Slightly updated from Dependable Software Systems Engineering (Marktoberdorf Summer School Lectures, 2016), ed. A. Pretschner, D. Peled and T. Hutzelmann, pp. 207-236. Published by IOS Press, Volume 50 of NATO Science for Peace and Security Series D, October 2017.

Assurance and Assurance Cases

John Rushby
Computer Science Laboratory
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025 USA

Abstract.
Assurance provides confidence that a system will work as required and not cause harm. Confidence is based on justified beliefs about the system and its environment, and justification can be developed and documented as an assurance case comprised of a structured argument grounded on evidence. For justification to be compelling, the argument must be indefeasible, meaning that we have so thoroughly considered everything that can go wrong (i.e., hazards to the system and defeaters to the argument) that there is no new information that could change our assessment.

I show how the obligation for indefeasible justification can guide construction and interpretation of the argument and the evidence in an assurance case and how confidence in the case translates to bounds on the risk posed by the system.

Assurance requires predictability in both the system and its environment; I speculate how credible assurance may be provided for recent and forthcoming systems where both kinds of predictability may be lacking.

Keywords.
Assurance, Assurance Case, Safety Case, Risk, Indefeasibility, Evidence, Argument

Critical Properties and System Assurance

It is natural that customers for a computerized system should expect assurance that it will perform as required. They, and society at large, will also expect assurance that the system will not cause unintended harm. To make this notion of "assurance" precise, we need to be careful about the terms we employ.

The system operates in an environment that we call the *world*; properties of the world that are relevant to the system are documented as *assumptions*. The system is intended to bring about some change in the world, and these intentions constitute its *requirements*. It is important to note that requirements are described entirely in terms of the change or effect that the system is to have on the world; requirements do not describe how this change is to be accomplished, that is the task of the system *specification* [31]. The specification is elaborated through possibly many levels of design to yield an *implementation* that realizes the requirements. The implementation may depend on components, or on interaction

with other systems; these are parts of the world and their properties should be documented among the assumptions for the system under consideration.

We need to be sure that the implemented system will satisfy the customer's intent; we break this overall goal into the four claims shown in Figure 1, which reflect the rôles of the intermediate artifacts: assumptions, requirements, and specification. We really want to know that these four claims are *true*, but truth is an ideal known only to the omniscient, so we need to find a weaker but acceptable alternative.

The closest we can come to *knowing* the truth of these claims is to have *confidence* in their truth. Confidence is a psychological state which, if rational, must be based on the reasons—that is, the justification—for believing the claims. Hence, *assurance* is the task of identifying and providing trustworthy means for *justified belief* in the four claims of Figure 1.

1. The requirements satisfy the customer's intent,
2. The assumptions correctly describe the world,
3. The specification satisfies the requirements, given the assumptions, and
4. The implementation satisfies the specification, given the assumptions.

Figure 1. Four Claims about the System

In addition to the explicit customer requirements, there may be additional expectations that are often left implicit: examples include "don't hurt me," "don't violate my privacy," "don't lose my data," and so on. Systems that have significant potential to cause harm are called *critical* systems, and the kinds of harm to be avoided are then made explicit as *critical requirements*.[1]

Because they are often stated as a negative (i.e., describe what must *not* happen) critical requirements seldom concern the *purpose* of the system and so they do not play quite the same rôle as ordinary requirements during systems development and engineering (e.g., we say "I require a banking system and it should be secure," rather than "I require a secure system and it should do banking") but they *do* play the same rôle in systems assurance: that is, we need justified belief in the four claims given earlier for both the critical and the ordinary requirements.

Due to the asymmetry in development, violations of critical requirements may come about because they conflict with, or are not ensured by, the ordinary requirements.[2] But a more common source of violation is an unanticipated attribute or behavior (often a failure, or mode of failure) in some component of the system, or in some other (sub)system with which it interacts. (The unanticipated attribute need not concern computational behavior: for example, the other component could overheat this one, or stress its power supply.) A deployed component or subsystem should satisfy its requirements but it may also have additional

[1] Some of these properties should surely be made explicit for every system, so to some extent all systems should be engineered as if critical: for example, the world would be a better place if all systems were secure.

[2] It is not mandatory that the ordinary requirements should (be augmented to) ensure the critical requirements—the ordinary and the critical requirements could be achieved by independent means—but it is one approach.

properties, including failures, due to choices made in its design and implementation; in an ideal world, these would be fully documented and then incorporated among the assumptions of the system under consideration. Part of the assurance for the system will be to show that it deals adequately with all these documented cases. But components and interacting systems (to say nothing of humans and physical machinery and other real-world elements of the environment) are often imperfectly documented, especially regarding their failures, so an essential part of critical systems engineering is to think of *every* failure and other previously unforeseen circumstance that could impact the ability of the system to achieve its critical requirements (these are the *hazards*) and to deal with them appropriately. The process of identifying hazards is referred to as *hazard analysis*, and assurance must establish that it has been performed completely and effectively and that all hazards are documented; the later stages of design must ensure that all hazards are adequately handled (e.g., eliminated or mitigated), and assurance must confirm that this is so.

We now know that assurance provides justified belief that a system will satisfy all its requirements, but how much justification do we need, particularly for the critical requirements? To answer this important question we need to explore the relationship between assurance and risk.

1. System Assurance and Risk

A *failure* occurs when a system violates its ordinary or critical requirements. The failure, particularly if it concerns critical requirements, will often cause harm or lead to some loss. Harm can occur in many dimensions (e.g., death and injury, theft of property, invasion of privacy, loss of service, reduced quality of life, or environmental damage) and on many scales (e.g., affecting individuals, communities, nations, or the world) so it is not easy to quantify severity of harm in a general way. But for any given system, we may assume the severity of harm associated with its potential failures can be ranked in some way. What we desire is that the severity and frequency of failures should be inversely related and, in particular, the really severe failures should be very rare indeed. The product of severity of harm and frequency of occurrence is termed *risk*, so our goal is to minimize risk. We then must ask: what is a reasonable threshold on acceptable risk?

Public perception and tolerance of risk are unrelated to statistical threat: for example, in recent years, the typical annual death toll in the United States due to medical errors is over 400,000, road accidents 35,000, firearms 12,000 (including about 400 mass shootings—defined as four or more victims, not necessarily fatal), terrorism 3–70 (highly variable), airplane crashes 0, and nuclear accidents 0 [66]. A significant element in public concern seems to be "dread factor": the degree to which there is mass impact, exposure is involuntary, and the evolution of events is outside individual control [44].

For any particular class of system or kind of harm, regulators, insurance companies, and other stakeholders generally establish some threshold on acceptable risk that pays due attention to both objective threat and public perception. Our

task is to identify means of assurance that are commensurate with these thresholds.

As an illustration of risk thresholds, we can briefly examine the regulations for commercial aircraft (see [50] for more details). The Federal Aviation Regulations (FAR) Part 25.1309 identifies five *failure condition categories* (i.e., severities of harm) from "catastrophic" down through "hazardous/severe-major," "major," and "minor" to "no effect." Catastrophic failure conditions are "those which would prevent continued safe flight and landing," while severe-major failure conditions can produce "a large reduction in safety margins or functional capabilities, higher workload or physical distress such that the crew could not be relied on to perform its tasks accurately or completely." Catastrophic failure conditions must be "extremely improbable" while hazardous/severe-major must be "extremely remote." Furthermore, no single failure may precipitate a catastrophic failure condition.

FAR 25.1309 does not define "extremely improbable" and related terms; these are explicated in FAA Advisory Circular (AC) 25.1309,[3] which states, for example, that "extremely improbable" means "so unlikely that they are not anticipated to occur during the entire operational life of all airplanes of one type," while "extremely remote" means "not anticipated to occur to each airplane during its total life, but which may occur a few times when considering the total operational life of all aeroplanes of the type." AC 25.1309 further states that "when using quantitative analyses... numerical probabilities... on the order of 10^{-9} per flight-hour may be used... as aids to engineering judgment... to... help determine compliance" with the requirement for extremely improbable failure conditions. The corresponding probabilities for hazardous/severe-major ("extremely remote") and severe ("remote") failure conditions are 10^{-7} and 10^{-5} per hour, respectively. Other industries provide similar guidance, often based on the ALARP principle ("As Low As Reasonably Practicable") [46].

Figures such as 10^{-9} per hour concern probability or frequency of failure, whereas assurance provides justified belief in claims that ultimately concern satisfaction of requirements. How are these related?

Violation of any of the four claims given in Figure 1 constitutes a *fault*, so we will say that a system that satisfies all four claims is "fault-free" or "nonfaulty." The only way a system can fail is by encountering a fault, which can happen if our confidence in its fault-freeness is misplaced. But a faulty system is not certain to fail: it depends on the circumstances of its execution whether it will encounter one of its faults, whether that encounter causes a departure from correct system state or behavior (i.e., an *error*), and whether that error ultimately leads to a failure.[4] Thus, we can speak of there being some *probability* that the system will fail, if it is faulty: i.e., $P(\text{system fails} \mid \text{system faulty})$.

We noted earlier that justified belief cannot guarantee truth of the claims for fault-freeness. But we can imagine that different procedures for justifying our belief will provide greater or lesser degrees of *confidence* in the truth of those claims.

[3]The European Aviation Safety Agency (EASA) Certification Specifications CS 25 and Acceptable Means of Compliance AMC 25 [18] are largely harmonized with FAR 25 and its Advisory Circulars.

[4]Fault-tolerant systems generally work by detecting errors and then correcting them before they escape their "fault containment unit," thereby preventing them from causing failure.

We can quantify that confidence as a subjective probability, P(system nonfaulty), denoted p_{nf}.[5] Then, by the formula for total probability

$$\begin{aligned} &P(\text{system fails [on a randomly selected demand]}) \\ &= P(\text{system fails} \,|\, \text{system nonfaulty}) \times P(\text{system nonfaulty}) \\ &\quad + P(\text{system fails} \,|\, \text{system faulty}) \times P(\text{system faulty}). \end{aligned} \tag{1}$$

The first term in this sum is zero, because the system does not fail if it is nonfaulty (which is why the theory needs these definitions[6]). Hence, with p_{nf} as the probability the system is nonfaulty (so that $P(\text{system faulty}) = 1 - p_{nf}$), and $p_{F|f}$ as the probability that it Fails, if faulty, its probability of failure on demand, *pfd* is given by

$$pfd = p_{F|f} \times (1 - p_{nf}). \tag{2}$$

Different industries make different assessments about the parameters to (2). Early nuclear protection, for example, seemed to presume the system *is* faulty, so it set p_{nf} to 0 and performed extensive random testing to substantiate (typically) $p_{F|f} < 10^{-3}$. If those regulators had accepted that modest amounts of assurance could deliver $p_{nf} \geq 0.9$, then by (2) the same probability of failure could be achieved[7] with the much less costly testing required to validate merely $p_{F|f} < 10^{-2}$.

