# Eliminative Argumentation for Arguing System Safety - A Practitioner's Experience

Simon Diemert
Critical Systems Labs Inc.
Vancouver, Canada
simon.diemert@cslabs.com

Jeff Joyce
Critical Systems Labs Inc.
Vancouver, Canada
jeff.joyce@cslabs.com

***Abstract* — Safety cases are an essential artifact for establishing the safety of complex systems. Industrial use of safety cases varies between industries. Due to inconsistent regulatory guidance, numerous different strategies, notations, and techniques have been developed for safety case construction. Eliminative Argumentation (EA) has been proposed as a technique to systematically improve confidence in a safety case via 'defeasible reasoning' wherein reasons to doubt safety claims are introduced and subsequently eliminated. Elimination of doubt results in increased confidence. This paper reports on the application of EA to seven different software-intensive systems in the automotive, rail, and industrial control industries. Our experiences suggest that EA's doubt-driven approach to safety argumentation increases confidence in a safety case and can be used to support activities such as independent safety assessments and safety verification and validation. From our experiences we synthesized into a set of lessons learned.**

***Keywords—complex systems, safety assurance, safety cases, eliminative argumentation***

## I. Introduction

Establishing the safety of complex systems is a significant challenge. Such systems are comprised of many elements, each with their own non-trivial behaviour. Since safety (defined for the purpose of this paper as "absence of unreasonable risk" [1]) is a system property, ensuring safe system behaviour requires understanding how each element contributes to the functionality of the system as a whole. In many projects, many individuals are involved with creation of a system. As a result, the rationale for the safety of a system is often distributed among subject matter experts. The problems associated with understanding system safety are heightened in software-intensive systems that contain complex functionality and rely on non-trivial mechanisms to achieve safe operation.

Safety cases are an accepted means of capturing the rationale for a system's safety. A safety case is "a clear, comprehensive and defensible argument, supported by evidence, that an item is free from unreasonable risk when operated in an intended context" [1]. Some safety standards such as ISO 26262 and EN 51026 require the creation of a safety case; however, they offer varying levels of guidance on the content of this critical artifact. ISO 26262 emphasizes the importance of sound relationship between the argument and supporting evidence: "an argument without supporting evidence is unfounded, and therefore unconvincing. Evidence without an argument is unexplained, resulting in a lack of clarity as to how the safety objectives have been satisfied" [1]. We emphasize the second point: a safety case must be more than a "box of test results". In our experience, lack of structure contributes to conceptual gaps in verification and validation activities. However, ISO 26262 does not provide additional guidance and the precise content of the safety case is left open to interpretation.

EN 51026 prescribes the content of a safety case as a collection of reports and analyses of various failure/fault conditions, i.e., it is heavily focused on evidence without an argument [2]. Other standards, such as IEC 61508 for industrial control and DO-178C for airborne software do not require the explicit creation of a safety case [3, 4]. These standards collectively rely an "implicit safety case" because they do not directly argue the safety of a product itself but instead depend on the implication that a rigorous process results in a safe product [5].

In the absence of authoritative and uniform regulatory guidance for creating safety cases, different conventions and techniques have been developed by academia and industry. Goal Structured Notation (GSN) was created by Tim Kelly and is the most widely recognized notation for describing safety arguments [6]; standards prescribing the syntax and semantics of GSN have been created in an effort to achieve uniformity across industries [7]. Other techniques for describing safety cases include "Claims-Argument-Evidence" (CAE) notation [8], and structured textual narrative. We have worked with some of these techniques in the past and have found them satisfactory.

Eliminative Argumentation has been proposed as an additional technique for the creation of safety cases [9]. The goal of EA is to systematically increase confidence in a safety case via the use of a defeasible reasoning. We have found that defeasible reasoning has changed the way we approach safety argumentation and results in a better understanding of the system under investigation. The main contribution of this paper is a set of "lessons learned" from the application of EA to several real-world safety-critical systems. The remainder of this paper is structured as follows. Section II provides a short introduction to EA. Section III summarizes our experience applying EA to real-world systems. Section IV provides the set of "lessons learned". Section V makes concluding remarks.

