Credible Autonomy Safety Argumentation

Philip Koopman, Aaron Kane, Jen Black

Carnegie Mellon University, Edge Case Research

Pittsburgh, PA, USA

Abstract *A significant challenge to deploying mission- and safety-critical autonomous systems is the difficulty of creating a credible assurance argument. This paper collects lessons learned from having observed both credible and faulty assurance argumentation attempts, with a primary emphasis on autonomous ground vehicle safety cases. Various common argumentation approaches are described, including conformance to a non-autonomy safety standard, proven in use, field testing, simulation, and formal verification. Of particular note are argumentation faults and anti-patterns that have shown up in numerous safety cases that we have encountered. These observations can help both designers and auditors detect common mistakes in safety argumentation for autonomous systems.*

1 Introduction

Ensuring that an autonomous vehicle will behave safely on public roads comes with a set of unique challenges. Established safety standards cover some, but not all aspects of vehicle operation. The use of machine learning technology generally results in aspects of the system for which there is no traditional design, making use of traditional V-model based safety methods problematic. Moreover, brute force testing approaches struggle to cope with ensuring the low failure rates required for critical applications.

Even though safety practices are still evolving to address the unique issues presented by autonomy, such systems are already being built and deployed on public roads. Autonomous systems are also poised for deployment in public airspace (e.g. drones), medical applications (e.g. robotic surgery), and other critical domains (e.g. cargo ships). Because of the pervasive secrecy of the autonomous vehicle market, it is difficult to say how robust the safety cases for such vehicles actually are – or even if a credible safety case has been created at all for any particular vehicle that is being deployed.

As the technology matures it will be imperative to establish standardized approaches to ensuring the safety of such systems. This paper proposes a set of patterns and anti-patterns for autonomous ground vehicle safety arguments that might

PREPRINT: 27th Safety-Critical Systems Symposium, Feb. 2019.

be used within an autonomous system safety case. While our emphasis is on autonomous ground vehicles, we expect a useful degree of overlap with autonomous system safety for other domains. This is intended as a starting point that should evolve over time as experienced is gained with how autonomous systems and autonomy safety arguments fail in practice.

1.1 Heterogeneous safety arguments

A safety case is "*a structured argument, supported by a body of evidence that provides a compelling, comprehensible and valid case that a system is safe for a given application in a given operating environment*" (UK Ministry of Defence 2017, page 26). For the purposes of this paper we assume that an autonomous ground vehicle designer wishes to produce a credible safety case, and in doing so needs to select argumentation strategies.

In practice, there are many ways to attempt to argue safety, including approaches such as: following a prescribed set of engineering practices, brute force testing, and claiming that something has been "proven in use." Whatever the strategy, it is important to use a methodical approach to documenting the arguments and evidence to ensure that the safety case is valid (Kelly 1998).

A common strategy is to argue that a system is sufficiently safe because an accepted applicable safety standard has been followed, implicitly adopting the underlying safety argument strategy inherent in the standard. However, because both the technology and safety strategies for autonomous systems are still evolving, there is no one-size-fits-all safety standard currently published. Therefore, it seems likely that a more explicit argumentation strategy will be required for these systems.

Because autonomous vehicles are a composite of comparatively mature technology (e.g. underlying conventional vehicle control systems) and novel technology (e.g. machine learning), it seems likely that safety arguments will be heterogeneous in nature. By this, we mean that different portions of the safety argument will likely take fundamentally different argumentation approaches for different system functions and components.

Consider an autonomy kit that has been added to a conventional consumer-grade automobile. The safety of the underlying production ground vehicle systems can be – and hopefully has been – argued via conformance to ISO 26262 (ISO 2011). However, some key aspects of autonomy functions are beyond the scope of that standard. Moreover, some assumptions made in determining conformance might be violated by the addition of autonomy, such as the availability of a human driver to exercise vehicle control in the event of a malfunction.

Some additional functionality might be argued safe via conformance to a different standard, such as a draft SOTIF standard (ISO 2018). However, making a case for the safety of higher-level autonomy functions, especially perception, is likely to require additional argumentation beyond currently available standards,

potentially encompassing approaches ranging from large testing campaigns to formal proofs of correctness.

There will be economic and time-to-market pressure to re-use safety arguments from other domains that have already been set up for components incorporated into autonomous systems. For example, industrial process equipment might be repurposed for autonomous ground vehicles, and automotive equipment might be repurposed for autonomous aircraft. In the end, an autonomous vehicle system-level safety argument seems likely to be an amalgamation of different safety assurance approaches for various functions, encompassing multiple safety standards as well as various techniques for areas in which mature standards are not yet available. We call this approach a heterogeneous safety argument.

1.2 Support for assessment

While the contents of a safety case likely differs for each system being designed, there needs to be a consistent way to evaluate the sufficiency of the argumentation and supporting evidence. While ultimately there is no substitute for an experienced and capable assessor, it can be helpful to support the assessment with a foundation of rules of thumb, lessons learned, and sets of good as well as bad practices to improve assessment consistency and reduce errors of omission in the assessment process.

To that end, this paper proposes a set of safety argument approaches and anti-patterns. A number of general approaches that we have seen used in practice or proposed for use in autonomous ground vehicles are briefly described. More importantly, we discuss the typical threats to validity we see in safety cases that have attempted each argumentation pattern.

While the emphasis in this discussion is on supporting assessment, the material should be beneficial to both developers (to avoid making argument mistakes in the first place) and assessors (to detect sometimes subtle, but common fallacies in arguments). We fully expect that this list can benefit from incorporating lessons learned as the technology matures, and intend it as a starting point to be built upon. Rather than attempting to set forth a library of formalized safety argument patterns, we concentrate on the main ideas and pitfalls behind typical argumentation strategies.

1.3 Previous work

Bishop and Bloomfield (1998) define a safety case as "*a documented body of evidence that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment.*" They argue that a safety case has four main elements (a claim, evidence, argument, and inference) and that each

element can be categorized by type. For example, the argument may be deterministic, probabilistic, or qualitative; and evidence may be design, process, testing, or historic experience. They also explain that the safety case for a safety claim about a system can be decomposed into safety cases for the supporting sub-claims, implying that each sub-claim might be supported by a different type of argument.

Kelly (1998) proposed the Goal Structuring Notation (GSN) approach to documenting safety arguments. A GSN community standard defines standardized terminology (ACWG 2018). Additionally, the Object Management Group has published the Structured Assurance Case Metamodel (SACM) for representing assurance cases (OMG 2018).

Rushby (2015) discusses inductive vs. deductive arguments in assurance cases as well as the role of confidence claims. An inductive argument means that the conjunction of sub-claims strongly suggests the claim. In contrast, a deductive argumentation approach means that the conjunction of sub-claims implies (or proves) the claim. A problem with inductive reasoning steps is that "*there is no effective way to estimate the size of the gap in our reasoning.*" He recommends that inductive arguments should be made deductive by explicitly stating or factoring out assumptions. In other words, a claim that is merely implied can be converted to a claim that is proven with the addition of an explicitly stated assumption required for the relevant proof to hold true. This has the virtue of making explicit all the assumptions required for a safety case to be valid.

