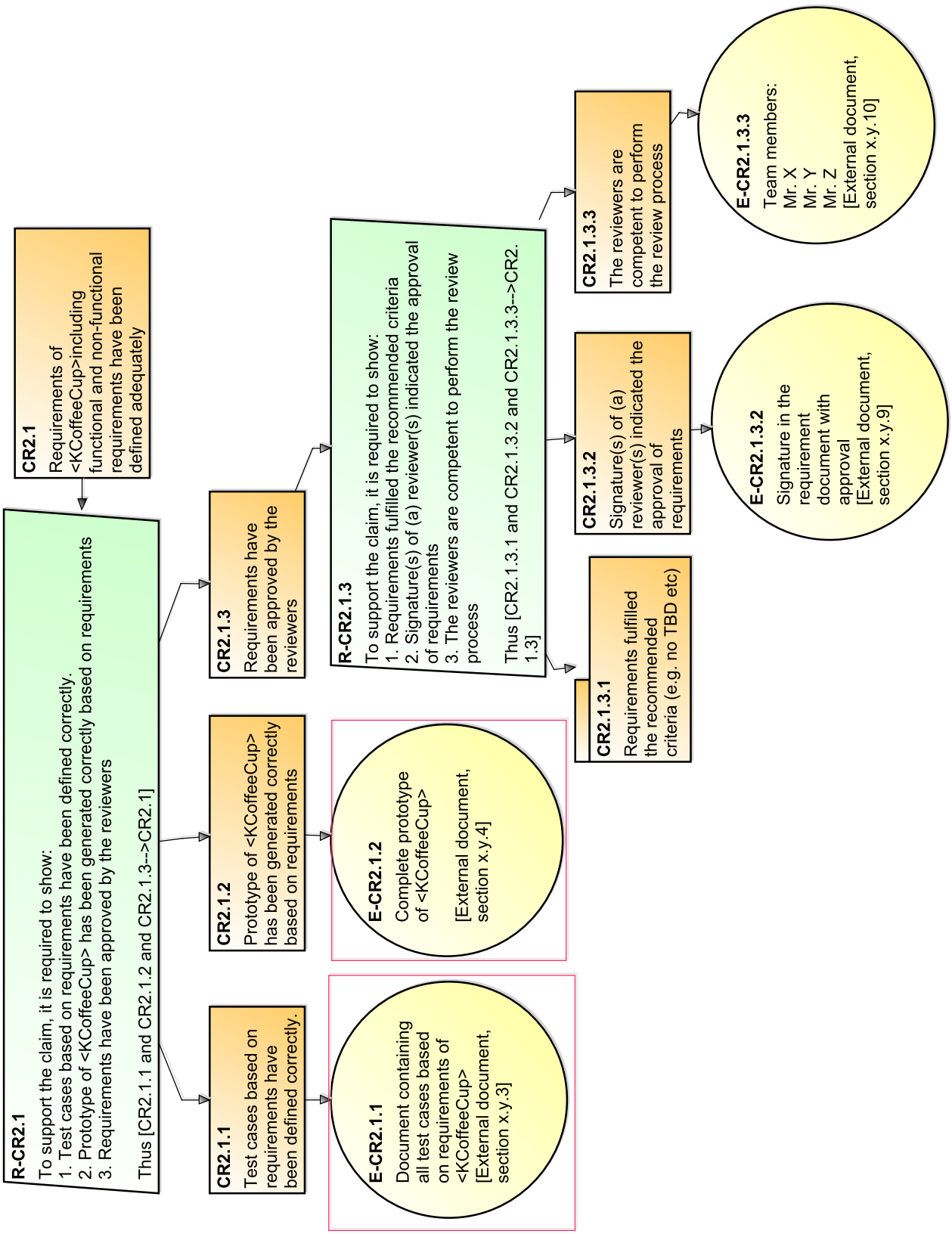


Figure 8.8: Requirement branch of an AC for a coffee cup-(part 1)

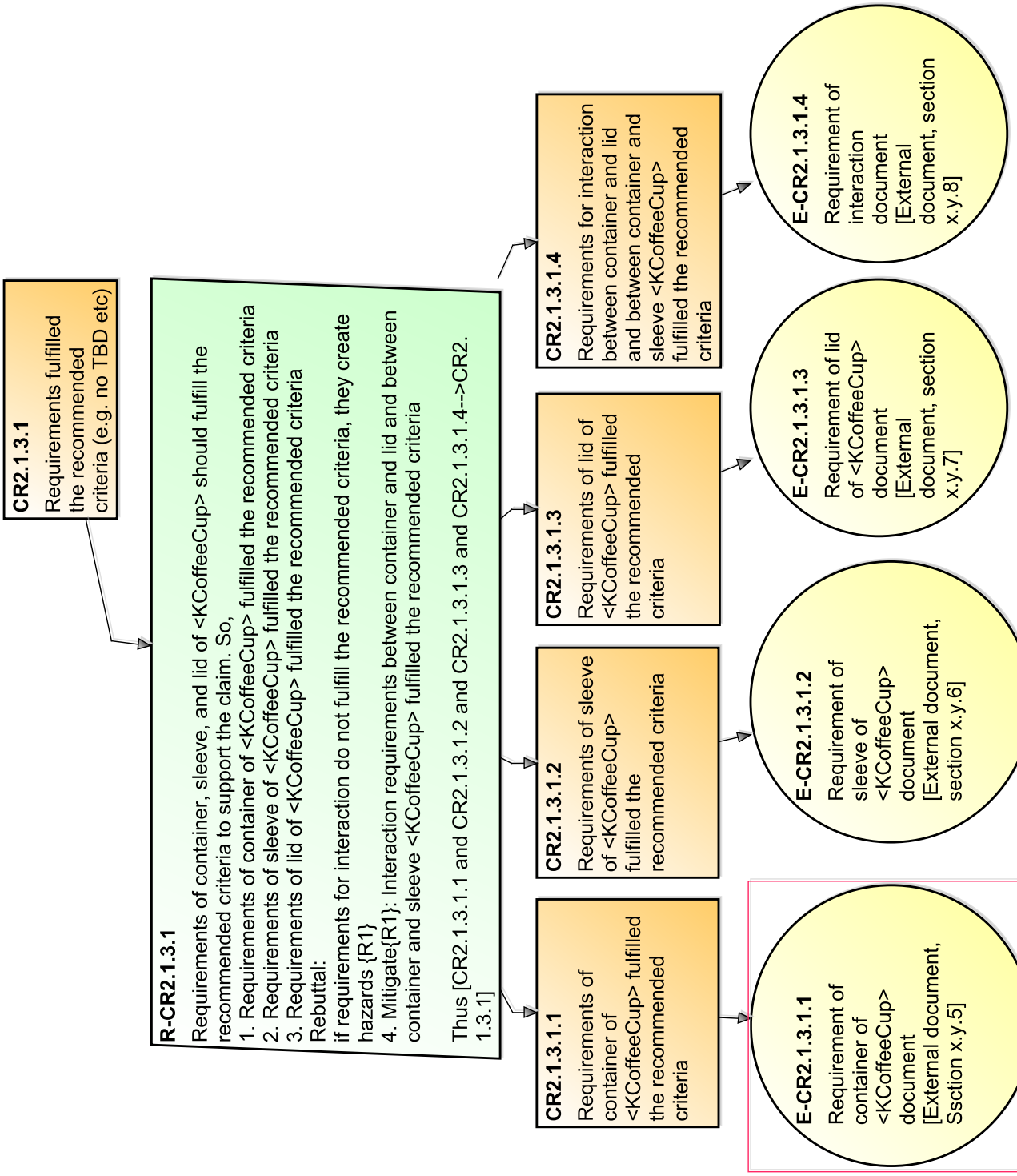


Figure 8.9: Requirement correctness branch of an AC for a coffee cup-(part 2)