## II. Overview of Elminative Argumentaiton

Eliminative Argumentation (EA) was introduced by Goodenough *et al.* as an adaption of Toulmin's notation [9, 10, 11]. EA provides an abstract framework for constructing an argument and assessing confidence in the argument based on the notion of defeasible reasoning where in claims are recursively

challenged. As reasons to doubt a claim are eliminated confidence in the claim increases. This section provides motivation for the use of EA, introduces the key concepts of technique, and provides a demonstration of EA on a toy chemical reactor system.

### A. The Role of Doubt in Safety Argumentation

Safety argumentation is invariably impacted by confirmation biases that arise when authors aim to directly "prove" a system is safe. A notable example of confirmation bias related to safety argumentation was revealed as part of a review following the fatal crash of the Nimrod, a UK military aircraft, in Afghanistan in 2006. In the investigation, it was found that:

> *"the Nimrod Safety Case [was] fatally undermined by an assumption by all the organisations and individuals involved that the Nimrod was 'safe anyway', because the Nimrod fleet had successfully flown for 30 years, and they were merely documenting something which they already knew. … The Nimrod Safety Case became essentially a paperwork and 'tick-box' exercise" [12].*

Typically, a safety case starts with a top-level safety claim and then recursively decomposes that claim into sub-claims which are eventually supported by evidence. Conventional techniques do not (at least as part of the standard notation) leave room for expression of doubt. As a result, authors of a safety case are not prompted to question their claims or validity of evidence. The task of creating a safety case is then reduced to making the minimum set of claims required to prove the top-level claim rather than seeking to demonstrate an acceptable level of residual risk.

However, in practice, engineers have many reasons to doubt the safety of a system. For example, in software engineering, there is a strong culture of doubting whether software is in fact defect free. This is so prevalent that the creation of defect free software is regarded as axiomatically impossible among software engineers. One would be hard pressed to find a software engineer who will claim the software they create is defect free. Doubting oneself and subsequently eliminating those doubts with further claims and evidence is central to the scientific and engineering approach to problem solving. The methods of safety argumentation should take advantage of this fact. There should be an explicit means to capture, express, and analyze doubts that engineers have about the systems they create. Only after the residual risk associated with the doubts is understood can a informed conclusion about the safety of the system be formulated.

Of course, enumeration of doubt is not, on its own, sufficient to defeat confirmation bias. Enumeration of doubt only shifts the question from "do the sub-claims completely support the top-level claim?" to "have all of the sources of doubt been identified?" However, doubt can also be expressed about the completeness of the doubts themselves which in turn necessitates further argument about the completeness. For example, one might doubt the completeness of a set of failure modes derived for a component in the system based on a Failure Modes and Effects Analysis (FMEA). An argument countering this doubt might claim that a combination of experienced persons and systematic methodology provide confidence in the completeness of the failure modes. Regardless, a residual doubt exists that maybe a failure mode was overlooked. This residual doubt should be captured and communicated to stakeholders as a risk associated with the design process used to create the component.

At this point, it should be noted that our concerns are not with individual safety cases (or their authors). It is not meant to imply that all safety cases developed by conventional methods are deeply flawed. Indeed, there are exceptional safety cases developed using conventional techniques that convincingly argue the safety of complex systems. The main concern here is the ability of engineers to express, analyze, and subsequently refute their natural doubts when using a conventional technique/notation and the role of this mode of thought in addressing confirmation bias.

EA addresses the question confirmation biases by including the notion of doubt as a first-class citizen. The EA notation calls these doubts "defeaters" in the sense that they are used to defeat aspects of an argument. There are three types of defeaters [9]:

- **Rebutting Defeaters** (RD) express doubt about a claim. For example, one might claim "*the valve will open when commanded"* which could be rebutted by *"unless the valve is stuck closed"*.
- **Undermining Defeaters** (UM) express doubt about evidence provided in support of some claim. For example, in a software project, one might provide evidence of laboratory test results to support the claim that a requirement as been satisfied. This evidence might be undermined by a defeater: "*unless the laboratory environment is not the same as the operational environment*".
- **Undercutting Defeaters** (UC) express doubt an inference rule used to combine multiple aspects of an argument. For example, an argument that adopts the strategy to argue that all hazards have been adequately mitigated has an (implicit) inference rule that *"if all failure modes have been adequately mitigated, then no hazard can occur"*. The undercutting defeater doubts the inference rule by questioning its premise: *"unless there was a failure mode that was not identified"*.