Goodenough et al. (2015) propose using an eliminative induction argumentation approach as a way to assess confidence in an assurance case argument based on inductive reasoning. Such an approach augments an inductive argumentation structure with sub-claims and evidence to demonstrate that various potential argumentation flaws are not present. In particular, they propose using *defeaters*, which are explicit statements of doubt regarding a particular inductive argument. They propose specific types of defeaters to attack claims (via looking for failure modes or possible counter-examples), undermine evidence (via looking for reasons the evidence might be invalid), and undercut inference rules (via looking for conditions under which the inference rule is not adequate). Pitfalls presented in this paper can be considered to be defeaters for various aspects of a safety argument.

Catapult (2017) presents GSN-notation safety cases for highway and urban pilot autonomous vehicle use cases that address road etiquette, operational envelope enforcement, and other functionality. That analysis leads to a number of observations including a potential need for roadway infrastructure support and a need to deal with residual risks after all reasonable risk mitigation approaches have been applied.

Wang et al. (2017) present a confidence assessment framework for safety arguments. They assume that the argument is valid and deal primarily with managing the confidence in evidence.

Burton et al. (2017) proposed a top level GSN-based safety case structure for autonomous vehicles, with follow up work that examines pedestrian detection (Gauerhof et al. 2018).

Graydon et al. (2012) propose publishing a catalogue of acceptable argumentation patterns. We extend this idea by proposing publishing a catalogue of commonly attempted but invalid argumentation patterns.

2 Safety argumentation strategies

We envision that safety cases for autonomous vehicles will be based upon a heterogeneous collection of safety arguments using different strategies. Assessing such safety cases could use an approach that initially consults a catalogue of strategies and accompanying argumentation patterns. That catalogue would capture previously analysed argument patterns, types of evidence that are acceptable for each argument pattern, and argumentation pitfalls observed in previous assessments.

Over time, such a catalogue could capture experience from lessons learned in assessments and field experience. The assessment process could then mature organically to one in which established and accepted patterns are well understood and documented, so there is a relatively stable baseline of expectations and assessment criteria for common patterns. New technological advances and architecture concepts would still require additions to the catalogue, and so continual growth and revision of the catalogue would need to be a part of any such approach. It is expected that any single system will require a combination of argumentation approaches. It is understood that standards can evolve slowly. Setting up such a catalogue might be done via a periodically updated assessment work aid that does not require a new parent standard document version to incorporate lessons learned.

We provide a starting point for such a process by listing argument strategies and pitfalls that we have observed in our work with autonomous ground vehicles. In analysing these patterns, we assume a common architectural approach of taking an existing production vehicle and adding an autonomy kit as an overlay. Such an overlay could completely replace a human vehicle driver (autonomous system), or require some aspects of human participation in vehicle operation (semi-autonomous system). Other approaches, such as bespoke design of complete autonomous vehicles, are likely to have many similarities, but also some differences that are beyond the scope of this paper.

The following sections describe general argumentation strategies as well as a number of anti-patterns, pitfalls, or other faults that we have observed in practice. A later section summarizes a collection of special cases and other observations beyond the primary set of strategies discussed.

2.1 Conformance to an existing standard

In this argumentation approach, a system component is constructed in accordance with a recognized, public safety standard, such as ISO 26262. Ideally, that component is independently assessed for conformance to the safety standard. Based on conformance, an argument can be made that the component safely carries out its intended function to an appropriate degree (e.g. to a specific Automotive Safety Integrity Level, or ASIL). Ensuring that conformance actually results in the desired system safety properties might require specific safety argumentation, and is subject to a number of additional pitfalls beyond those discussed herein (see Graydon et al. 2012).

This pattern can (and for the most part probably should) be used for the conventional software portions of the autonomy system. Moreover, this argumentation pattern can be used with a Doer/Checker pair (also called a monitor/actuator pair or a safety bag) to create a safety envelope around some types of autonomy functions (Koopman and Wagner 2016). The general idea is that a capable "Doer" satisfies the functional requirements of a system, potentially using difficult-to-validate technology such as machine learning. For example, the Doer might be a path planner that uses non-deterministic algorithms, heuristics or even machine-learning based approaches to find an optimal path. A "Checker" is designed with more conventional software techniques and used to enforce safety requirements. The Checker generally only checks for violations of safe operating envelopes and violations of assumptions made in the safety argument at run time. For a path planner example, the Checker simply needs to make sure that whatever plan has been selected does not intersect known obstacles. (We assume for this example that obstacle detection is a separate function that is dealt with outside the scope of the path planner safety argumentation.) Ideally the Checker has been designed to a high ASIL and is entirely responsible for ensuring safe operation of the function.

Assuming that the Checker can completely enforce safety, an appropriate safety argument could be that the Doer is not safety-relevant because the Checker (1) has a complete ability to ensure safety, and (2) has been developed to an appropriate integrity level in conformance with a functional safety standard. There are a number of variations to this approach that ultimately result in a "safety relevant" portion of the system being verified, and the remainder of the system not being verified with the same level of rigour. We use the Doer/Checker approach to illustrate issues that apply more generally to other types of systems with identified safety relevant portions.

2.1.1 Command override anti-pattern

A common pitfall when identifying safety relevant portions of a system is overlooking the safety relevance of sensors, actuators, software, or some other portion of a system. A common example is creating a system that permits the Doer to per-

form a command override of the Checker. (In other words, the designers think they are building a Doer/Checker pattern in which only the Checker is safety relevant, but in fact the Doer is safety relevant due to its ability to override the Checker's functionality.)

The usual claim being made is that a safety layer will prevent any malfunction of an autonomy layer from creating a mishap. This claim generally involves arguing that an autonomy failure will activate a safing function in the safety layer, and that an attempt by the autonomy to do something unsafe will be prevented. A typical (deficient) scheme is to have autonomy failure detected via some sort of self-test combined with the safety layer monitoring an autonomy layer heartbeat signal. It is further argued that the safety layer is designed in conformance with a suitable functional safety standard, and therefore acts as a safety-rated Checker as part of a Doer/Checker pair.

The flaw in that safety argumentation approach is that the autonomy layer has been presumed to fail silent via a combination of self-diagnosed fault detection and lack of heartbeat. However, self-diagnosis and heartbeat detection methods provide only partial fault detection (Hammett 2001). For example, there is always the possibility that the checking function itself has been compromised by a fault that leads to false negatives of checks for incorrect functionality. As a simple example, a heartbeat signal might be generated by a timer interrupt in the autonomy computer that continues to function even if significant portions of the autonomy software have crashed or are generating incorrect results. In general, such an architectural pattern is unsafe because it permits a non-silent failure in the autonomy layer to issue an unsafe vehicle trajectory command that overrides the safety layer. Fixing this fault requires making the autonomy layer safety-critical, which defeats a primary purpose of using a Doer/Checker architecture.

In practice, safety layer logic is usually less permissive than the autonomy layer. By less permissive, we mean that it under-approximates the safe state space of the system in exchange for simplifying computation (Machin et al. 2018). As a practical example, the safety layer might leave a larger spatial buffer area around obstacles to simplify computations, resulting in a longer total path length for the vehicle or even denying the vehicle entry into situations such as a tight alleyway that is only slightly larger than the vehicle.

A significant safety compromise can occur when vehicle designers attempt to increase permissiveness by enabling a non-safety-rated autonomy layer to say "trust me, this is OK" to override the safety layer. This creates a way for a malfunctioning autonomy layer to override the safety layer, again subverting the safety of the Doer/Checker pair architecture.

Eliminating this command override anti-pattern requires that the designers accept that there is an inherent trade-off between permissiveness and simplicity. A simple Checker tends to have limited permissiveness. Increasing permissiveness makes the Checker more complex, increasing the fraction of the system design work that must be done with high integrity. Permitting a lower integrity Doer to override the safety-relevant behaviour of a high integrity Checker in an attempt to avoid Checker complexity is unsafe.

