Patterns for certification standards

Kevin Delmas & Claire Pagetti & Thomas Polacsek*

Abstract

One of the absolute preconditions for a safety-critical system to enter the market is to be issued a certificate *by the regulating authorities. To this end, the "applicant" must demonstrate the compliance of its product with the domain's standards. The high complexity of this process has led applicants to rely on* assurance cases *made for certification in the medical, nuclear, or aeronautic domains. In this paper, we propose a generic method that guides the applicant through the specification of assurance cases for a complex standard. Unlike existing works focused on a single context, our objective is to provide an approach that is both generic and domain-agnostic. In order to illustrate this new approach, we present the results of its application on a real-world case study, which pointed out new issues and led to improvements.*

1. Introduction

Context. Safety-critical systems, i.e. systems with the potential to endanger a person's life, are often subject to a *certification process*. In practice, any *applicant* requesting the certification of a system is in charge of convincing a *certification authority* that their product is compliant with the regulatory requirements. When the authorities are positively convinced, they deliver a certificate that authorizes its operation. Examples of such authorities include: the European Medicines Agency (EMA) and the Food and Drug Administration (FDA), for drug evaluation; or the European Aviation Safety Agency (EASA) and the Federal Aviation Administration (FAA), for civil aviation safety.

To support applicants in this task, expert committees, composed of companies, certification authorities and academics, have defined standards, guidelines or recommendations (that will be simply referred as standards in the sequel) [1]. These standards are complex documents, which provide high-level certification objectives to be fulfilled and often require experts to understand precisely what is expected by the certification. Moreover, there are two main types of standards: those which only define objectives without imposing any method in order to give some leeway to applicants in their development and validation; and conversely those which impose some high-level process not easy to implement.

Assurance cases for certification. Practically, an applicant must provide all the elements concerning the design of the system and the Verification and Validation (V&V) operations that have been carried out. In addition, they must also *argue* why these are sufficient to address all of the certification authority's concerns. In this context, for applicants (and system designers), the problem is to argue well and, for the certification authority, the problem is to evaluate an argument. As [4] points out for reliable systems, the system must provide a service that can legitimately be trusted, with trust being established through plausible links between the evidence provided and the fact that the system provides the expected service.

In order to cope with the complex activities associated with certification, industries are increasingly relying on *assurance cases*. An assurance case can be defined as "*an organized argument*

*Authors version, Patterns for certification standards, 417-432, 32nd International Conference on Advanced Information Systems Engineering, CAiSE 2020, Grenoble, France, June 8-12, 2020, Proceedings. Lecture Notes in Computer Science 12127, Springer 2020, ISBN 978-3-030-49434-6

[1] Examples of standards are DO178, ARP4754 for aeronautics, ISO 26262 for automotive and EC 62366, EC 62304 for medical devices.

that a system is acceptable for its intended use with respect to specified concerns" [34]. In practice, to build an assurance case, the applicant is free to organize their argumentation and to use any kind of format. However, especially in the safety world, practitioners rely on dedicated formalisms such as the *Goal Structuring Notation* (GSN) [27, 19]. In addition, several works [39, 28, 8] suggest a pattern approach to design assurance case. In engineering, the design pattern approach is a way of describing a recurring problem and its associated solution based on best practices [2, 11]. In a certification context, these assurance case patterns consist of a generic assurance case that lists, for a given claim, the associated evidences and the justification of why the claim could be concluded. Those patterns are then instantiated for a particular product and usage domain.

Towards a generic method to build assurance cases. Even though the literature provides assurance case notations and consensus on the necessity of patterns approach, there is almost no work, apart from [14, 42], on how to make a pattern. In fact, designing assurance case patterns and instances is really challenging and requires numerous skills. So the aim of this paper is to propose a method for designing certification assurance case patterns.

Through various projects, we have already had the opportunity to design patterns in the medical field, embedded aeronautical systems and assembly line [8, 33, 5]. In all these projects, the design process was not clearly defined so the construction of the patterns was quite tedious and time-consuming. This is the reason why we tried to define a method that is as generic as possible. This method was designed using a trial and error approach. Of course, we did not design our process from scratch, but we gradually enriched the process and defined the practices (roles and wording) step by step.

After presenting the general context and notations in section 2, we define, in section 3, a method to design patterns for certification standards. In section 4, we detail the lessons learned when applying the method on a specific standard. Section 5 is dedicated to related work and we conclude in section 6.

