Uncertainty and Confidence in Safety Logic

Patrick J. Graydon, Ph.D.; Mälardalen University; Västerås, Sweden

Keywords: system safety, safety argument, standards, uncertainty, confidence

Abstract

Reasoning about system safety requires reasoning about confidence in safety claims. For example, DO-178B requires developers to determine the correctness of the worst-case execution time of the software. It is not possible to do this beyond any doubt. Therefore, developers and assessors must consider the limitations of execution time evidence and their effect on the confidence that can be placed in execution time figures, timing analysis results, and claims to have met timing-related software safety requirements. In this paper, we survey and assess five existing concepts that might serve as means of describing and reasoning about confidence: safety integrity levels, probability distributions of failure rates, Bayesian Belief Networks, argument integrity levels, and Baconian probability. We define use cases for confidence in safety cases, prescriptive standards, certification of component-based systems, and the reuse of safety elements both in and out of context. From these use cases, we derive requirements for a confidence framework. We assess existing techniques by discussing what is known about how well each confidence metric meets these requirements. Our results show that no existing confidence metric is ideally suited for all uses. We conclude by discussing implications for future standards and for reuse of safety elements.

Introduction

Reasoning about system safety requires describing and reasoning about confidence in safety claims. This is true regardless of whether the decision to release a system to service is based on the construction and evaluation of a safety case or conformance with a so-called "prescriptive" standard. For example, the DO-178B standard for airborne software requires developers to "determine the correctness and consistency of the Source Code, including … worst-case execution timing" (ref. 1). It is not possible to do this beyond any doubt (ref. 2). Developers and assessors must reason both about the confidence justified by various forms of timing evidence and about whether this confidence is sufficient. Ultimately, uncertainty in safety evidence creates uncertainty in the main safety claim that inevitably (although sometimes invisibly) underpins any decision to release a system to service, namely that the system is adequately safe to operate in a given context. Depending upon the certification regime used, developers, analysts, assessors, managers, and others will need to reason about uncertainty in various safety claims.

There are several existing concepts that might serve as means of describing and/or reasoning about confidence in safety claims. The concept of Safety Integrity Levels (SILs), familiar from many safety standards (refs. 1, 3–6), might be used to describe or reason about the confidence in claims about systems and their components that evidence must provide or does provide. Researchers have also proposed other means of describing and reasoning about confidence. Some recommend modeling uncertainty as probability distributions of dangerous failure rates (ref. 7). Others propose reasoning about confidence using Bayesian Belief Networks (BBNs) (refs. 8–12). Yet others propose modeling confidence as Safety Assurance Levels (refs. 13–14) or using argument defeaters or Baconian probability (refs. 15–19).

This paper makes the following contributions: (1) a survey of existing potential means of describing and reasoning about uncertainty in safety claims; (2) a set of use cases for confidence; (3) requirements for means of describing and reasoning about uncertainty based on these use cases; (4) an assessment of the suitability of existing means; and (5) suggestions for future research.

Potential Means of Describing and Reasoning About Confidence in Safety Claims

Some standards have defined Safety Integrity Levels. These relate to confidence in safety claims in several ways. Researchers have also proposed four other mechanisms for describing and reasoning about confidence: probability distributions of failure rates, Bayesian Belief Networks, Safety Assurance Levels, and Baconian probabilities.

Safety Integrity Levels: Many standards for software in safety-critical systems define and use Safety Integrity Levels (SILs), although the names and definitions of these vary. For example:

- The avionics standard RTCA DO-178C defines Software Development Assurance Levels SWDAL E (failures have no safety effect) through SWDAL A (failures would cause multiple fatalities) (ref. 4)
- The IEC 61508 standard for electrical, electronic, and programmable safety-related systems defines four SILs "corresponding to a range of safety integrity values, where safety integrity level 4 has the highest level of safety" and is associated with a rate of 10^{-8} to 10^{-9} dangerous failures per operating hour (ref 5).
- The ISO 26262 standard for road vehicles defines an Automotive Safety Integrity Level as "one of four levels to specify the item's or element's necessary requirements ... and safety measures to apply for avoiding an unreasonable residual risk, with D representing the most stringent and A the least stringent level" (ref. 6).

Safety standards typically use SILs in the development planning process (ref. 3). Developers (1) perform a system risk assessment, (2) define safety requirements so as to reduce risk, (3) associate SILs with either requirements or components, and then (4) develop, verify, and validate software with a rigor defined by these SILs.

For example, developers following DO-178C use a "system safety assessment process" (outside the scope of the standard) to define the SWDAL for each software component (ref. 4). Developers then plan a software development process that meets the standards' SWDAL-specific objectives and agree this plan with assessors.