To illustrate the above defeaters, consider a fictious chemical reactor, adapted from the example in [13]. A chemical reaction is carried out in a reaction vessel. The reaction temperature is managed by adding water from an external water reservoir. The flow of water is controlled by a control computer that actuates a water valve. The temperature of the reaction is measured by a sensor that provides feedback to the control computer. The control computer uses a sophisticated control algorithm to determine how much water to provide to the reactor. Figure 1 contains a sample EA-based safety argument that starts with the claim that the control system provides enough cooling to prevent the reactor from overheating (assuming overheating is a hazardous event). Note that this top-level claim might a sub-claim of a larger argument.

This example is not intended to be a complete argument, instead it demonstrates the key elements of the EA technique, namely: 1) enumeration of doubt, 2) elimination of doubt, 3) explicit description of inference rules, and 4) acceptance of residual doubt. We adopt a similar syntax and semantics to those defined by Goodenough *et al.* with only minor variation:

- **Claims** (C) are statements that must be supported further argumentation to demonstrate their validity.
- **Evidence** (E) describe observations, data, or artifacts that support claims.
- **Strategies** (S) describe the approach used to organize a collection claims or defeaters. Strategies are "top-down" in the sense that capture an overall approach to supporting a claim.
- **Inference Rules** (IR) describe how to logically combine a collection of one or more claims/defeaters to support a parent claim. Inference Rules are "bottom-up" and complement Strategies. Inference Rules are often obvious in an argument and do not need to be expressly stated every time. However, when non-obvious logic is employed or there is a reason to doubt the validity of an inference rule, then inference rules are explicitly included in the argument.
- **Context** (X) provide additional information, that is not required to create a sound argument but is helpful for orienting the reader. Context is not used in Figure 1.
- **Assumptions** (A) are explicit statements that are assumed to be true as part of an argument. If an assumption is not satisfied, then the argument would become invalid. Assumptions are not used in Figure 1.
- **Terminators** denote the end of a line of reasoning as either: complete (OK), a source of residual doubt (Res), or undeveloped (shown as a diamond).