2. Background and motivation

2.1. Certification

An applicant must provide a *compliance demonstration* that its product is compliant with the standards where a compliance demonstration is a set of assurance cases, each applying to a high-level objective. High-level objectives are usually defined as a sort of a reachable goal (sometimes process-oriented activities) and there is no indication on how to achieve the goal. Since nothing is imposed on the manner to develop or validate a product, applicants can rely on numerous solutions to fulfill an objective. For example, for the certification of a kettle, an objective may indicate that it is necessary to identify all scenarios where a user may be injured and show how those situations are mitigated. The ways to proceed (both for hazard identification and mitigation means validation) are not fixed by the standard. For instance, if, to reduce the risk of burns, the designer has put on a handle that remains always cold, it is up to them to demonstrate that this indeed mitigates the risk.

Any standard comes with an intrinsic complexity: high-level objectives are not always easy to understand and are very generic, rationales are not always provided, etc. Moreover, a compliance demonstration encompasses all the concerns of the certification authority, such as safety, security [3] or dependability [41]. This means that certification activities involve several people that need to have transverse and large spectrum knowledge of the product, the process and/or the V&V activities. Such a complexity can be a real obstacle, especially for small companies, to enter in safety critical markets. Thus, offering more tractable approaches is mandatory and our work is a

way.

2.2. Assurance cases

In order to help applicant organize their documentation, several works propose to structure argumentation demonstration with assurance cases and some adequate notations. We can cite for instance, on the academic side, GSN [27, 19], *Claim-Argument-Evidence* [9], *Justification Diagram* [33] and, on the standardization organism side, *Structured Assurance Case Meta-model* [31].

All of these notations organize in diagrammatic form the various elements, formal and informal, that contribute to the justification of a result. These frameworks are all based on the model of the British philosopher Stephen Toulmin [37]. His purpose was to define a structure to help assess the validity of a judgement issued on the basis of justifications. In Toulmin's model, any argumentation is composed of a conclusion, namely the *claim*, and facts on which the claim is based. Basically, Toulmin has a legalistic view: to argue well amounts to stating a claim based on facts. In addition to these facts, Toulmin adds information about the reasoning process. This information clarifies why the inference is acceptable, why a set of justifications lead to a conclusion. Typically, in the legal field, this information corresponds to a reference to an article of law. Toulmin writes that this distinction "*is similar to the distinction drawn in the law courts between questions of fact and questions of law*". Toulmin called this additional information a *warrant*. Warrants are therefore what allow the passage from facts to claim, they justify the inference. Distinguishing between facts and warrants is not always easy. Warrants relate to the strength of the argumentation, they are general, whereas reasons depend more on data related to the context. To these three concepts, Toulmin adds other notions for the qualification of the conclusion and the backing of the warrants.

All assurance case notations focus on the three concepts: claim, warrant and fact, although terminology is sometimes changed, for example *strategy* is used in place of warrant in GSN [19]. We have chosen an agnostic notation approach based on a textual syntax (kind of abstract syntax) compliant with all existing notations (kind of concrete syntax). We rename fact as *evidence* because our argumentation does not really refer to established facts but to documents, for instance calculation results, test reports or expert judgements. The notation is hierarchical since an evidence of one pattern may also be the claim of another one. A final evidence refers to a terminal element that does not become a claim for another pattern. Such a final evidence could be a document or an analysis.

`Claim:` All hazards identified
`Warrant:` Analysis acceptable by the authority
`Evidence:`
`(E1)` Means for correctness
`(E2)` Means for completeness

Figure 1: *Pattern example for the kettle*

`Claim:` All hazards identified
`Warrant:` Functional Hazard Analysis
`Evidence:`
`(E1)` Correctness: external safety experts reviews
`(E2)` Completeness: former accidents database

Figure 2: *Instance example for the kettle*

Figure 2 is a possible assurance case, for the kettle example, that answers part of the objective on identifying the hazards. To establish the claim, the justification relies on a Functional Hazard Analysis, a classical safety technique to extract hazards. Such an analysis, to be trustworthy, requires reaching a certain correctness level, based here on a double review by a second experts' team (E1), and also on a certain level of completeness, based here on checking the list with known accidents (E2).

For Toulmin, the notion of warrant is the cornerstone of reasoning. Indeed, it gives the rational and explains why a conclusion can be assessed. Even if some practitioners tend not to use the notion of warrant, it is difficult to evaluate an argument where the warrant is not explicit, in particular for an auditor. For us, even a simple aggregation with an "*and*", like a decomposition strategy for warrant, needs to be explicit. Indeed, a simple conjunction, such as "*and*" between evidence, can hide more complex mechanisms (e.g. check that the evidences are not contradictory or check whether they are sufficient).

2.3. Patterns notation

[22] promoted the use of a collection of assurance case patterns, with the aim of rationalizing and reusing elements from previous assurance cases. The authors of [20] provide a format, including meta-data, that allows to capture and reuse patterns. In the case of medical devices, the authors [40] explain all the advantages of using patterns in standards; and their arguments are valid in any application domain.