Related pitfalls are first a system in which the Checker only covers a subset of the safety properties of the system. This implicitly trusts the Doer to not have certain classes of defects, including potentially requirements defects. If the Checker does not actually check some aspects of safety, then the Doer is in fact safety relevant. A second pitfall is having the Checker supervise a diagnosis operation for a Doer health check. Even if the Doer passes a health check, that does not mean its calculations are correct. At best it means that the Doer is operating as designed – which might be unsafe since the Doer has not been verified with the level of rigour required to assure safety.

We have found it productive to conduct the following thought experiment when evaluating Doer/Checker architectural patterns and other systems that rely upon assuring the integrity of only a subset of safety-related functions. Ask this question: "Assume the Doer (a portion of the system, including software, sensors, and actuators, that are not 'safety related') maliciously attempts to compromise safety in the worst way possible, with full and complete knowledge of every aspect of the design. Could it compromise safety?" If the answer is yes, then the Doer is in fact safety relevant, and must be designed to a sufficiently high level of integrity. Attempts to argue that such an outcome is unlikely in practice must be supported by strong evidence.

2.1.2 The implicit controllability pitfall

A safety case must account for not only failures within the autonomy system, but also failures within the vehicle.

A subtle pitfall when arguing based on conformance to a safety standard is neglecting that the assumptions made when assessing the subsystem that have potentially been violated or changed by the use of autonomy. Of particular concern for ground vehicles is the "controllability" aspect of an ISO 26262 ASIL analysis. (Severity and exposure might also change for an autonomous vehicle due to different usage patterns and should also be considered, but are beyond the scope of this discussion.)

The risk analysis of an underlying conventional vehicle according to ISO 26262 requires taking into account the severity, exposure, and controllability of each hazard (ISO 2011). The controllability aspect assumes a competent human driver is available to react to and mitigate equipment malfunctions. Taking credit for some degree of controllability generally reduces the integrity requirements of a component. This idea is especially relevant for Advanced Driver-Assistance Systems (ADAS) safety arguments, in which it is assumed that the driver will intervene in a timely manner to correct any vehicle misbehaviour, including potential ADAS malfunctions.

With a fully autonomous vehicle, responsibility for managing potentially unsafe equipment malfunctions that were previously mitigated by a human driver falls to the autonomy. That means that all the assumed capabilities of a human

driver that have been built in to the safety arguments regarding underlying vehicle malfunctions are now imposed upon the autonomy system.

If the autonomy design team does not have access to the analysis behind underlying equipment safety arguments, there might be no practical way to know what all the controllability assumptions are. In other words, the makers of an autonomy kit might be left guessing what failure response capabilities they must provide to preserve the correctness of the safety argumentation for the underlying vehicle.

The need to mitigate some malfunctions is likely obvious, but we have found that "obvious" is in the eye of the beholder. Some examples of assumed human driver interventions we have noted, or even experienced first-hand include:

- Pressing hard on the brake pedal to compensate for loss of power assist
- Controlling the vehicle in the event of a tire blowout
- Path planning after catastrophic windshield damage from debris impact
- Manually pumping brakes when anti-lock brake mechanisms time out due to excessive activation
- Navigating by ambient light after a lighting system electrical failure at speed
- Attempting to mitigate the effects of uncommanded propulsion power

To the degree that the integrity level requirements of the vehicle were reduced by the assumption that a human driver could mitigate the risk inherent in such malfunction scenarios, the safety case is insufficient if an autonomy kit does not have comparable capabilities.

Creating a thorough safety argument will require either obtaining or reverse engineering all the controllability assumptions made in the design of the underlying vehicle. Then, the autonomy must be assessed to have an adequate ability to provide the safety relevant controllability assumed in the vehicle design, or an alternate safety argument must be made.

For cases in which the controllability assumptions are not available, there are at least two approaches that should both be used by a prudent design team. First, FMEA, HAZOP, and other appropriate analyses should be performed on vehicle components and safety relevant functions to ensure that the autonomy can react in a safe way to malfunctions. Such an analysis will likely struggle with whether or not it is safe to assume that the worst types of malfunctions will be adequately mitigated by the vehicle without autonomy intervention.

Second, defects reported on comparable production vehicles should be considered as credible malfunctions of the non-autonomous portions of any vehicle control system since they have already happened in production systems. Such malfunctions include issues such as the drivetrain reporting the opposite of the current direction of motion, uncommanded acceleration, significant braking lag, loss of headlights, and so on (Koopman 2018a).

2.1.3 Arguing compliance with an inadequate safety standard

Historically some car makers have used internal, proprietary software safety guidelines as their way of attempting to assure an appropriate level of safety (Koopman 2018b). If a safety argument is based in part upon conformance to a safety standard, it is essential that the comprehensiveness of the safety standard itself be supported by evidence. For public standards that argument can involve group consensus by experts, public scrutiny, assessments of historical effectiveness, improvement in response to loss events, and general industry acceptance.

On the other hand, proprietary standards lack at least public scrutiny, and often lack other aspects such as general industry acceptance as well as falling short of accepted practices in technically substantive ways. One potential approach to argue that a proprietary safety standard is adequate is to map it to a publicly available standard, although there might be other viable argumentation approaches.

Sometimes arguing compliance with a well-known and documented industry safety standard may be inadequate for a particular safety case. Although all passenger vehicle subsystems in the United States must comply with the Federal Motor Vehicle Safety Standards (FMVSS) issued by the National Highway Transportation Safety Administration (NHTSA), compliance with these standards alone concentrates on mechanical and electrical issues, and only high level electronics functionality. FMVSS compliance is insufficient for a safety case that must encompass software (Koopman 2018b).

Similarly, use of particular standards might be an accepted or recommended practice, but not a means of ensuring safety. For example, the use of AUTOSAR (Autosar.org 2018) and Model Based Design techniques are sometimes proposed as indicators of safe software, but use of these technologies has essentially no predictive power with respect to system safety.

For automotive systems with safety-critical software, ISO 26262 is typically an appropriate standard, although other standards such as IEC 61508 (IEC 1998) can be relevant as well. For some autonomous system functions, conformance to ISO PAS 21448 (ISO 2018) might well be appropriate.

Arguments of the form "technology X, standard Y, or methodology Z is adequate because it has historically produced safe systems in that past" should address the potential issues with "proven in use" argumentation considered in the following section.

2.2 Proven in use

The proven in use argumentation pattern uses field experience of a component (or, potentially, an engineering process) to argue that field failure rates have been sufficiently low to assure a necessary level of integrity.

Historically this has been used as a way to grandfather existing components into new safety standards (for example, in the process industry under IEC 61508).

The basis of the argument is generally that if a huge number of operational hours of experience in the actual application have been accumulated with a sufficiently low number of safety-relevant failures, then there is compelling experimental data supporting the integrity claim for a component. We assume in this section that sufficient field experience has actually been obtained to make a credible argument. A subsequent section on field testing addresses the issue of how much experience is required to make a statistically valid claim.

A plausible variant could be that a component is initially deployed with temporary additional level of redundancy or other safety backup mechanism such as using a Doer/Checker pair. (By analogy, the Checker is a set of training wheels for a child learning to ride a bicycle.) When sufficient field experience has been accumulated to attain confidence regarding the safety of the main system, the additional safety mechanism might be removed to reduce cost.