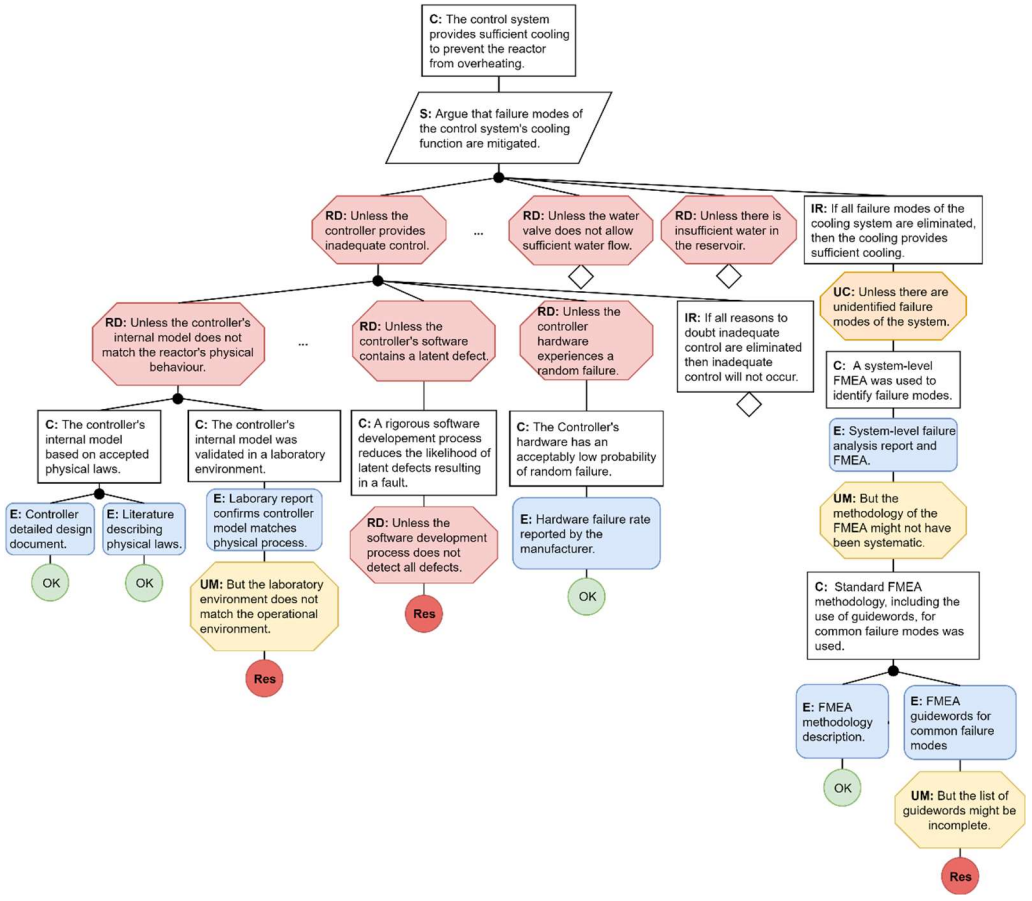


*Figure 1 - Sample argument for a chemical reactor.*

### *B. Assessing Confidence in an Argument*

Assessing confidence in a claim or evidence remains a concern [14]. How can a reviewer be confident that safety case adequately argues its stated claims? Enumeration and elimination of doubt might reduce confirmation bias, but to what extent does it do so? This quickly becomes a problem when developing a safety case for a complex system that combines multiple sub-arguments for different system elements and relies on a combination of qualitative and quantitative evidence.

A doubt-driven EA safety case may supplemented with numerical data in an attempt to quantify confidence. To this end, Goodenough *et al.* propose two methods [9]. In their first

method, probabilities are assigned to doubts in an argument and combined similarly to combining probabilities in a fault-tree analysis. This is also similar to other quantitative approaches for confidence assessment such as Bayesian Belief Networks (BBNs) [15]. However, a probabilistic approach suffers from concerns related to the choice of numerical values which are known to produce inconsistent results due to variability in judgement on the part of analysts [16]. To address this limitation, Goodenough *et al.* suggest using a qualitative scale and translating the selected categories into numerical values based on fuzzy logic, but this idea has not been developed [9].

Goodenough *et al.*'s second method is a semi-quantitative approach based on the notion of a Baconian confidence [11]. In this approach, the status of a defeater is either "fully eliminated" (i.e., no longer a concern) or "residual". Residual defeaters are counted and compared to the total number of defeaters in the argument to obtain a confidence level expressed as "X | Y" where X is the number of unaddressed defeaters (residual doubts) and Y is the total number of defeaters. This relatively simple technique is based on the premise that confidence in an argument increases as doubts in the argument is eliminated.

## III. Experiences Developing Safety Cases

This section describes our experience developing seven safety cases using EA. Each safety case is discussed in detail below. Discussion is limited to general commentary to avoid disclosure of private and proprietary information. In

Table 1 the number of nodes in the safety case is reported by EA node type. The total number of nodes is computed as the sum of all node types except for Residual Doubts ("Res."). The Residual Doubts are a subset of the Defeaters ("RD", "UM", "UC"); therefore, the Residual Doubts are excluded from the sum to avoid double counting.

### A. Safety Cases 1, 2, 3 - Embedded Automotive Software

Safety cases 1, 2, and 3 were all created for low-level embedded software products used in automotive systems. These safety cases have a broad scope that spans all phases of the software development lifecycle as defined by ISO 26262. The safety cases covered the definition of the software requirements, software architecture, detailed design, implementation, unit verification, and integration testing on target hardware. The low level of abstraction coupled with the complexity of software is responsible for their relatively large size.