Dually, AC 25.1309 seems to suggest that aircraft certification presumes the system *will* fail if it is faulty, and so sets $p_{F|f} = 1$. The whole burden for assurance then rests on the value assessed for p_{nf}. This suggests we need $p_{nf} \geq 1 - 10^{-9}$ for catastrophic failure conditions, which is completely implausible. In fact (and contrary to [50]), there is no credible assignment of values to the parameters of (2) that delivers $pfd \leq 10^{-9}$ per hour, which seems to be required for aircraft certification; an alternative model is needed.

Rather than the figure of 10^{-9} per hour, which is intended only as "an aid to engineering judgment," let us look at the fundamental requirement that a catastrophic failure condition is "not anticipated to occur during the entire operational life of all airplanes of one type." Extending (2), the probability of surviving n independent demands without failure, denoted $p_{srv}(n)$, is given by

$$p_{srv}(n) = p_{nf} + (1 - p_{nf}) \times (1 - p_{F|f})^n. \tag{3}$$

[5] For a frequentist interpretation think of the actual system as a random selection from the set of all systems for the same requirements, with the probability distribution of the selection being determined by the problem to be solved and the character of the system development and assurance processes employed. Each system can be assigned an indicator variable that takes the value 1 if it is fault free and 0 if it is not; p_{nf} is then the expected value of the indicator variable over the selection distribution [38, 41].

[6] There is a related notion of *quasi* fault-freeness, meaning the system is either fault free or is faulty but has only a minuscule probability of failure [60]; this is a more robust model and yields attractive results [69], but the details are more complicated.

[7] We are cutting a lot of corners here: the full treatment must distinguish *aleatoric* from *epistemic* assessment and must justify that beliefs about the two parameters can be separated; [39–41, 69] give details.

Demands can be interpreted as hours of operation, or flights, or some other measure of exposure and, whichever is chosen, a suitably large n can represent "the entire operational life of all airplanes of one type." The notable feature of (3) is that the first term establishes a lower bound for $p_{srv}(n)$ that is independent of n. Thus, if assurance gives us the confidence to assess, say, $p_{nf} \geq 0.9$ (or whatever threshold is meant by "not anticipated to occur") then it seems we have sufficient confidence to certify the aircraft as safe.

However, we can imagine using this procedure to provide assurance for multiple airplane types; if $p_{nf} = 0.9$ and we assure 10 types, then we can expect that one of them will have faults. In this case, we need confidence that the system will not suffer a critical failure despite the presence of faults, and this means we need to be sure that the second term in (3) will be well above zero even though it decays exponentially. This confidence could come from prior failure-free operation (e.g., flight tests). Calculating the overall $p_{srv}(n)$ can then be posed as a problem in Bayesian inference: we have assessed a value for p_{nf}, have observed some number r of failure-free demands, and want to predict the probability of seeing $n - r$ future failure-free demands. To do this, we need a prior distribution for $p_{F|f}$, which may be difficult to obtain and difficult to justify for certification. However, Strigini and Povyakalo [60] show there is a distribution (specifically, one in which $p_{F|f}$ is concentrated in a probability mass at some $q_n \in (0, 1]$) that delivers *provably worst-case* predictions; hence, we can make predictions that are guaranteed to be conservative, given only p_{nf}, r, and n. For values of p_{nf} above 0.9, their results show that $p_{srv}(n)$ is well above the floor given by p_{nf}, provided $r > \frac{n}{10}$.

If we regard a complete flight as a demand, then "the entire operational life of all airplanes of one type" might require n to be in the range 10^8 to 10^9 (e.g., the Airbus A320 series have already performed over 62 million flights). Flight tests prior to certification might provide only $r = 10^3$, so it appears this is insufficient for certification by the criterion above. However, it can be argued that when an airplane type is certified we do not require (and in fact cannot feasibly obtain) sufficient evidence to predict failure-free operation over the entire lifetime of the type; instead, we initially require sufficient confidence only for, say, the first six months of operation and the small number of aircraft that will be manufactured and deployed in that period. This will be a much smaller value of n, and our p_{nf} (from assurance) and our r (from flight tests) will be sufficient for confidence in its failure-free operation. Then we will need confidence in the next six months of operation, with a larger fleet, (i.e., a larger n) but now have the experience of the prior six months failure-free operation (i.e., a larger r) and in this way we can "bootstrap" our way forward.

It remains to consider what happens if experience in operation does reveal a fault (by manifesting a failure, hopefully not catastrophic—remember there is a requirement that no single fault may cause a catastrophic failure condition). Commercial airplanes operate in a legal and ethical framework where all incidents and accidents are promptly reported and dispassionately investigated. The FAA issues Airworthiness Directives mandating workarounds or corrections to detected faults; in extreme cases it may temporarily ground the fleet (as it did for Boeing 787 battery problems in January 2013). Bishop [11] constructs a statistical model

for this scenario and shows that, under plausible assumptions, detection and repair of faults significantly increases long run safety, even if the fleet continues to operate after a fault has been discovered, and even if repairs may be imperfect.

In its totality, the analysis above (which is based on research at Adelard and City University in London [10–12, 41, 60, 69]) provides—for the first time, I think—a plausible statistical model that retrospectively explains the success of aircraft certification, and other certification regimes based on similar practices. At the base of this analysis is an assessed confidence (e.g., $p_{nf} > 0.9$) that the system is nonfaulty with respect to critical requirements. Assurance delivers this through justified belief in the four claims of Figure 1, and so we now need to consider what manner of justification provides the required confidence.

2. Assurance Cases

It seems reasonable that justified belief in any of the claims of Figure 1 should rest on *evidence* about the artifacts concerned, and an *argument* that this evidence is sufficient to justify the claim. The four claims, and the artifacts they concern, are rather different from each other and we might expect that they will involve different kinds of evidence and argument. The first claim, for example, relates the requirements, which will be recorded in a document, to the customer's intent, which will be a psychological or institutional phenomenon. The evidence will presumably document the process of "elicitation" employed and the argument will justify the adequacy of this process and the means by which the requirements are probed for errors, inconsistencies, and oversights. Likewise, the second claim concerns the relationship between our assumptions and the actual world, and the evidence and argument will document the way the assumptions are developed and examined for utility, adequacy, and accuracy. The third and fourth claims, on the other hand, concern relationships between documents: namely, the requirements, the specification, and the implementation, given the assumptions. If the implementation is software, then all these documents are potentially formal and the justification can employ evidence whose construction can be partially automated using formal reasoning and program verification (or synthesis). However, the argument must then justify why we can trust the results of automated analysis, and how we can be sure that the formal documents accurately represent the informal interpretations we place upon them.

The differing nature of the artifacts involved in the four assurance claims and the correspondingly different character of their associated assurance tasks cause them to be referred to by different names: those associated with the first two claims are said to concern *validation* ("are we building the right system?") while the second two concern *verification* ("are we building the system right?"). But, in all cases, assurance uses argument and evidence to provide justified belief in the associated claim; where they differ is in the kind of evidence deployed. It is important to note that evidence is required not only for each primary claim (e.g., the process of elicitation used for requirements, or formal verification of specifications against requirements), but for all its attendant doubts and subsidiary

concerns, such as trust in the results of automated analysis and in the fidelity of formal specifications.[8]

Prior to the introduction of assurance cases [13,67], the selection of evidence was specified in standards and guidelines and in some industries this continues to the present. Thus DO-178C [47], the guidelines on assurance of software for civil aircraft (and which is entirely focused on items 3 and 4 of Figure 1), describes 71 "objectives" for which evidence may be required. For the most critical software (that which could lead to a catastrophic failure condition), known as Level A, all 71 objectives must be satisfied, 33 of them "with independence" (meaning assurance must be performed by personnel independent of the developers). For the less critical Level B software (that which could lead to a hazardous/severe-major failure condition) it is 69 objectives, 21 with independence; for Level C (major failure conditions) it is 62 and 8, and for Level D (minor failure conditions) it is 26 and 5.

Identification and selection of the objectives in DO-178C were performed in committee meetings, and through evolution from earlier versions of the document. The guidelines seem to be effective: no aircraft accident or serious incident has been traced to a flaw in software development (though some have implicated flawed requirements—more properly called specifications—which are assured by a different set of guidelines known as ARP 4754A [58]), but we do not have a good understanding of why they are effective, nor a rationale for how the objectives were selected.

Aircraft systems are changing (e.g., increasingly autonomous operation), as is their environment (e.g., closer integration with air traffic management in NextGen), and so are methods of software development and assurance (e.g., model based design, and automated verification and synthesis), so DO-178C and similar guidelines may need to be updated or replaced if they are to remain relevant [50]. We therefore need some principles to guide the selection of evidence that can meet these new challenges. Since we do not have a good account of the principles underlying DO-178C and its related guidelines, we need retrospectively to construct suitable principles from the ground up. Assurance cases can provide the intellectual scaffolding for this endeavor.

To provide justified belief in a claim, it seems reasonable that we should provide evidence that it is true. The difficulty is that there can be many reasons why proffered evidence might not be persuasive. Hence, the evidence in support of a claim needs to be organized in such a way that we can be sure that every concern is adequately addressed. In an assurance case, this organization is achieved through what is called a *structured argument*, which is a hierarchical collection of individual *argument steps*, each of which justifies a *local claim* on the basis of evidence and/or lower-level subclaims. A trivial example is shown on the left in Figure 2, where a top claim $\mathtt{C}$ is justified by an argument step $\mathtt{AS}_1$ on the basis of evidence $\mathtt{E}_3$ and subclaim $\mathtt{SC}_1$, which itself is justified by argument step $\mathtt{AS}_2$ on the basis of evidence $\mathtt{E}_1$ and $\mathtt{E}_2$.[9]

[8] It is instructive to review faults found in formally verified systems [22,68]: verification successfully eliminates one class of potential faults, but others remain.

[9] To make this concrete, the claim $\mathtt{C}$ might concern system correctness; $\mathtt{E}_1$ could then be test results, and $\mathtt{E}_2$ a description of how the tests were selected and the adequacy of their coverage, so

Here, **C** indicates a claim, **SC** a subclaim, and **E** evidence; **AS** indicates a generic argument step, **RS** a reasoning step, and **ES** an evidential step.