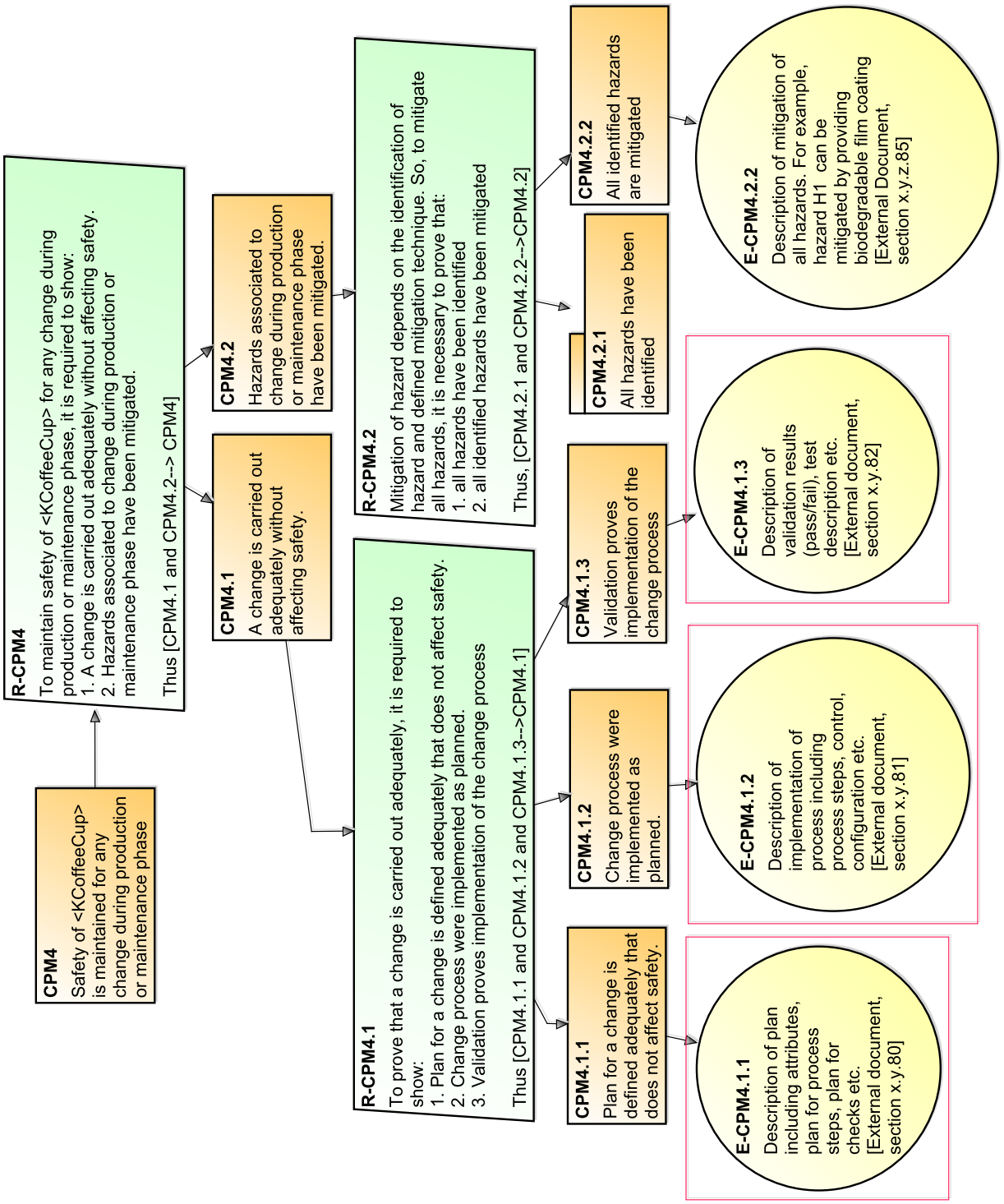


Figure 8.10: Maintenance argument branch of an AC for a coffee cup

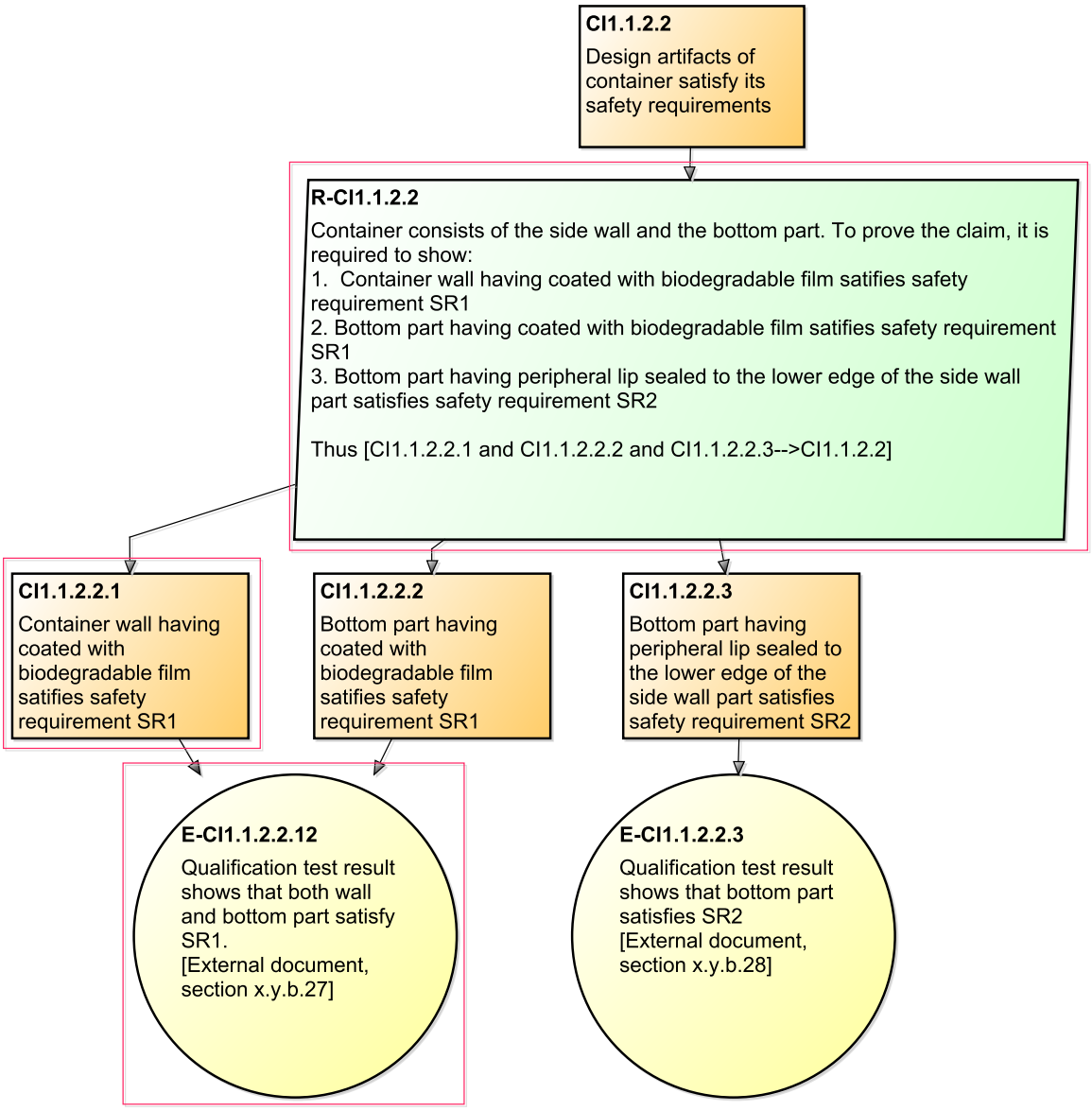


Figure 8.11: Design of container argument branch of an AC for a coffee cup

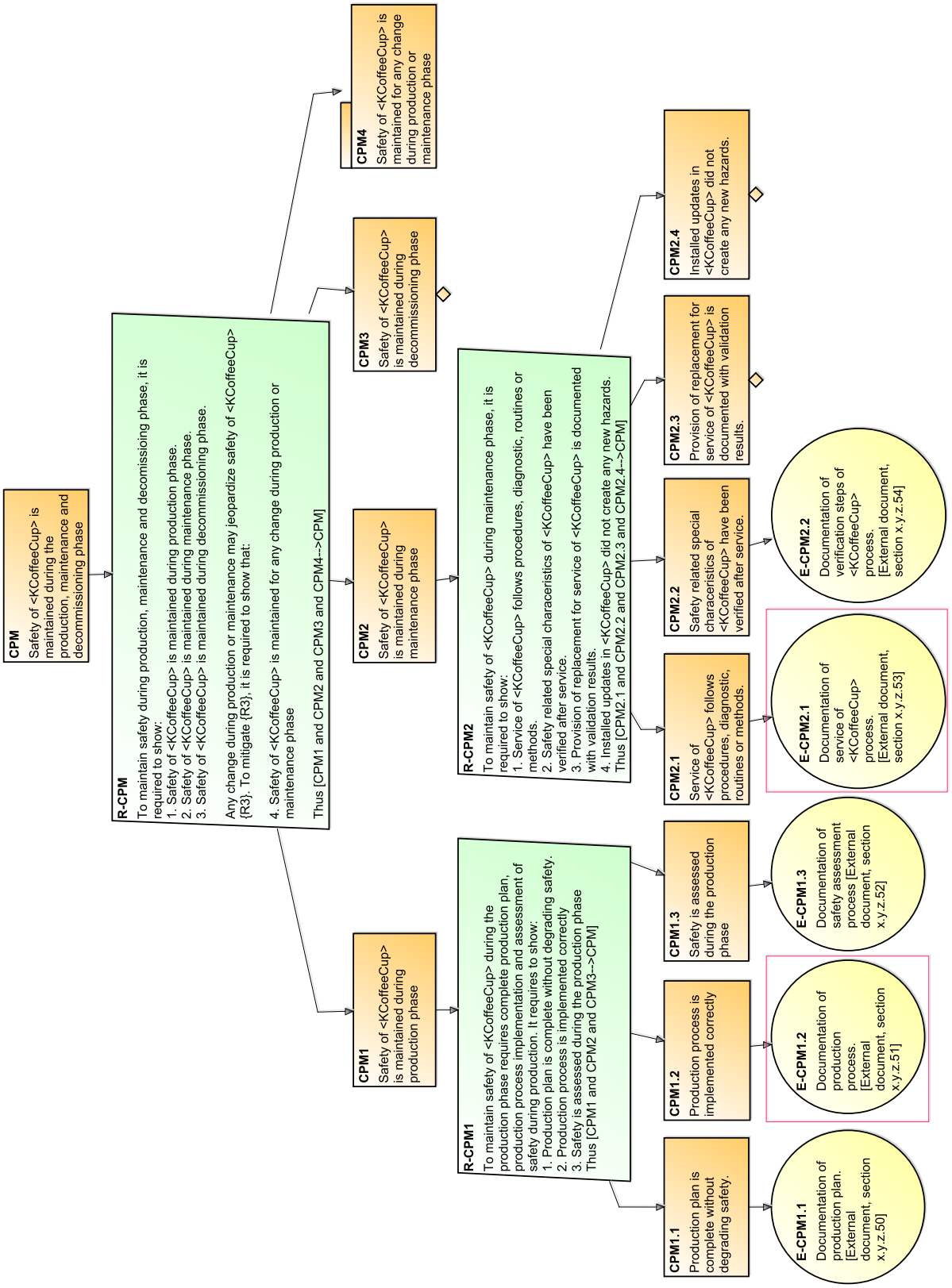


Figure 8.12: Production branch of an AC for a coffee cup

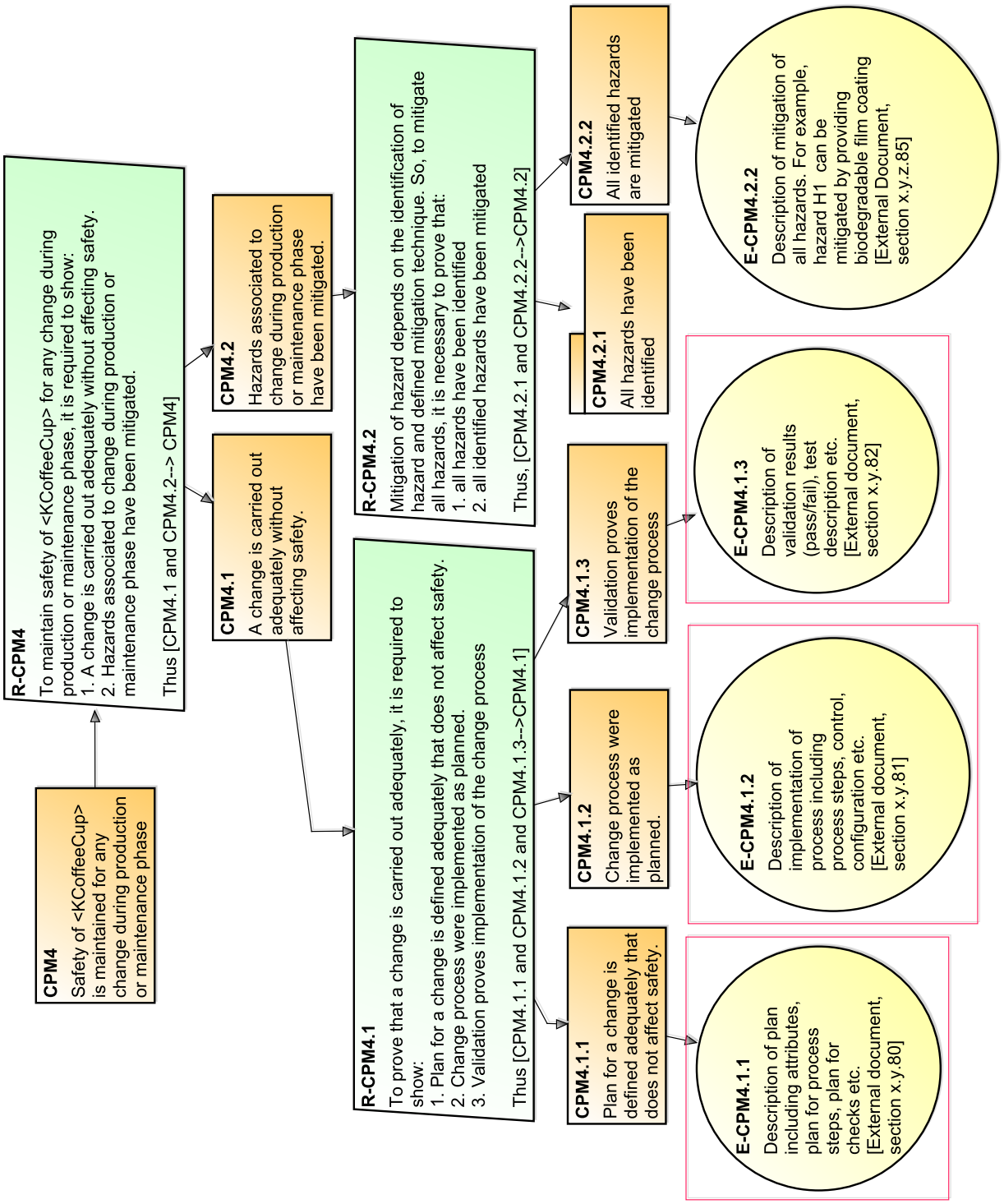


Figure 8.13: Change management argument branch of an AC for a coffee cup

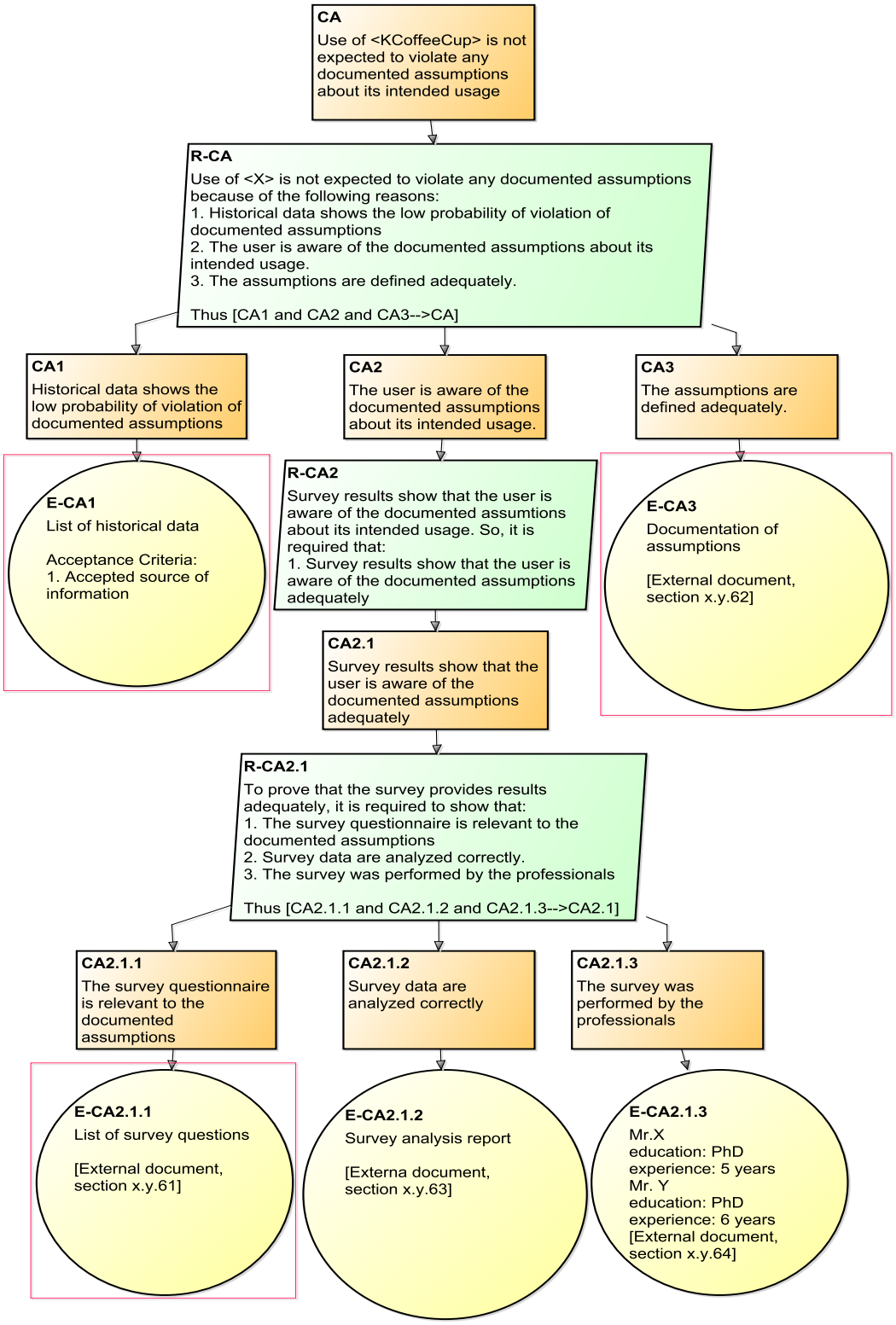


Figure 8.14: Operational assumption argument branch of an AC for a coffee cup

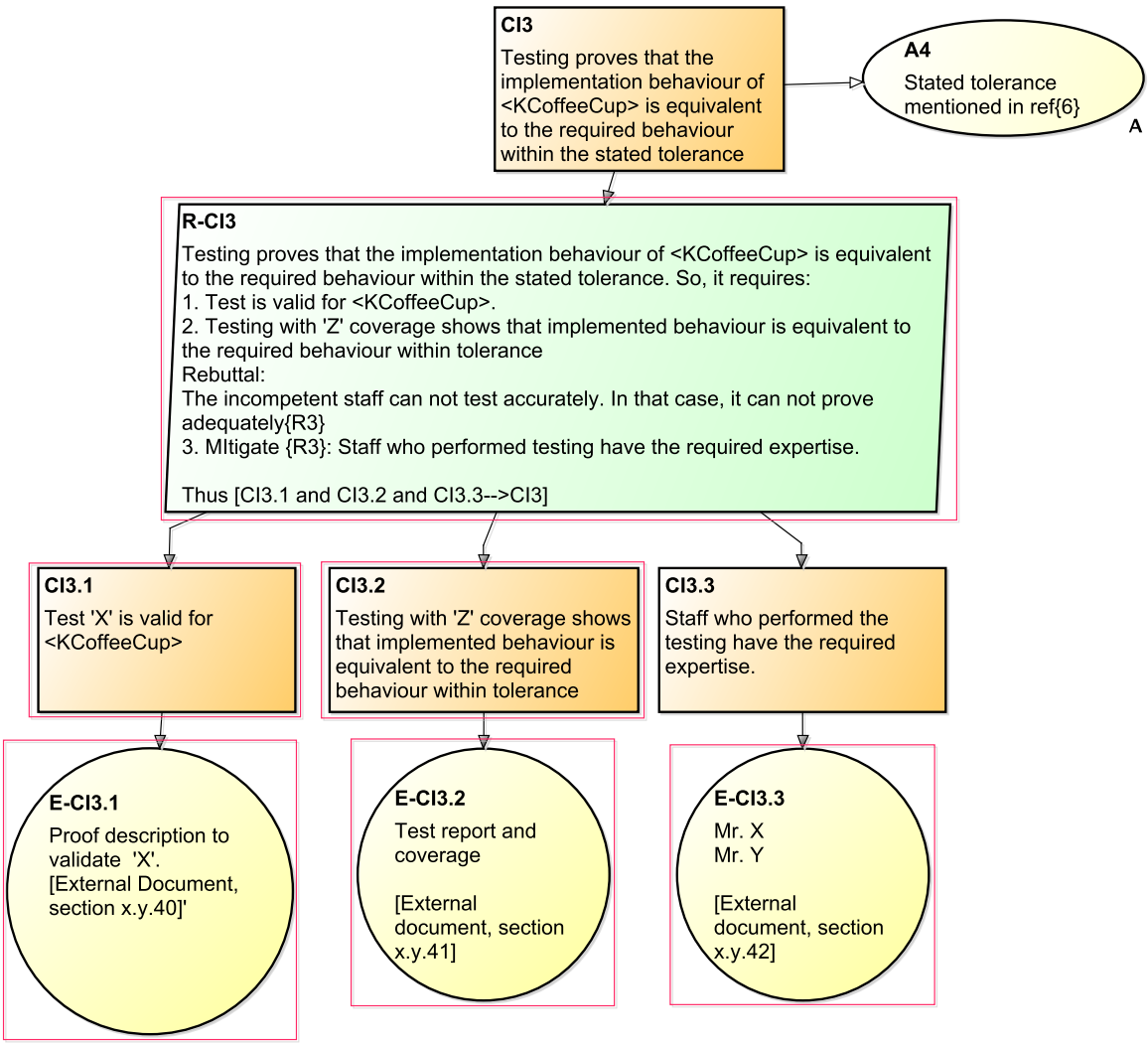


Figure 8.15: An argument of an AC for a coffee cup for alternative evidence

Figures 8.5, 8.6, 8.7, 8.8, 8.9, 8.10, 8.11 show different artefacts affected in hazard analysis argument branch by a likely change based on pairwise comparisons using "mental models" and/or experience. Figure 8.12 shows a production argument branch affected by a likely change. Figure 8.13 shows a change management argument branch in the AC. Figure 8.14 shows assumptions related argument branches. Figure 8.15 shows affected artefacts due to an alternative evidence in AC. There are also undeveloped claims, 'CPM3', 'CPM2.3' and 'CPM2.4' shown in figure 8.12.

We use '(Review) SystemVariabilityInAC' to review how an anticipated

change/variation in a system affects an AC.

Checking rules for *(Review) SystemVariabilityInAC*

Concerning Rule (1): We find that an anticipated change affects fewer artifacts in that AC. We found that 20 artifacts (e.g. claims, arguments, evidence) of AC need to be changed. Furthermore, we find that an anticipated change affects lower-level artifacts in that AC. For instance, in figure 8.12, evidence 'E-CPM1.2' and 'E-CPM2.1' need to be changed. Moreover, we find that a change management process for an anticipated change can be done in an independent branch of an AC.

We use '(Review) RebuttalsInAC' to check whether an anticipated rebuttal affects claims, arguments or evidence.

Checking rules for *(Review) RebuttalsInAC*

Concerning Rule (1): We find that three evidence ('E-CA1', 'E-CA3' and 'E-CA2.1.1' in figure 8.14) of that AC need to be changed. Moreover, we find that an anticipated rebuttal requires three evidence to be changed in an independent argument branch.

We use '(Review) AlternativeEvidence' to review changes due to alternative evidence.

Checking rules for *(Review) AlternativeEvidence*

Concerning Rule (1): We find that alternative evidence comply with acceptance criteria.

Concerning Rule (2): We find that evidence 'E-CI3.1' and 'E-CI3.2' affect terminal claims 'CI3.1' and 'CI3.2' but 'E-CI3.3' does not affect a terminal claim 'CI3.3' in figure 8.15. Furthermore, we find that terminal claims 'CI3.1' and 'CI3.2' may affect an argument 'R-CI3.

***GenerateRecommend* produces the following recommendations:**

- Suggestion ('sugDesc'): The rules and comparison reveal that AC is robust based on pairwise comparisons using "mental models" and/or experience. Furthermore, undeveloped claims, 'CR5', 'CI1.3', 'CL1.1.4', 'CI1.1.2.1', and 'CI1.1.2.3' are not affected due to changes. Thus, no

further action is required.

- Note delivered ('extra'):

  1. A likely change/variability in a system introduces changes in an AC artifacts. We consider the following anticipated change: the dimension of a container of coffee cup changes. This change may be height or diameter, or both of a container of a coffee cup.
  2. A likely rebuttal in an AC may violate claims, arguments or evidence; We consider the following anticipated change: A new operational assumption may violate the previous assumption and initiate changes in an AC.
  3. An alternative evidence in an AC may violate the existing terminal claims. We consider the following anticipated change: A new testing may affect terminal claims and/or upper claims and/or arguments.

8.1.1.4 Understandability

We use four checks to perform *understandability* evaluation.

We use '(Review) FontAttributes' to check font attributes of the AC.

Checking rules for *(Review) FontAttributes*

Concerning Rule (1): We identify that an acceptable font ('Arial') is used in documenting the AC.

Concerning Rule (2): We identify that 'font-size' is '18' with 'Plain' style and 'font-colour' is 'Black' and they are appropriate for viewing both on screen and in print.

Concerning Rule (3): It does not apply to the AC because it is a graphical notation.

We use '(Review) ModuleArgument' for reviewing module arguments.

Checking rules for *(Review) ModuleArgument*

Concerning Rule (1): We find that modules in the AC make it easier to comprehend. Long branches of arguments fit in one page due to modules. Following claims are formed into modules: 'CR', 'CI', 'CPM', 'CA', 'CR1',

'CR2', 'CR3', 'CR4', 'CR6','CR2.1', 'CR2.2', 'CR2.2.1.1', 'CI1', 'CI2', 'CI3', 'CI4', 'CI1.1', 'CI1.2', 'CI1.1.1', 'CI1.1.2', 'CI1.1.3', 'CI1.1.2.2'.

We use '(Review) IntersectAssociation' for reviewing intersection of association arcs.

Checking rules for *(Review) IntersectAssociation*

Concerning Rule (1): There is no intersection of association arcs in the AC, and it is comprehensible.

We use '(Review) SameLevelClaim' and apply rule (1) only because of graphical notation.

Checking rules for *(Review) SameLevelClaim*

Concerning Rule (1): Claims, argument, evidence are in the same horizontal position based on decomposition level. Furthermore, undeveloped claims, 'CPM3', 'CPM2.3', 'CPM2.4', 'CR5', 'CI1.3', 'CL1.1.4', 'CI1.1.2.1', and 'CI1.1.2.3' are also in the same horizontal position based on decomposition level.

***GenerateRecommend* produces the following recommendations:**

- Suggestion ('sugDesc'): As the structure does not violate any rule, there is no further action required.
- Note delivered ('extra'):
  1. Font 'Arial' is used with size '18', 'Plain' style, and 'Black' colour for ease of viewing.
  2. Module is formed when it does not fit on one page without clear readability.
  3. An argument branch in the AC complies with 'ALARP' pattern [99].

8.1.1.5 Efficiency

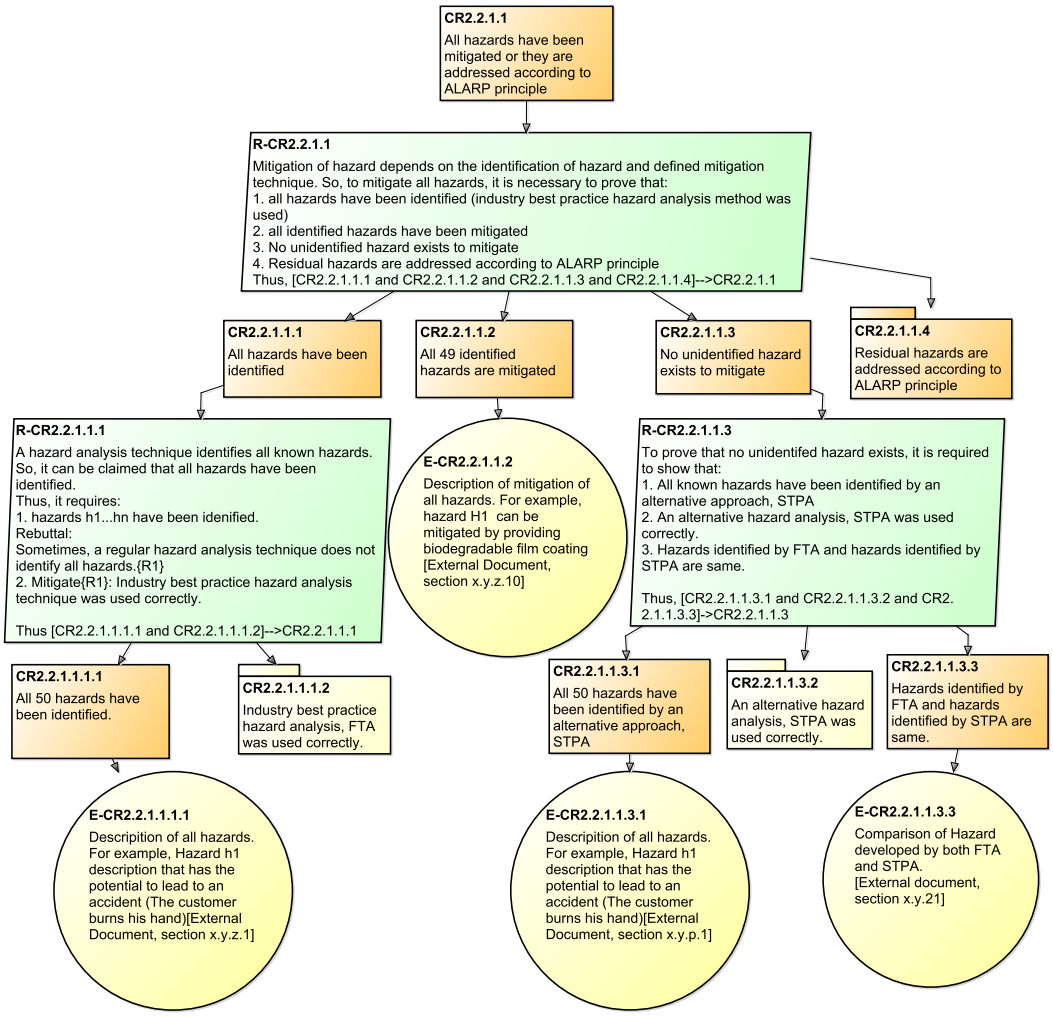


Figure 8.16: Hazard Analysis argument branch of an AC for a coffee cup

We use three checks to perform *efficiency* evaluation.

We use '(Review) FacilitateACDevelopment' to evaluate how it facilitates an AC development.

Checking rules for *(Review) FacilitateACDevelopment*

Concerning Rule (1): We identify that an argument pattern ALARP is instantiated in an argument branch starting with a claim 'CR2.2.1.1.4'. This pattern is instantiated manually with the help of 'Astah GSN' [98] tool.

Concerning Rule (2): We identify a single claim representing the same type of claims. For instance, claims 'CR2.2.1.1.1.1' and 'CR2.2.1.1.3.1' in figure 8.16 show all 50 hazards are identified by two different methods, FTA and STPA respectively. Similarly evidence 'E-CR2.2.1.1.1.1' and 'E-CR2.2.1.1.3.1' show description of all hazards by FTA and STPA respectively.

We use '(Review) FacilitateSystemDevelopment' to review how an AC facilitate system development.

Checking rules for *(Review) FacilitateSystemDevelopment*

Concerning Rule (1): We find that evidence guides the system development. For instance, evidence 'E-CR2.2.1.1.1.2.2' in figure 8.2 shows competency of people to perform FTA analysis. Similarly, an AC in appendix C shows evidence related to processes: E-CR2.1.2(prototype generation), E-CR2.2.1.1.1.2.1(FTA execution steps), E-CR2.2.1.1.3.2.1 (STPA execution steps), E-CR3.1 (development from requirements with current technology), E-CR3.2 (feasible development), E-CR6.2(interaction), E-CI1.1.1.1(design process of container), E-CI1.2.1.1 (design process of lid), E-CI4.1(safety assessment process during implementation), E-CPM1.2 (production process), E-CPM1.3(safety assessment process), E-CPM2.1 (service process); evidence related to product: E-CR2.1.1 (test case definition), E-CR2.1.3.1.1 (requirements of container), E-CR2.1.3.1.2 (requirements of sleeve), E-CR2.1.3.1.3 (requirement of lid), E-CR2.1.3.1.4 (requirement of interaction between all components), E-CR2.2.1.2 (description of mitigation), E-CR2.2.1.1.1.1 (all hazards description by FTA), E-CR2.2.1.1.3.1 (all hazards description by STPA), E-CR2.2.1.1.3.3 (comparison list of hazards), E-CR2.2.1.1.1.2.2 (review report for FTA), E-CR2.2.1.1.3.2.2 (review report for STPA), E-CR4 (testable criteria document), E-CR6.1 (consistent criteria proof for requirement), E-CR6.3 (traceability matrix for stakeholders), E-CI1.1.2.2.12(qualification test result of biodegradable film use), E-CI1.1.2.2.3 (bottom part satisfy safety requirement), E-CI1.2.2 (traceability matrix for lid), E-CI1.2.3 (specification of lid), E-CPM1.1 (production plan), E-CA2 (assumption descriptions); evidence related to credentials of people: E-CR2.1.3.3(people to review process), E-CR2.2.1.1.1.2.3 (people execute FTA), E-CR2.2.1.1.3.2.3 (people execute STPA), E-CI1.1.1.2 (people perform design of container), E-CI1.2.1.2(people perform design of lid),

E-CI2.3 (people perform mathematical validation of 'Z'), E-CI3.2 (people perform testing 'X'). E-CI4.2 (people perform safety assessment), E-CA1(the user aware documented assumption); evidence related to user acceptance test: E-CR1.2, E-1.2.2 and evidence related to validation/verification: 12. E-CI2.1 (validation of mathematical method 'Z'), E-CI3.1 (validation of test method 'X'), E-CI2.2 (verification of behaviour), E-CPM2.2 (verification steps). We do not find any evidence for following claims, 'CPM3', 'CPM2.3', 'CPM2.4', 'CR5', 'CI1.3', 'CL1.1.4', 'CI1.1.2.1', 'CI1.1.2.3' as they are undeveloped (i.e. they do not have any supporting argument branches).