Safety case 3 was created for a notably simpler embedded software product, hence the difference in size from safety cases 1 and 2. However, it is worth noting that the distribution of node types relative to the total number of nodes remains consistent between safety cases 1, 2, and 3.

Safety Cases 1, 2, and 3 cases were reviewed by a reputable independent authority and accepted as part of an ISO 26262 compliance audit for their respective products.

### B. Safety Cases 4, 5, and 6 – ADAS Sensor Function

Safety case 4 was created for an automotive Advanced Driver Assistance System (ADAS) sensor system. The safety case was created at the functional ("black box") level of abstraction and aimed to demonstrate the safety of the intended functionality of the product, particularly in relation to the behaviour of the external environment (e.g., weather conditions). In this project, an iterative EA-based approach was used refine the functional requirements of the system to better address concerns related to the external environment.

Safety cases 5 and 6 were for a signal processing function for use in a rail application. These safety cases were created to demonstrate the safety of a signal processing algorithm in a specific operational context. The emphasis of these two safety cases was on identifying validation activities that would increase confidence in the algorithm's behaviour given a wide range of inputs. The use of an iterative EA-based approach helped structure and focus the overall safety validation effort. Additionally, safety case 6 subsumes safety case 5, i.e., the algorithm related to safety case 5 is used as part of a more complex algorithm for safety case 6. Safety cases 5 and 6 were developed in the context of a broader EN 5012x compliance effort that focused on the product(s) that would incorporate the signal processing algorithms.

The smaller size of safety cases 4, 5, 6 is attributed to the higher level of abstraction used to create this safety case. If the arguments had been further developed to encompass the details of hardware and/or software development, it is anticipated that these safety cases would have grown much larger.

*Table 1 – Summary of safety cases developed using EA; includes number of each EA node type used for each argument.*

| | | | **Number of Each EA Node Type Used** | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **#** | **Application Type** | **Industry** | **C** | **E** | **RD** | **UM** | **UC** | **IR** | **A** | **X** | **S** | **Res.** | **Total** |
| 1 | Embedded Software | Automotive | 184 | 147 | 142 | - | 1 | 10 | 10 | 17 | 2 | 19 | **513** |
| 2 | Embedded Software | Automotive | 188 | 144 | 117 | 11 | 2 | 2 | 6 | 13 | 1 | 28 | **484** |
| 3 | Embedded Software | Automotive | 94 | 81 | 79 | - | - | - | - | 3 | - | 7 | **257** |
| 4 | Embedded Software | Automotive | 63 | 19 | 62 | - | - | - | 2 | - | 3 | 25 | **149** |
| 5 | Signal Processing Algorithm | Rail | 33 | 21 | 30 | - | 3 | 3 | - | 1 | 4 | - | **95** |
| 6 | Signal Processing Algorithm | Rail | 23 | 4 | - | - | 3 | 3 | - | 3 | 4 | - | **40** |
| 7 | Monitor & Interlock System | Industrial Control | 7 | 1 | 5 | - | - | - | - | 1 | - | - | **14** |

### C. Safety Case 7 – Industrial Interlock System

Safety case 7 is for a predictive safety monitor and interlock system used in an industrial application and is being developed as part of an IEC 61508 compliance effort. The system in question uses software to predict the behaviour of a physical process and engages an interlock if/when the process under control violates predetermined thresholds. The safety case's scope spans the entire monitor and interlocking system from top-level functional behaviours through to detailed hardware and software development. A key challenge associated with safety case 7 is demonstrating that the predictive functions of the monitoring system accurately predict the true behaviour of the process under control; early use of EA has been helpful in structuring the overall safety strategy in this regard.

This safety case is still in early development and is evolving with the development of the system. The number of nodes reported in

Table 1 represent the top two layers of the EA argument tree (i.e., root node and direct children) and are related to hazards and safety goals (system-level safety requirements). Safety case 7 will undergo independent review as part of the industrial system's overall certification effort.