Figure 1 shows an assurance case pattern (also referred as justification pattern) for the identification of all kettle hazards. A possible instantiation of the pattern is given Figure 2. The pattern is generic and could be reused for other products that need a risk analysis.

2.4. Justification pattern elicitation problems

Building an assurance case, pattern or instance, is not an easy task. Each pattern can be seen as a guide that lists the necessary elements to meet an objective. The design of a pattern must involve experts who will define the patterns according to their technical domain knowledge and of the established good practices, standards, quality requirements, etc. The main pitfall is the introduction of mistakes during the design of the patterns, which are meant to guarantee the validity of the reasoning.

The problem when it comes to making justification patterns is to think in terms of inference, that is, determine whether or not it is acceptable to pass from a set of justifications to a given claim and to elicitate why this inference is correct. Experts tend to cling to their technical knowledge and how different activities are organized; whereas claims often target quality and safety reached levels. *Critical Thinking* [17] and the usage of *guide words* (as done in some methodologies like HAZOP[2] [21]) may support the experts in their task.

There are many cognitive biases that influence human reasoning. Among them, there is a tendency to consider one's own subjective interpretation as the truth about reality. Research in psychology has shown that one of the implications of this cognitive bias is our inability to judge our understanding and ignorance of what we know. In other words, we think we understand and have valid explanations for phenomena that we do not really understand. On sensitive subjects, the situation is such that we can greatly overestimate the quality of our justifications and reasoning [10]. However, it is possible to compensate for this bias through dialogue. As many studies have confirmed, group reasoning in a collaborative way is more effective than individual reasoning, especially for reasoning and logic problems[3] [38, 24].

Regarding legitimacy, the experts must be considered as experts in their field by the people who will use the patterns. This legitimacy can only be acquired through credentials and recognition of competence by peers. In practice, the legitimacy comes from expert's resume, from the projects he

[2]HAZOP for *HAZard and OPerability analysis* is an industrial risk analysis method.

[3]Moshman and Geil showed on a reasoning problem, with a cohort of 20 groups and 32 individuals, that 75% of the groups found the right answer for only 9.4% of individuals [30]. It should also be noted that groups build more sophisticated, qualitatively, arguments than an individual.

has already collaborated on. Thus, an expert is most often someone who has already participated in system certification and/or made recognized contributions (usually in industrial conferences). The question of legitimacy arises with regard to the certification authority. It is the authority who will ultimately decide whether a person is an expert or not.

Finally, from our experience, patterns are very well received and accepted in a group (e.g. company) if they were collaboratively designed by experts working on the side of the applicant and experts belonging to the certification authority.

3. A method to design certification pattern

Our objective is to define a method to help applicant build a repository of justification patterns dedicated to their specific standard(s). To each objective is associated a pattern. Since correctness and completeness of a pattern can be altered by process flaws and psychological biases, the method concentrates on detecting and correcting these flaws as much as possible.

3.1. Process

Our method is based on a long process to construct justification patterns via several expert meetings. The process, given in Figure 3, is composed of four iterative steps described below. Note that for a given claim, several patterns may exist since a same claim may be justified in several ways.

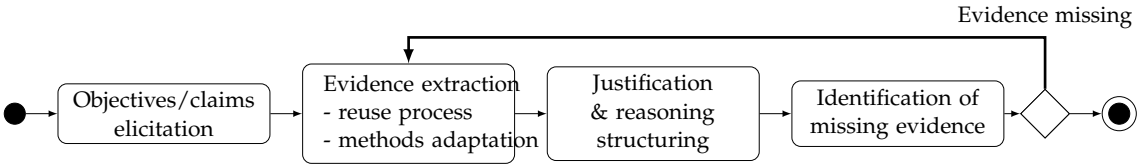

Figure 3: *A first pattern design process*

Objectives/claim elicitation. Identify the certification objectives the product or process must comply to. Each objective is considered to be a top-level claim. As the process is iterative, some justifications (evidences) defined during an iteration may become a claim.

Evidence extraction. There are mainly two cases for eliciting evidence: either the applicant has some experience on the claim and can rely on existing practices that have already been applied and convinced the authority. In which case, they can transform the process as a pattern and this corresponds typically to the classic design pattern approach where the pattern captures good practices and well-known solutions. Or the standard applies to a new technology or a new method, in which case experts have to find a fully new solution which can rely on methods coming from any other relevant domain. The result of this activity is an unorganized set of evidences (new claims or final evidences).

Justification & reasoning structuring. The activity consists in taking all identified evidences and articulating the inference, or different inferences, that lead(s) to the claim. This step defines the structure of the pattern and the associated warrants.