Concerning Rule (2): We find that the acceptance criteria guide in system development—for instance, acceptance criteria in figure 8.2 guide in choosing the right team by mentioning the required credentials. Similarly, appendix C shows acceptance criteria for all other evidence.

We use '(Check) FacilitateReview' to facilitate the review process.

Checking rules for *(Check) FacilitateReview*

Concerning Rule (1): We find that the argument with starting claim 'CR2.2.1.1.4' compies with the 'ALARP' pattern. The AC does not mention all attributes of the 'ALARP' pattern.

Concerning Rule (2): We do not find any rationale and context of using the graphical notation in documenting the AC.

***GenerateRecommend* produces the following recommendations:**

- Suggestion ('sugDesc'):
  1. It is recommended to add rationale and context of using a GSN in documenting an AC.
- Note delivered ('extra'):
  1. Reference [99] defines the argument pattern 'ALARP' that is instantiated in the AC.
  2. An ACT is provided in the appendix C to show acceptance criteria.
  3. We use 'Astah GSN' [98] tool to document our AC for a coffee cup.

8.1.2 Content evaluation

This section illustrates the validation of the proposed content evaluation processes from a developer's perspective.

8.1.2.1 Convincing basis

This section provides validation results of our *convincing basis* evaluation. We use four checks to perform an evaluation.

We use 'TopLevelClaimCheck' to evaluate the top-level claim in figure 8.1.

Checking rules for *TopLevelClaimCheck*

Concerning Rule (1): We find that the top-level claim, "TopClaim, C" consists of two parts: the subject "The coffee cup <KCoffeeCup>" specifies the system and the predicate "is safe in its intended environment, and its intended uses" specifies the critical 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.

We use 'SubClaimCheck' to check all sub-claims. The sub-claims 'CR,' 'CI,' 'CPM' and 'CA' are represented by modules (in figure 8.1) as they contain implicit argument branches and other claims shown in appendix C.

Checking rules for *SubClaimCheck*

Concerning Rule (1): The meaning of each sub-claim is clear and does not create any ambiguity. For instance, Claim 'CR2.2.1.1.1.2.2' assures the competency of people in performing 'FTA' in figure 8.2.

Concerning Rule (2): We do not find any clarification for any necessary term mentioned in the sub-claims.

Concerning Rule (3): , we find that all claims related to process, product and people are clarified to support upper-level claims; for instance, claim '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, decommissioning, and operational assumptions.

Concerning Rule (4): We find necessary assumptions. For instance, an assumption 'A3' supports claim 'CI2', and an assumption 'A4' supports claim 'CI3'.

Concerning Rule (5): We find that evidence support terminal claims. For instance, a terminal claim 'CR2.2.1.1.1.2.2' is supported by evidence 'E-CR2.2.1.1.1.2.2' and acceptance criteria 'E-CR2.2.1.1.1.2.2_ACT' are illustrated.

We use 'ExplicitArgument' to evaluate the explicitness of an argument.

Checking rules for *ExplicitArgument*

Concerning Rule (1): We find that arguments describe explicit reasoning of how sub-claims support upper-level claims. For instance, argument 'R' describes explicit reasoning of how sub-claims ('CR,' 'CI,' 'CPM' and 'CA') support top-level claim 'Top 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 for arguments.

We use 'ConfirmationBias' to review confirmation bias.

Checking rules for *ConfirmationBias*

Concerning Rule (1): We find that arguments demonstrate rebuttals wherever they are identified with possible counters explicitly. For instance, an argument 'R' demonstrates rebuttals with possible counters.

Concerning Rule (2): We do not find any evidence to support those rebuttals.

Concerning Rule (3): We find that all evidence comply with acceptance criteria. For instance, figure 8.2 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_ACT'.

***GenerateRecommend* produces the following recommendations:**

- Suggestion ('sugDesc'):

    1. It is recommended to clarify key terms: documented assumptions in claim 'CA', a standard for coffee cup requirements in claim 'CR1.2', industry best practice hazard analysis mentioned in claim 'CR2.2.1.1.1.2', academic/industry review in claim 'CR2.2.1.1.1.2.3', feasibility in claim 'CR3', a biodegradable film in claims 'CI1.1.2.2.1', 'CI1.1.2.2.2' and 'CI1.1.2.2.3' and method 'Z' in claim 'CI2.1'.

- Note delivered ('extra'):

    1. Rebuttals are mentioned in arguments to challenge our arguments, and they are resolved. For instance, an argument 'R' shows two rebuttals and two mitigations of those rebuttals.
    2. Acceptance criteria are shown in figure 8.2. It is necessary to show compliance with evidence. For all evidence, there are acceptance criteria for compliance checking.

8.1.2.2 Rigour of the argument

In our evaluation process, we identify that we have informal arguments. We use 'CheckInformalArgument.' We also use 'CheckClaimForValidity' and 'CheckRationaleForValidity.

We use 'CheckInformalArgument' for reviewing argument explicitly.

Checking rules for *CheckInformalArgument*

Concerning Rule (1): We find that the arguments are defined inductively with necessary steps, including rebuttals (they exist in some cases) to support the upper-level claims, and they are complete and consistent. For instance, an argument 'R' has four factors to demonstrate inductive reasoning (including two mitigation of two rebuttals). On the other hand, an argument 'R-CR1.2' has four factors to demonstrate inductive reasoning (there is no known rebuttal). However, claims 'CPM3', 'CPM2.3', 'CPM2.4', 'CR5', 'CI1.3', 'CL1.1.4', 'CI1.1.2.1', and 'CI1.1.2.3' do not have supporting argument branches.

Concerning Rule (2): We do not identify any justification for any argument. For instance, an argument 'R' is not justified by any justification.

Concerning Rule (3): We do not find any key term, e.g. context or assumption, to support arguments. For instance, an argument 'R' does not have any supporting contexts.

Concerning Rule (4): We find that some arguments also mentions rebuttals. For instance, an argument 'R' mentions two rebuttals, 'R1' and 'R2'. Similarly, others are 'R-CR', 'R-CR2.1.3.1', 'R-CR2.2.1.1.1.2', 'R-CI', 'R-CI1.1.1', 'R-CI1.1.2', 'R-CI1.2', 'R-CI1.2.1', 'R-CI2', 'R-CI3', 'R-CI4', 'R-CPM'. The rest of them do not have rebuttals. Rebuttals mentioned in those arguments are consistent.

Concerning Rule (5): We find that the mitigation of each rebuttal is illustrated. For instance, mitigations 'CPM' and 'CA' in argument 'R' resolve two rebuttals, 'R1' and 'R2'.

Concerning Rule (6): We find that an argument branch considers different phases of the development process.

We use 'CheckClaimForValidity' for reviewing validity of claims.

Checking rules for *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. For instance, sub-claims ('CR', 'CI,' 'CPM' and 'CA'-they are represented as modules) are adequately complete and consistent and valid supported by arguments.

Concerning Rule (2): We find that arguments, as mentioned earlier, have defined rebuttals and mitigations. Rebuttals are complete, but there is no proof to check the validity of those rebuttals. Claims 'CPM' and 'CA' are mitigation of rebuttals that may violate claims 'CR' and 'CI', but no proof exists to validate those rebuttals.

We use 'CheckRationaleForValidity' for reviewing rationale.

Checking rules for *CheckRationaleForValidity*

Concerning Rule (1): We find that no justification exists to support an argument.

***GenerateRecommend* produces the following recommendations:**

- Suggestion ('sugDesc'):

    1. It is recommended that justifications should exist to support arguments.
    2. It is recommended that environmental, operational conditions during the production and maintenance stage be clarified.
    3. It is recommended to develop supporting argument branches for claims: 'CPM3', 'CPM2.3', 'CPM2.4', 'CR5', 'CI1.3', 'CL1.1.4', 'CI1.1.2.1', 'CI1.1.2.3'.

- Note delivered ('extra'):

    1. An argument branch complies with 'ALARP' argument pattern [99].
    2. An argument 'R' provides reasoning that follows a system development life cycle.

8.1.2.3 Quality of the hazard analysis

We use four checks to perform *quality of the hazard analysis* evaluation. In our evaluation process, figure 8.16 shows a hazard analysis argument branch. Moreover, figures 8.17 and 8.18 show two different hazard analysis argument branches in an AC. Furthermore, figure 8.19 shows an argument branch for mitigation of hazards.

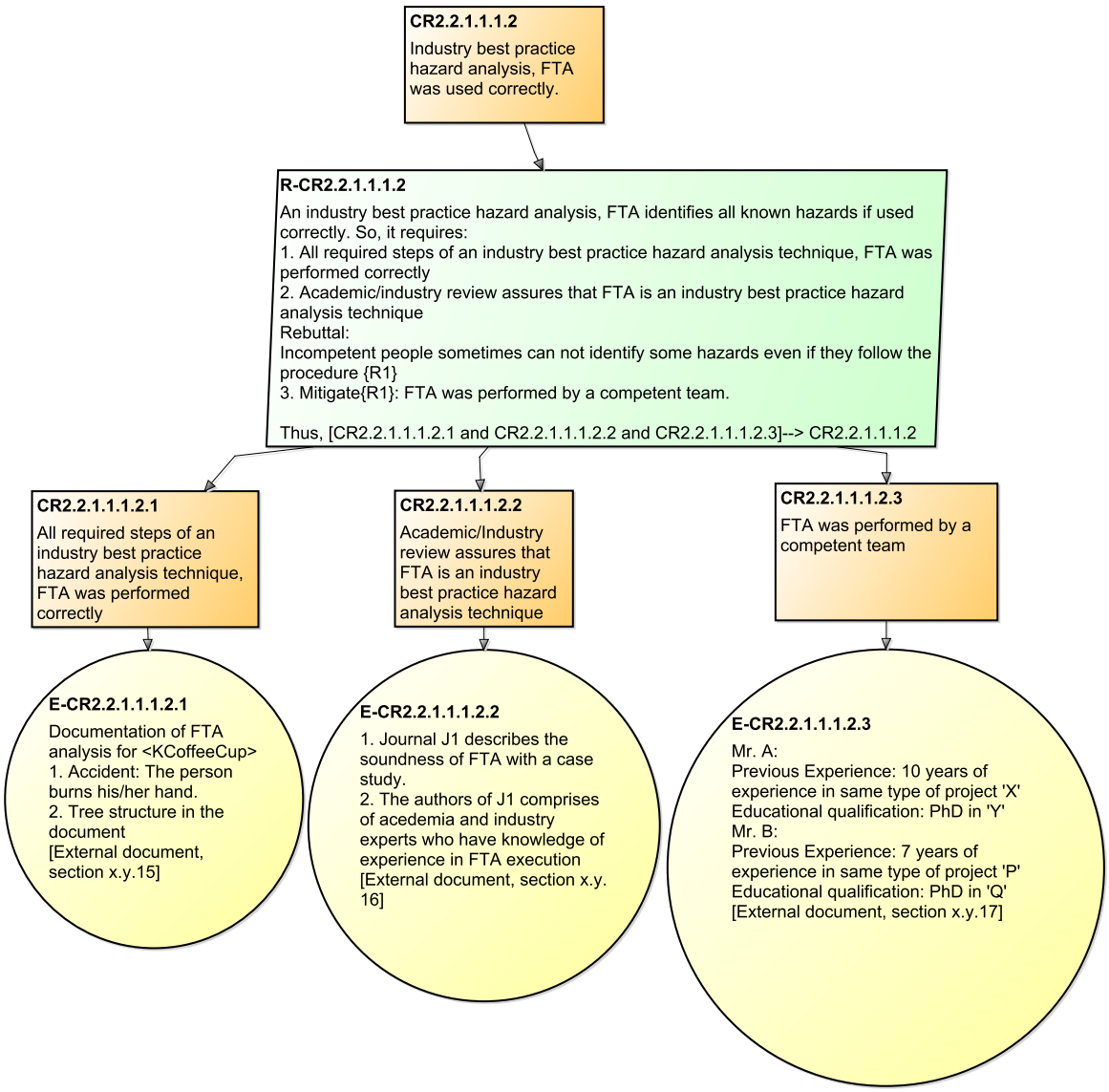


Figure 8.17: FTA argument branch of an AC for a coffee cup

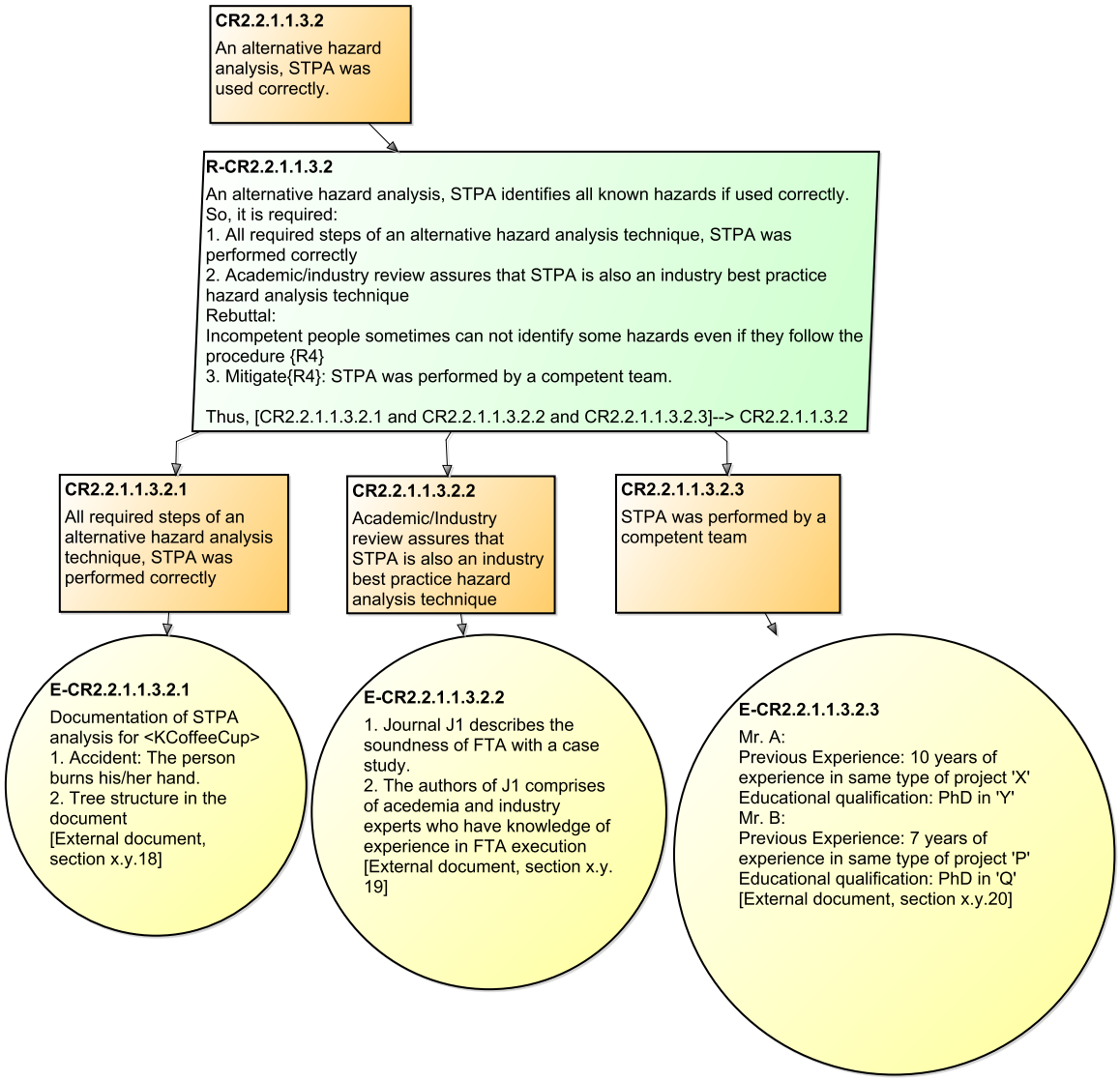


Figure 8.18: STPA argument branch of an AC for a coffee cup

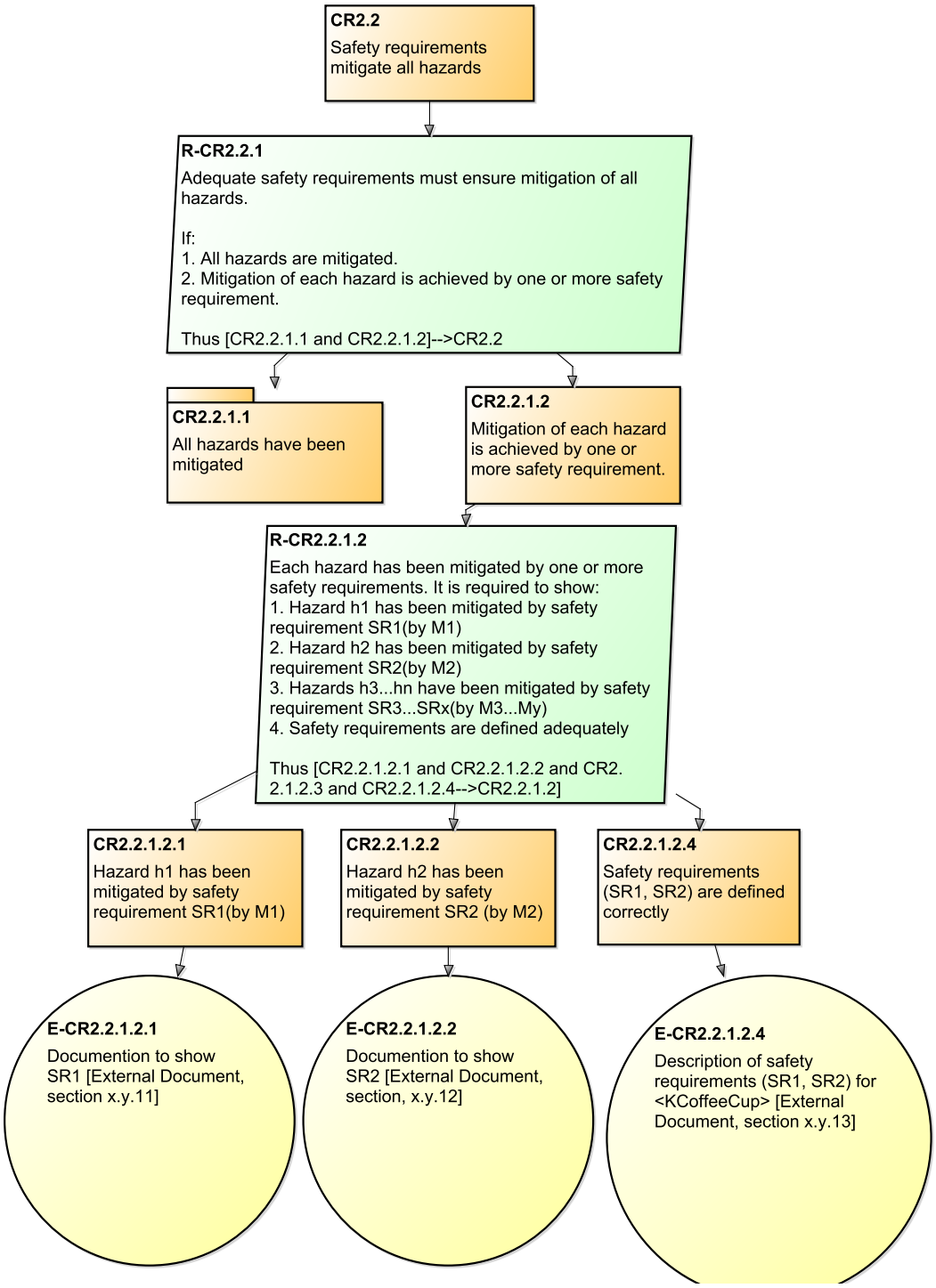


Figure 8.19: Mitigation of hazards in an AC for a coffee cup

Process 'ClaimEvidenceComparison' looks to compare identified hazards with a known hazard list adopted by regulatory organizations or standards. There is no regulator's recommended hazard list that can guide whether our hazard analysis method identifies all known hazards or not. Concerning this, we decided to use two different hazard analysis methods to identify hazards adequately, and thus we chose "Fault Tree Analysis (FTA)" and "System Theoretic Process Analysis (STPA)" because of their acceptability in industry and academia. We compare identified hazards by FTA with identified hazards by STPA to show we have identified all probable hazards.