### D. Remarks on Argument Size

Based on

Table 1, the safety cases discussed above are of varying size with the largest having 513 nodes and the smallest (excluding safety case 7 which is currently under development) having 40 nodes. We emphasize that the total number of nodes is not an absolute indicator of completeness or quality. In our experience, argument size (measured as number of EA nodes) depends on the system size, system complexity, and chosen level of abstraction (e.g., functional, software, hardware, etc.).

## IV. Lessons Learned

While developing the EA-based safety cases described above, we have made many observations about both the EA technique and safety case development in general. We have distilled these into the "lessons learned" provided below.

### A. Lesson 1: EA is easy to learn

We have had success introducing EA to both technical and non-technical individuals. The overall pattern of thought (claim, defeater, claim, defeater, … ) is a natural style that many people are already comfortable using. In one particular case, we lead an engineering meeting to introduce EA to a new project; we anticipated hesitant participation from the engineers in the room. However, in a matter of minutes, the participants began enthusiastically constructing an argument for their system with minimal guidance.

We have found that, especially when introducing EA to a new group, it is easier to use a simplified version of the notation that permits only claims, evidence, and a single generalized defeater. This reduces the notational barrier to adopting EA and helps focus on the primary purpose of the technique, namely, enumerating and eliminating doubt. More sophisticated notations can be applied later once the overall argument structure has established.

While we believe that EA is easy to learn for beginners, we also know from experience that mastery of the technique is challenging. Many aspects of EA require a nuanced perspective that comes with practice. For example, EA's Interference Rule (IR) node type allows the user the express a specific (possibly non-obvious) rule for combining multiple claims in support of a parent claim. When coupled with an Undercutting defeater (UC), this becomes a powerful means of eliminating doubt related to the soundness of the safety argument. However, overuse of this reasoning pattern becomes cumbersome and difficult to understand for a reviewer. Therefore, thoughtful use of this pattern is required. We have only begun to appreciate the usefulness of the more subtle aspects of EA.

### B. Lesson 2: EA supports independent review

We have found that the doubt-driven approach of EA helps engineering teams prepare for independent safety assessment/audit (e.g., an ISO 26262 compliance assessment). Doubts enumerated during EA-based safety case development often correspond to questions that a reviewer might pose to the engineering team. The systematic elimination of these doubts in the safety case means that there is often a thoughtful answer prepared in advance for questions from the reviewer. Moreover, since EA demands evidence be presented as part of elimination of doubt, the required documents/artifacts are prepared and organized in advanced. Having this information consolidated within the safety case results in a smoother review process and avoids a last minute "scramble" for answers.

On one occasion, an independent reviewing authority remarked that the EA-based safety case useful for structuring their thinking about the safety of the system under review. This demonstrates the value of a structured approach to safety case development, which is consistent with the overall value proposition of structured safety argumentation [6].

### C. Lesson 3: The importance of doubt

We have found the explicit permission to express doubt as defeaters is very helpful in communicating concerns and limitations of a system. Expression of doubt often leads to productive conflict amongst team members where one party expresses a doubt and the other refutes it. In our experience, such conflict (when appropriately managed) leads to a better understanding of the system and ultimately a stronger safety case. Of particular importance are residual doubts resulting from un-eliminated defeaters. The existence of residual doubt is not a "bad" outcome of a safety case. As discussed above, these are the basis for assessing the residual risk associated with a system. All safety-critical systems have some residual risk, the use of EA simply highlights this risk. Adopting EA as part of a safety engineering process necessitates a review of residual doubts by stakeholders as part of acceptance of the safety case.

While we view doubt as an essential part of EA, as a practical matter, it is important to know when to stop. Indeed, one could doubt themselves *ad infinitum*. While we cannot

offer specific criteria for when "enough is enough", we have found that a combination of a rigorously defined scope (e.g., application level software, not supporting software or hardware) in combination with reasonable engineering judgement to be effective.

In the end, safety case development is "messy" work. The literature leaves the impression that safety cases should be near-perfect tree structures that argue the safety of a system using a handful of non-specific claims. However, in our experience, reality rarely conforms to this idealized notion of a safety case. Complex systems demand complex and nuanced safety arguments that inevitably produce residual doubt. We would be suspicious of a safety case that was free of residual doubt.