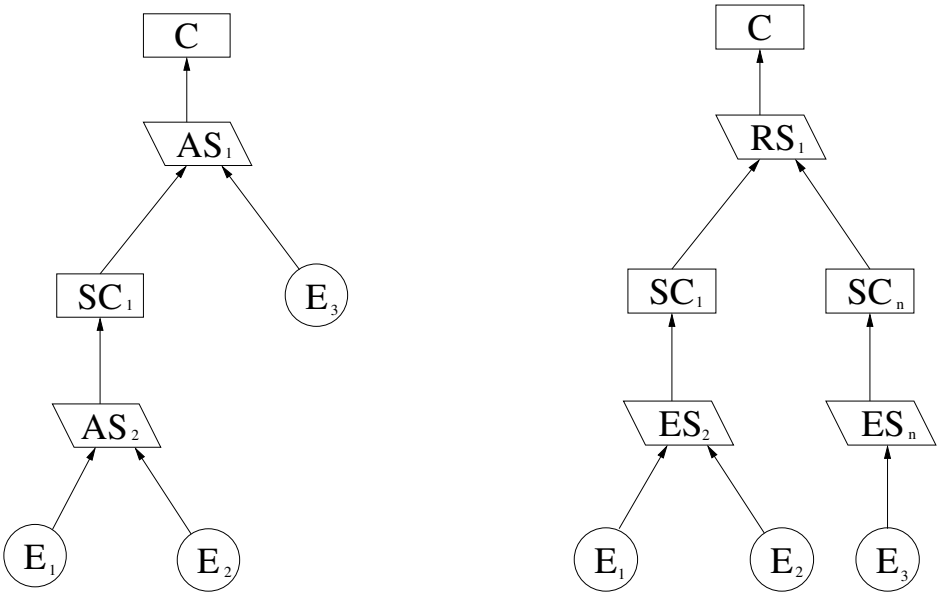

Figure 2. A Structured Argument in Free (left) and Simple Form (right)

Assurance cases often are portrayed graphically, as in the figure, and two such graphical notations are in common use: Claims-Argument-Evidence, or CAE [1], and Goal Structuring Notation, or GSN [34]; however, Figure 2 is generic and does not represent any specific notation. The boxes in the Figure contain, or reference, descriptions of the artifacts concerned: for evidence this may be substantial, including results of tests, formal verifications, etc., and for argument steps it will be a narrative explanation or "warrant" why the cited subclaims and evidence are sufficient to justify the local parent claim.

Our goal now is to determine when an assurance case delivers sufficient confidence to accept its claim—and to do that we need to determine how to evaluate its evidence and structured argument.

Recall that the goal of assurance is to deliver confidence in the truth of its claim, which we accomplish by means of justified belief. These concepts have been studied since antiquity: Plato equated *knowledge* with *justified true belief*, and that equivalence has been examined and debated for approximately 2,400 years as a central focus of *epistemology*, the study of knowledge. Plato's formulation was generally accepted until the beginning of the 20th Century, when doubts began to emerge. Russell posed the "stopped clock case" in 1912:

that $\texttt{SC}_1$ is a claim that the system is adequately tested; $\texttt{E}_3$ might then be version management data to confirm it is the deployed software that was tested. Of course, a real assurance case will concern more than testing and even testing will require dozens of items of supporting evidence, so real assurance cases are huge.

> Alice sees a clock that reads two o'clock, and believes that the time is two o'clock. It is in fact two o'clock. However, unknown to Alice, the clock she is looking at stopped exactly twelve hours ago.

The issue here is that although Alice has justified belief that the time is two o'clock, and that belief is true, it easily could have been false because her justification is weak. If this were an assurance case, Alice would be faulted for not considering the hazard of a faulty clock and seeking additional evidence to confirm or refute that possibility. A publication by Gettier in 1963 [24] gave additional examples in which poorly justified beliefs were "accidentally" true and this stimulated huge activity (over 3,000 citations), much of which attempts to adjust the definition of knowledge by replacing or augmenting "justified true belief."

In assurance, we do not seek to (re)define knowledge, but to find good criteria for belief to be adequately justified, and some epistemologists follow a similar direction. Among many proposed criteria, the one relevant to assurance is that of *indefeasibility* [36]. The idea is that for a belief to be justified indefeasibly we must be so sure that all contingencies have been identified and considered that there is no (or, more realistically, we cannot imagine any) new evidence that would change our belief. Paraphrasing Barker [8], if you have an indefeasibly justified belief, then what you don't know can't hurt you! Our task now is to apply the indefeasibility criterion to the evidence and arguments of an assurance case.

Observe that the argument step $\texttt{AS}_1$ on the left of Figure 2 uses both evidence $\texttt{E}_3$ and a subclaim $\texttt{SC}_1$. Elsewhere [52], I sketch how to interpret such "mixed" argument steps, but it is easier to understand the basic approach in their absence. By introducing additional subclaims where necessary, it is straightforward to convert arguments into *simple form* where each argument step is supported either by subclaims or by evidence, but not by a combination of the two. The "mixed" or free argument on the left of Figure 2 is converted to simple form on the right by introducing a new subclaim $\texttt{SC}_n$ and a new argument step $\texttt{ES}_n$ above $\texttt{E}_3$.

The benefit of simple form is that argument steps are now of two kinds: those supported by subclaims are called *reasoning steps* (in the example, argument step $\texttt{AS}_1$ is relabeled as reasoning step $\texttt{RS}_1$), while those supported by evidence are called *evidential steps* (in the example, these are the relabeled step $\texttt{ES}_2$ and the new step $\texttt{ES}_n$) and the key to our approach is that the two kinds of argument step are interpreted differently. Specifically, evidential steps are interpreted "epistemically," using ideas grounded in probability, while reasoning steps are interpreted "logically": subclaims supported by evidential steps that cross some threshold of credibility are accepted as premises in a classical deductive interpretation of the reasoning steps.

We now consider these two kinds of argument steps in more detail.

2.1. Evidential Steps

My recommended approach for evidential steps is described in a related paper [52]; here, I provide a summary and relate it to the indefeasibility criterion.

Evidential steps are the bridge between our assurance concepts, represented as claims and subclaims, and external reality consisting of the system, customers (and other stakeholders), and the world. Our concepts, such as correctness of an

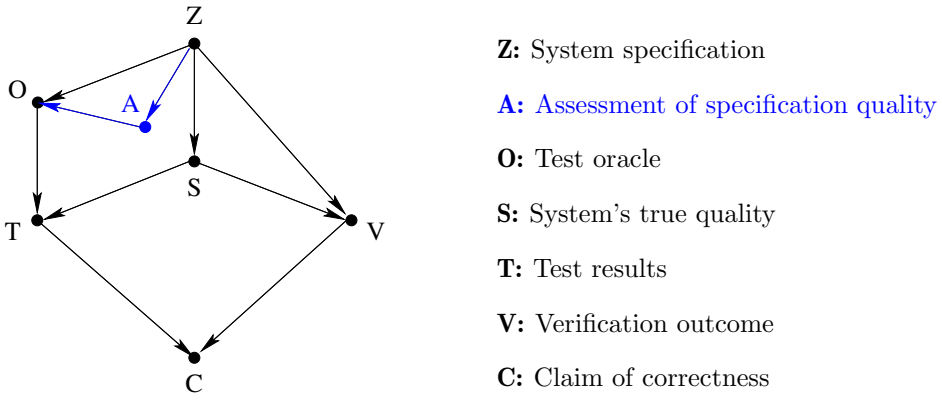

Figure 3. BBN for Testing and Verification Evidence

implementation with respect to its specification, might not be directly observable in the world, so we observe what we can—for example, we run tests, conduct walkthroughs, perform static analysis, and so on—and "weigh" this combination of evidence to see if it crosses some threshold of credibility that allows us to conclude the concept (i.e., claim or subclaim) concerned. There is often a choice between combining two (or more) items of evidence to support a single subclaim and using each to support its own subclaim, with those subclaims being combined in a higher level reasoning step. For example, on the right hand side of Figure 2, evidence $\mathtt{E}_1$ and $\mathtt{E}_2$ are combined in an evidential step, while $\mathtt{E}_3$ is combined with these in a higher level reasoning step. My opinion is that reasoning steps should be used to combine (subclaims corresponding to) evidence that is independent, while evidential steps combine evidence that has dependencies.

An example, developed in [52], considers evidence involving both empirical testing and formal verification of an implementation. These are not independent (although later I will revisit that assertion), since success or failure of one suggests a similar outcome is likely for the other. The question then, is how to combine and "weigh" these items of evidence, paying due attention to their lack of independence? We saw earlier that subjective probabilities provide a useful measure for confidence in assurance claims; we can extend this to subclaim concepts such as "implementation is correct," which is assessed by evidence that the system passed both formal verification and testing. Now there is a chance that a correct system will fail testing (the oracle, which judges whether a test succeeds or fails, could be flawed), and that an incorrect one will pass (testing samples only some of the possible runs), and likewise for verification (formal verification often has to use approximation), and we can allocate subjective probabilities to these. Bayesian Belief Nets (BBNs) then provide a method for combining probabilities such as these, and are supported by tools that facilitate construction of models and calculation of conditional probabilities.

The BBN for this example is shown in Figure 3, which is taken from [52] (and derived from [42]), where its analysis is described in more detail. The idea is that, given (presumably) successful test and verification outcomes, and conditional probabilities modeling how these are assumed to depend on the quality of the specification and the oracle and on the correctness of the implementation, the BBN allows us to calculate confidence in the claim of correctness C.

A key point that emerges from BBN analysis is that the value of testing evidence is highly dependent on the quality of the oracle, for which no evidence was initially provided. It is difficult to evaluate test oracles, but we might suppose they are more likely to be correct if the specification from which they are derived is "testable." Evidence for this (which might be expert opinion) is introduced to the BBN as the variable A; this kind of evidence is required by DO-178C [47, paragraph 6.3.1.d] and "what if" exploration with BBNs such as this can confirm its value. Observe that A is indirect or "confidence building" evidence: it does not measure a concept we are directly interested in, but "vouches for" the quality of some artifact (here, report of successful testing) used in that estimation; BBNs provide a way to use such confidence items in evidential steps.

It remains to determine what probabilistic quantities should be assessed in argument steps. When we have evidence E supporting a hypothesis or claim C, it seems plausible that our procedure should be to assess the conditional probability $P(C \mid E)$ and to accept C when this probability exceeds some threshold. Unfortunately, even experts find it difficult to directly assess a quantity such as $P(C \mid E)$. Furthermore, the significance of $P(C \mid E)$ depends on our prior assessment $P(C)$, which could be one of ignorance (or, in law, prejudice). For this reason, I recommend that we should take ideas from Bayesian Confirmation Theory [17], and instead use what are called *confirmation measures*.