For autonomous vehicles, the safety backup mechanism frequently manifests as a human safety driver, and the argument being made is that a sufficient amount of safe operation with a human safety driver enables removal of that safety driver for future deployments. The ability of a human safety driver to adequately supervise autonomy is itself a difficult topic (Koopman and Latronico 2019, Gao 2016), but is beyond the scope of this paper.

2.2.1 The violated assumptions pitfall

A significant threat to validity for proven in use argumentation is that the system will be used in an operational environment that differs in a safety-relevant way from its field history. In other words, a proven in use argument is invalidated if the component in question is exposed to operational conditions substantively different than the historical experience.

Diverging from historical usage can happen in subtle ways, especially for software components. Issues of data value limits, timing, and resource exhaustion can be relevant and might be caused by functionally unrelated components within a system. Differences in fault modes, sensor input values, or the occurrence of exceptional situations can also cause problems.

A proven in use safety argument must address how the risk of the system being deployed outside this operational profile is mitigated (e.g. by detecting distributional shifts in the environment, by external limitations on the environment, or by procedural mitigations). It is important to realize that an operational profile for a safety critical system must include not only typical cases, but also rare but expected edge cases as well. An even more challenging issue is that there might be operational environment assumptions that designers do not realize are being made, resulting in a residual risk of violated assumptions even after a thorough engineering review of the component reuse plan.

A classic example of this pitfall is the failure of Ariane 5 flight 501. That mishap involved reusing an inertial navigation system from the Ariane 4 design employing a proven in use argument (Lyons 1996). A contributing factor was that

software functionality used by Ariane 4 but not required for Ariane 5 flight was left in place to avoid having to requalify the component.

2.2.2 Depending upon COTS components

It is common to repurpose Commercial Off-The-Shelf (COTS) software or components for use in critical autonomous vehicle applications. These include components originally developed for other domains such as mine safety, low volume research components such as LIDAR units, and automotive components such as radars previously used in non-critical or less critical ADAS applications.

Generally such COTS components are being used in a somewhat different way than the original non-critical commercial purpose, and are often modified for use as well. Moreover, even field proven automotive components are typically customized for each vehicle manufacturer to conform to customer-specific design requirements. When arguing that a COTS item is proven in use, it is important to account for at least whether there is in fact sufficient field experience, whether the field experience is for a previous or modified version of the component, and other factors such as potential supply-chain changes, manufacturing quality fade, and the possibility of counterfeit goods.

In some cases we have seen proven in use arguments attempted for which the primary evidence relied upon is the reputation of a manufacturer based on historical performance on other components. While purchasing from a reputable manufacturer is often a good start, a brand name label by itself does not necessarily demonstrate that a particular component is fit for purpose, especially if a complex supply chain is involved.

2.2.3 The “small” change pitfall

Another threat to validity for a proven in use strategy is permitting “small” changes without revalidation (e.g. as implicitly permitted by (NHTSA 2016), which requires an updated safety assessment for significant changes). Change analysis can be difficult in general for software, and ineffective for software that has high coupling between modules and resultant complex interdependencies.

Because even one line of bad code can result in a catastrophic system failure, it is highly questionable to argue that a change is “small” because it only affects a small fraction of the source code base. Rigorous analysis should be performed on the change and its possible effects, which generally requires analysis of design artefacts beyond just source code.

A variant of this pitfall is arguing that a particular type of technology is generally “well understood” and implying based upon this that no special attention is required to ensure safety. There is no reason to expect that a completely new implementation – potentially by a different development team with a different engi-

neering process – has sufficient safety integrity simply because it is not a novel function to the application domain.

2.2.4 The discounted failure pitfall

A particularly tricky pitfall occurs when a proven in use argument is based upon a lack of observed field failures when field failures have been systematically under-reported or even not reported at all. In this case, the argument is based upon faulty evidence.

One way that this pitfall manifests in practice is that faults that result in low consequence failures tend to go un-reported, with system redundancy tending to reduce the consequences of a typical incident. It can take time and effort to report failures, and there is little incentive to report each incident if the consequence is small, the system can readily continue service, and the subjective impression is that the system is perceived as overall safe. Perversely, reporting numerous recovered malfunctions or warnings can actually increase user confidence in a system even while events go un-reported. This can be a significant issue when removing backup systems such as mechanical interlocks based on a lack of reported loss events. If the interlocks were keeping the system safe but interlock or other fail-safe engagements go unreported, removing those interlocks can be deadly. Various aspects of this pitfall came into play in the Therac 25 loss events (Leveson 1993).

An alternate way that this pitfall manifests is when there is a significant economic or other incentive for suppressing or mischaracterizing the cause of field failures. For example, there can be significant pressure and even systematic approaches to failure reporting and analysis that emphasize human error over equipment failure in mishap reporting (Koopman 2018b). Similarly, if technological maturity is being measured by a trend of reduced safety mechanism engagements (e.g. autonomy disengagements during road testing (Banerjee et al. 2018)), there can be significant pressure to artificially reduce the number of events reported. Claiming proven in use integrity for a component subject to artificially reduced or suppressed error reports is again basing an argument on faulty evidence.

2.2.5 The human filter pitfall

The operational history (and thus the failure history) of many systems is filtered by human control actions, pre-emptive incident avoidance actions and exposure to operator-specific errors. This history might not cover the entirety of the required functionality, might primarily cover the system in a comparatively low risk environment, and might under-represent failures that manifest infrequently with human operators present.

From a proven in use perspective, trying to use data from human-operated vehicles, such as historical crash data, might be insufficient to establish safety re-

quirements or a basis for autonomous vehicle safety metrics. The situations in which human-operated vehicles have trouble may not be the same situations that autonomous systems find difficult. It is easy to overlook the situations humans are good at navigating but which may cause problems for autonomous systems when looking at existing data of situations that humans get wrong. An autonomous system cannot be validated only against problematic human driving scenarios (e.g. NHTSA (2007) pre-crash typology). The autonomy might handle these hazardous situations perfectly yet fail often in otherwise common situations that humans regularly perform safely. Thus, an argument that a system is safe solely because it has been checked to properly handle situations that have high rates of human mishaps is incomplete in that it does not address the possibility of new types of mishaps.

This pitfall can also occur when arguing safety for existing components being used in a new system. For example, consider a safety shutdown system used as a backup to a human operator, such as an Automated Emergency Braking (AEB) system. It might be that human operators tend to systematically avoid putting an AEB system in some particular situation that it has trouble handling. As a hypothetical example, consider an AEB system that has trouble operating effectively when encountering obstacles in tight curves. If human drivers habitually slow down on such curves there might be no significant historical data indicating this is a potential problem, and autonomous vehicles that operate at the vehicle dynamics limit rather than a sight distance limit on such curves will be exposed to collisions due to this bias in historical operational data that hides an AEB weakness. A proven-in-use argument in such a situation has a systematic bias and is based on incomplete evidence. It could be unsafe, for example, to base a safety argument primarily upon that AEB system for a fully autonomous vehicle, since it would be exposed to situations that would normally be pre-emptively handled by a human driver, even though field data does not directly reveal this as a problem.

2.3 Field testing

In a field testing argumentation approach a fleet of systems is tested in real-world conditions to build up confidence. At a certain point the testers declare that the system has been demonstrated to be safe and proceed with production deployment.