### D. Lesson 4: Develop early, revise often

For safety-critical systems, safety influences all phases of the system development lifecycle, from early concept development through to deployment and maintenance. Such thinking is adopted by prominent functional safety standards such as ISO 26262 that intentionally "weave" safety engineering activities into conventional V lifecycle model for system development. As a result, the reason(s) a system is safe are often determined in the early phases of product development. That is, engineering teams begin creating an implicit safety case almost immediately. We have found it instructive to use EA to capture these preliminary safety arguments and have observed two main benefits of this practice. First, clearly articulating (in a structured format) the reason(s) that is a system is safe is productive exercise early in a project to ensure a team has a common understanding of the problem at hand. Second, expressing doubts about a design early in development highlights weakness early and can be used to focus subsequent design efforts.

However, we have also observed that defining the top-level claims of an EA-based safety case is the most conceptually challenging phase of safety case creation. Top-level claims make very broad assertions about the behaviour of the system in question and must be articulated carefully so as to not inadvertently extend the scope of the safety case beyond what can be demonstrated by evidence. Comparatively, lower-level argumentation tends to be more procedural often relying on well-established engineering processes/principles to support specific claims.

One of the primary principles of EA is "defeasible reasoning" wherein one repeatedly refines the argument based on doubt. Our experience supports this principle: safety cases should be developed early and revised often.

### E. Lesson 5: Support system verficiation and validation

As part of a "*develop early, revise often*" approach (Lesson 5), we have found that EA helps to identify, in advance, the types of verification and validation activities required to support safety claims. From a technical perspective, this provides a top-level view of how many different verification and validation activities fit together to form a cohesive safety argument. From project management perspective, this provides an input to planning verification and validation activities and identifying risk associated with each activity.

Additionally, doubt-driven thinking helps focus on meaningful verification and validation activities by permitting the expression of doubt about the verification and validation activities themselves. This is in contrast to following a standard engineering process that might (or might not) generate convincing evidence in support of safety claims. We have found this particularly helpful in the context of novel development where standard processes/techniques are insufficient and new techniques used to fill gaps. As part of EA-based safety case development, we often maintain a "shopping list" of evidence required to support the argument. This forms an interface between various stakeholders in the project (management, design team, quality assurance, etc.) and can be used as an input to verification and validation planning activities.

### F. Lesson 6: More than a fault tree analysis

On first appearance, an EA safety case resembles a Fault Tree Analysis (FTA). This is one of the first reactions we often receive when introducing EA to a new project. Indeed, there are many similarities between EA and FTA. Both EA and FTA are tree structures, both are deductive (top-down) styles of reasoning, and both permit the expression of negative events/concepts (defeaters in the case of EA, faults in the case of FTA). However, there are key differences between these two techniques.

First, the top-level node in FTA is usually a "bad event", e.g., "the reactor overheats". Conversely, EA (like all safety cases) starts with a top-level claim/goal that is positive, e.g., "the reactor will not overheat". This difference shifts the intent of the supporting sub-tree from finding causes of a "bad event" to defeasible reasoning on the top-level claim. Second, FTA focuses on components in a system and describing how the failure of the components, in conjunction with other failures, might result in the top-level event occurring. That is, FTA focuses on a specific type of defeater related to component failure. EA defeaters consider a wider set of doubts that indeed covers component failure but also include non-fault/failure functional behaviours, engineering processes, soundness of claims, and completeness of evidence. Due to these differences, we view FTA and complementary to EA. FTA is a specific analysis technique for generating evidence about failure scenarios that can be used to support specific claims in a safety argument regarding the tolerance of the system to faults and failures.