The idea underlying these measures is that we are really interested in the ability of E to discriminate between C and its negation $\neg C$, so the quantities we should look at concern the relationship between $P(E \mid C)$ and $P(E \mid \neg C)$; such measures may be said to "weigh" C and $\neg C$ "in the balance" provided by E. Notice that $P(E \mid C)$ is related to $P(C \mid E)$ by Bayes' Theorem but seems easier to assess: that is, it seems easier to estimate the likelihood of concrete observations E, given a claim about the world C, than vice-versa. There is no agreement in the literature on the best confirmation measure: Fitelson [20] considers several and makes a strong case for Good's measure $\log \frac{P(E \mid C)}{P(E \mid \neg C)}$, while Tentori and colleagues [62] perform an empirical comparison and generally approve of Kemeny and Oppenheim's measure $\frac{P(E \mid C) - P(E \mid \neg C)}{P(E \mid C) + P(E \mid \neg C)}$. Most of the measures support similar verdicts,[10] and these can differ significantly from $P(C \mid E)$; for example, addition of the confidence item A in Figure 3 makes little difference to the value of $P(C \mid E)$, but the Kemeny-Oppenheim confirmation measure improves from 0.49 to 0.76 [52].

I do not propose that numerical assessment of these measures should be performed in most assurance cases; rather, I think that "what if" experiments and sensitivity studies with BBN tools, such as that reported in [52], can help refine the recommendations for evidence and thresholds provided in standards and guidelines such as DO-178C, where 71 assurance "objectives" are identified for the most critical aviation software with little rationale for how these were selected or how they support or depend on each other.

[10] Shogenji [57] proposes a measure of *justfication* that is equivalent to $1 - \frac{\log P(C \mid E)}{\log P(C)}$. Atkinson [7] shows that this measure may support different verdicts than the standard confirmation measures and that it is unique in a certain sense.

In summary, the evidence supplied in an evidential step should be evaluated, either with the aid of explicit probabilistic modeling, or informally using judgment honed by experiments with such models, and accepted when its combined weight crosses some threshold that is deemed sufficient for its subclaim to be accepted as a "settled fact."

The indefeasibility criterion comes into play when we ask what *defeaters* could exist for the evidence supplied. Defeaters are discussed in more detail in the next subsection, but the basic idea is that a defeater to an argument step is like a hazard to a system: a reason or circumstance why things might go wrong. For example, testing evidence is defeated if it is not for exactly the same software as that under consideration, and verification evidence is defeated if its theorem prover might be unsound. Thus, each evidential step must be buttressed by additional evidence to negate all possible defeaters (we need *all possible* to ensure indefeasibility). According to the dependencies involved, this additional evidence might be combined in the same evidential step as the original evidence or, more likely, it will be used in individually dedicated evidential steps to support separate subclaims that are combined in higher-level reasoning steps. This process is described in the following subsection.

2.2. Reasoning Steps

In evidential steps, we have seen that multiple items of evidence are combined and "weighed" to justify (belief in) truth of the (sub)claim concerned. This combination may be performed informally or it can use probabilistic modeling with BBNs, as in the example, where verification and testing evidence were combined with "confidence" evidence about testability of the specification. We need to use some kind of formal or informal "weighing" because the items of evidence may be neither definitive nor independent.

When evidence is (or can be treated as) independent, it can be used to support separate subclaims that are then combined in reasoning steps. Independence is a matter of judgment: thus, although formal verification and testing are not generally independent, it could be argued that they can be treated as such when we are dealing with extremely high quality software, where the likelihood of testing or verification failure is infinitesimal and each can be considered to examine a different aspect of the sparse failure space. Thus, the verification evidence in the example of Figure 3 could be split from the testing and testability evidence so that each supports a separate subclaim. Because they are (now assumed) independent, the subclaims in a reasoning step are simply *conjoined* to deliver the truth of their parent claim: that is, the claim in a reasoning step is considered true if and only if all its subclaims are so. This interpretation could be "inductive," that is the conjunction of subclaims strongly *suggests* the claim is true, or it could be "deductive," meaning the conjunction *implies* (or entails, or proves) the claim.

We earlier stated that indefeasibly justified belief is the criterion that should apply to assurance cases and it is now time to expand on this. There are two reasons why an assurance case may be inadequate: one is that some subclaims may be supported by weak evidence (e.g., if the evidence is testing, the tests may be too few), and the other is that some legitimate concerns are overlooked, so

that necessary subclaims and their supporting evidence are entirely absent. Some may think these are very similar: the response of public officials and the press to embarrassing system failures is generally to blame inadequate testing. But this misunderstands the rôle that testing and other evidence plays in assurance: it is not to search for faults but to provide confidence in certain subclaims (which may be about the absence of some kinds of faults) for the system. Thus weak evidence is damaging not because it may have failed to detect a fault but because it provides inadequate justification for belief in its subclaim, and hence the system must be considered vulnerable to whatever consequences may follow from possible falsity of that subclaim.

Thus, the evidence in an assurance case must be both strong and complete. The previous subsection considered techniques for "weighing" evidence and our strength criterion for adequately justified belief is for the weight of each evidential step to exceed some threshold. For completeness, our criterion is indefeasibility: we need to be sure that no concern has been overlooked and we apply this to each step of the argument. For reasoning steps, this corresponds to the deductive interpretation. However, although we ultimately require reasoning steps to be deductive, I advocate approaching this via the methods and tools of an inductive interpretation.

The reason for this is that assurance cases are developed incrementally: at the beginning, we might miss some possible concerns and will not be sure that our reasoning steps are deductive. As our grasp of the problem deepens, we may add and revise subclaims and argument steps and only at the end will we be confident that each reasoning step is deductive. Yet even in the intermediate stages, we will want to have some way to evaluate the case, and an inductive interpretation can provide this.

Furthermore, even when we are satisfied that the case is deductively sound, we need to support review by others. The main objection to assurance cases is that they are prone to "confirmation bias" [37]: this is the human tendency to seek information that will confirm a hypothesis, rather than refute it. The most effective counterbalance to this and other fallibilities of human judgment is to subject assurance cases to vigorous examination by multiple reviewers with different points of view. Such a "dialectical" process of review can be organized as a search for potential *defeaters*. That is, a reviewer asks "what if this happens," or "what if that is not true." These challenges effectively assert that the reasoning step concerned is inductive rather than deductive because there is some missing case, or because some subclaim is too weak or needs to be bolstered by an assumption or is just plain wrong, and so on. Hence, although all reasoning steps should be (accepted as) deductive when an assurance case is in its finished state, they may be considered inductive during development and review.

The idea of defeaters comes from epistemology, where Pollock [45, page 40] defines a *rebutting defeater* as one that (in our terminology) contradicts the claim of an argument step (i.e., asserts it is false), while an *undercutting defeater* merely doubts it (i.e., doubts that the claim really does follow from the proffered subclaims or evidence); others subsequently defined *undermining defeaters* as those that doubt some of the evidence or subclaims used in an argument step. I do not find this particular taxonomy very helpful, but I do believe that tools to sup-

port development and evaluation of assurance cases should provide systematic methods for proposing defeaters or otherwise challenging an argument step.

The response to such challenges may be to adjust the case, or it may be to dispute the challenge (i.e., to defeat the defeater). I think the record of such challenges and responses (and the narrative justification that accompanies them) should be preserved as part of the assurance case to assist further revisions and additional reviews. The fields of defeasible and dialectical reasoning provide techniques for evaluating such "disputed" arguments. For example, *Carneades* [25] is a system that supports dialectical reasoning, allowing a subargument to be *pro* or *con* its conclusion: a claim is "in" if it is not the target of a *con* that is itself "in" unless ... (the details are unimportant here). Weights can be attached to evidence and a *proof standard* is calculated by "adding up" the *pro*s and *con*s supporting the conclusion and their attendant weights. Thus, a conclusion is supported to the *preponderance of evidence* proof standard if it has at least one *pro* argument that is "in" and weighs more than any "in" *con* argument. For assurance cases, we ultimately want the proof standard equivalent to a deductive argument, which means that no *con* may be "in" (i.e., every defeater must be defeated). Takai and Kido [61] build on these ideas to extend the *Astah GSN* assurance case toolset with support for dialectical reasoning [6].

The top-down motivation for insisting that reasoning steps, when completed, should be deductive is that the overall argument must support the indefeasibility criterion for justified belief. The indefeasibility criterion for assurance cases was introduced in [53]; prior to that I adduced bottom-up grounds for the deductive requirement [52] and these reinforce the derivation from indefeasibility.

One bottom-up motivation is that assurance cases are generally very large and cannot truly be comprehended *in toto*: a modular or compositional method is essential. Deductive reasoning steps can be assessed in just such a modular fashion, one step or one claim at a time. First, we check local soundness: that is, for each reasoning step, we must assure ourselves that the conjunction of subclaims truly implies the claim. Second, we must check that claims are interpreted consistently between the steps that establish them and the steps that use them; this, too, is a modular process, performed one claim at a time. In contrast, the first of these is not modular for inductive steps—for when a step is labeled inductive, we are admitting a "gap" in our reasoning: we must surely believe either that the gap is insignificant, in which case we could have labeled the step deductive, or that it is taken care of elsewhere, in which case the reasoning is not modular.

My second bottom-up reason for deprecating inductive reasoning steps is that they implicitly acknowledge the presence of an unknown defeater. We may surely assume that any inductive step is "almost" deductive. That is to say, the following generic inductive step

$$p_1 \texttt{ AND } p_2 \texttt{ AND } \cdots \texttt{ AND } p_n \texttt{ SUGGESTS } c \tag{4}$$

would become deductive if some missing subclaim or assumption a (which, of course, may actually be a conjunction of smaller subclaims) were added, as shown below.[11] (It may be necessary to adjust the existing subclaims p_1 to p_1' and so on

[11] There are other, logically equivalent, ways to interpret the repair: for example, we could suppose that the premises stay the same but the conclusion is weakened to the claim c `OR NOT` a.

if, for example, the originals are inconsistent with a.)

$$a \texttt{ AND } p'_1 \texttt{ AND } p'_2 \texttt{ AND } \cdots \texttt{ AND } p'_n \texttt{ IMPLIES } c. \tag{5}$$

If we cannot imagine such a "repair," then surely (4) must be utterly fallacious. It then seems that any estimation of the doubt in an inductive step like (4) must concern the gap represented by a which is, in effect, an undercutting defeater to its deductiveness. Now, if we knew anything at all about a it would be irresponsible not to add it to the argument. But since we did not do so, yet acknowledged that (4) is inductive, we must be ignorant of a and of the magnitude of doubt that it represents.

One way to buttress support for an inductive step could be to provide additional subclaims or evidence that strengthen some aspects of the justification, even though they do not change its inductive character. These intended strengthenings are called *confidence claims*. We saw earlier that confidence items can play a constructive rôle in evidential steps (recall the "testability" evidence A in Figure 3) but, there, we were in the context of "weighing" evidence that is interdependent. In a reasoning step, the subclaims are assumed to be independent and there seems no logical way to interpret the contribution of a confidence claim. Furthermore, Hawkins *et al* observe a tendency to include numerous confidence claims "just in case," resulting in "voluminous, rambling, ad infinitum arguments" [27] that fail to fulfill the primary purpose of an assurance case, which is to communicate a clear argument.