Developers following IEC 61508 perform a system hazard and risk analysis as per the standard, define "safety functions" to reduce risk, and define the SIL for each function (ref. 5). During software development, SILs determine which activities the standard recommended against, is neutral about, recommends, or highly recommends. Developers must either perform highly-recommended activities or justify to safety assessors their refusal to do so.

Developers following ISO 26262 identify "items" in a vehicle and subject these to hazard analysis and risk assessment as if they had "no internal safety mechanisms" (ref. 6). Based on the identified hazards, the severity and likelihood of potential harm, and the possible effect on vehicle controllability, developers define safety goals and set item ASILs. During software development, ASIL level determines which activities are neutral, recommended, or highly recommended in much the same manner as specified in IEC 61508 (ref. 5).

As a result of their complex role in software development, SILs are related to confidence in at least three ways:

- As a specification of the needed confidence in high-level safety claims: SILs are determined according to the harm that might result if systems do not behave as specified, and so represent the required degree of confidence that the system will operate adequately safely or according to its requirements
- As a specification of the needed confidence that components will function as specified: SILs are often flowed down to components during system design and safety requirements allocation, and so represent the required degree of confidence that components will operate as specified
- As an indication of the confidence inspired by safety evidence: SILs determine standards' objectives (and sometimes recommended means of satisfying those objectives), thus *influencing* (and so roughly indicating) the degree to which safety evidence produced during development justifies a given degree of confidence that the system and its components will operate according to their safety requirements

A Probability Distribution of Failure Rates: Bloomfield et al. propose modeling confidence in reliability as a probability distribution (ref. 7). In this scheme, the system or component is characterized by a rate at which it might fail in specific (dangerous) ways in expected use. Each point in the distribution represents a degree of belief in a possible dangerous failure rate. As testing, review, and analysis build confidence, the distribution narrows.

To simplify characterizing and reasoning about confidence, Bloomfield et al. suggest modeling distributions using a log normal model (ref. 7). This distribution is characterized by parameters μ and σ, has a mean of $\exp(\mu + 0.5\sigma^2)$, and has a mode (most believable failure rate) of $\exp(\mu - \sigma^2)$. The belief in any given failure rate λ is:

$$p(\lambda) = \frac{1}{\lambda\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(\ln\lambda - \mu\right)^2}{2\sigma^2}\right) \quad (1)$$

It might not be possible to model *every* contribution to uncertainty using a log normal distribution. However, Bloomfield et al. assert that the uncertainty operations they studied are relatively insensitive to which non-symmetric distribution is used. They report having repeated a portion of their work using a gamma distribution.

In some cases, it might be possible to measure a contribution to uncertainty (e.g., using life testing). In other cases, distributions must come from expert opinion. Bloomfield et al. describe having obtained distributions (to validate their approach) by asking a panel of experts for opinions on the likelihood of a specimen system having a probability of failure in each of six bands. The twelve experts conducted a Delphi phase and revised their estimates in light of the other's insights. "A minority of (3) doubters … expressed [their] doubts by giving the system a very high failure rate"; the remainder produced distributions whose means appear to spread over two or three orders of magnitude.

Bayesian Belief Networks: Several researchers propose using Bayesian Belief Networks to reason about uncertainty in safety claims (refs. 8–12). Developers, they say, should (1) create a Bayesian network with nodes representing safety evidence and claims, (2) populate the network with probabilities reflecting confidence in the evidence and the relationship between claims and evidence, and then (3) use software tools to compute the uncertainty in the highest-level safety claims. Like proponents of the probability distribution approach to uncertainty, proponents of the Bayesian Belief Network approach stress the importance of using quantified sources of uncertainty where available, but admit that these are generally not available in the case of high-consequence software (refs. 9, 11–12). Where quantified uncertainty is not available, developers are advised to use expert judgment.

Bayesian Belief Network approaches to confidence differ in terms of both the structure of the networks and the way confidence is encoded as probabilities in the network. In the SERENE method, the networks reflect a number of "idioms" (ref. 8). For example, in the "testing quality" idiom, "tester experience", "testing effort", and "test coverage" affect testing quality. In the "software failures" idiom, "problem difficulty" and "supplier quality" affect failures. While the method and its tools support various kinds of probability distributions, the examples given use a simple three-discrete-state model in which analysts rate the likelihoods of each node being "high", "medium", and "low". The authors provide no guidance on how to derive these estimates from commonly available information.

Littlewood and Wright (ref. 9) discuss their approach in terms of "safety arguments"—explicitly citing the UK Defence Standard 00-56 (ref. 20)—which suggests that their networks are meant to mirror the logical structure of a safety argument (ref. 21). The example they present represents only a small portion of a safety argument in which support for a claim about a system's probability of failure derives from both testing and verification. They model the probability of failure as a node with a real-numbered probability of failure on demand. They model the specification's correctness, verification conclusion correctness, test oracle correctness, and whether the test results show failures as nodes with two possibilities (e.g., correct and incorrect in the case of the specification), each with a real-numbered likelihood. Because the point of the paper is about the effects of "diverse evidence", rather than the validity of the BBN technique, they use artificial likelihood values to illustrate effects.