Identification of lack of evidence or end of the process. When structuring the pattern, the experts may observe that some elements are missing in their reasoning, meaning that evidences are missing. The most common problem is to *forget some final evidences or intermediate claims* to sustain the objective. Thus, between two meetings, the experts must individually think on the patterns they have designed together, looking for mistakes, problems and missing elements

possibly introduced during justification structuring. Alternating group and individual works is very important[4]. Indeed, collaborative reasoning facilitates individual cognitive progress, but it is also important for experts to take stock: team influences more individuals than individual influences team [23]. Any doubt should be discussed and traced at the next meeting, not to ask the same question several times. A lack of evidence can be a clue of some missing process, method or practices that, at first glance, seems not to sustain the objective but after deeper inspection provides some lack of evidences.

3.2. Organization

Designing patterns is both an individual and a collective task. To this end, meetings are organized. The purpose of these meetings is to engage in the construction of a common reference framework and to compare points of view. From there, a *justification pattern design team* (denoted *design team* in the remainder of the paper) will be able to collectively elicit justification patterns. The team should be small, three to five persons. Small teams encourage *dialogical* interaction (conversation between two people). To tackle the problem of deducting reasoning, psychological studies have shown that dialogical and small groups are very effective [38, 23, 24]. During the constitution of the design team, one must take into account the psychological biases of system experts. Especially for experts involved in the design of a system whose compliance to the certification objectives depends on the designed patterns. These experts are susceptible to confirmation bias (as identified in [25]) and thus may try to build assurance cases enforcing the compliance of their own system (a typical case of such a bias is illustrated in the accident report [13]).

During meetings, the experts must have all the necessary information: the standard, all the technical documentation, the past experiences. To create the patterns, the team must be able to share a common medium and "*draw*" patterns together (e.g. white board with markers). In order for the experts to work individually between meetings, it is also important to have minutes of meeting that include the patterns and detailed explanations of the elements of the patterns.

We recommend to have a design team composed of one *facilitator* managing the meetings and recording the patterns and *experts* designing the pattern.

Facilitator Role. The facilitator should help determine whether or not it is acceptable to pass from a set of justifications to a given claim. The study of such reasoning has expanded in North America since the 1970s, particularly since the publication of *Logical Self-defense* [17]. In this book, the authors attempt to define a systematic approach to studying informal argumentation. Thus, in recent years, all research that relates to non-formal reasoning has been called *Informal Logic*, *Critical Thinking* and *Argumentation*. To support the experts in their task of eliciting and explaining the inference, the facilitator must be very familiar with Critical Thinking. There is no need for the facilitator to be an expert in the areas covered by the standard, but they will still need to know the vocabulary and the context in order to communicate easily with the experts. Indeed, a minimum of technical knowledge is required for the experts to express their ideas without always having to explain technical issues. The facilitator is thus paramount in identifying a misuse of the pattern formalism that can lead to the following threats to pattern validity: introduction of *unnecessary evidence*, the *lack of evidence* and *fallacious inference*. If several members of the team are familiar with the Critical Thinking, we recommend alternating the role between meetings.

Expert Role. An expert must be a specialist in the field covered by the standard and, more precisely, a specialist in the V&V methods used to define the pattern. Indeed, the justifications

[4] In a sense, we are quite close to the Delphi method [29] here since, between two meetings, the experts think alone, in isolation, about what has been collectively produced, the synthesis, and give their feedback at the next meeting. However, unlike the Delphi method, in our method much of the work is done during group meetings.

and warrant of a pattern are generally related to V&V operations and results. To ensure the acceptance of the patterns, the expert must have credentials recognized by their peers and by the certification authority. Involving recognized experts prevents the design of incorrect patterns due to a *poor knowledge* of the application domain in which the pattern is intended to be used. An expert could be a well-known practitioner, a researcher or a member of the certification authority. Note that, it is better to have both practitioners and members of the authority in the team. Indeed the heterogeneity of the experts can address two threats by helping to identify *missing patterns* and avoiding a *non-holistic view*. A non-holistic view is when the pattern does not treat the whole problem but only adopts the point of view of the applicant or of the certification authority.

3.3. Wording

The way the design team brainstorms has a major impact on the avoidance of common mistakes. Hence, we define *guide words* and *avoid words* to promote an argumentation thinking mindset rather than a temporal thinking one and to ensure that warrants are not forgotten.