Hawkins *et al* [27] propose that confidence claims should comprise a separate part of the argument: the primary argument directly supports the top-level claim, while the confidence part provides arguments for believing the primary part. This is mistaken, in my view (the reasons for believing a reasoning step are not specified as claims but are part of its narrative justification), and derives from acceptance of inductive reasoning steps, which lack a clear criterion for adequacy. Deductive reasoning steps, on the other hand, have a strict criterion and it is very clear what their evaluation must accomplish: it must review the content and justification of the step and assent (or not) to the proposition that its subclaims truly imply the claim. The justification is provided as a narrative attached to each reasoning step. There is no rôle for confidence claims in deductive reasoning steps, and other superfluous subclaims are likely to complicate rather than strengthen the assessment.[12] Hence the requirement for deductive soundness encourages the formulation of precise subclaims and concise arguments.

Reasoning steps, as I have described them, have a simple and austere form: they comprise a claim, deductively supported by a conjunction of subclaims. Critics may argue that this simplicity is illusory, because it is impractical: they might cite argument over hazards as an example that is inherently inductive. The idea is that we identify all hazards and then show that the system design eliminates or adequately mitigates each one. A reasoning step will justify the claim "all hazards eliminated or adequately mitigated" by a conjunction of subclaims "$\texttt{hazard}_1$ eliminated or adequately mitigated" and similarly for $\texttt{hazard}_2$, ...,

[12] Context and assumptions are two additional kinds of claim that sometimes appear in assurance cases. In my opinion these are unnecessary because ordinary subclaims can fulfill the same rôles [52].

$\texttt{hazard}_n$; some will argue that this is necessarily an inductive step because we cannot know that these are the only hazards. But we must have performed hazard analysis to discover hazards $\texttt{hazard}_1$ to $\texttt{hazard}_n$, so we conjoin an additional subclaim that asserts "$\texttt{hazard}_1$, $\texttt{hazard}_2$, ..., $\texttt{hazard}_n$ are the *only* hazards" and the step is now deductive. This is not a trick: deductiveness of reasoning steps may sometimes be argued in the narrative justification for the step and other times, particularly when, as here, it involves an enumeration, in a subclaim. Doubts then focus, as they should, on the evidential support for this subclaim, which will describe the method of hazard analysis employed, the diligence of its performance, historical effectiveness, and so on.

From a strict viewpoint, all assurance cases must be inductive, even under the indefeasibility criterion, for we lack omniscience: the best we can achieve is adequately justified belief (i.e., confidence), not certainty. What my interpretation of indefeasible assurance cases accomplishes, however, is to isolate all inductive doubt in the evidential steps where, in principle, probabilistic measures for the weight of evidence provide the intellectual tools to manage confidence or its lack.[13] What we now need is a way to propagate measures of confidence from the evidential steps through the reasoning steps to the top claim.

3. Complete Arguments

A complete structured argument in simple form consists of evidential steps, all of which are deemed to have crossed some threshold of credibility, and reasoning steps, all of which are judged to be deductively valid. An argument that satisfies these criteria is considered sound and we then regard the evidential steps as premises in a conventional logical interpretation of the reasoning steps so that, together, they "prove" the top claim. However, some sound arguments may be considered "stronger" than others, so we need a way to evaluate argument strength; ideally, this should relate to p_{nf} so that it can be applied in the analysis following (3).

A sound argument must be indefeasible, so a measure of argument strength could, in principle, evaluate how persuasively this is justified. Justification for indefeasibility requires that both the developers of the assurance case and its reviewers actively search for defeaters and document these, including successfully resisted challenges (i.e., "defeated defeaters") as part of the case. The goal is to ensure that all assumptions and contingencies have been discovered and dealt with: there must be no surprises following deployment. I accept that there may be opportunity for undertaking these activities with greater or lesser degrees of rigor depending on the risk posed by the system, but I do not see how to attach a measure to this. Furthermore, a weak case for indefeasibility is similar to acceptance of inductive reasoning steps, and I caution against (in fact, deprecate) these for reasons presented in the previous section. Hence, I recommend that indefeasibility of each reasoning and evidential step should be justified "as rigorously as feasible."

[13] This can be seen as a more systematic version of the style of informal argumentation known as "natural language deductivism."

The other criterion for a sound argument is that its evidential steps must each cross a threshold of credibility, and these thresholds are a candidate measure for argument strength. It is debatable whether all the evidential steps in a given case should use the same threshold. Intuitively, it seems that some evidentially supported subclaims might be considered more important (and thus require higher thresholds) than others. But, on the other hand, all these subclaims are combined via reasoning steps higher up in the argument, and the method of combination is conjunction. Thus, the falsity of any subclaim in a reasoning step is sufficient to falsify its parent claim and it therefore seems inappropriate that some should be justified to different thresholds than others.

Independently of whether the thresholds are the same or different, we need to consider how to combine the assessed strength for each step into one for the overall argument. There seem several plausible ways to proceed. If the evidential steps are assessed by probabilities representing $P(C \mid E)$ (i.e., the probability of the local subclaim, given the local evidence) then we could say that a conservative assessment of the top claim is the product of all the local assessments (since subclaims are assumed independent, and are therefore conjoined in reasoning steps). However, we saw that there are good reasons for preferring other assessments for evidential steps, such as confirmation measures. There is little research on how confirmation measures should propagate through arguments, but there are indications that it may not be straightforward.

For example, we might hope that if evidence E supports a claim C and C deductively entails C', then E should also support C'. This expectation is refuted by the following counterexample. Let C be the claim that a certain playing card (drawn at random from a shuffled deck) is the Ace of Hearts, let C' be the claim that the card is red, and let E be the evidence that the card is an Ace. Certainly E (Ace) supports C (Ace of Hearts), which entails C' (red), but E (Ace) cannot be considered to support C' (red). However, we should note that the evidence "Ace" provides incomplete support for the claim "Ace of Hearts"—this is really a conjunction "Ace `AND` Heart" and we have evidence for Ace but not for Heart, so this counterexample is not persuasive when the indefeasibility criterion is imposed.

One plausible option is that confirmation measures for the evidential steps in an indefeasibly valid argument compose as a weakest link phenomenon. That is, the confirmation measure of the top claim is no worse than the least value for any evidential subclaim. This could still apply when confirmation assessments are applied informally, resulting in ordinal measures such as "low," "medium," and "high," but the simplest option seems to be that the top claim is simply assessed *true* when all its evidential subclaims exceed their thresholds.

It might seem that to exploit the line of argument following from (3) and thereby contribute to a quantitative case for certification, assurance does need to deliver a numerical assessment for p_{nf}. However, the rôle of p_{nf} in that equation is to provide acceptable confidence in the absence of faults (i.e., we need a judgment rather than a number), and the "simplest" option above can be interpreted as assessing that confidence directly.

Different systems and subsystems pose different risks and so their assurance targets should differ appropriately, as in the case mentioned earlier of aircraft

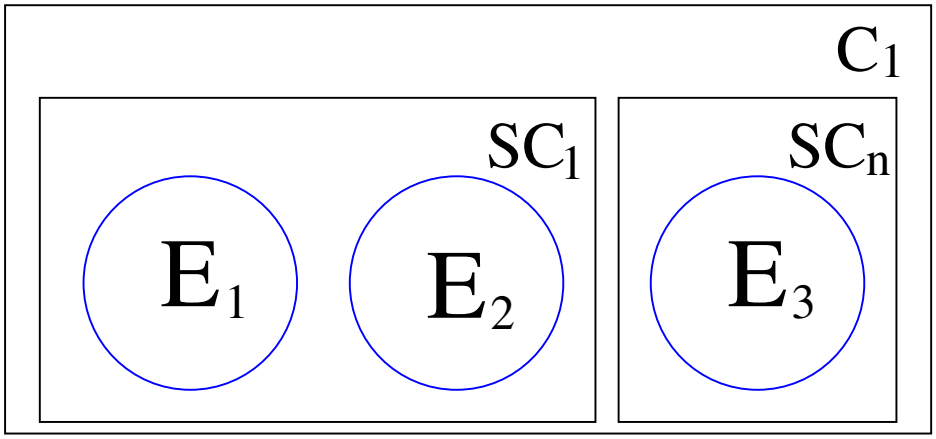

Figure 4. Venn Diagram of Assurance Case

software that is graduated from Level A to Level D. One way to accomplish this within a fixed argument structure is to lower the thresholds for acceptable credibility of evidential steps. With testing evidence, for example, we could accept fewer tests, or tests to a lesser coverage standard, for systems with lower assurance targets. Within an evidential step, we could even eliminate some items of evidence if these are considered strongly dependent on others (hence providing little diversity), or are merely confidence items (such as the testability evidence in Figure 3). Elimination of an item of evidence would also eliminate the need for subclaims and supporting evidence concerning its defeaters: for example, if formal verification evidence is eliminated, we no longer need evidence about the soundness of the verifier concerned. However, elimination of a subclaim and its attendant evidence other than as a consequence of the removal of evidence elsewhere seems unjustifiable: such removal would mean that its parent reasoning step is no longer deductive and hence the overall argument would no longer meet the indefeasibility criterion.

Although subclaims cannot be removed without sacrificing the indefeasibility criterion, they can certainly be rearranged. Figure 4 provides an alternative view of the assurance case on the right of Figure 2; here, the Venn diagram represents the space of "concerns" about the system. Their totality is covered by the claim $\texttt{C}_1$, which is partitioned into those covered by the subclaims $\texttt{SC}_1$ and $\texttt{SC}_n$, which are themselves covered by their nested evidence. Although the space would not be covered if we eliminated one of these subclaims, we can imagine it being divided up differently (and, indeed, it would be a different space) had we chosen other subclaims.

Michael Holloway has retrofitted an assurance case to DO-178C [28] that makes explicit the way its assurance is "graduated" from Level A down to D. Some of this is accomplished by lowering the threshold on evidence (for example, unit tests for Level A software must achieve MC/DC coverage, while Level B requires merely DC), and some involves changes to the argument. For example, Levels A, B, and C employ Low Level Requirements (LLR, these are actually specifications in our terminology), but these and all their attendant objectives are

eliminated at Level D. The reasoning seems to be that the correctness argument from High Level Requirements (HLR) to Executable Object Code (EOC) is more credible if it is broken into smaller steps via the LLR; it seems to me that this may be plausible if the argument is informal, but would not be so if the relation were formally verified, or if the EOC were generated from the HLR by automated synthesis. Some of the other changes are rather difficult to reconcile with my account, because Holloway employs a confidence argument (which I deprecate) in addition to the primary correctness argument. I discuss some of the difficulties in recasting DO-178C as an assurance case elsewhere [51, Section 3.1], and there is certainly scope for differing opinions on how this should be done. My hope is that the framework developed here will prove useful in formulating successors to DO-178C and similar guidelines directly as explicit assurance cases that achieve broad consensus.