Wu and Kelly pitch their method as a means of assessing proposed system architectures (ref. 10). At the highest level, their network's organization reflects the flow of information through the architecture. Nodes at this level model the likelihood that components will exhibit omission, commission, timing, or value errors. These nodes are supported by sub-structures representing influences. For example, a real-valued node representing the omission failure of a component receives support from nodes representing the "complexity of [the] function" and the "competency of [the] development team". To populate the node probability tables, the authors suggest using "machine learning over the past project data". They do not demonstrate the feasibility or validity of that data source.

Denney et al. suggest a network structure mirroring a safety argument (ref. 11). Each node represents confidence in a claim from the argument structure and may take one of five values: very low, low, medium, high, and very high. A real number for each value models the degree of belief that the confidence in that claim should have that value. For items of evidence, the authors appear to rely on expert judgment, encoding "a 95% confidence level in [a] calibration experiment" conducted on their example system's air data probe. To specify conditional probabilities for intermediate claims, the authors appear to rely on judgment, giving the confidence in the correctness of a computation as the mean of confidence in its specification and a related proof.

Zhao et al. also suggest a network structure whose high level mirrors a safety argument (ref. 12). They propose first refactoring the argument into reasoning steps that fit Toulmin's model of informal argument (ref. 22). For each reasoning step, analysts identify data (premises), a conclusion (possibly with qualifiers), a warrant (that allows drawing the conclusion based on the data), backing for the warrant, and rebuttals to the conclusion (if any). Each safety claim becomes a node with two states: "present" and "not present" (by which the authors presumably mean true and false). Each state has a real-valued likelihood of correctness. For each safety claim, analysts add additional nodes representing "justified premises", "adequate premises", "whether unprovided premises is [sic] obtainable", "coverage of equivalence partition of premises", and eight other factors derived from Toulmin's model. Analysts are directed to fill the probability tables using "expert judgment".

Given the breadth of the proposals, any discussion of the use of Bayesian Belief Networks to represent and/or reason about uncertainty in software safety claims needs to be qualified with a specification of how the BBNs are structured and populated. However, all of these approaches rely in large part on expert judgment to fill node probability tables.

Safety Assurance Levels: In addressing the problem of confidence in software safety arguments, Weaver deliberately eschewed the used of Bayesian Belief Networks (refs. 13–14). He objected to the "heavy demand that the relationship between all premises and conclusions can be expressed as a conditional probability", the inherent subjectivity of judgments of inference strength, the tendency of BBN probability tables to obscure argument structure, and the necessity of augmenting the BBN to justify the adequacy of its data (ref. 14). Weaver proposed instead using a set of Safety Assurance Levels (SALs) to represent "the level of confidence that a safety argument element (goal or solution) meets its objective". Noting that "there is no particular number of levels that should be used", he gives examples of a system of four levels ranging from SAL 1 (the weakest) to SAL 4 (the strongest).

In Weaver's scheme, developers building a system begin by sketching out the top level of a safety argument using the Goal Structuring Notation (GSN) (ref. 23). Developers assign a target SAL to the top-level goal of the safety argument depending upon "the severity associated with the failure of the top-level objective" (ref. 13) (i.e., the consequences of the system not meeting our operational definition of being adequately safe for use in its intended operating context). As developers refine and extend the safety argument, they decompose the SAL target into targets for each supporting sub-goal or intended item of evidence (i.e., GSN solution). Developers then reverse this process as they complete each item of evidence: they determine in a bottom-up fashion whether each item of evidence and each reasoning step supplies the requisite confidence in the safety claims (i.e., GSN goals) it is meant to support.

When decomposing confidence targets and when assessing complete arguments, developers must reason about the confidence inspired by evidence and by reasoning steps. Weaver notes that factors influencing the confidence derived from evidence include the number of faults known to be in the evidence, the competence of the people who prepared it, the fitness-for-use of any tools. However, he leaves a mechanism for deriving a confidence level from consideration of these factors to future work. Other researchers have since elaborated on these concepts but stopped short of defining a validated, repeatable, deterministic procedure for deriving a SAL for a given piece of evidence and a given claim (ref. 15).

To determine the confidence inspired by a reasoning step in an argument, Weaver advises analysts to first consider the relevance of each premise (i.e., GSN sub-goal or solution) to the conclusion (i.e., GSN parent goal). If necessary, developers should refactor the argument so that either each of the premises separately supports the conclusion or the premises collectively and interdependently support the conclusion. Weaver provides tables for each of those two patterns showing which combinations of premise relevance and premise SAL support which conclusion SALs. For example, a pair of conceptually diverse SAL 3 premises support a SAL 4 conclusion. Weaver offers no evidence showing that the tables are sound, saying instead that they are "only a reflection of a possible SAL decomposition approach" and should be used only "in combination with further justification" (ref. 13).