An appropriate argumentation approach for field testing is that a sufficiently large number of exposure hours have been attained in a highly representative real-world environment. In other words, this is a variant of a proven in use argument in which the “use” was via testing rather than a production deployment. As such, the same proven in use argumentation issues apply, including especially the needs for representativeness and statistical significance. However, there are additional pitfalls encountered in field testing that must also be dealt with.

2.3.1 The insufficient testing pitfall

Accumulating an appropriate amount of field testing data for high dependability systems is challenging. In general, real-world testing needs to last approximately 3 to 10 times the acceptable mean time between hazardous failure to provide statistical significance (Kalra and Paddock 2016). For life-critical testing this can be an infeasible amount of testing (Butler and Finelli 1993, Littlewood and Strigini 1993).

Even if a sufficient amount of testing has been performed, it must also be argued that the testing is representative of the intended operational environment. To be representative, the field testing must at least have an operational profile (Musa et al. 1996) that matches the types of operational scenarios, actions, and other attributes of what the system will experience when deployed. For autonomous vehicles this includes a host of factors such as geography, roadway infrastructure, weather, expected obstacle types, and so on.

While the need to perform a statistically significant amount of field testing should be obvious, it is common to see plans to build public confidence in a system via comparatively small amounts of public field testing. (Whether there is additional non-public-facing argumentation in place is unclear in many of these cases.)

To illustrate the magnitude of the problem, in 2016 there were 1.18 fatalities per 100 million vehicle miles travelled in the United States, making the Mean Time Between Failures (MTBF) with respect to fatalities by a human driver 85 million miles (NHTSA 2017). Assuming an exponential failure distribution, for a given MTBF the required test time in which r failures occur can be computed with confidence α using a chi square distribution (Morris 2018):

$$\text{Required Test Time} = \chi^2(\alpha, 2r + 2)(\text{MTBF})/2$$

Based on this, a single-occupant system needs to accumulate 255 million test miles with no fatalities to be 95% sure that the mean time is only 85 million miles. If there is a mishap during that time, more testing is required to distinguish normal statistical fluctuations from a lower MTBF: a total of 403 million miles to reach 95% confidence. If a second mishap occurs, 535 million miles of testing are needed, and so on. Additional testing might also be needed if the system is changed, as discussed in a subsequent section. Significantly more testing would also be required to ensure a comparable per-occupant fatality rate for multi-occupant vehicle configurations.

Attempting to use a proxy metric such as rate of non-fatal crashes and extrapolate to fatal mishaps requires also substantiating the assumption that the mishap profiles will be the same for autonomous systems as they are for human drivers. (See the human filter pitfall previously discussed.)

2.3.2 The fly-fix-fly anti-pattern

In practice, autonomous vehicle field testing has typically not been a deployment of a finished product to demonstrate a suitable integrity level. Rather, it is an iterative development approach that can amount to debugging followed by attempting to achieve improved system dependability over time based on field experience. In the aerospace domain this is called a "fly-fix-fly" strategy. The system is operated and each problem that crops up in operations is fixed in an attempt to remove all defects. While this can help improve safety, it is most effective for a rigorously engineered system that is nearly perfect to begin with, and for which the fixes do not substantially alter the vehicle's design in ways that could produce new safety defects.

A significant problem with argumentation based on fly-fix-fly occurs when designers try to take credit for previous field testing despite a major change. When arguing field testing safety, it is generally inappropriate to take credit for field experience accumulated before the last change to the system that can affect safety-relevant behaviour. (The "small change" pitfall has already been discussed.)

There is no denying the intuitive appeal to an argument that the system is being tested until all bugs have been fixed. However, this is a fundamentally flawed argument. That is because this amounts to saying that no matter how many failures are seen during field testing, none of them "count" if a story can be concocted as to how they were non-reproducible, fixed via bug patches, and so on.

To the degree such an argument could be credible, it would have to find and fix the root cause of essentially all field failures. It would further have to demonstrate (somehow) that the field failure root cause had been correctly diagnosed, which is no small thing, especially if the fix involves retraining a machine-learning based system. Additionally, it would have to argue that a sufficiently high fraction of field failures have actually been encountered, resulting in a sufficiently low probability of encountering novel additional failures during deployment.

A fly-fix-fly argument must also address both the fault reinjection problem. Fault reinjection occurs when a bug fix introduces a new bug as a side effect of that fix. Ensuring that this has not happened via field testing alone requires resetting the field testing clock to zero after every bug fix. (Hybrid arguments that include rigorous engineering analysis are possible, but simply assuming no fault reinjection without any supporting evidence is not credible.)

It is difficult to believe an argument that claims that a fly-fix-fly process alone (without any rigorous engineering analysis to back it up) will identify and fix all safety-relevant bugs. If there is a large population of bugs that activate infrequently compared to the amount of field testing exposure, such a claim would clearly be incorrect. Generally speaking, fly-fix-fly requires an infeasible amount of operation to achieve the ultra-dependable results required for life critical systems, and typically makes unrealistic assumptions such as no new faults are injected by fixing a fault identified in testing (Littlewood and Strigini 1996). A specific issue is the matter of edge cases, discussed in the next section.

2.3.3 Dealing with edge cases

A significant limitation to a field testing argument is the assumption of random independent failures inherent in the statistical analysis. Arguing that software failures are random and independent is clearly questionable, since multiple instances of a system will have identical software defects. Moreover, arguing that the arrival of exceptional external events is random and independent across a fleet is clearly incorrect in the general case. A few simple examples of correlated events between vehicles in a fleet include:

- Timekeeping events (e.g. daylight savings time, leap second)
- Extreme weather (e.g. tornado, tsunami, flooding, blizzard white-out, wildfires) affecting multiple systems in the same geographic area
- Appearance of novel-looking pedestrians occurring on holidays (e.g. Halloween, Mardi Gras)
- Security vulnerabilities being attacked in a coordinated way

For life-critical systems, proper operation in typical situations needs to be validated. But this should be a given. Progressing from baseline functionality (a vehicle that can operate acceptably in normal situations) to a safe system (a vehicle that safely handles unusual situations and unexpected situations) requires dealing with unusual cases that will inevitably occur in the deployed fleet.

We define an *edge case* as a rare situation that will occur only occasionally, but still needs specific design attention to be dealt with in a reasonable and safe way. The quantification of "rare" is relative, and generally refers to situations or conditions that will occur often enough in a full-scale deployed fleet to be a problem but have not been captured in the design or requirements process. (It is understood that the process of identifying and handling edge cases makes them – by definition – no longer edge cases. So in practice the term applies to situations that would not have otherwise been handled had special attempts not be made to identify them during the design and validation process.)

It is useful to distinguish edge cases from corner cases. Corner cases are combinations of normal operational parameters. Not all corner cases are edge cases, and the converse. An example of a corner case could be a driving situation with an iced over road, low sun angle, heavy traffic, and a pedestrian in the roadway. This is a corner case since each item in that list ought to be an expected operational parameter, and it is the combination that might be rare. This would be an edge case only if there is some novelty to the combination that produces an emergent effect with system behaviour. If the system can handle the combination of factors in a corner case without any special design work, then it's not really an edge case by our definition. In practice, even difficult-to-handle corner cases that occur frequently will be identified during system design. Only corner cases that are both infrequent and present novelty due to the combination of conditions are edge cases. It is worth noting that changing geographic location, season of year, or other factors can result in different corner cases being identified during design and test, and leave different sets of edge cases unresolved. Thus, in practice, edge cases

that remain after normal system design procedures could differ depending upon the operational design domain of the vehicle, the test plan, and even random chance occurrences of which corner cases happened to appear in training data and field trials.