4. Future Challenges

The purpose of assurance is to provide guarantees about the future, so that we can be confident our system will do what we want and will not cause or allow bad things to happen. These guarantees are based on two bedrock assumptions: first, that it *is possible* to predict the future behavior of our system in the world and, second, that with enough effort we can obtain *near-omniscient* (i.e., indefeasible) insight into those behaviors.

Those bedrock assumptions are credible only in highly constrained circumstances: those where both our system and the world in which it operates are fully predictable. These constraints do apply to traditional aircraft systems, for example, where the world consists of well-understood physical phenomena (nothing is more accurately modeled than the flow of air over a wing), simple physical devices (flaps, landing gear, etc.) with well-understood failures, other computerized systems, and trained and disciplined human operators (pilots and air traffic controllers).

But even in this simple space, additional constraints are imposed to ensure predictability. In particular, the FAA requires aircraft systems to be deterministic. This affects software design (e.g., choice of scheduling strategy), hardware exploitation (e.g., use of caches and multiple cores), and algorithm design. For example, control laws for the ailerons and other control surfaces need to change as the aircraft transitions from slow flight in warm dense air at takeoff to transonic flight in cold thin air at cruise. A single adaptive control law could do this, but these are considered insufficiently predictable so, instead, as many as 30 separate, individually certified, control laws are used with complex rules to smooth the transitions between them. Concern about predictability is not unreasonable: adaptive control is implicated in the crash of one of NASA's X-15 spaceplanes and the death of its pilot [16] (the problem was not in adaptive control considered alone, but its behavior in the presence of other failures, as one of the elevons had lost 80% of its control effectiveness).

The constraints that ensure the bedrock assumptions are becoming less tenable in modern systems, so on what alternative foundations can credible assurance

be constructed? There are two dimensions to this question: one concerns properties and predictability (a strong property may not be indefeasibly predictable, but a weaker one might), and the other concerns the location of unpredictability: is it in the system (where unpredictability can sometimes be managed by architectural means), or in the world, or both?

An example where unpredictability in the system is transformed into predictability for a weaker property is provided by the so-called "simplex" architecture [56]. Here, we have a sophisticated primary component that provides attractive but unassurable behavior (e.g., an adaptive control law), and a secondary component that is less attractive, but assurable (e.g., a verifiably safe, but crude control law). The secondary component guarantees safety as long as the system is within some envelope; the primary component generally drives the system but the architecture transitions to the secondary component when monitoring indicates the system is approaching the edge of the secondary component's envelope.

The simplex architecture can be seen as the extension to reactive systems of an earlier technique for sequential systems called (provably) "safe programming" [4], and both are related to the "detectors and correctors" treatment of fault tolerance [5] and similar treatments for security [63].

All these techniques depend on a monitor function that detects incipient failure of the primary. A monitor can be very simple: it does not need to do anything active, just observe the values of sensors and internal variables and periodically evaluate some predicate. The predicate need not be a full check on the system's operation, but rather some property that is relevant to its safety claim. Due to its simplicity, it is plausible that assurance can deliver strong confidence that a monitor is fault-free with respect to the monitored property. Now it is a remarkable fact that when we have a system with two "channels," one that is reliable with *pfd* p_A, and one for which we have confidence p_B in its nonfaultiness, then the reliability of their combination has $pfd \leq p_A \times p_B$ [41].[14] Notice that the *pfds* of two reliable channels *cannot* be multiplied together because their failures may not be independent.

The simplest arrangement of this type is a "monitored architecture" where the first channel provides the standard operation of the system and the second is a monitor that signals failure when its property is violated. This prevents transitions into hazardous states and notifies higher-level fault management (e.g., human pilots) that something is amiss. In a monitored architecture, we must consider failures of commission (i.e., the monitor signals failure when it should not) as well as omission (i.e., the monitor does not signal when it should). These details are developed and analyzed in the paper cited above [41, Sections 4 and 5] and illustrated for the case of an aircraft incident that involved failure of a massively redundant fuel management system. The notable conclusion is that confidence in nonfaultiness of the monitor provides strong assurance for reliability of the overall system. The closer the monitored property is to a top-level claim, the stronger and more direct this assurance will be.

[14] This is an *aleatoric* analysis; to apply it, we need *epistemic* estimates of the parameters involved and these may introduce dependencies. However, it is often possible to manage these by making conservative estimates [69].

In its most basic form, the monitored property can be a simple predicate; in the fuel management example mentioned above this would check that the fuel is distributed among the aircraft tanks in a way that is laterally balanced, has the center of gravity in the correct place, is distributed along the wings in a way that minimizes structural loads, and keeps an adequate supply in the engine tanks. But in more complex systems, we may need to check stimulus-response properties: for example, the automation of an "Increasingly Autonomous" (IA) aircraft (one where automation can stand in for a human pilot) may request the human pilot to perform some task (e.g., set radio frequencies) and must then listen for the confirmation. Temporal logics provide suitable foundations for describing such properties and languages based on these (e.g., [9]) have been developed for a variant of monitoring known as "runtime verification" [26,35]. For more complex properties, the property specification language may need to be further enriched to include real-time and other quantitative attributes [3,15]. In the presence of imperfect sensors (e.g., a vision understanding system) the monitored property may need to be expressed probabilistically [32]. When human interaction is considered, it will generally be necessary to include some representation of the humans' "mental models" [48] and their states of knowledge and belief [2], and even trust [29]. These concepts require ever-richer logics, and combinations of logics, as the foundations for languages to describe them.

The simplex architecture and its variants extend monitored architectures by adding a secondary operational channel that can take over when the monitor detects failure of the primary. Assurance for this architecture can be based on that for the monitor and the secondary channel alone, with little or no reliance on the primary operational channel. (Presumably the primary channel delivers attractive behavior, so weak assurance for the ordinary requirements may derive from this channel, while strong assurance for safety derives from the monitor and secondary channel.)

The simplex architecture can be extended to multiple levels of fall-back with less and less desirable, but still safe, backup channels or behaviors. But what if there is no safe behavior? The crash of Air France flight 447 in 2009 provides an example: here, the Pitot tubes were blocked by ice crystals, depriving the airplane of airspeed data. The autopilot detected the problem and disconnected handing control to the pilots (i.e., the standard behavior of a monitored architecture). In the absence of airspeed data, pilots are instructed to hold pitch and thrust constant, although this is difficult (AF 447 was in turbulence and cloud) and delicate (at altitude there is a small margin between stall and Mach buffet/overspeed). However, the pilot flying misinterpreted the situation[15] and stalled the airplane all the way down to the ocean, resulting in the deaths of everyone on board [14]. The autopilot of a future IA aircraft, could offer to hold the plane straight and level (i.e., a form of simplex architecture) while the pilot troubleshoots—indeed, the autothrottle of AF 447 did automatically enter thrust lock, but the pilot flying disabled it. But what would be the basis for assuring the safety of this procedure? In the absence of reliable airspeed data, it is impossible to give guarantees on the safety of flight.

[15]The formal analysis of [2] attributes this to a lack of "negative introspection," meaning the pilot was unaware he lacked necessary information (i.e., he did not know what he did not know).

Assurance, as developed in the previous sections, requires indefeasible confidence in strong claims for critical properties; here, we cannot achieve this, so should we relax indefeasibility or the strength of the claims? My opinion is that we should retain indefeasibility—we must remain sure that nothing has been overlooked—but lower the threshold on evidential claims. In the absence of airspeed data, we cannot be sure of the performance of the wing, but we can attempt to estimate that data from other sensors (e.g., ground speed and altitude from GPS) and recent history (e.g., stale air data), with fixed (or table-lookup) pitch and thrust as a final fallback. Although these estimates cannot support strong assurance, it seems to me that this approach is the best we can do and it retains the basic outline and justification we have developed for trust in assurance cases.

Monitoring provides a systematic way to deal with unpredictability: if we cannot predict how our system or the world will behave, we monitor them and base our assurance case on claims that are guaranteed *assuming* the monitored properties. However, this requires that we can accurately monitor properties of interest, which may not be so, especially when uncertainty is located in the world. Automated driving provides an instructive example. Here, we have cameras and possibly LiDARs and other sensors, and a sensor fusion and vision understanding system that attempts to discern the layout of the road and the presence of traffic signals and obstacles and other hazards. The vision system uses machine learning, so its operation is opaque and unsuited to conventional assurance[16] and we have no independent way to monitor its correct interpretation of the scene. However, although we cannot monitor a safety claim directly related to its top-level function, we can monitor related properties. For example, given the car's location (from GPS) and map data, we can calculate static features of the expected scene, such as a gas station on the left and a mailbox on the right, or an intersection ahead. A monitor can calculate these predicted features and check that the vision system reports them. Such a monitor does not guarantee that the vision system is operating safely, but at least we know it is awake and interpreting the world around it with some accuracy [43, 59].

A variant is to monitor the state of the system against its own history. The idea (a generalization of [63]) is that we observe the system during (incident free) operation, either in test or in prior operation, and construct a compact representation that approximates the states encountered. We then monitor the system in execution against that representation and raise a signal when we encounter a state that differs from those encountered previously. The idea is to alert higher-level functions, or a human operator, that the system is now entering a space that it has not previously encountered.

Beyond monitoring and backup channels to ensure satisfaction of assumptions, safety properties, or other weaker claims, are systems that adapt their behavior to ensure these. This can be appropriate in highly unpredictable envi-

[16]There is much recent interest in "Explainable AI" and in verification of AI systems that use deep neural nets. Some of this was prompted by "adversarial perturbations" where small changes to an image cause a vision system to misclassify it ([30] gives examples) leading to concern, for example, that an automated driving system might miss a stop sign. There has been some success in formal analysis and verification of these topics (e.g., [30, 33]), so this is an area where we can expect rapid change.

ronments, such as those involving collaboration with humans, as in IA aircraft where the automation may replace one of the pilots. In addition to mandated and airline-specific standard operating procedures (SOPs), there is a huge amount of "implicit knowledge" underlying the interaction of human pilots. Attempting to program suitable behavior for each scenario into an IA system seems difficult and fault prone. It might be better to monitor the total system state and pilot activity against a model of the aircraft, SOPs, checklists, and high-level "rules of flight" and to employ alternative courses of action when the state does not evolve as expected.[17] For example, if the IA system operating as the "pilot flying" requests the human as "pilot monitoring" to deploy the landing gear, it should expect to see the state equivalent to "three greens" (gear down and locked) within some reasonable time and, if not, take some alternative action (e.g., repeat the request, try to diagnose the problem, release the gear itself, initiate a go-around, switch roles etc.).