Defeaters and Baconian Probabilities: Some researchers suggest assessing confidence in a given safety claim by searching for factors that would lower it and then reasoning about whether they do (refs. 16–18). Because safety arguments are generally inductive, claims might be false despite supporting evidence. For example, unless it is known for certain that a test plan is exhaustive, a test oracle correct, the testing mechanism trustworthy, etc., testing shows only that it is *likely* that the tested artifact has the desired properties. An unperformed test might refute the conclusion drawn from testing. Knowledge about how a conclusion might be false despite the given support can be used to express confidence in the form of a Baconian probability (ref. 19).

In the language of informal argumentation, the factors that would lower confidence in a conclusion are called "defeaters" (refs. 17–18, 24). Some defeaters directly rebut safety claims. For example, a documented observation of software violating a requirement rebuts a claim that the software satisfies that requirement. Other defeaters are said to "undercut" the reasoning supporting a safety claim by showing how the conclusion might be false even if the premises are true. For example, the observation that a development organization's configuration management practices are slipshod casts doubt upon the argument that testing shows that the software meets its requirements: the developers might have used the wrong test artifact or oracle.

The Baconian probability of a safety claim is the degree to which "all identified reasons for doubt [have been] eliminated" (ref. 17). This is not the Pascalian probability used by the probability distribution and Bayesian Belief Network approaches: if significant defeaters are unknown, the Baconian probability of a claim might be high while the (true) Pascalian probability is low. (In such cases, expert judgments of the Pascalian probability might also be high.) Baconian probability is sometimes expressed numerically (e.g., "3 of 4 defeaters eliminated").

To assess confidence in a safety claim, Hawkins et al. advise developers to (1) identify the reasoning supporting the claim, (2) identify "assurance deficits" (i.e., possible means of rebutting the claim or undercutting the reasoning), (3) fix unacceptable assurance deficits, and (4) reason about why each remaining assurance deficit is acceptable. Developers describe the reasoning behind the claim, the rigor of the search for assurance deficits, and the acceptability of the deficits in a tripartite "confidence argument" structure parallel to the original safety argument.

Goodenough et al. advocate a similar procedure but a slightly different argument structure to document it (ref. 17). In place of Hawkins et al.'s tripartite structure, they advocate associating with each reasoning step in the safety argument a single claim: "all defeaters for [this reasoning step] have been eliminated".

Uses Cases for a Confidence Metric

Confidence descriptions serve many purposes. In this section, we define five use cases of interest: (1) assessing the confidence justified by a safety case, (2) assessing a prescriptive safety standard; (3) assessing conformance to a standard, (4) using a software component in an expected context, and (5) using a software component out of context. For each use case, we define requirements for an ideal means of describing confidence.

Assessing The Confidence Justified by a Safety Case: In this use case, a person decides whether a safety case justifies sufficient confidence that the system is acceptably safe to operate in its intended operating context to warrant releasing that system into service. In some domains, developers are required to construct a safety case containing safety evidence and an argument linking this evidence to the main safety claim or to safety requirements (refs. 6, 20). An independent analyst then critically examines the safety case (refs. 20, 24–25). Evidence is imperfect and safety arguments use inductive reasoning; this analyst must work from the bottom of the argument up, estimating the confidence in each claim in turn. The analyst—or someone informed by the analyst's findings—then makes a decision based on estimated confidence and the potential consequences of being wrong. To be fit for this use, a means of describing and reasoning about confidence must satisfy three requirements:

R1. It must be possible to accurately assess the confidence in a safety claim justified by the safety evidence typically used to support that claim (e.g., the confidence in software module functionality justified by requirements-based unit testing that achieves Modified Condition/Decision Coverage [ref. 26])

R2. It must be possible to use knowledge of how development artifacts have been constructed and used to reason about uncertainty in safety claims based upon uncertainty in related claims (e.g., reason from uncertainty in detailed claims about specific development artifacts to uncertainty in the main safety claim)

R3. It must be possible to determine whether the confidence in a safety claim is sufficient to justify releasing a system into service given the potential consequences of being wrong

Assessing a Prescriptive Safety Standard: In this use case, a person must decide whether conformance with a standard as written is sufficient grounds for releasing a conforming system to service. The technique most often used to assess software safety standards is ad hoc review by standardization committee members. However, researchers have proposed a more rigorous argument-based review technique (ref. 27). Using that technique, an analyst extracts from the standard an argument linking the evidence demanded by the standard's requirements to the claim represented by the decision the standard is used to support (e.g., adequate safety). Determining whether a standard demands strong enough evidence requires making judgments about the strength of categories of evidence, but is otherwise similar to assessing the confidence justified by a safety case and has similar requirements:

R4. It must be possible to accurately assess the confidence in a safety claim justified by the evidence provided by conformance to a safety standard's requirement/objective

Assessing Conformance to a Standard: In this use case, a person must decide whether conformance documentation establishes with sufficient confidence that each of a standard's requirements has been satisfied. So-called "prescriptive" standards do not generally prescribe a fixed process that all developments must follow. Instead, they set a series of goals or objectives and leave it to developers to define and follow a plan for meeting these. (See the section on SILs for examples.) In this use case, an assessor must determine whether a developer's efforts provide sufficient confidence that the standard's objectives have been met.

For example, objective 6.2.4f of DO-178B requires developers to perform "reviews and analyses of the source code ... to determine the correctness and consistency of the Source Code, including ... worst-case execution timing" (ref. 1). There are several approaches to measuring or analyzing worst-case execution time, and subsequent clarifications of the standard explicitly allow testing as a means of satisfying this objective (ref. 28). Some approaches justify greater confidence than other approaches in a claim that the worst-case execution time of a task is no greater than a specified limit (ref. 2). An assessor determining whether a given applicant has conformed with DO-178B must determine whether there is sufficient confidence that objective 6.2.4f has been satisfied.

This use case gives rise to two additional requirements:

R5. It must be possible to describe the confidence with which a standard's requirement/objective must be met

R6. It must be possible to accurately determine whether a realistic means of complying with a standard's requirement or objective provides the specified confidence

Using a Software Component in an Expected Context: Software developers sometimes build components for use in a limited number of discrete systems (e.g., a product line). In this use case, developers must determine whether evidence accompanying a component built for use in a set of well-defined systems provides sufficient confidence that the component has the safety properties needed for use in a specific system. For example, parts of a turbofan engine controller might be built for use across a number of different aircraft to which the engine might be fitted. The system description such software is built against is well-defined save for specially-defined variants. The traditional, conservative approach to certifying such software for use in a set of systems is to recertify it for use in each new system. Doing so forces certifiers to consider the safety implications of using the software in each system.

However, cost and time-to-market concerns make it desirable to certify a component in isolation as having a specified set of safety properties so that these can be assumed when certifying systems built using those components. Incremental and/or component-based certification is the subject of active research (refs. 28–31). However, it is already clear that such certification mechanisms will require means of describing and reasoning about confidence in safety claims. In this use case, a developer building a component specifies a safety contract. As in Component-Based Software Engineering generally, this contract specifies the safety properties as pairs of assumptions and guarantees (ref. 32). Each assume–guarantee pair should be associated with a confidence level. Certification of the component should establish that the developer's safety evidence is sufficient to provide the specified confidence. Certification of the complete system should show system safety based on those guarantees.

This use case shows a different need for existing requirements R1, R2, R3, and R5 (with the assume–guarantee pair standing in for a standard's requirement).

Using a Software Component Out of Context: This use case is identical to the previous use case save for one crucial difference: the component in question is used in a context for which it was not originally intended. A component might be created for use in one context, and then re-used in others. Alternatively, a component such as an embedded operating system might be created for use in nearly unbounded circumstances. Since safety requirements derive from systems and applications, such components have only *assumed* requirements: system developers determine safety requirements in their specific contexts and consider a component fit for use if the component supplies the needed safety functionality without introducing any new and unacceptable dangerous failure possibilities (ref. 6).

One of the main challenges of assuring the safety of systems built from components used out of context is to understand the implications of the change in context on what is known about the component's safety properties. Many traditional software safety lifecycle activities are defined for use in context; performing these for a component used out of context requires the use of an assumed system context that can be described only broadly. For instance, reviews and analyses intended to show software compatibility with a target computer underpin guarantees of the software behavior but require some description of the target computer hardware. Existing guidance for the use of safety-related software out of context call for developers to record these assumptions and then reconcile them with each actual system the software is used in (ref. 6). The reconciliation step is crucial because software engineers asked to do so do not always consistently enumerate all—or even most—of the assumptions that actually underpin their software (ref. 33). The goal of the active research on component-based safety-related software development and certification (e.g., ref. 28) is to partially-certify components for use out of context, and then to re-use this effort (re-examining as little as practicable) when certifying systems built from those components. To do this would require developers to examine the confidence inspired by evidence, reason about combinations of evidence, and reason about the implications of confidence on the decision to release a system to service, much as the other use cases. However, need to do this based on necessarily broad descriptions of system context gives rise to four additional requirements for a means of describing and reasoning about confidence:

R7. It must be possible to accurately assess the confidence in a safety claim justified by the safety evidence typically used to support that claim in cases where that evidence was produced using only broad assumptions about possible system contexts

R8. It must be possible to determine which additional assumptions about context would markedly improve the confidence that would be inspired by safety evidence produced out of context

R9. It must be possible to reason about which differences in context greatly affect confidence in safety claims

R10. It must be possible to reason about the (small) degree to which differences in context that are judged to be minor affect confidence in safety claims

Discussion of The Available Approaches

We have been describing, estimating, and reasoning about confidence for many years in the course of building and certifying systems. Nevertheless, there is little evidence that any existing confidence framework is fit for purpose.