Classically an edge case refers to a type of boundary condition that affects inputs or reveals gaps in requirements. More generally, edge cases can be wholly unexpected events, such as the appearance of a unique road sign, or an unexpected animal type on a highway. They can be a corner case that was thought to be impossible, such as an icy road in a tropical climate. They can also be an unremarkable (to a human), non-corner case that somehow triggers an autonomy fault or stumbles upon a gap in training data, such as a light haze that results in perception failure. The thing that makes something an edge case is that it unexpectedly activates a requirements, design, or implementation defect in the system.

There are two implications to the occurrence of such edge cases in safety argumentation. One is that fixing edge cases as they arrive might not improve safety appreciably if the population of edge cases is large due to the heavy tail distribution problem (Koopman 2018c). This is because removing even a large number of individual defects from an essentially infinite-size pool of rarely activated defects does not materially improve things. Another implication is that the arrival of edge cases might be correlated by date, time, weather, societal events, micro-location, or combinations of these triggers. Such a correlation can invalidate an assumption that losses from activation of a safety defect will result in small losses between the time the defect first activates and the time a fix can be produced. (Such correlated mishaps can be thought of as the safety equivalent of a "zero day attack" from the security world.)

It is helpful to identify edge cases to the degree possible within the constraints of the budget and resources available to a project. This can be partially accomplished via corner case testing (e.g. Ding 2017). The strategy here would be to test essentially all corner cases to flush out any that happen to present special problems that make them edge cases. However, some edge cases also require identifying likely novel situations beyond combinations of ordinary and expected scenario components. And other edge cases are exceptional to an autonomous system, but not obviously corner cases in the eyes of a human test designer.

Ultimately, it is unclear if it can ever be shown that all edge cases have been identified and corresponding mitigations designed into the system. (Formal methods could help here, but the question would be whether any assumptions that needed to be made to support proofs were themselves vulnerable to edge cases.) Therefore, for immature systems it is important to be able to argue that inevitable edge cases will be dealt with in a safe way frequently enough to achieve an appropriate level of safety. One potential argumentation approach is to aggressively monitor and report unusual operational scenarios and proactively respond to near misses and incidents before a similar edge case can trigger a loss event, arguing that the probability of a loss event from unhandled edge cases is sufficiently low. Such an argument would have to address potential issues from correlated activation of edge cases.

2.4 Vehicle simulation

In this pattern, vehicle-level simulation rather than on-road operation is used as a proxy field testing strategy. Simulation offers a number of potential advantages over field testing of a real vehicle including lower marginal cost per mile, better scalability, and reduced risk to the public from testing. Ultimately, simulation is based upon data that generates scenarios used to exercise the system under test, commonly called the simulation workload. The validity of the simulation workload is just as relevant as the validity of the simulation models and software.

Simulation-based validation is often accomplished with a weighting of scenarios that is intentionally different than the expected operational profile. Such an approach has the virtue of being able to exercise corner cases and known rare events with less total exposure than would be required by waiting for such situations to happen by chance in real-world testing (Ding 2017). To the extent that corner cases and known rare events are intentionally induced in physical vehicle field testing or closed course testing, those amount to simulation in that the occurrence of those events is being simulated for the benefit of the test vehicle.

A more sophisticated simulation approach should use a simulation "stack" with layered levels of abstraction. High level, faster simulation can explore system-level issues while more detailed but slower simulations, bench tests, and other higher fidelity validation approaches are used for subsystems and components.

Regardless of the mix of simulation approaches, simulation fidelity and realism of the scenarios is generally recognized as a potential threat to validity. The simulation must be validated to ensure that it produces sufficiently accurate results for aspects that matter to the safety case. This might include requiring conformance of the simulation code and model data to a safety-critical software standard.

2.4.1 Missing rare events

Even with a conceptually perfect simulation, the question remains as to what events to simulate. Even if simulation were to cover enough miles to statistically assure safety, the question would remain as to whether there are gaps in the types of situations simulated. This corresponds to the representativeness issue with field testing and proven in use arguments. However, representativeness is a more pressing matter if simulation scenarios are being designed as part of a test plan rather than being based solely on statistically significant amounts of collected field data.

Another way to look at this problem is that simulation can remove the need to do field testing for rare events, but does not remove determine what rare events matter. All things being equal, simulation does not reduce the number of road miles needed for data collection to observe rare events. Rather, it permits a substantial fraction of data collection to be done with a non-autonomous vehicle. Thus, even if simulating billions of miles is feasible, there needs to be a way to

ensure that the test plan and simulation workload exercise all the aspects of a vehicle that would have been exercised in field testing of the same magnitude.

As with the fly-fix-fly anti-pattern, fixing defects identified in simulation requires additional simulation input data to validate the design. Simply re-running the same simulation and fixing bugs until the simulation passes invokes the "pesticide paradox." (Beizer 1990) This paradox holds that a system which has been debugged to the point that it passes a set of tests can't be considered completely bug free. Rather, it is simply free of the bugs that the test suite knows how to find, leaving the system exposed to bugs that might involve only very subtle differences from the test suite.

2.4.2 Nondeterministic behaviour and legibility

The nature of the algorithms used by autonomy systems creates problems for modelling and testing that go beyond typical safety critical software. Some autonomy algorithms, such as randomized path planning, are inherently nondeterministic. Others can be brittle, failing dramatically with subtle variations in data, such as perception false negatives induced by adversarial attacks (Szegedy at al. 2013) or false negatives induced by slight image degradation due to haze or defocus (Pezzementi et al. 2018).

A related issue is over-fitting to the test, in which an autonomy system over-fits and learns how to beat a fixed test. By analogy, this is the pitfall of the system cheating by having memorized the correct answers. A proposed way to deal with this risk is by randomly varying aspects of test cases. In such a fuzzing or variable testing approach it is important to randomly vary all relevant aspects of a problem. For example, varying geometries for traffic situations can be helpful, but probably does not address potential over-fitting for perception algorithms that perform object classification.

The use of potentially non-deterministic test scenarios combined with nondeterministic system behaviours and opaque system designs means it is difficult to know whether a system has passed a test, because there is no single correct answer. Rather, there must be some algorithmic way to determine whether a particular system response is acceptable or not, making that test oracle algorithm safety critical.

Moreover, it is possible that a system has passed a particular test by chance. For example, a pedestrian might be avoided due to a properly functioning detection and avoidance algorithm. But a pedestrian might also be avoided merely because a random path planner by chance picked a path that did not intersect the pedestrian, or responded to a completely unrelated aspect of the environment that caused it to pick a fortuitously safe path. Similarly, a pedestrian might be detected in one image, but undetected in another that differs in ways that are essentially imperceptible to a human.

It is unclear if resolving this issue requires solving the difficult problem of explainable AI (Gunning 2018). As a minimum, a credible safety argument will need

to address the problem of how plans to test vehicles with less than a statistically valid amount of real-world exposure data can avoid these pitfalls. It seems likely that a credible argument will also have to establish that each type of test has been passed due to safe operation of the system rather than simply by chance.

2.4.3 Human test scenario bias

Simulation-based testing (including especially closed-course testing of real vehicles) can suffer from a test planning bias. The problem is that a test plan is often made according to human perception of the scenario being tested. For example, a test scenario might be "child crossing in a painted cross-walk." Details of the test scenario might explore various corner cases involving child clothing, size, weather conditions, scene clutter, and so on.