Some do expect these behaviors to be pre-programmed and propose to verify their scripts against high-level requirements such as "rules of flight" [19, 65] or financial regulations [23], whereas others advocate their generation at runtime by an automated planner or other (verifiably sound) synthesis procedure driven from those requirements and the modeled and observed environment. (Among other benefits, a system of this kind can explain the reasons for its actions.) Assurance then derives from a combination of the validity of the rules (assured statically, at design time), and their interpretation and monitoring at runtime. Whereas a traditional system might use formal verification to provide design-time assurance for some aspect of design, a system of the kind envisaged here uses formal synthesis and monitoring to construct and verify the design at runtime. Because some of the assurance is derived from runtime analysis and synthesis, this approach is provocatively named "runtime certification" [49]. Also provocatively, we can speculate that advanced and future systems of this kind might evaluate their actions not merely against SOPs and safety rules, but against models of social norms and ethics.[18] And beyond systems that, in effect, generate assurance evidence at runtime, we can anticipate those that construct parts of the assurance argument at runtime. Such behavior is very plausible in the Internet of Things, where separately assured systems discover each other and integrate to achieve some coordinated purpose (e.g., multiple medical devices attached to the same patient). Safety may require that the separate systems exchange their assurance cases and adjust their behavior to ensure compatibility (e.g., synthesize wrappers to constrain behavior by one that would be a hazard to the other), or synthesize a new collective case [54, 64].

It is unlikely that autonomous systems operating in unpredictable environments can be assured to the level of today's highly constrained systems. A plausible goal would be "at least as good as a human"; in the case of automated driving this might be quantified as more than 165,000 miles between crashes.[19] As in

[17] Inadequate monitoring and challenging are implicated in the majority of aircraft accidents attributed to (human) crew error [21].

[18] Already there is an annual workshop on "Social Trust in Autonomous Robots".

[19] To be clear, we would expect classical car and aircraft systems to be assured to the level they are today, or better; it is their automated driving and IA autopilots that might be assured to the

cases where failures render the system unsafe (e.g., the AF 447 scenario described earlier), assurance cases for autonomous systems operating in unpredictable environments cannot be watertight. So we have to decide whether some alternative means of assurance should be adopted (e.g., licensing, or massive testing) or, if assurance cases are retained, whether we should relax the indefeasibility criterion or other elements of the traditional case.

In my opinion, equation (3) and its subsequent analysis retain their value even when assurance goals are relaxed (i.e., we use a smaller value for n), because some confidence in absence of faults is needed to "bootstrap" the Bayesian analysis that demonstrates the value of (even massive) pre-deployment testing and subsequent experience in operation. An assurance case is surely the only intellectually credible means to obtain that confidence: what other basis can there be apart from evidence and argument? Similarly, indefeasibility seems the only credible basis for justified belief: unless we (and our dialectical challengers) have made a comprehensive attempt to think of *everything* that can go wrong with both our system and our argument (i.e., all hazards and defeaters), then we are vulnerable to faults that may reside in the overlooked cases.

5. Summary and Conclusions

We have seen that assurance provides justified belief that a system is free of serious faults. Confidence in that belief can be expressed as a subjective probability, which then translates into a similar probability that the software will suffer no critical failures in its entire operational lifetime. However, there is a possibility that our confidence is misplaced and so we also need to be sure that the system will not fail even if it does contain faults. Confidence in that property is acquired incrementally through predeployment testing and operational experience that is justified by a Bayesian analysis bootstrapped under worst-case assumptions from the initial confidence in fault-freeness, and is therefore conservative.

Justified belief in the quality of the system is based on evidence about its design, construction, behavior, and other relevant attributes. Evidence is organized into an assurance case using a structured argument, which is partitioned into evidential and reasoning steps. The former, which are interpreted epistemically, combine related items of evidence to deliver probabilistic confidence in properties referred to as local claims, while the latter, which are interpreted logically, conjoin these to justify higher level claims and, ultimately, the top claim, which is the safety property of interest.

To provide truly justified confidence in its top claim, the argument steps of an assurance case must be indefeasible, meaning we must be so sure that all objections and difficulties (i.e., hazards to the system and defeaters to the argument) have been identified and considered that there is no new information that would change our assessment. Each argument step is supported by a narrative justification of its indefeasibility and, for evidential steps, of the thresholds on

human-equivalent level. Note also that a crash by an automated car should lead to investigation of the precipitating scenario and improvement in the automated driving software that benefits the safety of all automated cars, whereas human drivers learn only from their own experience.

evidence (e.g., if the evidence is testing, the number of tests required and the criteria for their selection).

As a human construct, there is always the possibility that an assurance case may be flawed; thus it must be subjected to active scrutiny and challenge by independent reviewers. Reviewers may contest the indefeasibility of each argument step and the thresholds on evidential steps. When an assurance case is accepted as sound, its evidential steps are treated as premises in a deductive logical interpretation of its reasoning steps. During development and review, however, an assurance case may not yet be indefeasible, and there can be utility in tools that support inductive and defeasible interpretations of the partial case.

Systems that pose less risk may be assured to lower criteria; a standardized assurance case can be graduated to deliver reduced assurance by lowering the thresholds on evidential steps. This may allow the elimination of some evidence and, hence, of subclaims to support its indefeasibility (e.g., if we drop static analysis, we no longer need evidence for the soundness of its analyzer).

This model of assurance can be used, retrospectively, to explain the success of current assurance practices, such as those for commercial aircraft and, prospectively, to guide their evolution to cope with the challenges of new developments. I recommend that future guidelines and standards for assurance should be explicitly constructed as (templates for) assurance cases. These could allow sufficient flexibility to accommodate novel systems and methods while retaining the industry-wide scrutiny that seems to have been effective in ensuring the effectiveness of current practices.

However, some current and many likely future systems break key assumptions underlying our basic model for assurance: namely, that we have a good understanding of the operation of our system, and that its environment is predictable. Self-driving cars exhibit both these difficulties: they use vision understanding systems based on machine learning to detect the layout of the road and its obstacles, and they must cope with the often unpredictable behavior of other road users.

My opinion is that the basic framework of assurance cases can be employed for these systems, but much of the evidence must be based on runtime monitoring, and some of the assurance argument also may need to be constructed at runtime.

The challenge for the future, then, is to develop practical methods for assurance of these new systems that are as effective, and based on foundations as credible, as those for the best current systems.

Acknowledgments.

This work was partially funded by SRI International and builds on previous research that was funded by NASA under a contract to Boeing with a subcontract to SRI International.

I have benefited greatly from extensive discussions on these topics with Robin Bloomfield, Bev Littlewood, Lorenzo Strigini, and others at Adelard and City University, and with John Knight, tragically now departed, of Dependable Computing and University of Virginia.

References

[1] *ASCAD: Adelard Safety Case Development Manual*. Adelard LLP, London, UK, 1998. Available from https://www.adelard.com/resources/ascad.html.

[2] Seth Ahrenbach. Reasoning about safety-critical information flow between pilot and computer. In *NASA Formal Methods*, Volume 10227 of Springer-Verlag *Lecture Notes in Computer Science*, pages 342–356, Mountain View, CA, May 2017.

[3] Rajeev Alur, Dana Fisman, and Mukund Raghothaman. Regular programming for quantitative properties of data streams. In *European Symposium on Programming Languages and Systems*, Volume 9632 of Springer-Verlag *Lecture Notes in Computer Science*, pages 15–40, Eindhoven, The Netherlands, April 2016.

[4] T. Anderson and R. W. Witty. Safe programming. *BIT*, 18:1–8, 1978.

[5] Anish Arora and Sandeep S. Kulkarni. Detectors and correctors: A theory of fault-tolerance components. In *18th International Conference on Distributed Computing Systems*, pages 436–443, IEEE Computer Society, Amsterdam, The Netherlands, 1998.

[6] Astah. *Astah GSN home page*. http://astah.net/editions/gsn.

[7] David Atkinson. Confirmation and justification. A commentary on Shogenji's measure. *Synthese*, 184(1):49–61, January 2012.

[8] John Barker. What you don't know won't hurt you. *American Philosophical Quarterly*, 13(4):303–308, October 1976.

[9] Howard Barringer, Allen Goldberg, Klaus Havelund, and Koushik Sen. Rule-based runtime verification. In *Verification, Model Checking, and Abstract Interpretation, VMCAI 2004*, Volume 2937 of Springer-Verlag *Lecture Notes in Computer Science*, pages 44–57, Venice, Italy, January 2004.

[10] Antonia Bertolino and Lorenzo Strigini. Assessing the risk due to software faults: Estimates of failure rate vs. evidence of perfection. *Software Testing, Verification and Reliability*, 8(3):156–166, 1998.

[11] Peter Bishop. Does software have to be ultra reliable in safety critical systems? In SafeComp [55], pages 118–129.

[12] Peter Bishop and Robin Bloomfield. A conservative theory for long-term reliability-growth prediction. *IEEE Transactions on Reliability*, 45(4):550–560, 1996.

[13] Peter Bishop and Robin Bloomfield. A methodology for safety case development. *Safety and Reliability*, 20(1):34–42, 2000.

[14] *Final Report on the Accident on 1st June 2009 to the Airbus A330-203 registered F-GZCP operated by Air France flight AF 447 Rio de Janeiro–Paris*. Bureau d'Enquêtes et d'Analyses (BEA), Paris, France, July 2012.

[15] Radu Calinescu, Carlo Ghezzi, Marta Kwiatkowska, and Raffaela Mirandola. Self-adaptive software needs quantitative verification at runtime. *Communications of the ACM*, 55(9):69–77, 2012.

[16] Zachary T. Dydek, Anuradha M. Annaswamy, and Eugene Lavretsky. Adaptive control and the NASA X-15-3 flight revisited. *IEEE Control Systems Magazine*, 30(3):32–48, March 2010.

[17] John Earman. *Bayes or Bust? A Critical Examination of Bayesian Confirmation Theory*. MIT Press, 1992.

[18] *Certification Specifications and Acceptable Means of Compliance for Large Aeroplanes, CS-25 and AMC-25*. The European Aviation Safety Agency (EASA), June 2016. Amendment 18; available at https://www.easa.europa.eu/document-library/certification-specifications.

[19] Michael Fisher, Louise Dennis, and Matt Webster. Verifying autonomous systems. *Communications of the ACM*, 56(9):84–93, September 2013.

[20] Branden Fitelson. *Studies in Bayesian Confirmation Theory*. PhD thesis, Department of Philosophy, University of Wisconsin, Madison, May 2001. Available at http://fitelson.org/thesis.pdf.

[21] *A Practical Guide for Improving Flight Path Monitoring: Final Report of The Active Pilot Monitoring Working Group*. Flight Safety Foundation, November 2014.

[22] Pedro Fonseca, Kaiyuan Zhang, Xi Wang, and Arvind Krishnamurthy. An empirical study on the correctness of formally verified distributed systems. In *Proceedings of the Twelfth*

European Conference on Computer Systems (EuroSys'17), pages 328–343, Association for Computing Machinery, Belgrade, Serbia, April 2017.