Safety Integrity Levels: There can be no doubt that it is feasible to use SILs to reason about prescriptive standards and conformance to them: we have been doing so for several decades. Nevertheless, there is little evidence to show the validity of some of the ways in which they are presently being used. Moreover, there are known difficulties that make SILs less useful as means of describing and reasoning about uncertainty in the component-based use cases.

Developers following some standards use SILs as a way of reasoning about how uncertainty about component behavior affects system safety given the system architecture. For example, IEC 61508 allows developers to combine two independent, SIL-x implementations of a function so as to achieve a SIL-x+1 implementation (ref. 5). This concept is intuitive. However, we are not aware of evidence showing that it is valid wherever the standard is applied. This casts doubt upon whether SILs satisfy R2.

While safety assessors routinely determine whether applicants have satisfied standards' objectives, anecdotes about shopping for lenient assessors casts doubt about whether SILs satisfy R6. Combining objectives with recommended means of compliance might help address this problem, but present standards using this approach might still be improved. For example, consider paragraph 7.4.18a of Part 6 of ISO 26262, which requires developers to verify the architectural design to demonstrate "compliance with the software safety requirements" (ref. 6). The standard recommends or highly recommends "inspection of the design", "simulation of the dynamic parts of the design", "prototype generation", "formal verification", "control flow analysis", and "data flow analysis". Given the short, vague, and confusing recommendations—control flow analysis and data flow analysis are both formal analyses, yet they are highly recommended while "formal verification" is only recommended—developers and assessors could be forgiven for disagreeing over whether an approach to verifying a given architecture justified sufficient confidence. More detail in the recommendations might better convey the needed confidence.

Using a SIL to describe the confidence in a component's contract requires reasoning about how well confidence evidence that was deemed appropriate for that SIL in one context justifies confidence in actual safety requirements in a different context. SILs in a given standard are part of a complex relationship between development practice requirements and what is accepted as "adequately safe" in the domain in which the standard is applied. Because of this, it is not clear whether deeming a particular evidence package good enough in one context tells us that it is good enough in another. This is particularly complicated when a component is reused in a domain in which a different standard is typically used. Not only do standards define SILs differently, they do so in terms of a basic vocabulary that disagrees on the meanings of fundamental terms like 'safety' and 'fault' (ref. 34). It is not clear that SILs support requirements R7–R10.

Probability Distributions of Failure Rates and Bayesian Belief Networks: The concept of modeling uncertainty as a probability distribution and reasoning about it using existing theories of probability is appealing because it seems to offer both precision and a sound theoretical footing. However, there is little evidence that the uncertainty values produced by any of the proposed approaches are usefully accurate. Proponents of these approaches readily admit that, in the case of high consequence software, much of the input data used in calculations is simply a codification of expert opinion. Given the use of opinion as data and the possibility that several functionally different networks might plausibly model the same system, calculations might produce "superficially plausible nonsense" (ref. 35). Because opinion underlies both the probabilities representing evidence and the joint probability distributions governing how confidence in one claim relates to confidence in others, it is not clear whether the probabilistic approach supports requirements R1 and R2.

Anecdotes—we know of no proper studies—of variability in safety expert opinion cast doubt on probabilistic approaches. But if we must rely on opinion for lack of anything better, one might ask whether BBN modeling adds value by reducing the degree to which opinions reflect misunderstanding of how confidence propagates. We are aware of no data showing that it does, or that this value is worth the cost of building the network and populating its probability tables. It should be feasible to assess this experimentally. Randomly divide a volunteer sample of safety experts into two groups. Present each with certification evidence and ask each to estimate confidence in the main safety claim (using their own opinions), one using a BBN and the other a simpler system such as "traffic-lighting" argument nodes with Safety Assurance Levels. Measure inter-assessor agreement. Estimates that agree need not be right, but some estimates that disagree must be wrong. Does the use of a BBN produce greater agreement? It would be interesting to also ask participants to rate their confidence in their confidence estimate on a 1–10 scale, to see if the process inspires unjustified trust in the resulting confidence figure.