We use 'ClaimEvidenceMitigation' to evaluate mitigation strategies.

Checking rules for *ClaimEvidenceMitigation*

Concerning Rule (1): We find that a claim mentions mitigation of all identified hazards. Claim 'CR2.2.1.1.2' mentions the mitigation of all identified hazards.

Concerning Rule (2): We find that evidence support mitigation claims. Evidence 'E-CR2.2.1.1.2' shows the mitigation steps of each hazard.

Concerning Rule (3): We find that a claim of safety requirements relates to mitigation steps. Claim 'CR2.2.1.2' mentions safety requirements for mitigation.

Concerning Rule (4): We find that evidence supports the claim containing safety requirement description for each mitigation of hazards, and it complies with the acceptance criteria.

We use 'ClaimEvidenceHAMethod' to check the hazard analysis method.

Checking rules for *ClaimEvidenceHAMethod*

Concerning Rule (1): We find that two claims mention the effectiveness of hazard analysis methods. In figure 8.17, claim 'CR2.2.1.1.1.2.2' mentions that FTA is an industry best practice hazard analysis by expert appraisal.

Concerning Rule (2): We find that evidence supports the claim with an expert appraisal to prove soundness. Evidence 'E-CR2.2.1.1.1.2.2' and 'E-CR2.2.1.1.3.2.2' show proof to support claims 'CR2.2.1.1.1.2.2' and 'CR2.2.1.1.3.2.2' respectively. Both evidence comply with the acceptance criteria mentioned in appendix C.

Concerning Rule (3): We find that evidence document the execution steps of a hazard analysis. Evidence 'E-CR2.2.1.1.1.2.1' and 'E-CR2.2.1.1.3.2.1' document implemented procedure steps of hazard analyses FTA and STPA respectively. Both evidence comply with the acceptance criteria.

We use 'ClaimEvidencePeopleCredentials' to check the qualification of people performing hazard analysis.

Checking rules for *ClaimEvidencePeopleCredentials*

Concerning Rule (1): We find that claim mentions competent people execute the hazard analysis. Claim 'CR2.2.1.1.1.2.3' mentions a competent team involve in FTA. Similarly, claim 'CR2.2.1.1.3.2.3' mentions a competent team involve in STPA analysis.

Concerning Rule (2): We find that evidence illustrates the credentials of people to support claims of competent teams. Evidence 'E-CR2.2.1.1.1.2.3' documents the credentials of people involve in FTA analysis. Similarly, evidence 'E-CR2.2.1.1.3.2.3' documents the credentials of people involve in STPA analysis. Both evidence comply with the acceptance criteria.

We use 'ClaimEvidenceComparison' to compare hazards identified by FTA with regulatory organizations or standards recommended known hazard list. We mentioned that we do not find any regulator's approved known hazard list. Thus, Rule (2) does not apply to our example.

Checking rules for *ClaimEvidenceComparison*

Concerning Rule (1): Instead of known hazards, we find the comparison between hazards identified by two different hazard analysis methods, FTA and STPA. Claim 'CR2.2.1.1.3.3' shows that both methods identified the same hazards.

***GenerateRecommend* produces the following recommendations:**

- Suggestion ('sugDesc'):
    1. It is recommended to add justifications to support arguments.
- Note delivered ('extra'):

1. Appendix C demonstrates the acceptance criteria for evidence.
2. There is no known hazard list found for a coffee cup adopted by regulators or standards. So, it is not possible to check with known hazards.
3. We choose FTA as a primary hazard analysis technique and STPA as a secondary hazard because of their acceptability in industry and academia; thus, selecting a hazard analysis method is subjective.

8.1.2.4 Arguing completeness

Three checks evaluate *arguing completeness* of an AC. Figures 8.20 and 8.21 show requirement argument branches.

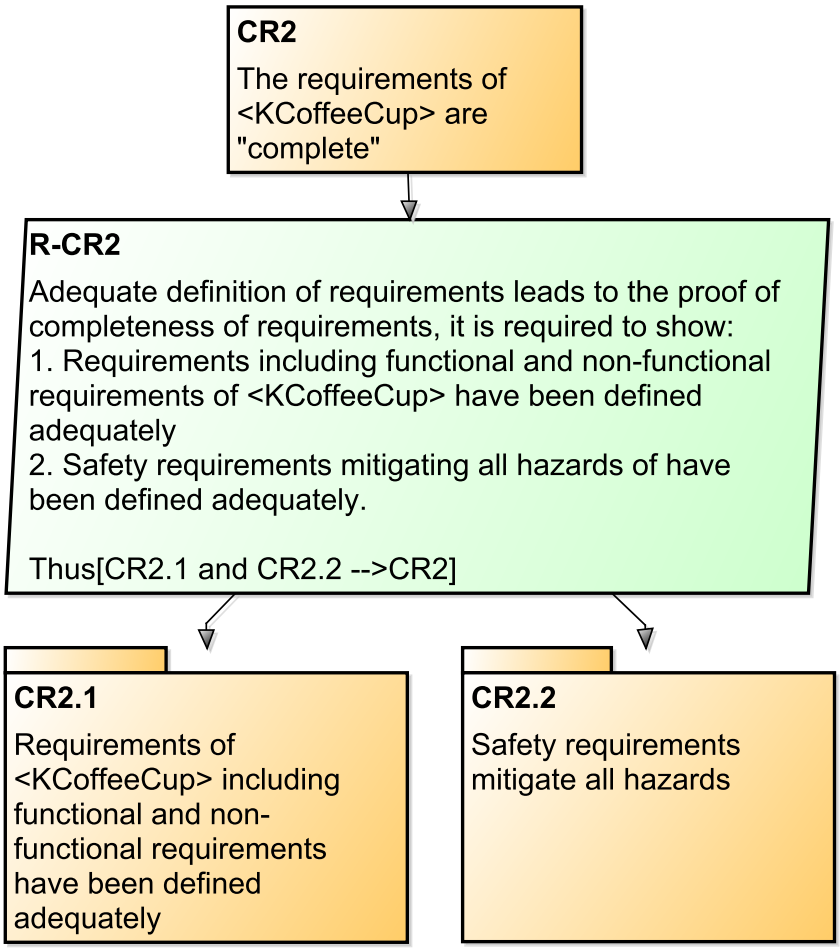


Figure 8.20: Start of requirements complete argument branch of an AC for a coffee cup

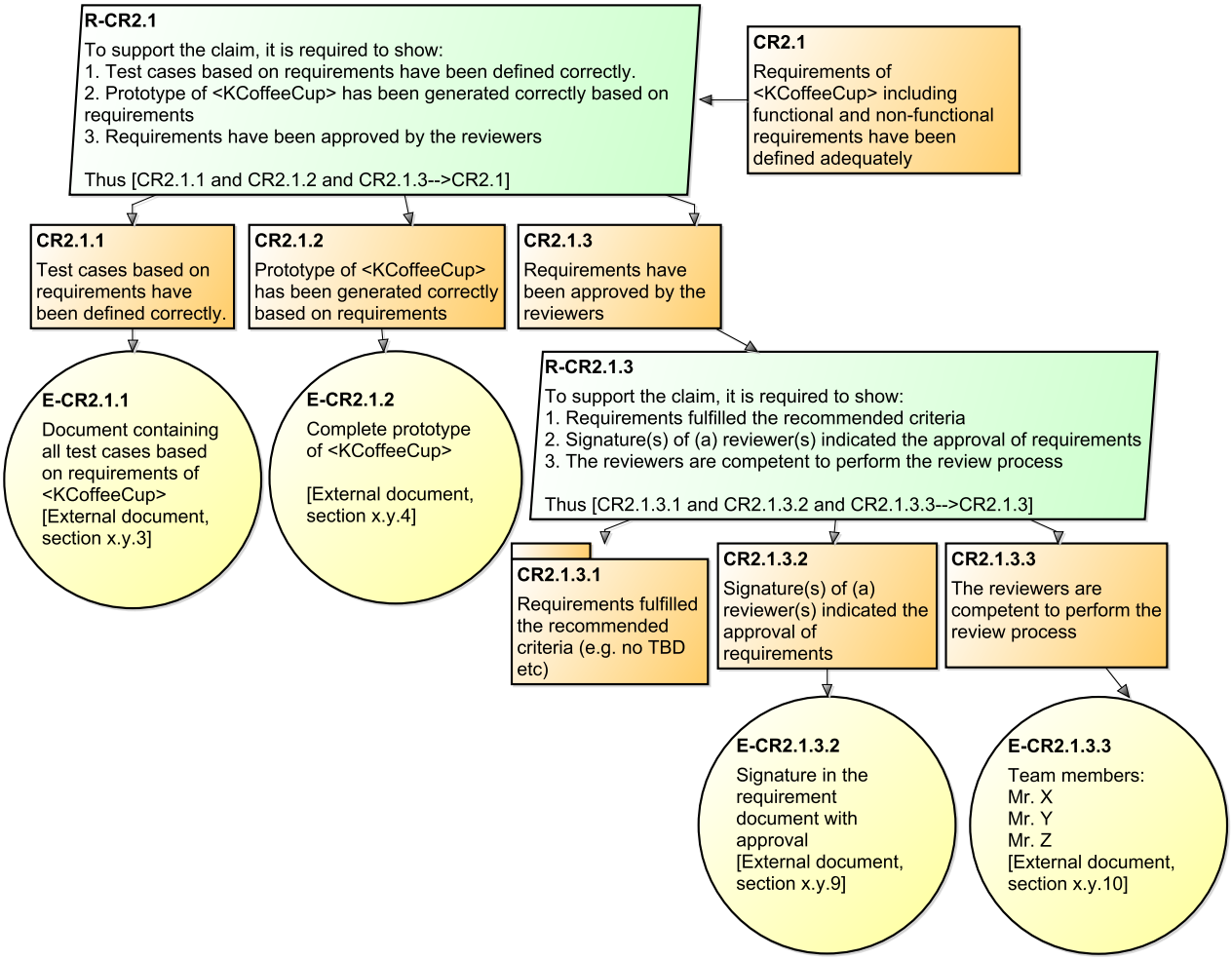


Figure 8.21: More of requirements complete argument branch of an AC for a coffee cup

<KCoffeeCup> is considered a feasible system due to simple structure, thus 'LargeSystemCompleteness' process does not apply to our example. We use 'HazardIdentificationComplete' to evaluate completeness of hazards. Previous criterion *Quality of the hazard analysis* discusses completeness considering methodology oriented approach along with competency of people.

Checking rules for *HazardIdentificationComplete*

Concerning Rule (1): We find arguments. Figure 8.16 shows that argument 'R-CR2.2.1.1.1' mentions that all known hazards have been identified.

Similarly, figures 8.17 and 8.18 show that arguments 'R-CR2.2.1.1.1.2' and 'R-CR2.2.1.1.3.2'.

Concerning Rule (2): We do not find any clarification or proof for the arguments.

Concerning Rule (3): We find claims related to mitigation of hazards. Figure 8.16 shows that claim 'CR2.2.1.1.1.2' mentions mitigation. Similarly, figures 8.17 and 8.18 show that claims 'CR2.2.1.1.1.2.3' and 'CR2.2.1.1.3.2.3' mention about mitigation of hazards.

Concerning Rule (4): We find that evidence illustrates mitigation to support claims mentioning mitigation. In figures 8.17 and 8.18, evidence 'E-CR2.2.1.1.1.2.3' and 'E-CR2.2.1.1.3.2.3' support claims 'CR2.2.1.1.1.2.3' and 'CR2.2.1.1.3.2.3' respectively.

We use 'FeasibleSystemCompleteness' to review completeness of a feasible system.

Checking rules for *FeasibleSystemCompleteness*

Concerning Rule (1): We find that arguments provide reasoning of completeness to support upper level claims. In figures 8.20, 8.21 and 8.19 arguments 'R-CR2', 'R-CR2.1' and 'R-CR2.2.1'show reasoning of complete requirements.

Concerning Rule (2): We do not find any terms clarified for arguments.

Concerning Rule (3): We find that claims mention proof or test to support completeness. In figures 8.20, 8.21 and 8.19, claims 'CR2', 'CR2.1', 'CR2.2', 'CR2.1.1', 'CR2.1.2', 'CR2.1.3', 'CR2.1.3.1', 'CR2.1.3.1.1', 'CR2.1.3.1.2', 'CR2.1.3.1.3' and 'CR2.1.3.1.4' discuss test, prototype and criteria for completeness of arguments.

Concerning Rule (4): We do not find any terms clarified for claims.

Concerning Rule (5): We find that evidence support claims mentioning test, prototypes. In figure 8.21 evidence 'E-CR2.1.1', 'E-CR2.1.2', 'E-CR2.1.3.1.1','E-CR2.1.3.1.2', 'E-CR2.1.3.1.3' and 'E-CR2.1.3.1.4' support claims mentioned above.

***GenerateRecommend* produces the following recommendations:**

- Suggestion ('sugDesc'):

1. It is recommended to describe necessary contexts, assumptions and justifications to support arguments for completeness.
2. It is recommended to develop supporting argument branches for claims: 'CPM3', 'CPM2.3', 'CPM2.4', 'CR5', 'CI1.3', 'CL1.1.4', 'CI1.1.2.1', 'CI1.1.2.3' for completeness evaluation.

- Note delivered ('extra'):

1. Arguments follow the inductive reasoning to support completeness;
2. In this scenario, testing checks the completeness of the requirements. On the other hand, both testing and mathematical analysis (argument 'R-CI') support completeness in the design phase.
3. Arguments for hazard analysis also argue why no unidentified hazard exists by using an alternative approach (already mentioned in *Quality of the hazard analysis* criterion).

8.1.2.5 Repeated arguments

Two checks evaluate *repeated arguments* of an AC. Figure 8.22 contains a hazard analysis argument branch. Figures 8.23 and 8.24 show an argument branch complying with ALARP pattern [99].

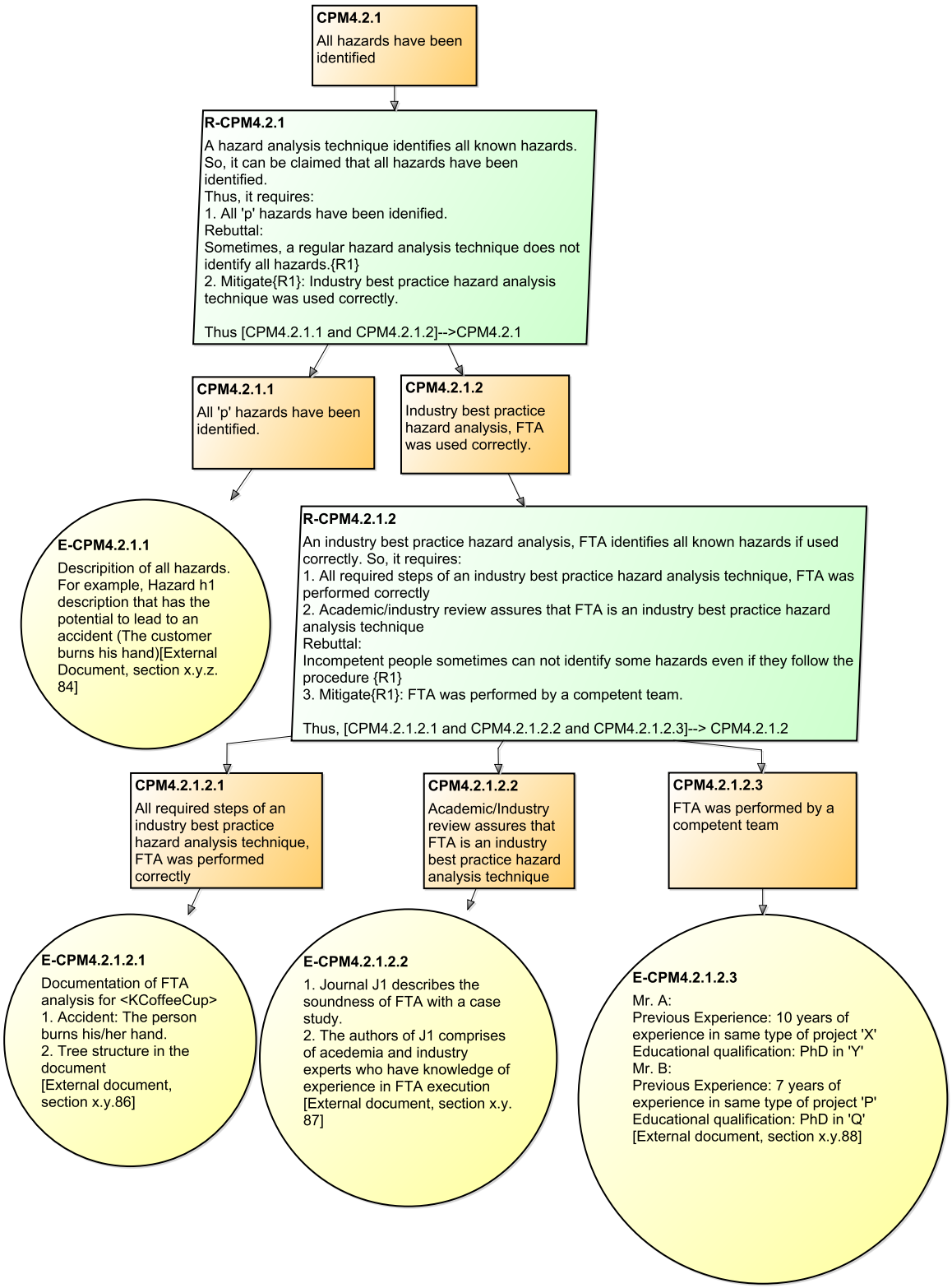


Figure 8.22: Hazard analysis argument branch due to likely changes of an AC for a coffee cup

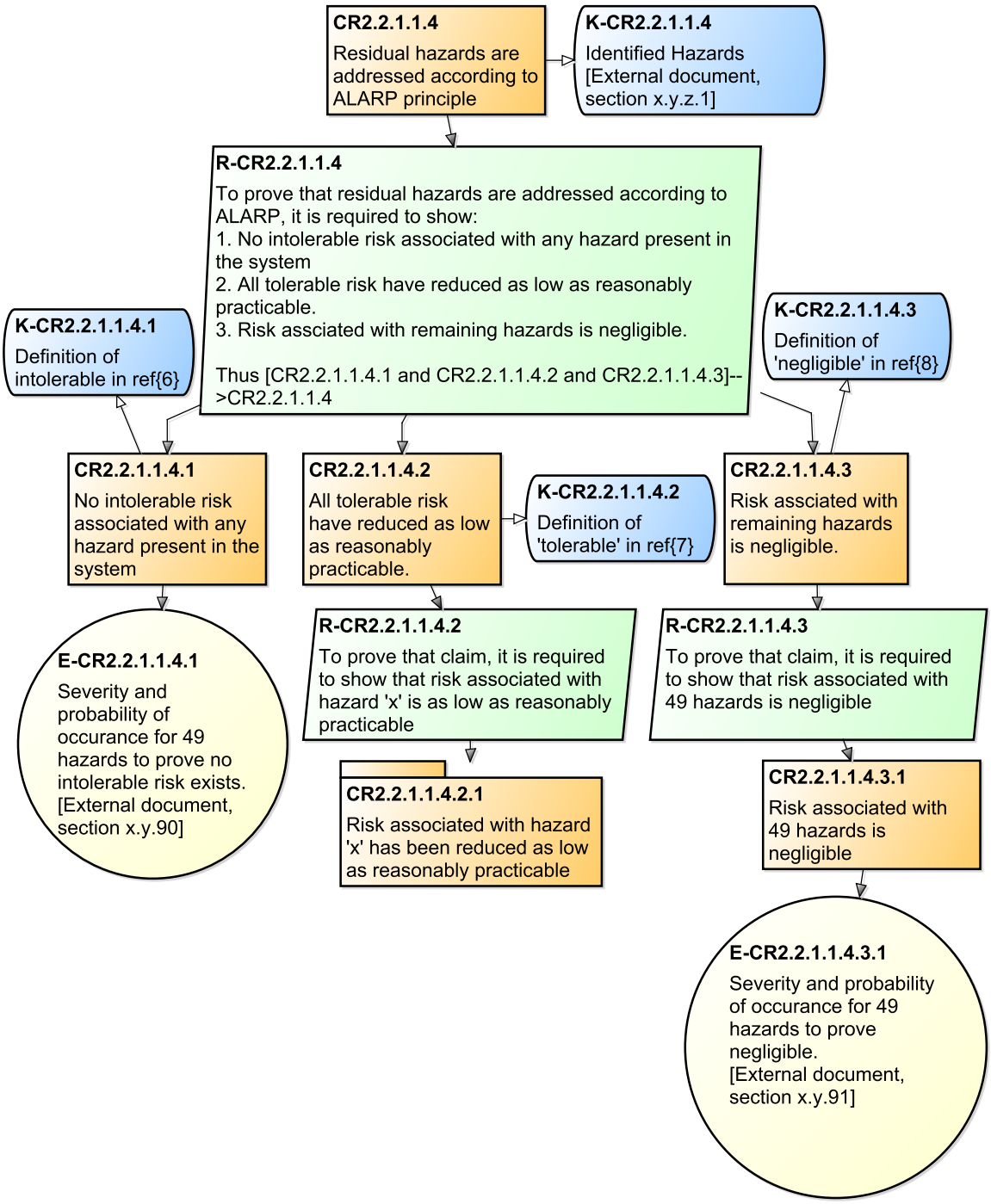


Figure 8.23: Instantiated ALARP argument branch of an AC for a coffee cup (part 1)