[23] Reginald Ford, Grit Denker, Daniel Elenius, Wesley Moore, and Elie Abi-Lahoud. Automating financial regulatory compliance using ontology+rules and Sunflower. In *Proceedings of the 12th International Conference on Semantic Systems*, pages 113–120, Association for Computing Machinery, Leipzig, Germany, September 2016.

[24] Edmund L. Gettier. Is justified true belief knowledge? *Analysis*, 23(6):121–123, 1963.

[25] Thomas F. Gordon, Henry Prakken, and Douglas Walton. The Carneades model of argument and burden of proof. *Artificial Intelligence*, 171(10):875–896, 2007.

[26] Klaus Havelund and Grigore Roşu. Synthesizing monitors for safety properties. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2002)*, Volume 2280 of Springer-Verlag *Lecture Notes in Computer Science*, pages 342–356, Grenoble, France, April 2002.

[27] Richard Hawkins, Tim Kelly, John Knight, and Patrick Graydon. A new approach to creating clear safety arguments. In Chris Dale and Tom Anderson, editors, *Advances in System Safety: Proceedings of the Nineteenth Safety-Critical Systems Symposium*, pages 3–23, Southampton, UK, February 2011.

[28] C. Michael Holloway. Explicate '78: Discovering the implicit assurance case in DO-178C. In Mike Parsons and Tom Anderson, editors, *Engineering Systems for Safety. Proceedings of the 23rd Safety-critical Systems Symposium*, pages 205–225, Bristol, UK, February 2015.

[29] Xiaowei Huang and Marta Kwiatkowska. Reasoning about cognitive trust in stochastic multiagent systems. In *Proceedings, AAAI-2017*, San Francisco, CA, February 2017.

[30] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. *arXiv preprint arXiv:1610.06940*, 2017.

[31] Michael Jackson. *Software requirements & specifications: A Lexicon of Practice, Principles and Prejudices*. ACM Press/Addison-Wesley Publishing Co., 1995.

[32] Susmit Jha and Vasumathi Raman. Automated synthesis of safe autonomous vehicle control under perception uncertainty. In *NASA Formal Methods*, Volume 9690 of Springer-Verlag *Lecture Notes in Computer Science*, pages 117–132, Minneapolis, MN, June 2016.

[33] Guy Katz, Clark Barrett, David Dill, Kyle Julian, and Mykel Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. *arXiv preprint arXiv:1702.01135*, 2017.

[34] Tim Kelly. *Arguing Safety—A Systematic Approach to Safety Case Management*. DPhil thesis, Department of Computer Science, University of York, UK, 1998.

[35] Insup Lee, Sampath Kannan, Moonjoo Kim, Oleg Sokolsky, and Mahesh Viswanathan. Runtime assurance based on formal specifications. In *Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 279–287, Las Vegas, NV, June 1999.

[36] Keith Lehrer and Thomas Paxson. Knowledge: Undefeated justified true belief. *The Journal of Philosophy*, 66(8):225–237, April 1969.

[37] Nancy Leveson. The use of safety cases in certification and regulation. *Journal of System Safety*, 47(6):1–5, 2011.

[38] B. Littlewood and D. R. Miller. Conceptual modeling of coincident failures in multiversion software. *IEEE Transactions on Software Engineering*, 15(12):1596–1614, December 1989.

[39] Bev Littlewood and Andrey Povyakalo. Conservative bounds for the pfd of a 1-out-of-2 software-based system based on an assessor's subjective probability of "not worse than independence". *IEEE Transactions on Software Engineering*, 39(12):1641–1653, 2013.

[40] Bev Littlewood and Andrey Povyakalo. Conservative reasoning about the probability of failure on demand of a 1-out-of-2 software-based system in which one channel is "possibly perfect". *IEEE Transactions on Software Engineering*, 39(11):1521–1530, 2013.

[41] Bev Littlewood and John Rushby. Reasoning about the reliability of diverse two-channel systems in which one channel is "possibly perfect". *IEEE Transactions on Software Engineering*, 38(5):1178–1194, September/October 2012.

[42] Bev Littlewood and David 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.

[43] Ayhan Mehmed, Sasikumar Punnekkat, Wilfried Steiner, Giacomo Spampinato, and Martin Lettner. Improving dependability of vision-based advanced driver assistance systems using navigation data and checkpoint recognition. In SafeComp *2015: Proceedings of the 34th International Conference on Computer Safety, Reliability, and Security*, Volume 9337 of Springer-Verlag *Lecture Notes in Computer Science*, pages 59–73, Delft, The Netherlands, September 2015.

[44] Charles Perrow. *Normal Accidents: Living with High Risk Technologies*. Basic Books, New York, NY, 1984.

[45] John L. Pollock. *Cognitive Carpentry: A Blueprint for How to Build a Person*. MIT Press, 1995.

[46] Felix Redmill. ALARP explored. Technical Report CS-TR-1197, Department of Computing Science, University of Newcastle upon Tyne, UK, March 2010.

[47] *DO-178C: Software Considerations in Airborne Systems and Equipment Certification*. Requirements and Technical Concepts for Aviation (RTCA), Washington, DC, December 2011.

[48] John Rushby. Using model checking to help discover mode confusions and other automation surprises. *Reliability Engineering and System Safety*, 75(2):167–177, February 2002.

[49] John Rushby. Runtime certification. In Martin Leucker, editor, *Eighth Workshop on Runtime Verification: RV08*, Volume 5289 of Springer-Verlag *Lecture Notes in Computer Science*, pages 21–35, Budapest, Hungary, April 2008.

[50] John Rushby. New challenges in certification for aircraft software. In Sanjoy Baruah and Sebastian Fischmeister, editors, *Proceedings of the Ninth ACM International Conference On Embedded Software: EMSOFT*, pages 211–218, Taipei, Taiwan, 2011.

[51] John Rushby. The interpretation and evaluation of assurance cases. Technical Report SRI-CSL-15-01, Computer Science Laboratory, SRI International, Menlo Park, CA, July 2015. Available at http://www.csl.sri.com/users/rushby/papers/sri-csl-15-1-assurance-cases.pdf.

[52] John Rushby. On the interpretation of assurance case arguments. In *New Frontiers in Artificial Intelligence: JSAI-isAI 2015 Workshops, LENLS, JURISIN, AAA, HAT-MASH, TSDAA, ASD-HR, and SKL, Revised Selected Papers*, Volume 10091 of Springer-Verlag *Lecture Notes in Artificial Intelligence*, pages 331–347, Kanagawa, Japan, November 2015.

[53] John Rushby. The indefeasibility criterion for assurance cases. In *Shonan Workshop on Implicit and Explicit Semantics Integration in Proof Based Developments of Discrete Systems*, Kanagawa, Japan, November 2016. Postproceedings to be published in Springer LNCS.

[54] John Rushby. Trustworthy self-integrating systems. In Nikolaj Bjørner, Sanjiva Prasad, and Laxmi Parida, editors, *12th International Conference on Distributed Computing and Internet Technology, ICDCIT 2016*, Volume 9581 of Springer-Verlag *Lecture Notes in Computer Science*, pages 19–29, Bhubaneswar, India, January 2016.

[55] SafeComp *2013: Proceedings of the 32nd International Conference on Computer Safety, Reliability, and Security*, Volume 8153 of Springer-Verlag *Lecture Notes in Computer Science*, Toulouse, France, September 2013.

[56] Lui Sha. Using simplicity to control complexity. *IEEE Software*, 18(4):20–28, July 2001.

[57] Tomoji Shogenji. The degree of epistemic justification and the conjunction fallacy. *Synthese*, 184(1):29–48, January 2012.

[58] *Aerospace Recommended Practice (ARP) 4754A: Certification Considerations for Highly-Integrated or Complex Aircraft Systems*. Society of Automotive Engineers, December 2010. Also issued as EUROCAE ED-79.

[59] Wilfried Steiner, Ayhan Mehmed, and Sasikumar Punnekkat. Improving intelligent vehicle dependability by means of infrastructure-induced tests. In *Dependable Systems and Networks Workshops (DSN-W), 2015 IEEE International Conference*, pages 147–152, 2015.

[60] Lorenzo Strigini and Andrey Povyakalo. Software fault-freeness and reliability predictions. In SafeComp [55], pages 106–117.

[61] Toshinori Takai and Hiroyuki Kido. A supplemental notation of GSN to deal with changes of assurance cases. In *4th International Workshop on Open Systems Dependability (WOSD)*, pages 461–466, IEEE International Symposium on Software Reliability Engineering Workshops, Naples, Italy, November 2014.

[62] Katya Tentori, Vincenzo Crupi, Nicolao Bonini, and Daniel Osherson. Comparison of confirmation measures. *Cognition*, 103:107–119, 2007.

[63] Ashish Tiwari, Bruno Dutertre, Dejan Jovanović, Thomas de Candia, Patrick D. Lincoln, John Rushby, Dorsa Sadigh, and Sanjit Seshia. Safety envelope for security. In *Proceedings of the 3rd International Conference on High Confidence Networked Systems (HiCoNS)*, pages 85–94, Association for Computing Machinery, Berlin, Germany, April 2014.

[64] Mario Trapp and Daniel Schneider. Safety assurance of open adaptive systems—a survey. In Nelly Bencomo, Robert France, Betty H.C. Cheng, and Uwe Assmann, editors, *Models@Run.Time: Foundations, Applications, and Roadmaps*, volume 8378 of *Lecture Notes in Computer Science*, pages 279–318. Springer-Verlag, 2014.

[65] Matt Webster, Michael Fisher, Neil Cameron, and Mike Jump. Formal methods for the certification of autonomous unmanned aircraft systems. In SafeComp *2011: Proceedings of the 30th International Conference on Computer Safety, Reliability, and Security*, Volume 6894 of Springer-Verlag *Lecture Notes in Computer Science*, pages 228–242, Naples, Italy, September 2011.

[66] Wikipedia. Various entries, 2016.

[67] S. P. Wilson, T. P. Kelly, and J. A. McDermid. Safety case development: Current practice, future prospects. In Roger Shaw, editor, *Safety and Reliability of Software Based Systems (Twelfth Annual CSR Workshop)*, pages 135–156, Bruges, Belgium, September 1995.

[68] Xuejun Yang, Yang Chen, Eric Eide, and John Regehr. Finding and understanding bugs in C compilers. In *Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pages 283–294, San Jose, CA, June 2011.

[69] Xingyu Zhao, Bev Littlewood, Andrey Povyakalo, Lorenzo Strigini, and David Wright. Modeling the probability of failure on demand (pfd) of a 1-out-of-2 system in which one channel is quasi-perfect. In *Reliability Engineering and System Safety*. pages 230–245, February 2017.