Safety Assurance Levels: The concept of using a small set of discrete Safety Assurance Levels to describe confidence in safety claims was developed in reaction to the perceived shortcomings of the probabilistic approach. However, there is little evidence to show that Safety Assurance Levels and related methods address our requirements. Again, variability in expert opinion casts doubt on whether this mechanism satisfies R1. The admitted lack of evidence for the validity of adding one level of confidence by replicating a subsystem casts doubt on whether this mechanism satisfies R2. Moreover, without a well-defined absolute scale for confidence—Weaver's four-level scale was an example, not a universal definition—it is not clear whether this concept would satisfy the requirements surrounding the component-based use cases (R7–R10).

Defeaters and Baconian Probabilities: The notion of improving arguments by identifying defeaters and acting to eliminate them is well accepted by philosophers studying informal argumentation. Anecdotal evidence from the

safety community suggests that these techniques can help to find problems in safety arguments and evidence that can then be fixed, increasing confidence in system safety. However, it is not clear that Baconian probabilities are suited to describing confidence as needed to satisfy R5. That 3 of 4 identified defeaters have been eliminated is only a meaningful measure of confidence if we know that there are no more defeaters of interest, which can only be judged if we know what the four defeaters are (or how thoroughly appropriate people have searched for them). The argument structures that researchers propose contain this information (refs. 16–18), but are far too bulky to serve as a summary of confidence in a claim. Baconian probability might be a poor solution to requirements R7–R10.

Conclusions

Confidence is a crucial part of reasoning about system safety. However, it is not clear that any existing method for representing or reasoning about confidence is suitable for all of the use cases that we might have for it. This is particularly true for the use cases surrounding incremental and component-based certification. Such certification demands more of a confidence method than existing argument-based and standards-based certification processes because it demands a universally understood, absolute scale for confidence.

Statistical approaches to confidence and defeater-based approaches to confidence have the advantage of underlying theories that are well developed in the relevant literature. However, it is not clear that we know how to quantify all of the sources of uncertainty that affect software in safety-related systems. It is also not clear that Baconian probability figures encapsulate everything that we might want to know about confidence in a given safety claim. Further research is needed to establish the validity and relative utility of the various means of describing and reasoning about confidence in safety claims. Once a clear winner emerges, further research will be needed to define and validate means of precisely assessing the confidence inspired by each of the forms of safety evidence used in modern software safety practice.

Acknowledgment

The work was funded by the Swedish Foundation for Strategic Research (SSF) as part of the SYNOPSIS project and the EU's Artemis-funded SafeCer project.

References

1. RTCA DO-178B. *Software Considerations in Airborne Systems and Equipment Certification*. Washington, DC, USA: RTCA, Inc., December 1992.
2. Graydon, P. and I. Bate. "Realistic Safety Cases for the Timing of Systems". *Computer Journal*, in press.
3. Redmill, F. "Safety Integrity Levels—Theory and Problems". In *Proc. 8th Safety-critical Systems Symposium (SSS)*. Springer, February 2000.
4. RTCA DO-178C. *Software Considerations in Airborne Systems and Equipment Certification*. Washington, DC, USA: RTCA, Inc., December 2011.
5. IEC 61508. *Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems*. Second edition, International Electrotechnical Commission, 2010.
6. ISO 26262. *Road vehicles — Functional safety*. International Organization for Standardization, 2011.
7. Bloomfield, R., B. Littlewood, and D. Wright. "Confidence: Its Role in Dependability Cases for Risk Assessment". In *Proc. 37th Int'l Conference on Dependable Systems and Networks (DSN)*, 338–346, 2007.
8. The SERENE Partners. *The SERENE Method Manual*. Task Report SERENE/5.3/CSR/3053/R/1, The SafEty and Risk Evaluation using bayesian NEts (SERENE) project, May 1999.
9. Littlewood, B. and D. Wright. "The Use of Multi-Legged Arguments to Increase Confidence in Safety Claims for Software-Based Systems: A Study Based on a BBN Analysis of an Idealised Example". *IEEE Transactions on Software Engineering* 33(5), 347–365, May 2007.
10. Wu, W. and T. Kelly. "Combining Bayesian Belief Networks and the Goal Structuring Notation to Support Architectural Reasoning About Safety". In *Proc. 26th Int'l Conference on Computer Safety, Reliability, and Security (SAFECOMP)*, Nuremberg, Germany, 172–186. Springer, September 2007.
11. Denney, E., G. Pai, and I. Habli. "Towards Measurement of Confidence in Safety Cases". In *Proc. 5th Int'l Symposium on Empirical Software Engineering and Measurement (ESEM)*, Banff, Alberta, Canada, 2011.