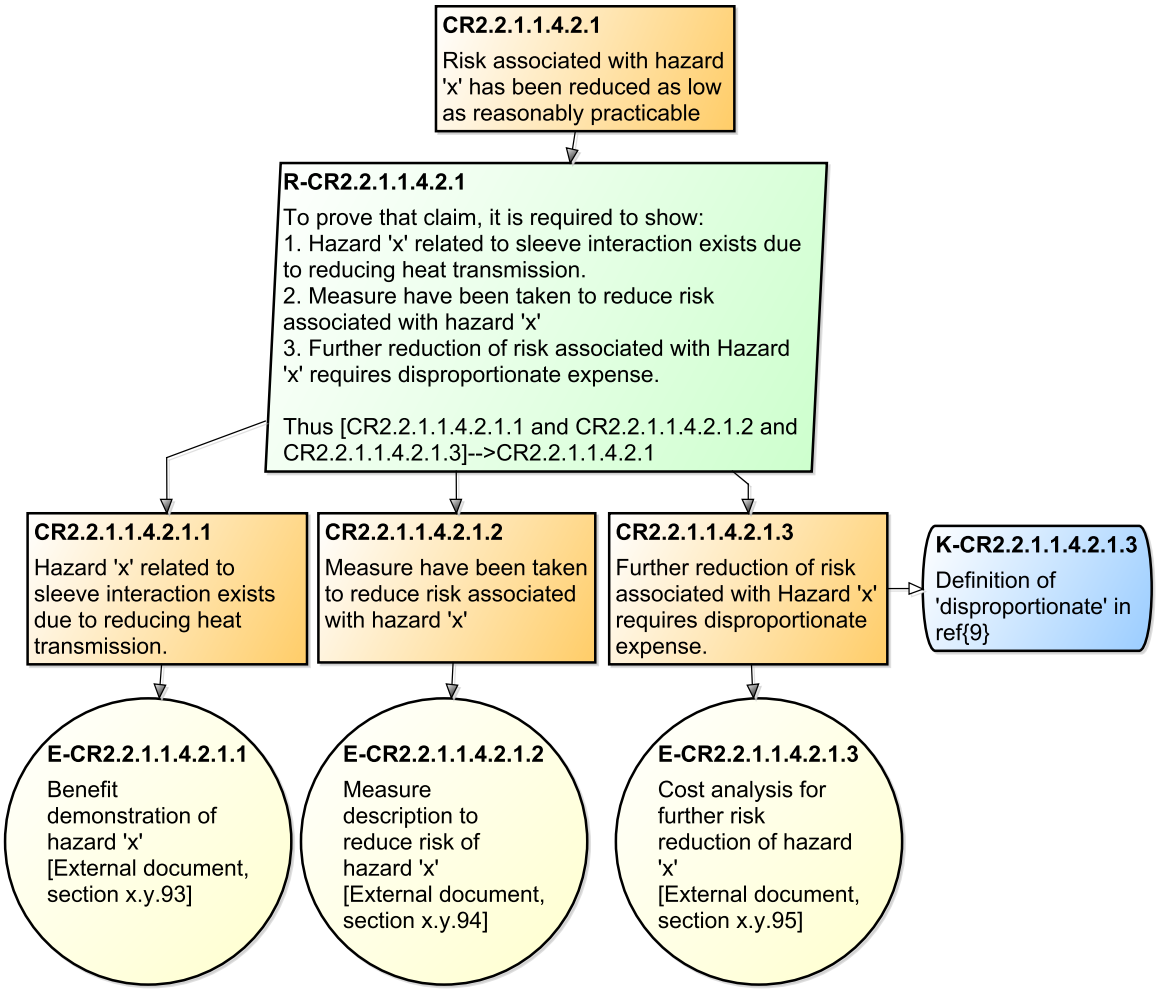


Figure 8.24: Instantiated ALARP argument branch of an AC for a coffee cup (part 2)

We use 'SimilarArgument' to check similar argument in an AC.

Checking rules for *SimilarArgument*

Concerning Rule (1): We find that an argument pattern is instantiated to a specific argument branch. An argument pattern *ALARP* is instantiated to an argument branch staring with claim 'CR2.2.1.1.4'. and it is supported by three subclaims 'CR2.2.1.1.4.1', 'CR2.2.1.1.4.2' and 'CR2.2.1.1.4.3' through argument 'R-CR2.2.1.1.4'.

Concerning Rule (2): We find that external note is mentioned to show

target argument branch.

Concerning Rule (3): We find that target argument branch complies with source branch. For instance, it is demonstrated that argument branch starting with claims 'CPM4.2.1' and 'CPM4.2.2' are target branches those are copied from argument branch with claims 'CR2.2.1.1.1' and 'CR2.2.1.1.2'.

We use 'ClaimArgumentEvidencePattern' to check claim, argument, evidence compliance with source argument with consistency.

Checking rules for *ClaimArgumentEvidencePattern*

Concerning Rule (1): We find that claims and evidence of an instantiated argument branch comply with claims and evidence of a pattern. The pattern does not provide any explicit argument. But instantiated one has explicit arguments to support upper claims. All claims 'CR2.2.1.1.4','CR2.2.1.1.4.1', 'CR2.2.1.1.4.2', 'CR2.2.1.1.4.3', 'CR2.2.1.1.4.2.1', 'CR2.2.1.1.4.3.1', 'CR2.2.1.1.4.2.1.1', 'CR2.2.1.1.4.2.1.2' and 'CR2.2.1.1.4.2.1.3' comply with claims 'G1', 'G2', 'G3', 'G4', 'G8', 'G9', 'G10', 'G11' and 'G12' respectively. Evidence 'E-CR2.2.1.1.4.1' complies with 'Sn1'. More evidence 'E-CR2.2.1.1.4.3.1' complies with criteria in [99].

Concerning Rule (2): We find that all terms in an instantiated branch comply with those of that argument pattern. Contexts 'K-CR2.2.1.1.4', 'K-CR2.2.1.1.4.1', 'K-CR2.2.1.1.4.2', 'K-CR2.2.1.1.4.3', 'K-CR2.2.1.1.4.2.1.3' comply with 'C1', 'C2', 'C3', 'C4' and 'C5' respectively.

Concerning Rule (3): We find that argument branch copied or instantiated is consistent with a neighbouring argument branch. Claims 'CR2.2.1.1.4', 'CR2.2.1.1.4.1', 'CR2.2.1.1.4.2', 'CR2.2.1.1.4.3', 'CR2.2.1.1.4.2.1', 'CR2.2.1.1.4.3.1', 'CR2.2.1.1.4.2.1.1', 'CR2.2.1.1.4.2.1.2' and 'CR2.2.1.1.4.2.1.3', 'E-CR2.2.1.1.4.1', 'E-CR2.2.1.1.4.3.1' are consistent with 'CR2.2.1.1.1', 'CR2.2.1.1.2', 'CR2.2.1.1.3' and their corresponding argument branches.

***GenerateRecommend* produces the following recommendations:**

- Suggestion ('sugDesc'):

    1. It is recommended to describe contexts, assumptions and justifications to support target arguments.

2. It is recommended to add an external note for a target argument branch to trace to the source argument branch.

- Note delivered ('extra'):

1. Claim 'CR2.2.1.1.4.1' is supported by evidence 'E-CR2.2.1.1.4.1' only because there is no intolerable risk associated with hazards.
2. Informal arguments exist in the compliant argument branch to show explicit reasoning to support upper-level claims.

8.1.2.6 ALARP

One check evaluates *ALARP*. An argument branch in figures 8.23 and 8.24 comply with ALARP pattern [99].

We use 'TolerableRiskArgument' to review tolerable risk arguments.

Checking rules for *TolerableRiskArgument*

Concerning Rule (1): We find that a claim mentions the benefit of having a hazard. Claim CR2.2.1.1.4.2.1.1 mentions the benefit of having a hazard.

Concerning Rule (2): We find that evidence support claim of benefits. Evidence, 'E-CR2.2.1.1.4.2.1.1, ' documents the benefit to support the claim, 'CR2.2.1.1.4.2.1.1'.

Concerning Rule (3): We find that claim mentions measures to reduce risk to tolerable. Claim 'CR2.2.1.1.4.2.1.2' mentions measures to mitigate a hazard 'x.'

Concerning Rule (4): We find that evidence document measures to reduce risk. Evidence 'E-CR2.2.1.1.4.2.1.2' documents measures to reduce risk.

Concerning Rule (5): We find a claim mentioning further reduction introduces a disproportionate expense. Claim 'CR2.2.1.1.4.2.1.3' mentioning further reduction introduces a disproportionate expense.

Concerning Rule (6): We find that evidence refers to expense report to support the claim. Evidence 'E-CR2.2.1.1.4.2.1.3' shows a cost analysis to support the claim.

Concerning Rule (7): We find that key terms are clarified. Context 'K-CR2.2.1.1.4.2.1.3' shows clarification of 'disproportionate'. There is no context, justification to support an argument.

***GenerateRecommend* produces the following recommendations:**

- Suggestion ('sugDesc'):
    1. It is recommended to describe necessary contexts, assumptions and justifications to support arguments for tolerable risk using ALARP principle.
- Note delivered ('extra'):
    1. An argument branch complies with the ALARP pattern [99].
    2. Context 'K-CR2.2.1.1.4' provides a reference of hazard list based on identified hazards from another claim, 'CR2.2.1.1.1'.
    3. Contexts 'K-CR2.2.1.1.4.1', 'K-CR2.2.1.1.4.2', 'K-CR2.2.1.1.4.3' and 'K-CR2.2.1.1.4.2.1.3' clarify 'intolerable', 'tolerable', 'negligible' and 'disproportionate' based on a company's perspective (e.g. in this scenario company 'K').
    4. The pattern does not provide any explicit argument to support upper-level claims. We developed explicit arguments to provide explicit reasoning.

8.1.2.7 Confidence

Figure 8.25 shows a confidence assessment of an argument branch using quantitative confidence assessment method [77]. The authors propose a quantitative approach to measure sufficiency, insufficiency and uncertainty using Dempster-Shafer Theory (DST). We apply that approach to measure confidence and later apply our approach to evaluate confidence.

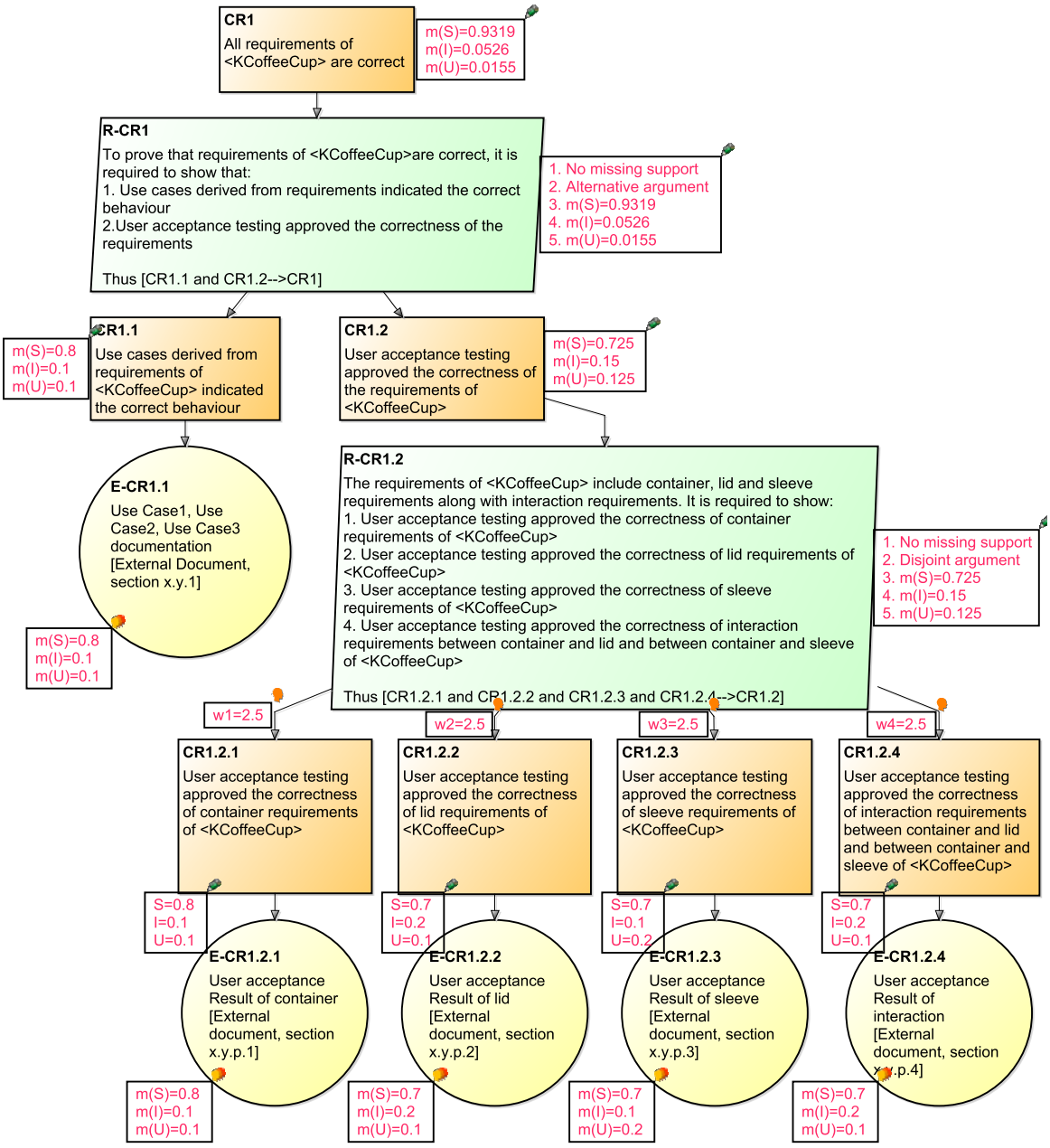


Figure 8.25: Confidence assessment of an argument of an AC for a coffee cup using a method proposed in [77]

We have four processes consisting of rules to review *confidence* assessment. We use 'EvidenceConfidence' to review the probability of evidence.

Checking rules for *EvidenceConfidence*

Concerning Rule (1): Experts estimate the probability of evidence by any means, e.g. discussion, experience. Evidence 'E-CR1.1', 'E-CR1.2.1', 'E-CR1.2.2', 'E-CR1.2.3', 'E-CR1.2.4' have probability of sufficiency, (m(S)), insufficiency (m(I)) and uncertainty (m(U)). This degree of belief is derived based on domain expert appraisal.

We use 'ClaimConfidence' to review quantitative confidence of claims.

Checking rules for *ClaimConfidence*

Concerning Rule (1): We find that a recommended approach estimates probabilities of claims. A proposed approach [77] estimates the probabilities of claims 'CR1' and 'CR1.2'.

Concerning Rule (2): We do not find any term for clarification.

Concerning Rule (3): We find the probability of terminal claims is the same as evidence as there is no explicit argument between them. Probabilities of sufficiency, insufficiency and uncertainty of terminal claims 'CR1.2.1', 'CR1.2.2', 'CR1.2.3', 'CR1.2.4', 'CR1.1' are same as probabilities of those of evidence 'E-CR1.2.1', 'E-CR1.2.2', 'E-CR1.2.3', 'E-CR1.2.4' and 'E-CR1.1'.

We use 'ArgumentConfidence' to review argument confidence assessment.

Checking rules for *ArgumentConfidence*

Concerning Rule (1): We find that probabilities of arguments are derived using a recommended approach. Probabilities sufficiency, insufficiency and uncertainty of arguments 'R-CR1' and 'R-CR1.2' are derived using a proposed approach by [77].

Concerning Rule (2): We do not find any term for clarification.

We use 'AssociationWeight' to review the weight of the association link. This process applies to graphical notation only.

Checking rules for *AssociationWeight*

Concerning Rule (1): We find that domain experts estimate the weights of associations. Weights of association among claims and arguments (R-CR1.2 and CR1.2.1), (R-CR1.2 and CR1.2.2), (R-CR1.2 and CR1.2.3) and (R-CR1.2 and CR1.2.4) are estimated by domain experts.

***GenerateRecommend* produces the following recommendations:**

- Suggestion ('sugDesc'):
  1. It is recommended that contexts, assumptions and justifications should exist to support arguments (R-CR1 and R-CR1.2)
  2. It is recommended to provide description of confidence assessment methodology in an external document.
- Note delivered ('extra'):
  1. The confidence assessment follows the methodology defined in [77].
  2. Probabilities of sufficiency, insufficiency of evidence are estimated by domain experts.
  3. There are two types of arguments (alternative and disjoint) are considered to measure sufficiency, insufficiency and uncertainty.
  4. For confidence assessment of an alternative argument, probability of sufficiency, insufficiency and uncertainty can be measured in the following ways: if claims $n2$ and $n3$ support claim $n1$ then probability of sufficiency, insufficiency and uncertainty can be measured by following [77]:

$$m_{n1}(S) = \frac{m_{n2}(S) * m_{n3}(S) + m_{n2}(S) * m_{n3}(U) + m_{n2}(U) * m_{n3}(S)}{1 - (m_{n2}(S) * m_{n3}(I) + m_{n2}(I) * m_{n3}(S))}$$
$$m_{n1}(I) = \frac{m_{n2}(I) * m_{n3}(I) + m_{n2}(I) * m_{n3}(U) + m_{n2}(U) * m_{n3}(I)}{1 - (m_{n2}(S) * m_{n3}(I) + m_{n2}(I) * m_{n3}(S))}$$
$$m_{n1}(U) = 1 - (m_{n1}(S) + m_{n1}(I)) \tag{8.1}$$

  5. For confidence assessment of a disjoint argument, probability of sufficiency, insufficiency and uncertainty can be measured in the following ways: if claims n_i support claim n then probability of sufficiency, insufficiency and uncertainty can be measured by following

[77]:

$$
\begin{aligned}
m_n(S) &= \frac{\sum_{i=1}^{n} w_i * m_i(S)}{\sum_{i=1}^{n} w_i} \\
m_n(I) &= \frac{\sum_{i=1}^{n} w_i * m_i(I)}{\sum_{i=1}^{n} w_i} \\
m_n(U) &= 1 - (m_n(S) + m_n(I))
\end{aligned}
\tag{8.2}
$$

6. All weights to measure probabilities of sufficiency, insufficiency and uncertainty are equally distributed for disjoint argument.

8.2 Evaluation performed by an external reviewer

This section illustrates the validation of our evaluation process of an AC for a coffee cup submitted by developers from the external reviewer's perspective. Developers may provide optional notes that facilitate the evaluation process by external reviewers. External reviewers offer recommendations based on knowledge and experience, along with notes voluntarily supplied by developers. Thus, it increases belief in stakeholders.

8.2.1 Structure evaluation

This section illustrates the validation of the proposed structure evaluation process from an external reviewer's perspective.

8.2.1.1 Syntax check

Notation attribute in a class "AssuranceCase" denotes 'GSN'; thus rules for 'CheckGraphSyntax' apply.

Checking rules for *CheckGraphSyntax*
Concerning Rule (1): We consider "GSN community Standard 2.0" [12] for checking syntax.
Concerning Rule (2): We note that shapes of goal, strategy, evidence,

context, assumption, and justification comply with the standard by review.
Concerning Rules (3) and (4): There is one and only one valid association ('SupportedBy' and 'InContextOf') that exists between any two nodes.
Concerning 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.
Concerning Rule (6): The label of goals, strategies and evidence follows a hierarchy.

***GenerateRecommend* produces the following recommendations:**

- Based on the standard and a developer's provided note, we find no syntax error; thus, no further action is recommended.

8.2.1.2 Traceability

We use three checks to perform traceability evaluation.

We use '(Review) EvidenceToSystemTrace' to evaluate evidence to system artefacts traceability.