### G. Lesson 7: Confidence assessment remains challenging

One of the value propositions of EA is increased confidence in safety based on the idea of enumerating and eliminating doubts about claims. We agree that this, in combination with consideration of residual doubt, qualitatively increases confidence. However, we have yet to see value in the Baconian or probabilistic confidence measures proposed by Goodenough *et al.* For both measures, our criticism is similar to Graydon's [16, 17]. First, Baconian confidence is blind to relative importance of residual doubts and assessment of confidence still requires expert judgement to resolve/accept the associated residual risk. Second, measures of probabilistic confidence are strongly influenced by the choice of numerical values, a task that is challenging

for all but the simplest types of failures (e.g., how does one accurately quantify the probability of a software defect occurring?). Regardless, we think that assessing confidence in an argument is an important pursuit and merits further work.

## V. CONSLUSIONS

This paper has motivated and introduced the essential concepts of Eliminative Argumentation (EA) as a technique for the development of safety cases. This technique improves confidence in a safety case by allowing the creators of a safety case to enumerate and eliminate doubt in a safety claims in a structured manner. We have recounted our experience developing seven EA-based safety cases and offered seven "lessons learned" based on our experience. We view safety cases as an important part of a mature safety engineering processes and we have found that EA is a good technique for the development of such arguments. While we have observed many benefits of EA, we do not mean to imply that this technique should be used as a sole means of safety analysis for a safety-critical system; the opposite is in fact true. EA, like any safety case, should be used to structure and complement established engineering practices to improve overall system development outcomes.

## REFERENCES


[1] International Organization for Standardization, "ISO 26262 -- Road Vehicles -- Functional Safety," 2011.

[2] European Committee for Electrotechnical Standardization, "EN50126 - Railway Applications - The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS)," 2017.

[3] International Electrotechnical Commision, "IEC 61508 - Functional safety of electrical/electronic/programmable electronic safety-related systems," International Electrotechnical Commision, 2010.

[4] Radio Technical Commision for Aeronautics, "DO-178C - Software Considerations in Airborne Systems and Equipment Certification," 2011.

[5] J. Birch and et al., "Safety Cases and Their Role in ISO 26262 Functional Safety Assessment," in *Computer Safety, Reliability, and Security*, Toulouse, 2013.

[6] T. P. Kelly, "Arguing safety - A Systematic Approach to Safety Case Management," University of York, 1998.

[7] Safety-Critical Systems Club, "SCSC-141B - Goal Structuring Notation Community Standard - Version 2," 2018.

[8] L. Emmet and G. Cleland, "Graphical notations, narratives and persuasion: a Pliant Systems approach to Hypertext Tool Design," in *Proceedings of the 13th ACM conference on Hypertext and hypermedia*, College Park, Maryland, USA, 2002.

[9] J. B. Goodenough, C. B. Weinstock and A. Z. Klein, "Eliminative Argumentation: A Basis for Arguing Confidence in System Properties," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, 2015.

[10] J. B. Goodenough, C. B. Weinstock and A. Z. Klein, "Toward a Theory of Assurance Case Confidence," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, USA, 2012.

[11] J. B. Goodenough, C. B. Weinstock and A. Z. Klein, "Eliminative induction: A basis for arguing system confidence," in *Proceedings of the 2013 International Conference on Software Engineering*, San Francisco, 2013.

[12] C. Haddon-Cav, "The Nimrod Review," London Stationary Office, London, UK, 2009.

[13] J. Thomas, *STAMP/STPA Intermediate Tutorial - Guided Exercise: Applying STPA to a real system.*

[14] A. Wassyng, T. Maibaum, M. Lawford, H. Bherer, R. Calinescu and E. Jackson, "Software Certification: Is There a Case against Safety Cases?," in *16th Monterey Workshop*, Redmond, WA, USA, 2011.

[15] C. Hobbs, M. Llyod, C. Dale and T. Anderson, "The Application of Bayesian Belief Networks to Assurance Case Preparation," in *Achieving Systems Safety*, London, UK, 2012.

[16] P. J. Graydon and M. C. Holloway, "An investigation of proposed techniques for quantifying confidence in assurance arguments," *Safety Science,* vol. 92, pp. 53-65, 2017.

[17] P. J. Graydon, "Defining Baconian Probability for Use in Assurance Argumentation," NASA, 2016.