12. Zhao, X., D. Zhang, M. Lu, and F. Zeng. "A New Approach to Assessment of Confidence in Assurance Cases". In *Proc. 31st Int'l Conference on Computer Safety, Reliability, and Security (SAFECOMP) Workshops*. Springer, September 2012.
13. Weaver, R., J. Fenn, and T. Kelly. "A pragmatic approach to reasoning about the assurance of safety arguments". In *Proc. 8th Australian Workshop on Safety Critical Systems and Software (SCS)*, Darlinghurst, Australia, 57–67. Australian Computer Society, Inc., 2003.
14. Weaver, R. A. *The Safety of Software—Constructing and Assuring Arguments*. Ph.D. thesis, University of York, York, UK, September 2003.
15. Sun, L. and T. Kelly. "Elaborating the Concept of Evidence in Safety Cases". In *Proc. 21st Safety-Critical Systems Symposium (SSS)*. Bristol, UK, February 2013.
16. Hawkins, R., T. Kelly, J. Knight, and P. Graydon. "A New Approach to Creating Clear Safety Arguments". In *Proc. 19th Safety-Critical Systems Symposium (SSS)*, Southampton, UK, February 2011.
17. Goodenough, J. B., C. B. Winstock, and A. Z. Klein. *Toward a Theory of Assurance Case Confidence*. Technical Report CMU/SEI-2012-TR-002, Software Engineering Institute, Pittsburgh, PA, USA, 2012.
18. Weinstock, C. B., J. B. Goodenough, and A. Z. Klein. "Measuring Assurance Case Confidence Using Baconian Probabilities". In *Proc. 1st Int'l Workshop on Assurance Cases for Software-Intensive Systems (ASSURE)*, San Francisco, CA, USA, May 2013.
19. Cohen, L. J. "Some Historical Remarks on the Baconian Conception of Probability". *Journal of the History of Ideas* 41(2), 219–231, 1980.
20. Def. Stan. 00-56. *Safety Management Requirements for Defence Systems, Issue 4*. UK Ministry of Defence.
21. Kelly, T. P. *Arguing Safety—A Systematic Approach to Managing Safety Cases*. D.Phil. thesis, University of York, York, UK, September 1998.
22. Toulmin, S. E. *The Uses of Argument*. Cambridge, UK: Cambridge University Press, 1958.
23. Attwood, K. et al. *GSN Community Standard, Version 1*. York, UK: Origin Consulting Limited, 2011.
24. Kelly, T. "Reviewing Assurance Arguments—A Step-by-Step Approach". In *Proc. Workshop on Assurance Cases for Security—The Metrics Challenge*. Edinburgh, UK, July 2007.
25. Graydon, P., J. Knight, and M. Green. "Certification and Safety Cases". In *Proc. 28th Int'l Systems Safety Conference (ISSC)*. Minneapolis, MN, USA, 2010.
26. Hayhurst, K., D. Veerhusen, J. Chilenski, and L. Rierson. *A Practical Tutorial on Modified Condition/ Decision Coverage*. Technical memorandum TM-2001-210876. Hampton, VA, USA: NASA, May 2001.
27. Graydon, P. and T. Kelly. "Using Argumentation to Evaluate Software Assurance Standards". *Information and Software Technology*, in press.
28. RTCA DO-248B. *Final Report For Clarification Of DO-178B "Software Considerations In Airborne Systems And Equipment Certification"*. Washington, DC, USA: RTCA, Inc., October 2001.
29. "SafeCer: Safety Certification of Software-Intensive Systems with Reusable Components". Accessed 7 May 2013. http://www.safecer.eu
30. "MRTC Research Projects: SYNOPSIS". Accessed 7 May 2013. http://www.mrtc.mdh.se/index.php?choice=projects&id=0356
31. "Welcome to OPENCOSS". Accessed 7 May 2013. http://www.opencoss-project.eu
32. Carlson, J., C. Ekelin, J.-L. Gilbert, Á. Herranz, and S. Puri. *Generic Component Meta Model*. Deliverable D2.2.4, SafeCer project, October 2012.
33. Spiegel, M., P. Reynolds, Jr., and D. Brogan. "A Case Study of Model Context for Simulation Composability and Reusability". In *Proc. Winter Simulation Conference*, 2005.
34. Manni, V. et al. *Baseline for the Common Certification Language*. Deliverable report D4.1, OPENCOSS project, April 2012.
35. Littlewood, B. "Dependability Assessment of Software-Based Systems: State of the Art". In *Proc. 27th Int'l Conference on Software Engineering (ICSE)*. (Invited talk.) May, 2005.

Biography

Patrick J. Graydon, Ph.D., Postdoctoral Research Fellow, Mälardalen University, Box 883, 731 23 Västerås, Sweden, telephone +46 21 1014 21, e-mail – patrick.graydon@mdh.se.

Dr. Graydon's research interests include safety and security argumentation, dependable software engineering, and certification processes. He is presently investigating component-based software engineering for critical systems.