Checking rules for *(Review) EvidenceToSystemTrace*
Concerning Rule (1): We identify that an explicit link or reference exists between evidence supporting a process related claim and a specific section of a document related to that process. Evidence 'E-CR2.2.1.1.1.2.1' (FTA execution) refers to 'x.y.15', 'E-CR2.1.2'(prototype generation) refers to 'x.y.4', 'E-CR2.2.1.1.3.2.1'(STPA execution) refers 'x.y.q.18', 'E-CR3.1'(development from requirements with current technology) refers to 'x.y.19', 'E-CR3.2'(built feasibly) refers to 'x.y.18', 'E-CR6.2'(interaction) refers to 'x.y.20', 'E-CI1.1.1.1' (Design process of container) refers to 'x.y.25', 'E-CI1.2.1.1' (design process of lid) refers to 'x.y.28', 'E-CI4.1'(safety assessment process during implementation) refers to 'x.y.51', 'E-CPM1.2' (production process) refers to 'x.y.z.51', 'E-CPM1.3'(safety assessment process) refers to 'x.y.z.52', 'E-CPM2.1' (service process) refers to 'x.y.z.53', 'E-CA2.1.2' (survey analysis) refers to 'x.y.62'.
Concerning Rule (2): We find that explicit links or references exist between evidence representing credentials of people for the process and a spe-

cific section of a document related to credentials of people involved in a process. Evidence 'E-CR2.1.3.3' (competency of people involve in review process) refers to 'x.y.10', 'E-CR2.2.1.1.1.2.3' (people execute FTA) refers to 'x.y.17', 'E-CR2.2.1.1.3.2.3' (people execute STPA) refers to 'x.y.q.20', 'E-CR1.1.2.2' (competent people) refers to 'x.y.c.2', 'E-CI1.1.1.2' (people perform design of container) refers to 'x.y.26', 'E-CI1.2.1.2'(people perform design of lid) refers to 'x.y.29', 'E-CI2.3' (people perform mathematical validation of 'Z') refers to 'x.y.33', 'E-CI3.2' (people perform testing 'X') refers to 'x.y.43'. 'E-CI4.2' (people perform safety assessment) refers to 'x.y.52', 'E-CA2.1.3'(competency of people to perform survey) refers to 'x.y.64'.

Concerning Rule (3): We find that explicit links or references exist between evidence supporting a product-related claim and a specific section of a document related to that product. Evidence 'E-CR1.1.2.1' (signature proof) refers to 'x.y.c.1', 'E-CR2.1.1' (test case definition) refers to 'x.y.3', 'E-CR2.1.3.1.1' (requirements of container) refers to 'x.y.5', 'E-CR2.1.3.1.2' (requirements of sleeve) refers to 'x.y.6', 'E-CR2.1.3.1.3' (requirement of lid) refers to 'x.y.7', 'E-CR2.1.3.1.4' (requirement of interaction between all components) refers to 'x.y.8', 'E-CR2.2.1.1.1.1' (all hazards description by FTA) refers to 'x.y.z.1', 'E-CR2.2.1.1.3.1' (all hazards description by STPA) refers to 'x.y.p.1', 'E-CR2.2.1.1.3.3' (comparison list of hazards) refers to 'x.y.21', 'E-CR2.2.1.1.1.2.2' (review report for FTA) refers to 'x.y.16', 'E-CR2.2.1.1.3.2.2' (review report for STPA) refers to 'x.y.q.19', 'E-CR4' (testable criteria document) refers to 'x.y.a.19', 'E-CR6.1' (consistent criteria proof for requirement) refers to 'x.y.b.19', 'E-CR6.2' (dependency graph) refers to 'x.y.b.20', 'E-CR6.3' (traceability matrix for stakeholders) refers to 'x.y.b.21', 'E-CI1.1.2.2.12' (qualification test result of biodegradable film use) refers to 'x.y.b.27', 'E-CI1.1.2.2.3' (bottom part satisfy safety requirement) refers to 'x.y.b.28', 'E-CI1.2.2' (traceability matrix for lid) refers to 'x.y.b.30', 'E-CI1.2.3' (specification of lid) refers to 'x.y.b.31', 'E-CPM1.1' (production plan) refers to 'x.y.z.50', 'E-CA1' (historical data) refers to 'x.y.60', 'E-CA2.1.1'(survey questionnair) refers to 'x.y.61', 'E-CA3' (assumption descriptions) refers to 'x.y.62', 'E-CR1.1.1' (use case) refers to 'x.y.1'.

Concerning Rule (4): We find that explicit links or references exist between evidence supporting a claim related to the validation of a product/process and

the description of validation of that product/process in a document. Evidence 'E-CI2.2' supports a terminal claim 'CI2.2' related to verification of behaviour. It refers to a specific section 'x.y.32' of the implementation chapter in a document. Similarly, evidence 'E-CI2.1' (validation of mathematical method 'Z') refers to x.y.31, E-CI3.1 (validation of test method 'X') refers to 'x.y.40', 'E-CPM2.2' (verification steps) refers to 'x.y.z.54', 'E-CR1.2.1' refers to 'x.y.p.1', 'E-CR1.2.2' refers to 'x.y.p.2', 'E-CR1.2.3' refers to 'x.y.p.3', 'E-CR1.2.4' refers to 'x.y.p.4'.

Concerning Rule (5): We find that evidence description comply with acceptance criteria. Evidence 'E-CR2.2.1.1.1.2.2' complies with acceptance criteria 'E-CR2.2.1.1.1.2.2_ACT'.

Concerning Rule (6): We do not find any explicit link or reference exists between counter-evidence and valid proof (deductive or inductive) defined in a document.

Concerning Rule (7): We find that explicit links or references exist between evidence supporting claims related to change management and change management documentation of that system. Evidence 'E-CPM4.1.1' supports a terminal claim 'CPM4.1.1' related to a plan for change management. It refers to a specific section 'x.y.80' of the change management chapter in a document. Similarly, evidence 'E-CPM4.1.2' refers to 'x.y.81', 'E-CPM4.1.3' refers to 'x.y.82'.

We use '(Review) RecentToPastVerTrace' to review trace between previous version and recent version of an AC.

Checking rules for *(Review) RecentToPastVerTrace*

Concerning rules (1) and (2): We do not find any traceability between artefacts of previous version and recent version of an AC.

We use '(Review) ArgumentPatternTrace' to review compliance of instantiated argument with the pattern.

Checking rules for *(Review) ArgumentPatternTrace*

Concerning Rule (1): We find that an instantiated claims/arguments/evidence shall comply with an argument pattern. Claim 'CR2.2.1.1.4' complies with claim 'G1' from 'ALARP' pattern [99]. Other claims comply with claims from

that pattern.

***GenerateRecommend* produces the following recommendations:**

- It is recommended to show explicit links or references between counter-evidence and a specific section of a document.
- It is recommended to develop supporting argument branches for claims, 'CPM3', 'CPM2.3' and 'CPM2.4', 'CR5', 'CI1.3', 'CL1.1.4', 'CI1.1.2.1', and 'CI1.1.2.3'

8.2.1.3 Robustness

We use three checks to perform the robustness evaluation. For this evaluation, we use pairwise comparisons relying on "mental models" and/or experience. We take into account the following changes provided by developers for robustness evaluation:

1. We consider the following anticipated change: the dimension of a container of coffee cup changes. This change may be height or diameter, or both of a container of a coffee cup.
2. We consider the following anticipated change: a new operational assumption may violate the previous assumption and initiate changes in an AC.
3. We consider the following anticipated change: new testing may affect terminal claims and/or upper claims and/or arguments.

We review the affected branches due to changes by using pairwise comparisons. Pairwise comparisons are guided by "mental models" and/or experience from previous examples. Thus, we consider figures. 8.5, 8.6, 8.7, 8.8, 8.9, 8.10, 8.11, 8.12, 8.13, 8.14, and 8.15.

We use '(Review) SystemVariabilityInAC' to review how an anticipated change/variation in a system affects the AC.

Checking rules for *(Review) SystemVariabilityInAC*

Concerning Rule (1): We find that an anticipated change affects fewer

artifacts. Besides, we find that an anticipated change affects lower-level artifacts. Moreover, we find that change management for an anticipated change can be performed in an independent branch.

We use '(Review) RebuttalsInAC' to check whether an anticipated rebuttal affects claims, arguments or evidence.

Checking rules for *(Review) RebuttalsInAC*

Concerning Rule (1): We find that three evidence ('E-CA1', 'E-CA3' and 'E-CA2.1.1' in figure 8.14) of AC need to be changed. Furthermore, we find that an anticipated rebuttal requires three evidence to be changed in an independent argument branch.

We use '(Review) AlternativeEvidence' to review change due to alternative evidence.

Checking rules for *(Review) AlternativeEvidence*

Concerning Rule (1): We find that alternative evidence comply with acceptance criteria.

Concerning Rule (2): We find that evidence 'E-CI3.1' and 'E-CI3.2' affect the terminal claims 'CI3.1' and 'CI3.2' but 'E-CI3.3' does not affect terminal claim 'CI3.3' in figure 8.15 for AC. Besides, we find that terminal claims 'CI3.1' and 'CI3.2' may affect the argument 'R-CI3' in AC.

***GenerateRecommend* produces the following recommendations:**

- It is recommended to add justification, context to support arguments. No argument is supported by context, justification.
- Undeveloped claims, 'CR5', 'CI1.3', 'CL1.1.4', 'CI1.1.2.1', and 'CI1.1.2.3' are not affected due to changes.
- Estimating the affected nodes and their position reveals that AC is robust.

8.2.1.4 Understandability

We use four checks consisting of rules to perform an *Understandability* evaluation.

We use '(Review) FontAttributes' for reviewing fonts.

Checking rules for *(Review) FontAttributes*

Concerning Rule (1): We find that an acceptable font ('Arial') is used in documenting the AC.

Concerning Rule (2): We find that 'font-size' is'18' with 'Plain' style and 'font-colour' is'Black' and they are appropriate for viewing both on screen and in print.

Concerning Rule (3): It does not apply to the AC because it is a graphical notation.

We use '(Review) ModuleArgument' for module arguments.

Checking rules for *(Review) ModuleArgument*

Concerning Rule (1): We find that modules in the AC make it easier to comprehend. Following claims are formed into modules: 'CR', 'CI', 'CPM', 'CA', 'CR1', 'CR2', 'CR3', 'CR4', 'CR6','CR2.1', 'CR2.2', 'CR2.2.1.1', 'CI1', 'CI2', 'CI3', 'CI4', 'CI1.1', 'CI1.2', 'CI1.1.1', 'CI1.1.2', 'CI1.1.3', 'CI1.1.2.2'.

We use '(Review) IntersectAssociation' because AC in consideration is documented using GSN-like notation.

Checking rules for *(Review) IntersectAssociation*

Concerning Rule (1): There is no intersection of association arcs in the AC, and it is comprehensible.

We use '(Review) SameLevelClaim' and apply rule (1) only because of GSN-like notation.

Checking rules for *(Review) SameLevelClaim*

Concerning Rule (1): Claims, argument, evidence are in the same horizontal position based on decomposition level. Furthermore, undeveloped claims, 'CPM3', 'CPM2.3', 'CPM2.4', 'CR5', 'CI1.3', 'CL1.1.4', 'CI1.1.2.1', and 'CI1.1.2.3' are also in the same horizontal position based on decomposition level.

***GenerateRecommend* produces the following recommendations:**

- We find that the AC is understandable and does not create any ambiguity. Thus, no further action is required.

8.2.1.5 Efficiency

We use three checks to perform an *Efficiency* evaluation.

We use '(Review) FacilitateACDevelopment' to evaluate how it facilitates an AC development.

Checking rules for *(Review) FacilitateACDevelopment*
Concerning Rule (1): We find that an argument pattern ALARP is instantiated to an argument branch starting with a claim 'CR2.2.1.1.4'.
Concerning Rule (2): We find a single claim representing the claims that assure the same artifact. Claims 'CR2.2.1.1.1.1', and 'CR2.2.1.1.3.1' show all 50 hazards are identified by two different methods, FTA and STPA respectively. Evidence 'E-CR2.2.1.1.1.1' and 'E-CR2.2.1.1.3.1' show description of all hazards by FTA and STPA respectively.

We use '(Review) FacilitateSystemDevelopment' to review how an AC facilitate system development.

Checking rules for *(Review) FacilitateSystemDevelopment*
Concerning Rule (1): We find that evidence guide the system development. Evidence 'E-CR2.2.1.1.1.2.2' shows the competency of people to perform FTA analysis. Similarly, evidence related to processes: E-CR2.1.2(prototype generation), E-CR2.2.1.1.1.2.1(FTA execution steps), E-CR2.2.1.1.3.2.1 (STPA execution steps), E-CR3.1 (development from requirements with current technology), E-CR3.2 (feasible development), E-CR6.2 (interaction), E-CI1.1.1.1(design process of container), E-CI1.2.1.1 (design process of lid), E-CI4.1(safety assessment process during implementation), E-CPM1.2 (production process), E-CPM1.3(safety assessment process), E-CPM2.1 (service process); evidence related to product: E-CR2.1.1 (test case definition), E-CR2.1.3.1.1 (requirements of container), E-CR2.1.3.1.2 (requirements of sleeve), E-CR2.1.3.1.3 (requirement of lid), E-CR2.1.3.1.4 (requirement of interaction between all components), E-CR2.2.1.2(description of mitigation), E-CR2.2.1.1.1.1 (all hazards description by FTA), E-CR2.2.1.1.3.1 (all hazards description by STPA),

E-CR2.2.1.1.3.3 (comparison list of hazards), E-CR2.2.1.1.1.2.2 (review report for FTA), E-CR2.2.1.1.3.2.2 (review report for STPA), E-CR4 (testable criteria document), E-CR6.1 (consistent criteria proof for requirement), E-CR6.3 (traceability matrix for stakeholders), E-CI1.1.2.2.12(qualification test result of biodegradable film use), E-CI1.1.2.2.3 (bottom part satisfy safety requirement), E-CI1.2.2 (traceability matrix for lid), E-CI1.2.3 (specification of lid), E-CPM1.1 (production plan), E-CA2 (assumption descriptions); evidence related to credentials of people: E-CR2.1.3.3(people to review process), E-CR2.2.1.1.1.2.3 (people execute FTA), E-CR2.2.1.1.3.2.3 (people execute STPA), E-CI1.1.1.2 (people perform design of container), E-CI1.2.1.2(people perform design of lid), E-CI2.3 (people perform mathematical validation of 'Z'), E-CI3.2 (people perform testing 'X'). E-CI4.2 (people perform safety assessment), E-CA1(the user aware documented assumption); evidence related to user acceptance test: E-CR1.2, E-1.2.2 and evidence related to validation/verification: 12. E-CI2.1 (validation of mathematical method 'Z'), E-CI3.1 (validation of test method 'X'), E-CI2.2 (verification of behaviour), E-CPM2.2 (verification steps). We do not find any evidence for following claims, 'CPM3', 'CPM2.3', 'CPM2.4', 'CR5', 'CI1.3', 'CL1.1.4', 'CI1.1.2.1', 'CI1.1.2.3' as they are undeveloped (i.e. they do not have any supporting argument branches).

Concerning Rule (2): We find that the acceptance criteria that guide system development— acceptance criteria in figure 8.2 guide in choosing the right team by prescribing the required credentials. Similarly, acceptance criteria for all other evidence guide system development.

We use '(Check) FacilitateReview' to facilitate the review process.

Checking rules for *(Check) FacilitateReview*

Concerning Rule (1): We find that the argument with starting claim 'CR2.2.1.1.4' complies with the 'ALARP' pattern. The AC does not mention all attributes of the 'ALARP' pattern explicitly.

Concerning Rule (2): We do not find any necessary rationale/justification and context of using the GSN-like notation in documenting an AC.

***GenerateRecommend* produces the following recommendations:**

- It is recommended to provide the rationale and context of using GSN adequately.
- It is recommended to justify arguments, add contexts for clarification.

8.2.2 Content evaluation

This section illustrates the validation of the proposed content evaluation process from an external reviewer's perspective.

8.2.2.1 Convincing basis

We follow the rules for four checks to evaluate a convincing basis.

We use 'TopLevelClaimCheck' for reviewing a top claim.

Checking rules for *TopLevelClaimCheck*

Concerning Rule (1): We find that the top-level claim, "TopClaim, C" consists of two parts: the subject "The coffee cup <KCoffeeCup>" specifies the system and the predicate "is safe in its intended environment, and its intended uses" specifies a critical property 'safe.'

Concerning Rule (2): We find that the meaning of the top-level claim is unambiguous and easy to understand.

Concerning Rule (3): All necessary terms (e.g. 'safe,' "intended environment," "intended uses," "coffee cup specification") are clarified with external references.

Concerning Rule (4): We find that necessary assumptions (e.g. non-toxic material for a coffee cup and tolerable temperature) are clarified, including temperature range.

We use 'SubClaimCheck' to check all sub-claims.

Checking rules for *SubClaimCheck*

Concerning Rule (1): The meaning of all sub-claims is unambiguous.

Concerning Rule (2): We do not find any clarification for any necessary term mentioned in the sub-claims.

Concerning Rule (3): We find that all claims related to process, product

and people are clarified to support upper-level claims; claim '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, decommissioning, and operational assumptions.

Concerning Rule (4): We find assumptions.

Concerning Rule (5): We find that evidence support terminal claims.

We use 'ExplicitArgument' to evaluate the explicitness of an argument.

Checking rules for *ExplicitArgument*

Concerning Rule (1): We find that arguments describe explicit reasoning of how sub-claims support upper-level claims.

Concerning Rules (2), (3) and (4): We do not find any necessary justification, context or assumption for arguments.

We use 'ConfirmationBias' to review confirmation bias.

Checking rules for *ConfirmationBias*

Concerning Rule (1): We find that arguments demonstrate rebuttals wherever they are identified with possible counters explicitly.

Concerning Rule (2): We do not find any evidence to support those rebuttals.

Concerning Rule (3): We find that all evidence comply with acceptance criteria provided by developers.

***GenerateRecommend* produces the following recommendations:**

- It is recommended to provide clarification of necessary terms.
- It is recommended to provide necessary justifications for arguments. For instance, the argument 'R' requires justification of why they follow this decomposition structure.
- The arguments have known rebuttals and evidence comply with acceptance criteria (provided by developers), so no further action is recommended for those.

8.2.2.2 Rigour of the argument

The AC has informal arguments. Thus, we use 'CheckInformalArgument.' We also perform 'CheckClaimForValidity' and 'CheckRationaleForValidity.

We use 'CheckInformalArgument' to review informal arguments.

Checking rules for *CheckInformalArgument*

Concerning Rule (1): We find that the arguments are defined inductively with adequate steps, including rebuttals (when they are identified) to support the upper-level claims, and they are adequate and consistent. However, claims 'CPM3', 'CPM2.3', 'CPM2.4', 'CR5', 'CI1.3', 'CL1.1.4', 'CI1.1.2.1', and 'CI1.1.2.3' do not have supporting argument branches.

Concerning Rule (2): We do not find any justification for any argument.

Concerning Rule (3): We do not find any necessary term, e.g. context or assumption, to support arguments.

Concerning Rule (4): We find that some arguments mention rebuttals. Arguments 'R' mentions two rebuttals, 'R1' and 'R2'. Similarly, others are 'R-CR', 'R-CR2.1.3.1', 'R-CR2.2.1.1.1.2', 'R-CI', 'R-CI1.1.1', 'R-CI1.1.2', 'R-CI1.2', 'R-CI1.2.1', 'R-CI2', 'R-CI3', 'R-CI4', 'R-CPM'. Rebuttals mentioned in those arguments are consistent.

Concerning Rule (5): We find that the mitigation of each rebuttal is provided. Claims relating to mitigations 'CPM' and 'CA' resolve two rebuttals, 'R1' and 'R2' mentioned in an argument 'R.'

Concerning Rule (6): We find that an argument branch considers different phases of the development process.

We use "CheckClaimForValidity" for reviewng validity.

Checking rules for *CheckClaimForValidity*

Concerning Rule (1): We find that the sub-claims are valid and consistent because arguments are valid and evidence complying with acceptance criteria support terminal claims.

Concerning Rule (2): We find that arguments, as mentioned earlier, have identified rebuttals and corresponding mitigations. Rebuttals are consistent, but there is no proof to check the validity of those rebuttals.

We use"CheckRationaleForValidity" for reviewing rationale.

Checking rules for *CheckRationaleForValidity*
Concerning Rule (1): We find that no justification exists to support the argument.

***GenerateRecommend* produces the following recommendations:**

- It is recommended to include justification for each argument to comprehend the reasoning. For instance, the system development life-cycle guides the argument formation.
- It is recommended to provide a rationale that no known rebuttal is identified.
- It is recommended to provide context, assumptions or justifications for claims /arguments; for instance, claim 'CR1.2' deals with user acceptance testing. The claim should have the context about user acceptance testing with assumptions along with preconditions or postconditions. This recommendation applies to all tests mentioned in the AC.
- It is recommended to develop supporting argument branches for claims: 'CPM3', 'CPM2.3', 'CPM2.4', 'CR5', 'CI1.3', 'CL1.1.4', 'CI1.1.2.1', 'CI1.1.2.3'.

8.2.2.3 Quality of the hazard analysis

We use four checks having rules to perform an evaluation.

We use 'ClaimEvidenceMitigation' to evaluate mitigation strategies.

Checking rules for *ClaimEvidenceMitigation*
Concerning Rule (1): We find that claim mentions mitigation of all identified hazards.
Concerning Rule (2): We find that evidence support mitigation claims.
Concerning Rule (3): We find that a claim of safety requirements relates to mitigation steps. Claim 'CR2.2.1.2' mentions safety requirements for mitigation.
Concerning Rule (4): We find that evidence document safety requirements for each mitigation of hazards, and it complies with the acceptance criteria.