Temporal thinking. One of the major difficulties when developing a pattern is to elicit an inference and not a process. Again, experts know how the system has been designed and they are tempted to graft the development process to the justification pattern. Writing a sequence of actions can lead to simply paraphrasing a process and thus concealing the underlying rationale justifying the claim. The claim is no longer the result of an argument, but of a series of activities and this does not sustain the claim. This threat of *temporal thinking* can be mitigated if the meeting participants avoid using all vocabulary relating to time. In other words, experts should try not to use the words: follow, after, before, then, etc. Instead, the facilitator should question the experts and direct them towards reformulation using the wording: "*the conclusion of*", "*needs*", "*is based on*", etc.

Warrantless approach. Experts may be familiar with formal logic and tend to build a proof tree instead of a pattern representing informal argumentation. This formal thinking usually leads to *logical* warrants, a symptomatic case is logical decomposition (the claim is the conjunction of the evidence). Of course, if one is able to express the argumentation in a formal way then this formal proof should be a final evidence and does not need to be represented as an argumentation pattern. Nevertheless, the facilitator must seek carefully this kind of warrants since it may conceal the actual warrant that allows the passage from evidence to claim. In the context of argumentation, the experts should avoid warrants containing only logical connectors: "*and*", "*or*", "*entails*", etc.

4. Case study

We have applied our method on the CAST-32A [6], that serves as a guideline to certify multi-core processor-based systems in avionics. All embedded platforms until now relied on mono-processor hardware or very specific dual-core. In the coming years, only multi-core processor hardware will be available on the market and the airframers will have no choice but to embed these new architectures. Since the CAST-32A is a new guideline, there is currently no process to refer to and applicants must create their argumentation from scratch. This is a perfect opportunity to apply our method.

4.1. Application of the method

The *Design team* was composed of: 1. a senior expert on multi-core processor architectures, predictable programming and the mainstream aeronautics validation and verification process; 2. a junior safety expert of the safety assessment of technical systems; 3. a facilitator with a solid

experience in justification pattern design and familiar with the overall V&V process used in aeronautics.

During the project, the justification pattern design team had a meeting once every two weeks, and each member individually took some time to ponder on the work that was done.

By the end, the design team defined 15 patterns[5] that address 5 high-level objectives of the guideline (some objectives, such as those that are purely organizational, have not been addressed in the context of this project).

4.2. Justification pattern for RU3

Let us describe one of the objectives, namely RU3 (for *resource usage 3*) and part of the associated patterns. This objective concerns interference situations, which are feared situations where software can encounter strong slowdowns.

Objective RU3 *The applicant has identified the interference channels that could permit interference to affect the software applications hosted on the multi-core processor cores, and has verified the applicant's chosen means of mitigation of the interference.*

`Claim:` RU3
`Warrant: (W1)` Check completeness of interference and mitigation
`Evidence:`
`(E1)` Identification and classification of interferences
`(E2)` Verified mitigation means

Figure 4: *Pattern for RU3*

`Claim:` E1
`Warrant: (W2)` Platform stressing strategy
`Backing:` Architecture mastering
`Evidence:`
`(E3)` Interference identification
`(E4)` Effect classification
`Given:` Configuration, temporal constraints

Figure 5: *Pattern for E1*

Figure 4 shows its transcription as a pattern. Evidence (E1) states that the existing interferences have been identified and classified. Focusing on (E1), Figure 5, it has been achieved because there was a stressing benchmark analysis that has collected the effects of each interference (strategy (W2)). Those effects can be expressed in different units (e.g. delay, bandwidth). Evidence (E3) points to a report that summarizes which interferences have been identified, how they have been identified, and why the identification is sound and complete. Evidence (E4) points to a safety report that details the acceptable effects on the hosted applications. From this information, the applicant has defined adequate means of mitigation to prevent, for instance, unacceptable effects. Evidence (E2) collects all those means of mitigation, how they mitigate each unacceptable interference and how they were verified. The applicant can argue the compliance with RU3 because an expert, who masters the architecture, has reviewed and double-checked that each interference has been correctly mitigated (W1).

4.3. Lessons learned

The facilitator supports the elicitation of patterns

Both experts clearly reported that the facilitator helped them understand the argumentation approach. When designing the first patterns, experts tended to not know how to express the warrant, to skip it, and to describe a process rather than an argument. Interestingly, the further the project progressed, the more the experts understood how to operate. However, although the

[5] available at `https://w3.onera.fr/phylog/patterns`

experts became familiar with the approach, a facilitator was always needed. By being outside of the context, facilitators rephrase the discussions and question the foundations of what may seem obvious to experts (by using, for example, the Douglas Walton's critical questions [12]). In the future, it would be preferable to define more precisely the skills of the facilitator as well as the way in which meetings should be conducted. To do this, we can take inspiration from, for example, [29].

Wording importance

The wording was really necessary to prevent the experts falling in false reasoning. It helped counter the tendency to express what needs to be done rather than what leads to a justification.