Commonly test scenarios map to a human-interpretable taxonomy of the system and environmental state space. However, autonomy systems might have a different internal state space representation than humans, meaning that they classify the world in ways that differ from how humans do so. This in turn can lead to a situation in which a human believes apparent complete coverage via a testing plan has been achieved, while in reality significant aspects of the autonomy system have not been tested. As a hypothetical example, the autonomy system might have deduced that a human's shirt colour is a predictor of whether that human will step into a street because of accidental correlations in a training data set. But the test plan might not specify shirt colour as a variable, because test designers did not realize it was a relevant autonomy feature for pedestrian motion prediction.

Machine-learning based systems are known to be vulnerable to learning bias that is not recognized by human testers, at least initially. Some such failures have been quite dramatic (e.g. Grush 2015). Thus, simplistic tests such as having an average body size white male in neutral summer clothing cross a street to test pedestrian avoidance do not demonstrate a robust safety capability. Rather, such tests tend to demonstrate a minimum performance capability.

Interpreting the results of human-constructed test designs, including humans interpreting why a particular on-road scenario failed, are also subject to human test scenario bias. A credible safety argument that relies upon human-constructed tests or human interpretation of root cause analysis in claiming that test failures have been fixed should address this pitfall.

2.4.4 Data validity

Any safety argument that relies upon simulation must not only argue that the simulation is sufficiently accurate, but also that the workload is sufficiently accurate.

For example, consider a workload derived from a collection of on-road data. The sensors that captured the data, the data recording mechanism, the data translation programs, and the data handling procedures that eventually produce the simu-

lation workload are all safety relevant. This is because any biases or faults in that data handling process will produce inaccurate testing data, potentially resulting in flawed simulation evidence of safety. Relevant guidance exists on the topic of safety relevant data, and should be taken into account (DSIWG 2018).

As a simple example of how simulation data collection can go wrong, consider a data handling approach that discards outliers or seemingly invalid data. That approach might usually discard legitimate outliers. On the other hand, the outlier rejection method might also discard legitimate edge cases that are then excluded from simulation, resulting in a deficient safety case.

2.5 Formal proof of correctness

In this pattern, formal methods are used to define and prove system safety properties. These might include model checking, control system analysis, use of validated synthesis tools, kinematic analysis, and correct by construction approaches. It is unlikely that any such technique will scale up to the entirety of an autonomous vehicle in the near future, but such techniques can be a valuable part of appropriate aspects of a safety argument.

2.5.1 Assumption validity

Any assumptions made in formal analysis must be shown to be valid – or at least not safety relevant – in deployed systems. Assumptions vary widely and can include items such as: defect-free computational hardware, defect free computation execution (e.g. no soft errors), defect-free sensors, 100% fault diagnosis coverage, objects obeying expectations such as following traffic laws, absence of unexpected obstacles, model accuracy, and perfect vehicle maintenance. Additionally, any specific model or proof is likely to have other assumptions or simplifications required to make a formal proof practicable.

A way to mitigate the threats to validity posed by incorrect assumptions is to monitor them for validity during deployment using some type of runtime monitoring (Koopman and Wagner 2016). When an assumption violation does occur that does not quite mean that the system is unsafe, nor does it mean a mishap is inevitable. But an assumption violation does mean that the safety case is invalid while the assumption is being violated.

2.5.2 Proving safety

While formal methods can be an extremely useful piece of verifying correctness of a system, safety is an emergent property that touches the system's implementation and environment in ways that often reach beyond the formal verification.

There are many subtle ways in which formal verification can provide less assurance than is assumed, including model or implementation bugs (e.g. Vanhoef and Piessens 2017) or requirements gaps (failures which occur outside the model/proof or outside the hypothesized fault model). These types of issues highlight the value of an explicit, heterogeneous safety case that can clearly delineate what evidence the formal methods are actually providing as well as the strengths and limitations of how that evidence can be used in a credible manner.

3 Other autonomous vehicle safety argument observations

While not necessarily argumentation patterns, we have observed a number of other common issues that are worthy of comment.

Defective disengagement mechanisms. Generally this involves the ability of an arbitrary fail-active autonomy failure to prevent successful disengagement by a human supervisor. As a concrete example, a system might read the state of the disengagement activation mechanism (the "big red button") as an I/O device fed directly into the primary autonomy computer rather than using an independent safing mechanism. This is a special case of a single point of failure in the form of the autonomy computer.

Assuming perception failures are independent. Some arguments assume independent failures of multiple perception modes. While there is clearly utility in creating a safety case for the non-perception parts of an autonomous vehicle, one must argue rather than assume the safety of perception to create a credible safety case at the vehicle level.

Requiring perfect human supervision of autonomy. Humans are well known to struggle when assigned such monitoring tasks. Koopman et al. (2019) cover this topic in more detail as it relates to autonomous vehicle road testing safety.

Dismissing a potential fault as "unrealistic" without supporting data. For example, argumentation might state that a lightning strike on a moving vehicle is unrealistic or could not happen in the "real world," despite data to the contrary (e.g. Holle 2008). To be sure, this does not mean that something like a lightning strike must be completely mitigated via keeping the vehicle fully operational. Rather, such faults must be considered in risk analysis. Dismissing hazards without risk analysis based on a subjective assertion that they are "unrealistic" results in a safety case with insufficient evidence.

Using multi-channel comparison approaches for autonomy. In general autonomy algorithms are nondeterministic, sensitive to initial conditions, and have many acceptable (or at least safe) behaviours for any given situation. Architectural

approaches based on voting diverse autonomy algorithms tend to run into a problem of deciding whether the outputs are close enough to be valid. Averaging and other similar approaches are not necessarily appropriate. As a simple example, the average of veering to the right and veering to the left to avoid an obstacle could result in hitting the obstacle dead-on.

Confusion about fault vs. failure. While there is a widely recognized terminology document for dependable system design (Avizienis 2004), we have found that there is widespread confusion about the terms fault and failure in practical use. This is especially true when discussing malfunctions that are not due to a component fault, but rather a requirements gap or an excursion from the intended operational environment. It is beyond the scope of this paper to attempt to resolve this, but we note it as an area worthy of future work and particular attention in interdisciplinary discussions of autonomy safety.

4 Conclusions

Creating a credible safety case for an autonomous vehicle seems likely to require a heterogeneous approach, with various aspects argued via different methods. It would be no surprise if a safety case for a complete autonomous vehicle required using all of the strategies described in this paper and more. It is important to note that addressing all the pitfalls in this paper is necessary, but is likely insufficient for creating a safe autonomous vehicle. A number of other factors must be resolved including understanding the role of engineering rigor, software quality, and the special needs of validating system functionality based upon machine learning techniques. Along the way, we expect that additional pitfalls and argumentation anti-patterns will be identified that should be added to the list given in this paper.

We believe that assuring the validity and credibility of autonomous vehicle safety cases can benefit from a library of both valid and invalid argumentation strategies as well as examples and pitfalls. Such a library at the very least provides fodder for assessors to ensure that the most common argumentation and evidence pitfalls are avoided in safety cases. This paper provides a starting point for such an approach.

5 References

ACWG (2018) Goal Structuring Notation Community Standard, version 2, The Assurance Case Working Group (ACWG), SCSC-141B, January 2018.

Autosar.org (2018) The Standard Software Framework for Intelligent Mobility, https://www.autosar.org/ accessed Nov. 5, 2018.