Process 'ClaimEvidenceComparison' compares identified hazards with a known hazard list adopted by regulatory organizations or standards. There is no regulator's recommended hazard list for a coffee cup that can verify whether the hazard analysis performed by developers identifies all known hazards or not. Instead, we compare two hazard lists identified by FTA and STPA respectively.

We use 'ClaimEvidenceHAMethod' to check the hazard analysis method.

Checking rules for *ClaimEvidenceHAMethod*

Concerning Rule (1): We find that two claims mention the effectiveness of hazard analysis methods. Claims 'CR2.2.1.1.1.2.2' and 'CR2.2.1.1.3.2.2' mention that FTA and STPA are industry best practice hazard analyses by expert appraisal.

Concerning Rule (2): We find that evidence supports the claim along with an expert appraisal to prove soundness. Evidence 'E-CR2.2.1.1.1.2.2' shows proof to support claim 'CR2.2.1.1.1.2.2'. Similarly, evidence 'E-CR2.2.1.1.3.2.2' supports claim 'CR2.2.1.1.3.2.2'. Both evidence comply with acceptance criteria.

Concerning Rule (3): We find that evidence demonstrates the execution steps of a hazard analysis. Evidence 'E-CR2.2.1.1.1.2.1' documents implemented procedure steps of a hazard analysis FTA. Similarly, evidence 'E-CR2.2.1.1.3.2.1' documents an implemented hazard analysis procedure, STPA. Both evidence comply with acceptance criteria.

We use 'ClaimEvidencePeopleCredentials' to check the qualification of people performing hazard analysis.

Checking rules for *ClaimEvidencePeopleCredentials*

Concerning Rule (1): We find that claim mentions competent people execute the hazard analysis. Claim 'CR2.2.1.1.1.2.3' mentions a competent team involve in FTA. Similarly, claim 'CR2.2.1.1.3.2.3' mentions a competent team involve in STPA analysis.

Concerning Rule (2): We find that evidence document credentials of people to support claims of competent teams. Evidence 'E-CR2.2.1.1.1.2.3' documents the credentials of people involve in FTA analysis. Similarly, evi-

dence 'E-CR2.2.1.1.3.2.3' documents the credentials of people involve in STPA analysis. Both evidence comply with the acceptance criteria.

We use 'Claim Evidence Comparison' to compare identified hazards by FTA with regulatory organizations or standards recommended hazard list. We already mentioned that we do not find any regulator's approved known hazard list. Thus, Rule (2) does not apply to the AC.

Checking rules for *Claim Evidence Comparison*
Concerning Rule (1): Instead of known hazards, we find the comparison between identified hazards identified by two different hazard analysis methods, FTA and STPA. Claim 'CR2.2.1.1.3.3' shows that two identified hazard list is the same by comparison.

***GenerateRecommend* produces the following recommendations:**

- It is recommended to describe justifications to support arguments.
- It is recommended to clarify the selection of hazard analysis based on system description.
- It is recommended to clarify necessary terms to support claims, arguments.
- It is recommended to clarify why FTA is better than STPA in this scenario.

8.2.2.4 Arguing Completeness

We use three checks to evaluate *arguing completeness*. *Coffee Cup* is considered a feasible system to analyze, thus 'LargeSystemCompleteness' does not apply to the AC for a coffee cup.

We use 'HazardIdentificationComplete' to review the completeness of hazards.

Checking rules for *HazardIdentificationComplete*
Concerning Rule (1): We find arguments. Argument 'R-CR2.2.1.1.1' mentions that all known hazard have been identified.

Concerning Rule (2): We do not find any clarification or proof for the arguments.
Concerning Rule (3): We identify claims related to the mitigation. Claim 'CR2.2.1.1.1.2' mentions the mitigation. Similarly, claims 'CR2.2.1.1.1.2.3' and 'CR2.2.1.1.3.2.3' mention mitigation.
Concerning Rule (4): We find that evidence show mitigation to support claims related to mitigation. Evidence 'E-CR2.2.1.1.1.2.3' and 'E-CR2.2.1.1.3.2.3' support claims 'CR2.2.1.1.1.2.3' and 'CR2.2.1.1.3.2.3' respectively.

We use 'FeasibleSystemCompleteness' to review the completeness of a feasible system.

Checking rules for *FeasibleSystemCompleteness*
Concerning Rule (1): We find that arguments provide reasoning to support upper-level claims. Arguments 'R-CR2', 'R-CR2.1' and 'R-CR2.2.1' show the reasoning of complete requirements.
Concerning Rule (2): We do not find any necessary terms clarified to support arguments.
Concerning Rule (3): We find that claims mention proof or test to support completeness. Claims 'CR2', 'CR2.1', 'CR2.2', 'CR2.1.1', 'CR2.1.2', 'CR2.1.3', 'CR2.1.3.1', 'CR2.1.3.1.1', 'CR2.1.3.1.2', 'CR2.1.3.1.3', 'CR2.1.3.1.4' discuss test, prototype and criteria for completeness of arguments.
Concerning Rule (4): We do not find any terms clarified for claims.
Concerning Rule (5): We find that evidence support claims of tests, prototypes. Evidence 'E-CR2.1.1', 'E-CR2.1.2', 'E-CR2.1.3.1.1','E-CR2.1.3.1.2', 'E-CR2.1.3.1.3' and 'E-CR2.1.3.1.4' support claims 'CR2.1.1', 'CR2.1.2', 'CR2.1.3.1.1', 'CR2.1.3.1.2', 'CR2.1.3.1.3' and 'CR2.1.3.1.4'.

***GenerateRecommend* produces the following recommendations:**

- It is recommended to provide justifications for arguments of completeness.
- It is recommended to provide an explicit rationale of testing alone in the requirement completeness check.
- It is recommended to develop supporting argument branches for claims:

'CPM3', 'CPM2.3', 'CPM2.4', 'CR5', 'CI1.3', 'CL1.1.4', 'CI1.1.2.1', 'CI1.1.2.3' for completeness evaluation.

8.2.2.5 Repeated arguments

We use 'SimilarArgument' to check *repeated arguments* in an AC.

Checking rules for *SimilarArgument*

Concerning Rule (1): We find that an argument pattern is instantiated to a specific argument branch. An argument pattern *ALARP* is instantiated to an argument branch staring with claim 'CR2.2.1.1.4'. and it is supported by three subclaims 'CR2.2.1.1.4.1', 'CR2.2.1.1.4.2' and 'CR2.2.1.1.4.3' through argument 'R-CR2.2.1.1.4'.

Concerning Rule (2): We find that an external note is mentioned to show target argument branch.

Concerning Rule (3): We find that a target argument branch complies with a source branch. It is demonstrated that argument branch starting with claims 'CPM4.2.1' and 'CPM4.2.2' are target branches those are copied from argument branch with similar claims 'CR2.2.1.1.1' and 'CR2.2.1.1.2'.

We use 'ClaimArgumentEvidencePattern' to check claim, argument, evidence in compliance with source argument with consistency.

Checking rules for *ClaimArgumentEvidencePattern*

Concerning Rule (1): We find that claims and evidence of an instantiated argument branch comply with claims and evidence of a pattern. The pattern does not provide any explicit argument. But instantiated one has arguments to support upper claims. An argument branch for an ALARP principle complies with ALARP pattern [99]. All claims 'CR2.2.1.1.4','CR2.2.1.1.4.1', 'CR2.2.1.1.4.2', 'CR2.2.1.1.4.3', 'CR2.2.1.1.4.2.1', 'CR2.2.1.1.4.3.1', 'CR2.2.1.1.4.2.1.1', 'CR2.2.1.1.4.2.1.2' and 'CR2.2.1.1.4.2.1.3' comply with claims 'G1', 'G2', 'G3', 'G4', 'G8', 'G9', 'G10', 'G11' and 'G12' respectively of that pattern. Evidence 'E-CR2.2.1.1.4.1' complies with 'Sn1'. Another evidence 'E-CR2.2.1.1.4.3.1' complies with guidance provided by Kelly [99].

Concerning Rule (2): We find that all terms in an instantiated branch comply with those of the pattern. Contexts 'K-CR2.2.1.1.4', 'K-CR2.2.1.1.4.1',

'K-CR2.2.1.1.4.2', 'K-CR2.2.1.1.4.3', 'K-CR2.2.1.1.4.2.1.3' comply with 'C1', 'C2', 'C3', 'C4' and 'C5' respectively.

Concerning Rule (3): We find that argument branch copied or instantiated from a source argument branch is consistent with a neighbouring argument branch. Claims 'CR2.2.1.1.4', 'CR2.2.1.1.4.1', 'CR2.2.1.1.4.2', 'CR2.2.1.1.4.3', 'CR2.2.1.1.4.2.1', 'CR2.2.1.1.4.3.1', 'CR2.2.1.1. 4.2.1.1', 'CR2.2.1.1.4.2.1.2' and 'CR2.2.1.1.4.2.1.3', evidence 'E-CR2.2.1.1.4.1', and 'E-CR2.2.1.1.4.3.1' are consistent with claims 'CR2.2.1.1.1', 'CR2.2.1.1.2', and 'CR2.2.1.1.3' and their corresponding argument branches.

***GenerateRecommend* produces the following recommendations:**

- It is recommended to provide necessary contexts, assumptions and justifications to support target arguments.
- It is recommended to provide a rationale of using an argument pattern in the AC.

8.2.2.6 ALARP

We have one check consisting of rules to review *ALARP*.

We use 'TolerableRiskArgument' to review tolerable risk arguments.

Checking rules for *TolerableRiskArgument*

Concerning Rule (1): We find that a claim mentions the benefit of having a hazard. Claim CR2.2.1.1.4.2.1.1 mentions the benefit of having a hazard.

Concerning Rule (2): We find that evidence support claim of benefits. Evidence 'E-CR2.2.1.1.4.2.1.1' documents the benefit to support the claim, 'CR2.2.1.1.4.2.1.1'.

Concerning Rule (3): We find that claim mentions measures to reduce risk to tolerable. Claim 'CR2.2.1.1.4.2.1.2' mentions measures to mitigate a hazard 'x.'

Concerning Rule (4): We find that evidence document measures to reduce risk. Evidence 'E-CR2.2.1.1.4.2.1.2' documents measures to reduce risk.

Concerning Rule (5): We find a claim related to the introduction of a disproportionate expense. Claim 'CR2.2.1.1.4.2.1.3' mentions the introduction of

a disproportionate expense.

Concerning Rule (6): We find that evidence shows expense report to support the claim. Evidence 'E-CR2.2.1.1.4.2.1.3' shows a cost analysis to support the claim.

Concerning Rule (7): We find that key terms are clarified. Context 'K-CR2.2.1.1.4.2.1.3' shows clarification of 'disproportionate'. There is no other context or justification to support an argument.

***GenerateRecommend* produces the following recommendations:**

- It is recommended that contexts, assumptions and justifications shall exist to support arguments for tolerable risk using the 'ALARP' principle.
- It is recommended to provide a rationale for using the pattern in a document with the necessary attributes of the pattern.

8.2.2.7 Confidence

Developers used a quantitative confidence assessment developed by [77]—this approach measures probabilities of sufficiency, insufficiency and uncertainty using DST. We have four processes to review *confidence* assessment.

We use 'EvidenceConfidence' to review the probability of evidence.

Checking rules for *EvidenceConfidence*

Concerning Rule (1): We find that experts estimate the probability of evidence. Evidence 'E-CR1.1', 'E-CR1.2.1', 'E-CR1.2.2', 'E-CR1.2.3', 'E-CR1.2.4' have probability of sufficiency, (m(S)), insufficiency (m(I)) and uncertainty (m(U)). This degree of belief is derived based on domain expert appraisal.

We use 'ClaimConfidence' to review the quantitative confidence of claims.

Checking rules for *ClaimConfidence*

Concerning Rule (1): We find that a recommended approach estimates probabilities of claims. A proposed approach [77] estimates the probabilities of claims 'CR1' and 'CR1.2'.

Concerning Rule (2): We do not find any term for clarification.

Concerning Rule (3): We find the probability of terminal claims is the same as evidence as there is no explicit argument between them. Probabilities of sufficiency, insufficiency and uncertainty of terminal claims 'CR1.2.1', 'CR1.2.2', 'CR1.2.3', 'CR1.2.4', 'CR1.1' are same as probabilities of those of evidence 'E-CR1.2.1', 'E-CR1.2.2', 'E-CR1.2.3', 'E-CR1.2.4' and 'E-CR1.1'.

We use 'ArgumentConfidence' to review argument confidence assessment.

Checking rules for *ArgumentConfidence*

Concerning Rule (1): We find that probabilities of arguments are derived using a recommended approach. Probabilities sufficiency, insufficiency and uncertainty of arguments 'R-CR1' and 'R-CR1.2' are derived using a proposed approach by [77].

Concerning Rule (2): We do not find any term for clarification.

We use 'AssociationWeight' to review the weight of the association link.

Checking rules for *AssociationWeight*

Concerning Rule (1): We find that domain experts estimate the weights of associations. Weights of association among claims and arguments (R-CR1.2 and CR1.2.1), (R-CR1.2 and CR1.2.2), (R-CR1.2 and CR1.2.3), and (R-CR1.2 and CR1.2.4) are estimated by domain experts.

***GenerateRecommend* produces the following recommendations:**

- It is recommended to provide necessary contexts, assumptions and justifications to support arguments (R-CR1 and R-CR1.2)
- It is recommended to provide advantages of this quantitative approach in an external document.
- It is recommended to provide the rationale for using that confidence assessment method.

Chapter 9

Conclusion

This chapter summarizes the progress I have made with respect to the research outcomes mentioned in Chapter 1:1.2. It also presents a brief recap of the contributions I have made to the existing body of knowledge (Section 9.2). Each of the 4 major topics in the thesis has been published in high quality conferences, namely the *International Symposium on Software Reliability Engineering (ISSRE)* [84, 88], and the *International Conference on Computer Safety, Reliability, and Security (SafeComp)* [87, 97]. Finally, the section on Future Work (Section 9.3) presents a brief discussion on guidelines for extending the research work to resolve the issues that are out of scope in this thesis.

9.1 Research Objectives Revisited

Assurance cases are a generalization of safety cases to demonstrate assurance of critical properties (e.g. safety, security) in complex software-intensive systems. It is a widespread belief that a 'good' AC provides adequate confidence to stakeholders about the critical properties of a product. A lack of standardized patterns results in differently structured assurance cases for different products in a specific product family. This thesis focuses on developing effective ACs. Concerning this, we develop principles to document an ACT from existing safety and security standards/guidance (e.g. *ISO 26262*, *SAE J3061*). An ACT that complies with a relevant standard can be used to guide the development of systems that must comply with that standard. We have demonstrated

ten principles with examples to show the development of an ACT. We also developed an example feature diagram of an ACC product family.

We have demonstrated that an ACTs can be developed by integrating functional safety and security for automotive vehicles. We applied our principles to *ISO 26262* (functional safety for automotive vehicles) and *SAE J3061* (cybersecurity guidebook). We used the resulting ACT's specialization in a case study to guide us in mitigating potential vulnerabilities in OTA update implementations pre-emptively. In particular, we specialized the ACT to include specific arguments that apply when OTA Updates take place in maintenance. Concerning security, we have performed a threat analysis using the 'STRIDE' method.

Without a valid and convincing argument, it is evident that an assurance case cannot provide adequate confidence to stakeholders about their product. What is not apparent is that many other shortcomings may hinder our belief in an assurance case. To some extent, these problems may not indicate inherently unsafe systems. They may simply result in a system/product not being used when it really could have been deployed to the benefit of society. In the case of medical devices or treatments, for example, not deploying these adequately safe systems may cause actual harm to people, in addition to the financial loss inflicted on the manufacturer. On the other hand, what if the AC presents a compelling argument – but the argument is fatally flawed in some way? It is thus vital that we be able to evaluate ACs effectively. We have systematically presented a set of criteria for evaluating assurance cases, focusing on structural and content evaluation, and considering both developers and external reviewers' perspectives. The evaluation has to be repeatable and not overly sensitive to the reviewer. It also has to identify many different kinds of weaknesses in the AC – and suggest ways of rectifying those weaknesses. Our proposed approach incorporates rules to identify known weaknesses in an AC. We illustrated the application of these evaluation rules, via refinement and instantiation of a generic evaluation process, for all criteria related to the content of the AC, and all criteria related to the structure of the AC. These processes are then validated using one *GSN* example from two different perspectives: developers and external reviewers. We have thus shown that systematic and comprehensive evaluation of ACs is feasible.

9.2 Personal Contributions

This section briefly highlights the key contributions of this research:

R1. We have identified a lack of methods to develop an ACT complying with standards. Our way of dealing with this was to define principles to develop an ACT that complies with a standard/guidebook such as *ISO 26262*, *SAE J3061*. Thus, we proposed ten principles to develop an ACT.

R2. We applied our principles to *ISO 26262* (functional safety for automotive vehicles) and *SAE J3061* (cybersecurity), and used the resulting ACT's specialization in a case study to guide us to pre-emptively mitigate against potential vulnerabilities in automotive over-the-air update implementations.

R3. We defined effective criteria for both structure and content evaluation of an AC. We also defined evaluation criteria from two different perspectives: developers and external reviewers.

R4. To perform an effective evaluation using these criteria, we proposed a systematic evaluation process to make it less subjective. Later, these processes were then validated using one *GSN* example from two different perspectives: developers and external reviewers.

9.3 Future Work

We showed that semi-automation of the application of development principles should be possible. Tools to help achieve this would be extremely useful.

We did not develop a complete ACT complying with ISO 26262 and SAE J3061. If we can develop this it would be useful in its own right. Also, we could use our evaluation criteria on that ACT and thus learn more about the effectiveness of these principles. It may lead to the development of additional principles.

Furthermore, we are exploring how to integrate functional safety and security better. As mentioned earlier, one of the best ways is to integrate hazard

and threat analyses. If we achieve this, we will then re-evaluate the argument in the assurance case. This should eventually result in changes to *ISO 26262* and *SAE J3061* – or even better, the integration of cyber-security into *ISO 26262*. We intend to derive more examples of evidence, and especially acceptance criteria for that evidence. In this thesis, we only considered the integration of security of OTA and functional safety in a combined ACT.

ACs are developed by following different processes, often based on past practice, and influenced by relevant safety and security standards. We are currently considering integrating these criteria into representative processes (e.g. the V-model as influenced by DO-178C) to provide best practice guides and concrete checklists to better support engineers in their work.

The proposed evaluation process focuses on the evaluation of different aspects of an AC. We have used rebuttals to improve some of the evaluations methods. This can be made more systematic. We believe this is better than using defeaters as sometimes proposed, especially when the defeaters are developed by the developers of the AC.

Bibliography

[1] RE Bloomfield, PG Bishop, CCM Jones, and PKD Froome. *ASCAD: Adelard Safety Case Development Manual*. 1998 (cit. on pp. 1, 7).

[2] BBC News. *Faulty update breaks Lexus cars' maps and radio systems*. http://www.bbc.com/news/technology-36478641. Accessed: 2016-06-08 (cit. on pp. 3, 73).

[3] ISO. "26262: Road vehicles-Functional safety". In: *International Standard ISO* 26262 (2011) (cit. on pp. 4, 14, 15, 32, 43, 44, 48, 63, 64).

[4] Nancy Leveson. "Cost-effective safety certification of software-intensive systems". In: *Seventh Software Certification Consortium (SCC), Annapolis, May* (2011) (cit. on pp. 5, 124).

[5] US Food and Drug Administration et al. "Infusion pumps total product life cycle: guidance for industry and FDA staff". In: *Food and Drug Administration Standard* (2014), pp. 910–766 (cit. on pp. 7, 28, 98).

[6] Katrina Attwood, Paul Chinneck, Martyn Clarke, George Cleland, Mark Coates, Trevor Cockram, and P Williams. *GSN community standard version 1*. 2011 (cit. on p. 7).

[7] John Rushby, Xidong Xu, Murali Rangarajan, and Thomas L Weaver. *Understanding and evaluating assurance cases*. Tech. rep. SRI International, 2015 (cit. on pp. 8, 10, 28, 95).

[8] David J Reinhart, John C Knight, and Jonathan Rowanhill. *Current practices in constructing and evaluating assurance cases with applications to aviation*. Tech. rep. NASA, 2015 (cit. on pp. 8, 9).

[9] Stephen P Wilson, John A McDermid, Clive H Pygott, and David J Tombs. "Assessing complex computer based systems using the goal structuring notation". In: *Engineering of Complex Computer Systems, 1996. Proceedings., Second IEEE International Conference on* (1996), pp. 498–505 (cit. on p. 8).

[10] Robin Bloomfield and Peter Bishop. "Safety and assurance cases: Past, present and possible future–an Adelard perspective". In: *Making systems safer* (2010), pp. 51–67 (cit. on p. 8).

[11] Tim P Kelly. "Arguing Safety - A Systematic Approach to Managing Safety Cases". In: *Department of Computer Science, The University of York* (1998) (cit. on pp. 8, 10).

[12] Assurance Case Working Group et al. *Goal structuring notation community standard (version 2)*. 2018 (cit. on pp. 9, 147, 149, 198, 28).

[13] Stephen E Toulmin. *The uses of argument*. Cambridge University Press, 2003 (cit. on p. 10).