Process / Patterns evaluation

To evaluate the process, we must turn to an evaluation of the produced patterns. At the end, the design team presented the justification patterns in a workshop, the participants of which were: two contributors to the CAST-32A, five well-known experts from the aeronautics industry and three certification authority members.

The overall feedback was very positive. For industrial experts, the patterns are very useful and help clarify some implicit / ambiguous textual rationales. Moreover, because they give concrete evidence, they simplify discussion between stakeholders. Industrial experts also gave some suggestions to prepare certification audits with the patterns. For CAST-32A contributors, the patterns were compliant with the writers' perspective. They confirmed that patterns highlight some elements that were only in the writers' minds. In fact, the design team has extracted the implicit structure of the sentences, the main elements expected to be supplied and made explicit the reasoning of the writers. For certification authority members, patterns provide a framework for legible and clear presentation of justifications and their rationale.

Of course, the patterns were not free of defects (some evidences were missing and some warrants were not explicit enough). In addition, it appeared that an additional pattern would be useful for easing the discussion and moving around the other patterns. The conclusion we can draw from this evaluation is that there is one step missing from our process. We could add an expert committee assessment to our process. In this new process, the assessment committee would become the validation team. At the end, this team would be involved in a validation activity and would address the following challenges:

- *fallacious reasoning:* find conditions where the warrants do not sustain the claim. Those conditions can either be considered as rebuttal and must be integrated into the patterns, or disclose a flaw to be corrected;
- *lack of evidence:* find conditions where the evidences are not sufficient to sustain the claim. In that situation, the design team should identify them out of the processes, methods and practices;
- *missing patterns:* find another way to establish the claim. This may look challenging since this requires designing a new pattern but it can be addressed by trying some slight modifications of the existing patterns and assessing the validity of this new version.

5. Related work

If there are many notations to structure an assurance case, there are fewer works addressing the justification patterns elicitation. In [42], the authors focus on security requirements and

propose a tool to manage these requirements. In addition, they are interested in capturing the rationality of these requirements by using Toulmin's scheme. While they give some key elements to produce such models, they do not go into the details (role, wording, etc.) of the elicitation method. In another field, [14] are interested in safety arguments and provide a guide on how to build a GSN diagram properly, but no elicitation method is proposed. In the avionic context, the authors of [43, 44] propose a UML profile, namely SafeUML, dedicated to safety requirements for an aeronautics guideline. This profile defines a set of stereotypes to model specific concepts associated to safety. The purpose of their approach is to facilitate communication between safety experts, software developers and certification authorities. Regarding the links with our approach, the different certification objectives are seen as requirements in SafeUML. Tractability between requirements and design choices is achieved by a stereotype "*rationale*" which has a text field to give an explanation. So, the use of our patterns could easily be added to SafeUML. Indeed, their application, linked to the rationale, would model more precisely this explanation of why a design meets a certification objective.

This idea of having a modeling framework to organize the certification elements is not new. It was particularly highlighted by [26, 1]. Among the works on compliance to a regulation, we can mention, for example, the SafetyMet meta-model safety oriented [7] or the UML stereotype developed by [32]. In the second case, with the UML stereotype-based approach, the authors give a generic approach to model a safety certification standard and make the link between the concepts of the system designer and those of the standard. Their method consists in supporting modeling a safety standard in their UML profile, then to make the link, according to precise rules materialized by OCL constraints, between the domain model and the certification model. This work, as identified by [1], models the *structure* of the standard to provide an organization of the elements provided by the applicant to satisfy the standard. However this work does not clarify the *intent* of the objectives of standard, this task being assigned here to the experts who will model the safety standard.

Still in the field of modeling, [15] propose to add an argumentative dimension to a combine model of i* and Nomòs [36] with the *Acceptability Evaluation framework* [18]. The purpose here is to capture expert discussions to determine whether the requirements in systems are compliant with a standard or whether there are irregularities. Unlike us, the authors focus here on an argumentation with contradictory points of view and the certification of a specific system, not to eliciting requirements from the standard.

Seeking to capture variability in regulatory texts, [35] propose a formalism to model conditions and exceptions in a regulation. In addition, their framework also allows them to express alternatives that are compliant with the standard. We could imagine a link between their approach and ours. Indeed, sometimes, for one certification objective, several justification patterns could be applicable. Depending on the chosen pattern, it is necessary to guarantee new sub-objectives which are the evidences of the pattern. Representing these alternatives and all the possible solutions could be a significant help for system designers.