Avizienis, A., Laprie, J.-C., Randell B., Landwehr, C. (2004) “Basic concepts and taxonomy of dependable and secure computing,” IEEE Trans. Dependability, 1(1):11-33, Oct. 2004.

Bannerjee, S., Jha, S., Cyriac, J., Kalbarczyk, Z, Iyer, R., (2018) “Hands Off the Wheel in Autonomous Vehicles?: A Systems Perspective on over a Million Miles of Field Data,” 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2018.

Beizer, B. (1990) Software Testing Techniques, 2nd Ed., 1990.

Bishop, Bloomfield (1998) “A methodology for safety case development,” Proceedings of the Sixth Safety-Critical Systems Symposium, Feb. 1998.

Burton, S., Gauerhof, L., Heinzemann, C. (2017) Making the case for safety of machine learning in highly automated driving, 5th International Workshop on Assurance Cases for Software-Intensive Systems (ASSURE 2017), 2017, pp. 5-16

Butler, Finelli (1993) “The infeasibility of experimental quantification of life-critical software reliability,” IEEE Trans. SW Engr. 19(1):3-12, Jan 1993.

Catapult (2017) Taxonomy of Scenarios for Automated Driving, v1.2, Technical Report, Transport Systems Catapult, UK, April 2017.

Ding, Z., “Accelerated evaluation of automated vehicles,” http://www-personal.umich.edu/~zhaoding/accelerated-evaluation.html on 10/15/2017.

DSIWG (2018) Data Safety Guidance, The Data Safety Initiative Working Group [DSIWG], January 2018.

Gao, F. (2016) Modeling Human Attention and Performance in Automated Environments with Low Task Loading, Ph.D. Dissertation, Massachusetts Institute of Technology, Feb. 2016.

Gauerhof, L., Munk, P., Burton, S. (2018) “Structuring Validation Targets of a Machine Learning Function Applied to Automated Driving,” International Conference on Computer Safety, Reliability, and Security (SAFECOMP), 2018, pp. 45-58

Goodenough, J., Weinstock, C., Klein, A., (2015) Eliminative Argumentation: A Basis for Arguing Confidence in System Properties, Software Engineering Institute, Technical Report CMU/SEI-2015-TR-005, Feb. 2015.

Graydon, P., Habli, I., Hawkins, R., Kelly, T., Knight, J. (2012) “Arguing conformance,” IEEE Software, May/June 2012, pp. 50-57.

Grush, L. (2015) “Google engineer apologizes after Photos app tags two black people as gorillas,” The Verge, July 1, 2015. https://www.theverge.com/2015/7/1/8880363/google-apologizes-photos-app-tags-two-black-people-gorillas (Accessed Oct. 27, 2018)

Gunning, D. (2018), Explainable Artificial Intelligence (XAI), Defense Advanced Research Projects Agency, https://www.darpa.mil/program/explainable-artificial-intelligence (accessed October 27, 2018).

Hammett, R., (2001) “Design by extrapolation: an evaluation of fault-tolerant avionics, 20th Conference on Digital Avionics Systems, IEEE, 2001.

Holle, R. (2008) Lightning-caused deaths and injuries in the vicinity of vehicles, American Meteorological Society Conference on Meteorological Applications of Lightning Data, 2008.

IEC (1998) IEC 61508: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems, International Electronic Commission, December, 1998.

ISO (2011) Road vehicles -- Functional Safety -- Management of functional safety, ISO 26262, International Standards Organization, 2011.

ISO (2018) Road vehicles – Safety of the Intended Function, ISO/PRF PAS 21448 (under development), International Standards Organization, 2018.

Kalra, N., Paddock, S., Driving to Safety: how many miles of driving would it take to demonstrate autonomous vehicle reliability? Rand Corporation, RR-1479-RC, 2016.

Kelly, T., (1998) Arguing Safety – a systematic approach to managing safety cases, Ph.D. Thesis, University of York, Sept. 1998.

Koopman, P. and Latronico, B., (2019) Safety Argument Considerations for Public Road Testing of Autonomous Vehicles, SAE WCX, 2019. (In press.)

Koopman, P. and Wagner, M., (2016) "Challenges in Autonomous Vehicle Testing and Validation," SAE Int. J. Trans. Safety 4(1):2016

Koopman, P., (2018a), Potentially deadly automotive software defects, https://betterembsw.blogspot.com/2018/09/potentially-deadly-automotive-software.html, Sept. 25, 2018.

Koopman, P. (2018b), "Practical Experience Report: Automotive Safety Practices vs. Accepted Principles," SAFECOMP, Sept. 2018.

Koopman, P. (2018c) "The Heavy Tail Safety Ceiling," Automated and Connected Vehicle Systems Testing Symposium, June 2018.

Leveson, N. (1993) An investigation of the Therac-25 Accidents, IEEE Computer, July 1993, pp. 18-41.

Littlewood, B., Strigini, L. (1993) “Validation of Ultra-High Dependability for Software-Based Systems,” Communications of the ACM, 36(11):69-80, November 1993.

Lyons, J.L. (1996) Ariane 5: Flight 501 Failure Report By The Inquiry Board, Paris, July 1996.

Machin, M., Guiochet, J., Waeselynck, H., Blanquart, J-P., Roy, M., Masson, L., (2018) “SMOF – a safety monitoring framework for autonomous systems,” IEEE Trans. System, Man and Cybernetics Systems, 48(5) May 2018, pp. 702-715.

Morris, S. (2018) Reliability Analytics Toolkit: MTBF Test Time Calculator. https://reliabilityanalyticstoolkit.appspot.com/mtbf_test_calculator. Reliability Analytics. (accessed December 12, 2018)

Musa, J., Fuoco, G., Irving, N., Kropfl, D., Juhlin, B., (1996) “The Operational Profile,” Handbook of Software Reliability Engineering, pp. 167-216, 1996.

NHTSA (2007) Pre-Crash Scenario Typology for Crash Avoidance Research, National Highway Traffic Safety Administration, DOT HS-810-767, April 2007

NHTSA (2016) Federal Automated Vehicles Policy, National Highway Transportation System Administration, U.S. Department of Transportation, September 2016.

NHTSA (2017) Traffic Safety Facts Research Note: 2016 Fatal Motor Vehicle Crashes: Overview. U.S. Department of Transportation National Highway Traffic Safety Administration. DOT HS-812-456.

OMG (2018) Structured Assurance Case Metamodel (SACM) version 2.0, Object Management Group, March 2018.

Pezzementi, Z., Tabor, T., Yim, S., Chang, J., Drozd, B., Guttendorf, D., Wagner, M., Koopman, P., "Putting image manipulations in context: robustness testing for safe percep-

tion," IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), Aug. 2018.

Rushby (2015) “On the interpretation of assurance case arguments,” AI Workshop on Argument for Agreement and Assurance (AAA 2015), November 2015, Springer LNAI Vol. 10091, pp. 331-347.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R. (2013) "Intriguing properties of neural networks." arXiv preprint arXiv:1312.6199 (2013).

UK Ministry of Defence (2017) Defence Standard 00-56 Issue 7 (Part 1): Safety Management Requirements for Defence Systems, Feb. 2017.

Vanhoef, M., Piessens, F. (2017) Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17). ACM, New York, NY, USA.

Wang, R., Guiochet, J., Motet, G., (2017) “Confidence assessment framework for safety arguments,” SAFECOMP 2017, Sept. 2017.