[14] Alan Wassyng, Neeraj Kumar Singh, Mischa Geven, Nicholas Proscia, Hao Wang, Mark Lawford, and Tom Maibaum. "Can Product-Specific Assurance Case Templates Be Used as Medical Device Standards?" In: *IEEE Design & Test* 32.5 (2015), pp. 45–55 (cit. on pp. 10, 11).

[15] Alan Wassyng, Paul Joannou, Mark Lawford, Thomas SE Maibaum, and Neeraj Kumar Singh. "Chapter 13 New Standards for Trustworthy Cyber-Physical Systems". In: *Trustworthy Cyber-Physical Systems Engineering* (2016), pp. 337–368 (cit. on pp. 11, 89, 95, 115).

[16] Don Batory. "Feature models, grammars, and propositional formulas". In: *International Conference on Software Product Lines* (2005), pp. 7–20 (cit. on pp. 12, 13).

[17] Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker. "Formalizing cardinality-based feature models and their specialization". In: *Software process: Improvement and practice* 10.1 (2005), pp. 7–29 (cit. on p. 12).

[18] Kyo C Kang, Sholom G Cohen, James A Hess, William E Novak, and A Spencer Peterson. *Feature-oriented domain analysis (FODA) feasibility study*. Tech. rep. DTIC Document, 1990 (cit. on p. 12).

[19] ISO/SAE AWI. "21434: Road vehicles-Cybersecurity Engineering". [Under development] (cit. on p. 14).

[20] Angela Barber. *Status of Work in Process on ISO/SAE 21434 Automotive Cybersecurity Standard*. https://www.sans.org/summit-archives/file/summit-archive-1525889601.pdf. Accessed: 2018-05-28 (cit. on p. 14).

[21] IEC International Electrotechnical Commission et al. "IEC 61508". In: (2010) (cit. on p. 14).

[22] SAE International. "SAE J3061-Cybersecurity Guidebook for Cyber-Physical Automotive Systems". In: *SAE-Society of Automotive Engineers* (2016) (cit. on pp. 16, 58, 70, 72).

[23] Trishank Karthik, Akan Brown, Sebastien Awwad, Damon McCoy, Russ Bielawski, Cameron Mott, Sam Lauzon, André Weimerskirch, and Justin Cappos. "Uptane: Securing Software Updates for Automobiles". In: *Int. Conf. Embeded Security in Car.* (2016), pp. 1–11 (cit. on pp. 17, 23, 74, 76, 81).

[24] Adam Shostack. *The Threats to Our Products*. Ed. by Microsoft SDL Blog. https://www.microsoft.com/security/blog/2009/08/27/the-threats-to-our-products/. Accessed: 2019-08-27. Microsoft (cit. on p. 17).

[25] Microsoft. *The STRIDE threat model*. https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20). Accessed: 2019-08-28 (cit. on p. 17).

[26] Remy Spaan, Lejla Batina, Peter Schwabe, and Sjoerd Verheijden. "Secure updates in automotive systems". In: *Nijmegen: Radboud University* (2016), pp. 1–71 (cit. on pp. 18, 19, 23, 25, 75).

[27] Alex Summers and Chris Tickner. *What is Security Analysis*. http://www.doc.ic.ac.uk/~ajs300/security/CIA.htm. Accessed: 2017-09-20 (cit. on p. 19).

[28] US Dept of Transportation. *Architecture Reference for Cooperative and Intelligent Transportation*. https://local.iteris.com/arc-it/. Accessed: 2018-02-24 (cit. on p. 19).

[29] Microsoft. *Microsoft Threat Modeling Tool*. https://www.microsoft.com/en-us/download/details.aspx?id=49168. Accessed: 2017-09-20 (cit. on pp. 19, 23, 75).

[30] NCC Group. *The Automotive Threat Modeling Template*. https://www.nccgroup.trust/us/about-us/newsroom-and-events/blog/2016/july/the-automotive-threat-modeling-template/. Accessed: 2017-09-20 (cit. on pp. 20, 76).

[31] T Scott Ankrum and Alfred H Kromholz. "Structured assurance cases: Three common standards". In: *Ninth IEEE International Symposium on High-Assurance Systems Engineering (HASE'05)* (2005), pp. 99–108 (cit. on p. 22).

[32] Stefan Wagner, Bernhard Schatz, Stefan Puchner, and Peter Kock. "A case study on safety cases in the automotive domain: Modules, patterns, and models". In: *2010 IEEE 21st International Symposium on Software Reliability Engineering* (2010), pp. 269–278 (cit. on p. 22).

[33] C Michael Holloway. "Making the implicit explicit: Towards an assurance case for do-178c". In: (2013) (cit. on p. 22).

[34] C Michael Holloway. "Explicate'78: Uncovering the Implicit Assurance Case in DO-178C". In: *Safety-Critical Systems Symposium* (2015) (cit. on p. 22).

[35] Ashlie B Hocking, John Knight, M Anthony Aiello, and Shinichi Shiraishi. "Arguing software compliance with ISO 26262". In: *2014 IEEE International Symposium on Software Reliability Engineering Workshops* (2014), pp. 226–231 (cit. on p. 22).

[36] Rajwinder Kaur Panesar-Walawege, Mehrdad Sabetzadeh, Lionel Briand, and Thierry Coq. "Characterizing the chain of evidence for software safety cases: A conceptual model based on the IEC 61508 standard". In: *2010 Third International Conference on Software Testing, Verification and Validation* (2010), pp. 335–344 (cit. on p. 22).

[37] Raghad Dardar, Barbara Gallina, Andreas Johnsen, Kristina Lundqvist, and Mattias Nyberg. "Industrial experiences of building a safety case in compliance with ISO 26262". In: *2012 IEEE 23rd International Symposium on Software Reliability Engineering Workshops* (2012), pp. 349–354 (cit. on p. 22).

[38] Rob Palin, David Ward, Ibrahim Habli, and Roger Rivett. "ISO 26262 safety cases: Compliance and assurance". In: (2011) (cit. on p. 23).

[39] John Birch, Roger Rivett, Ibrahim Habli, Ben Bradshaw, John Botham, Dave Higham, Peter Jesty, Helen Monkhouse, and Robert Palin. "Safety cases and their role in ISO 26262 functional safety assessment". In: *International Conference on Computer Safety, Reliability, and Security* (2013), pp. 154–165 (cit. on p. 23).

[40] Lukasz Cyra and Janusz Gorski. "Supporting Compliance with Security Standards by Trust Case Templates". In: *2nd International Conference on Dependability of Computer Systems (DepCoS-RELCOMEX '07)* (June 2007), pp. 91–98 (cit. on p. 23).

[41] Georg Macher, Eric Armengaud, Eugen Brenner, and Christian Kreiner. "Threat and risk assessment methodologies in the automotive domain". In: *Procedia computer science* 83 (2016), pp. 1288–1294 (cit. on pp. 23, 24).

[42] Microsoft. *The STRIDE Threat Model*. https://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx. Accessed: 2017-09-20 (cit. on pp. 23, 75).

[43] Adam Shostack. *Threat modeling: Designing for security*. John Wiley & Sons, 2014 (cit. on p. 23).

[44] Webtrend. *Threat Modeling with STRIDE*. https://www.webtrends.com/blog/2015/04/threat-modeling-with-stride/. Accessed: 2017-09-20 (cit. on pp. 23, 75).

[45] Sam Lauzon. *Secure Software Updates for Automotive Systems: Introduction to the Uptane SOTA Solution*. 2017 (cit. on pp. 23, 74).

[46] Nancy Leveson. *Engineering a safer world: Systems thinking applied to safety*. MIT press, 2011 (cit. on pp. 24, 92, 94).

[47] William Young and Nancy Leveson. "Systems thinking for safety and security". In: *Proc. of the 29th Annual Computer Security Applications Conference* (2013), pp. 1–8 (cit. on p. 24).

[48] Ivo Friedberg, Kieran McLaughlin, Paul Smith, David Laverty, and Sakir Sezer. "STPA-SafeSec: safety and security analysis for cyber-physical systems". In: *Journal of Information Security and Applications* 34 (2017), pp. 183–196 (cit. on p. 24).

[49] Sam Procter, Eugene Y Vasserman, and John Hatcliff. "SAFE and secure: Deeply integrating security in a new hazard analysis". In: *ARES* (2017), p. 66 (cit. on p. 24).

[50] Daniel Pereira, Celso Hirata, Rodrigo Pagliares, and Simin Nadjm-Tehrani. "Towards combined safety and security constraints analysis". In: *SAFECOMP* (2017), pp. 70–80 (cit. on p. 24).

[51] Emmanuel Aroms et al. "NIST Special Publication 800-30 Risk Management Guide for Information Technology Systems". In: (2012) (cit. on p. 24).

[52] Tiago Amorim, Helmut Martin, Zhendong Ma, Christoph Schmittner, Daniel Schneider, Georg Macher, Bernhard Winkler, Martin Krammer, and Christian Kreiner. "Systematic Pattern Approach for Safety and Security Co-engineering in the Automotive Domain". In: *SAFECOMP* (2017), pp. 329–342 (cit. on p. 24).

[53] Kateryna Netkachova, Kevin Müller, Michael Paulitsch, and RE Bloomfield. "Security-informed safety case approach to analysing MILS systems". In: (2015) (cit. on p. 25).

[54] Robin Bloomfield, Kateryna Netkachova, and Robert Stroud. "Security-informed safety: if it's not secure, it's not safe". In: *Int. Workshop on Software Engineering for Resilient Systems* (2013), pp. 17–32 (cit. on pp. 25, 57).

[55] Tim Kelly. "Reviewing assurance arguments-a step-by-step approach". In: *Workshop on assurance cases for security-the metrics challenge, dependable systems and networks (DSN)* (2007) (cit. on pp. 25, 88, 115).

[56] Tim Kelly. "Chapter 16: Safety Cases". In: *Handbook of Safety Principles* (2018), pp. 361–385 (cit. on p. 26).

[57] Tangming Yuan, Tim Kelly, Tianhua Xu, Haifeng Wang, and Lin Zhao. "A dialogue based safety argument review tool". In: *Proceedings of the 1st International Workshop on Argument for Agreement and Assurance (AAA-2013), Kanagawa, Japan* (2013) (cit. on p. 26).

[58] Shuichiro Yamamoto and Shuji Morisaki. "A system theoretic assurance case review". In: *2016 11th International Conference on Computer Science & Education (ICCSE)* (2016), pp. 992–996 (cit. on p. 26).

[59] Jose Luis de la Vara, Gabriel Jiménez, Roy Mendieta, and Eugenio Parra. "Assessment of the Quality of Safety Cases: A Research Preview". In: *International Working Conference on Requirements Engineering: Foundation for Software Quality* (2019), pp. 124–131 (cit. on p. 27).

[60] Shuichiro Yamamoto. "Assuring security through attribute GSN". In: *2015 5th International Conference on IT Convergence and Security (ICITCS)* (2015), pp. 1–5 (cit. on p. 27).

[61] Tangming Yuan, Suresh Manandhar, Tim Kelly, and Simon Wells. "Automatically Detecting Fallacies in System Safety Arguments". In: *Principles and Practice of Multi-Agent Systems* (2015), pp. 47–59 (cit. on p. 27).

[62] John Spriggs. *GSN-The Goal Structuring Notation: A Structured Approach to Presenting Arguments*. Springer Science & Business Media, 2012 (cit. on p. 27).

[63] Yihai Chen, Mark Lawford, Hao Wang, and Alan Wassyng. "Insulin pump software certification". In: *International Symposium on Foundations of Health Informatics Engineering and Systems* (2013), pp. 87–106 (cit. on p. 27).

[64] Yaping Luo, Mark van den Brand, Zhuoao Li, and Arash Khabbaz Saberi. "A systematic approach and tool support for GSN-based safety case assessment". In: *Journal of Systems Architecture* 76 (2017), pp. 1–16 (cit. on p. 27).

[65] Zarrin Langari and Tom Maibaum. "Safety Cases : A Review of Challenges". In: *2013 1st International Workshop on Assurance Cases for Software-Intensive Systems, ASSURE 2013 - Proceedings* (2013), pp. 1–6 (cit. on p. 28).

[66] European Organization for the safety of air navigation. *Safety Case Development Manual*. 2.2. DAP/SSH/091. European Air Traffic Management (cit. on p. 28).

[67] Charles B Weinstock and John B Goodenough. *Towards an assurance case practice for medical devices*. Tech. rep. Carnegie-Mellon Univ. Pittsburgh, PA, Software Engineering Institute., 2009 (cit. on p. 28).

[68] PR Mayo. "Structured safety case evaluation: a systematic approach to safety case review". In: *Proceedings of the first IET international conference on system safety* (2006), pp. 164–173 (cit. on p. 28).

[69] March HSE. *Assessment Principles for Offshore Safety Cases (APOSC)*. 2006 (cit. on p. 29).

[70] Office for Nuclear Regulation. *The purpose, scope, and content of safety cases*. Revision 4. NS-TAST-GD-051 (cit. on p. 29).

[71] Office of Nuclear Regulatory Research (RES). *Technical Basis to Review Hazard Analysis of Digital Safety Systems*. Office of New Reactors (NRO). 2011. Chap. RIL-1101 (cit. on p. 29).

[72] Alvine Boaye Belle, Timothy C Lethbridge, Sègla Kpodjedo, Opeyemi O Adesina, and Miguel A Garzón. "A Novel Approach to Measure Confidence and Uncertainty in Assurance Cases". In: *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)* (2019), pp. 24–33 (cit. on p. 30).

[73] Xingyu Zhao, Dajian Zhang, Minyan Lu, and Fuping Zeng. "A new approach to assessment of confidence in assurance cases". In: *International Conference on Computer Safety, Reliability, and Security* (2012), pp. 79–91 (cit. on p. 30).

[74] Chunchun Yuan, Ji Wu, Chao Liu, and Haiyan Yang. "A Subjective Logic-Based Approach for Assessing Confidence in Assurance Case." In: *International Journal of Performability Engineering* 13.6 (2017) (cit. on p. 30).

[75] Sunil Nair, Neil Walkinshaw, Tim Kelly, and Jose Luis de la Vara. "An evidential reasoning approach for assessing confidence in safety evidence". In: *2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE)* (2015), pp. 541–552 (cit. on p. 30).

[76] Silviya Grigorova and TSE Maibaum. "Argument evaluation in the context of assurance case confidence modeling". In: *2014 IEEE International Symposium on Software Reliability Engineering Workshops* (2014), pp. 485–490 (cit. on p. 30).

[77] Anaheed Ayoub, Jian Chang, Oleg Sokolsky, and Insup Lee. "Assessing the overall sufficiency of safety arguments". In: (2013) (cit. on pp. 30, 194–198, 216, 217).

[78] Rui Wang, Jérémie Guiochet, and Gilles Motet. "Confidence assessment framework for safety arguments". In: *International Conference on Computer Safety, Reliability, and Security* (2017), pp. 55–68 (cit. on pp. 30, 144).

[79] Chung-Ling Lin, Wuwei Shen, Steven Drager, and Betty Cheng. "Measure confidence of assurance cases in safety-critical domains". In: *2018 IEEE/ACM 40th International Conference on Software Engineering: New Ideas and Emerging Technologies Results (ICSE-NIER)* (2018), pp. 13–16 (cit. on p. 31).

[80] Lian Duan, Sanjai Rayadurgam, Mats Heimdahl, Oleg Sokolsky, and Insup Lee. "Representation of confidence in assurance cases using the beta distribution". In: *2016 IEEE 17th International Symposium on High Assurance Systems Engineering (HASE)* (2016), pp. 86–93 (cit. on p. 31).

[81] Lian Duan, Sanjai Rayadurgam, Mats PE Heimdahl, Oleg Sokolsky, and Insup Lee. "Representing confidence in assurance case evidence".

In: *International Conference on Computer Safety, Reliability, and Security* (2014), pp. 15–26 (cit. on pp. 31, 27).

[82] Ewen Denney, Ganesh Pai, and Ibrahim Habli. "Towards measurement of confidence in safety cases". In: *2011 International Symposium on Empirical Software Engineering and Measurement* (2011), pp. 380–383 (cit. on p. 31).

[83] Victor R. Basili, Gianluigi Caldiera, and H Dieter Rombach. "The goal question metric approach". In: *Encyclopedia of software engineering* (1994), pp. 528–532 (cit. on p. 31).

[84] Thomas Chowdhury, Chung-Wei Lin, BaekGyu Kim, Mark Lawford, Shinichi Shiraishi, and Alan Wassyng. "Principles for systematic development of an assurance case template from ISO 26262". In: *2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)* (2017), pp. 69–72 (cit. on pp. 32, 40, 43, 44, 47, 49, 51, 57, 218).

[85] Valentin Cassano and Tom Maibaum. *Conceptual Models of ISO 26262 Work Products and Their Role in Complying with the Standard*. Tech. rep. McSCS-2016-002. McMaster Centre For Software Certification, 2016, pp. 1–24 (cit. on pp. 33, 35).

[86] Kateryna Netkachova, Kevin Müller, Michael Paulitsch, and RE Bloomfield. "Security-informed safety case approach to analysing MILS systems". In: *International Workshop on MILS: Architecture and Assurance for Secure Systems* (2015), pp. 19–21 (cit. on p. 58).

[87] Thomas Chowdhury, Eric Lesiuta, Kerianne Rikley, Chung-Wei Lin, Eunsuk Kang, BaekGyu Kim, Shinichi Shiraishi, Mark Lawford, and Alan Wassyng. "Safe and secure automotive over-the-air updates". In: *International Conference on Computer Safety, Reliability, and Security* (2018), pp. 172–187 (cit. on pp. 60, 67, 79, 81, 82, 218).

[88] Thomas Chowdhury, Alan Wassyng, Richard F Paige, and Mark Lawford. "Criteria to Systematically Evaluate (Safety) Assurance Cases". In: *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)* (2019), pp. 380–390 (cit. on pp. 83, 218).

[89] Richard Hawkins, Ibrahim Habli, Dimitrios S. Kolovos, Richard F. Paige, and Tim Kelly. "Weaving an Assurance Case from Design: A Model-Based Approach". In: *16th IEEE International Symposium on High Assurance Systems Engineering, HASE 2015, Daytona Beach, FL, USA, January 8-10, 2015* (2015), pp. 110–117 (cit. on p. 87).

[90] Peter Bishop and Robin Bloomfield. "A Methodology for Safety Case Development". In: *Industrial Perspectives of Safety-critical Systems* (1998). Ed. by Felix Redmill and Tom Anderson, pp. 194–203 (cit. on p. 88).

[91] David Lorge Parnas. "On the Criteria to Be Used in Decomposing Systems into Modules". In: *Communications of the ACM* 15.12 (Dec. 1972), pp. 1053–1058. ISSN: 0001-0782 (cit. on pp. 88, 115).

[92] John Rushby. "Formalism in safety cases". In: *Making Systems Safer* (2010), pp. 3–17 (cit. on p. 95).

[93] Ying Jin and David Lorge Parnas. "Defining the meaning of tabular mathematical expressions". In: *Science of Computer Programming* 75.11 (2010), pp. 980–1000 (cit. on p. 100).

[94] Timothy Patrick Kelly. "Arguing safety: a systematic approach to managing safety cases". PhD thesis. University of York York, UK, 1999 (cit. on pp. 100, 102).

[95] Monika Szczygielska and Aleksander Jarzebowicz. "Assurance case patterns on-line catalogue". In: *Advances in Dependability Engineering of Complex Systems* (2017), pp. 407–417 (cit. on p. 100).

[96] Patrick J Graydon and C Michael Holloway. "An investigation of proposed techniques for quantifying confidence in assurance arguments". In: *Safety science* 92 (2017), pp. 53–65 (cit. on p. 102).

[97] Thomas Chowdhury, Alan Wassyng, Richard F Paige, and Mark Lawford. "Systematic Evaluation of (Safety) Assurance Cases". In: *International Conference on Computer Safety, Reliability, and Security* (2020), pp. 18–33 (cit. on pp. 104, 106, 125, 147–149, 218).

[98] ChangeVision. *Astah GSN*. https://astah.net/products/astah-gsn/. Accessed: 2020-04-09 (cit. on pp. 150, 171, 173).

[99] Tim Kelly and John McDermid. "Safety case patterns-reusing successful arguments". In: *IEE Colloquium on Understanding Patterns and their Application to Systems Engineering* (1998) (cit. on pp. 154, 170, 173, 178, 188, 192–194, 201, 214).

[100] European Organisation for the Safety of Air Navigation. *Preliminary safety case for ADS-B airport surface surveillance application. PSC ADS-B-APT*. 1.2. 2011, pp. 1–188. 214 pp. (cit. on p. 22).

[101] EUROCAE ED-163/RTCA DO-321. *Safety Performance and Interoperability Requirements Document for ADS-B-APT Application* (cit. on pp. 23, 26).

[102] EUROCAE ED-78A / RTCA DO-264. *Guidelines for approval of the provision and use of Air Traffic Services supported by data communications*. 2000 (cit. on p. 26).

[103] ALTRAN-ATM Division, National Aerospace Laboratory NLR and AFI RVSM Project Management Team. *AFI RVMS Pre-Implementation Safety Case*. Final. 2008. 167 pp. (cit. on p. 28).