Finally, close to our work, [16] use a Goal-Oriented approach to refine guideline objectives. This method allows clarify law and certification terms, that are subject to interpretation. However, unlike us, they do not attempt to highlight the rationality that allows us to conclude from sub-objectives to the main claim. Clearly explaining this, in particular by means of a warrant, is crucial for the certification authority side that is rarely taken into account as identified by [1].

6. Conclusion

This paper introduced a method to guide the design justification patterns by experts. The method has been applied to a new position paper written for multi-core processor and allowed design several patterns accepted by end users.

Repeatability and reproducibility are the main limitations of our approach. For the moment, even if the method results from a long standing experience, we have only used it on the CAST-32A. In the future, to consolidate the method, we will ask a new team to define patterns for the same standard and compare the results. As there are many ways to develop an argument, we will have to define the notion of equivalence between two patterns. A second axis of consolidation is to define patterns for another standard with the same team.

Future work will also need to address more deeper the problems of biases (anchoring, availability, bandwagon effect, halo effect, overconfidence, etc.) that may arise and their mitigation. To do this, we will have to rely on methods and works on expert knowledge elicitation.

Eventually, the current method does not characterize the assurance level provided by a given pattern nor an assessment of its cost. Our future works need to provide guidelines to document such impact to support the trade-off analysis of the applicant when several patterns are applicable. The question of how to instantiate a pattern is also an important issue and we will provide guidelines to help applicants on this matter as well as a method to conduct efficient certification audits with justification patterns and instances.

References

[1] Okhaide Akhigbe, Daniel Amyot, and Gregory Richards. A systematic literature mapping of goal and non-goal modelling methods for legal and regulatory compliance. *Requirements Engineering*, 24(4):459–481, 2019.

[2] Christopher Alexander, Sara Ishikawa, and Murray Silverstein. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, 1977.

[3] Rob Alexander, Richard David Hawkins, and Tim Kelly. *Security Assurance Cases: Motivation and the State of the Art*. Department of Computer Science, University of York, 2011.

[4] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl E. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable Sec. Comput.*, 1(1), 2004.

[5] Pierre Bieber, Frédéric Boniol, Guy Durrieu, Olivier Poitou, Thomas Polacsek, Virginie Wiels, and Ghilaine Martinez. MIMOSA: Towards a model driven certification process. In *Proc. of European Congress Embedded Real Time Software And Systems*, 2016.

[6] Certification Authorities Software Team. Multi-core Processors - Position Paper. Technical Report CAST 32-A, Federal Aviation Administration, 2016.

[7] Jose Luis de la Vara and Rajwinder Kaur Panesar-Walawege. SafetyMet: A metamodel for safety standards. In *Model-Driven Engineering Languages and Systems*, 2013.

[8] Clément Duffau, Thomas Polacsek, and Mireille Blay-Fornarino. Support of justification elicitation: Two industrial reports. In *Proceedings of International Conference Advanced Information Systems Engineering, CAiSE 2018*, 2018.

[9] Luke Emmet and George Cleland. Graphical notations, narratives and persuasion: a pliant systems approach to hypertext tool design. In *Proceedings of Hypertext and Hypermedia, HYPERTEXT 2002*, 2002.

[10] Matthew Fisher and Frank C Keil. The illusion of argument justification. *Journal of Exumlperimental Psychology: General*, 143(1), 2014.

[11] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing, 1995.

[12] David M. Godden and Douglas Walton. Argument from expert opinion as legal evidence: Critical questions and admissibility criteria of expert testimony in the american legal system. *Ratio Juris*, 19(3), 2006.

[13] Charles Haddon-Cave. *The Nimrod Review: an independent review into the broader issues surrounding the loss of the RAF Nimrod MR2 aircraft XV230 in Afghanistan in 2006, report*, volume 1025. DERECHO INTERNACIONAL, 2009.

[14] Richard Hawkins, Tim Kelly, John Knight, and Patrick Graydon. A new approach to creating clear safety arguments. In *Advances in systems safety*. Springer, 2011.

[15] Silvia Ingolfo, Alberto Siena, John Mylopoulos, Angelo Susi, and Anna Perini. Arguing regulatory compliance of software requirements. *Data Knowl. Eng.*, 87, 2013.

[16] F. Ishikawa, R. Inoue, and S. Honiden. Modeling, analyzing and weaving legal interpretations in goal-oriented requirements engineering. In *Proceedings of International Workshop on Requirements Engineering and Law*, 2009.

[17] Ralph Henry Johnson and J. Anthony Blair. *Logical Self-defense*. Key titles in rhetoric, argumentation, and debate series. International Debate Education Association, 2006. first edition 1977.

[18] Ivan Jureta, John Mylopoulos, and Stéphane Faulkner. Analysis of multi-party agreement in requirements validation. In *Proceedings of International Requirements Engineering Conference - RE'09*, 2009.

[19] Tim Kelly and Rob Weaver. The goal structuring notation - a safety argument notation. In *DNS 2004 Workshop on Assurance Cases*, 2004.

[20] Tim P. Kelly and John A. McDermid. Safety case construction and reuse using patterns. In *Proceedings of Safe Comp*, 1997.

[21] Trevor Kletz. *Hazop & Hazan – Identifying and Assessing Process Industry Hazards*. Institution of Chemical Engineers, 1999.

[22] John Knight. Advances in software technology since 1992. In *National Software and Airborne Electronic Hardware Conference, ser. FAA*, 2008.

[23] Patrick R Laughlin. Collective induction: Twelve postulates. *Organizational Behavior and Human Decision Processes*, 80(1), 1999.

[24] Patrick R Laughlin, Erin C Hatch, Jonathan S Silver, and Lee Boh. Groups perform better than the best individuals on letters-to-numbers problems: effects of group size. *Journal of Personality and social Psychology*, 90(4), 2006.

[25] Nancy G Leveson. The use of safety cases in certification and regulation. 2011.

[26] Robert Lewis. Safety case development as an information modelling problem. In *Safety-Critical Systems: Problems, Process and Practice*. Springer, 2009.

[27] John A McDermid. Support for safety cases and safety arguments using sam. *Reliability Engineering & System Safety*, 43(2), 1994.

[28] Dominique Méry, Bernhard Schätz, and Alan Wassyng. The pacemaker challenge: Developing certifiable medical devices (dagstuhl seminar 14062). In *Dagstuhl Reports*, volume 4:2, 2014.

[29] Mary A Meyer and Jane M Booker. *Eliciting and analyzing expert judgment: a practical guide*. SIAM, 2001.

[30] David Moshman and Molly Geil. Collaborative reasoning: Evidence for collective rationality. *Thinking & Reasoning*, 4(3), 1998.

[31] OMG. Structured assurance case meta-model (SACM). Technical report, Object Management Group, 2013.

[32] R. K. Panesar-Walawege, M. Sabetzadeh, and L. Briand. A model-driven engineering approach to support the verification of compliance to safety standards. In *Proceedings of International Symposium on Software Reliability Engineering*, 2011.

[33] Thomas Polacsek. Validation, Accreditation or Certification: a New Kind of Diagram to Provide Confidence. In *Proceedings of International Conference on Research Challenges in Information Science, RCIS*, 2016.

[34] David J Rinehart, John C Knight, and Jonathan Rowanhill. Current practices in constructing and evaluating assurance cases with applications to aviation. Technical report, NASA, 2015.

[35] Alberto Siena, Ivan Jureta, Silvia Ingolfo, Angelo Susi, Anna Perini, and John Mylopoulos. Capturing variability of law with NóMos 2. In *Proceedings of International Conference on Conceptual Modeling - ER'12*, 2012.

[36] Alberto Siena, John Mylopoulos, Anna Perini, and Angelo Susi. Designing law-compliant software requirements. In *Conceptual Modeling - ER 2009*, 2009.

[37] Stephen E. Toulmin. *The Uses of Argument*. Cambridge University Press, Cambridge, UK, 2003. Updated Edition, first edition 1958.

[38] Alain Trognon, Martine Batt, and Jennifer Laux. Why is dialogical solving of a logical problem more effective than individual solving?: A formal and experimental study of an abstract version of WasonâĂŹs task. *Language and Dialogue*, 1(1), 2011.

[39] Alan Wassyng, Paul Joannou, Mark Lawford, Maibaum Thomas, and Neeraj Kumar Singh. New standards for trustworthy cyber-physical systems. In *Trustworthy Cyber-Physical Systems Engineering*, chapter 13, pages 337–368. Addison-Wesley Longman Publishing, 2016.

[40] 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? *IEEE Design & Test*, 32(5), 2015.

[41] Charles B Weinstock, John B Goodenough, and John J Hudak. Dependability cases. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 2004.

[42] Yijun Yu, Virginia NL Franqueira, Thein Than Tun, Roel J Wieringa, and Bashar Nuseibeh. Automated analysis of security requirements through risk-based argumentation. *Journal of systems and software*, 106, 2015.

[43] Gregory Zoughbi, Lionel Briand, and Yvan Labiche. A UML profile for developing airworthiness-compliant (RTCA DO-178B), safety-critical software. In *Model Driven Engineering Languages and Systems*, 2007.

[44] Gregory Zoughbi, Lionel Briand, and Yvan Labiche. Modeling safety and airworthiness (RTCA DO-178B) information: conceptual model and UML profile. *Software & Systems Modeling*, 10(3), 2011.