Identifying Run-time Monitoring Requirements for Autonomous Systems through the Analysis of Safety Arguments*

Richard Hawkins[0000−0001−7347−3413] and Philippa Ryan Conmy[0000−0003−1307−5207]

Assuring Autonomy International Programme, Department of Computer Science, University of York, Deramore Lane, York, England, YO10 5GH
richard.hawkins@york.ac.uk, philippa.ryan@york.ac.uk

Abstract. It is crucial that safety assurance continues to be managed for autonomous systems (AS) throughout their operation. This can be particularly challenging where AS operate in complex and dynamic environments. The importance of effective safety monitoring in ensuring the safety of AS through-life is already well documented. These current approaches often rely on utilising monitored information that happens to be available, or are reliant solely on engineering judgement to determine the requirements. Instead, we propose to use a systematic analysis of the safety case as the basis for determining the run-time monitoring requirements.

Safety cases are created for AS prior to deployment in order to demonstrate why they are believed to be sufficiently safe to go into operation. The safety case is therefore inevitably based upon predictions and assumptions about the system and its operation which may become untrue due to changes post-deployment. Our approach identifies specific run-time monitoring requirements for AS based upon a dialectic analysis of the safety case developed for the system. The advantage of the approach described is that it is systematic (through explicit consideration of elements of the safety case for the AS) and provides a way to justify the sufficiency of the resulting monitoring requirements (through creating explicit links the safety claims made about the AS).

Keywords: Monitors · Safety arguments · Run-time

1 Introduction

It is crucial for the assurance of safety-related and safety-critical systems that the safety of the system can be demonstrated throughout its entire operational life. Many safety assurance activities (such as design analysis and testing) can be undertaken during development prior to the deployment of the system to

* This work is funded by the Assuring Autonomy International Programme https://www.york.ac.uk/assuring-autonomy. Parts of this work were undertaken as part of the "LOPAAS" project of the Fraunhofer-Gesellschaft "ICON" programme.

operation. However, it is also important that safety continues to be managed and assessed post-deployment, particularly to identify and respond to unanticipated changes in the system or the operating environment. As part of this it is important to ensure that effective monitoring is in place during operation that can identify when a response is required to ensure acceptable safety is maintained. Effective monitoring is important for all safety-related systems, however it is of particular importance for autonomous systems (AS) [7], [3], since it is expected that AS will experience more change during operation. This may be in the form of changes to the AS itself (updates to machine learning models or unanticipated failure modes of system components), or changes in the complex, dynamic operating environment in which AS are required to operate.

Other work has previously discussed the need for safety monitors for AS. For example, in [6] the use of safety performance indicators (SPIs) is suggested as a way of defining safety metrics for an AS that can be monitored during operation. In [2], the authors propose the use of runtime monitors to assess assurance properties of AS by measuring confidence/uncertainty in those properties at runtime. And in [9], the use of probabilistic runtime risk monitors is proposed as a way of supporting dynamic risk assessment of AS during operation. A lot of literature relating to run-time monitoring is focused on highly situational monitoring to mitigate specific hazards. For example, [8] provides a method for ensuring safe distance between platooning trucks. In [5], the authors propose a safety concept, which monitors internal health and plausibility checking for an autonomous driving control system. However, underpinning the successful use of any safety monitors is the need to be able to identify and justify the selection of what is required to be monitored, and how the use of the information from that monitor can be shown to be effective in maintaining safety during operation. Whereas existing approaches rely largely on engineering judgement to define monitoring requirements, this paper describes an approach for identifying specific run-time monitoring requirements for AS based upon an analysis of the safety case developed for the system.

The advantage of the approach described in this paper is that it is systematic (through explicit consideration of elements of the safety case for the AS) and it provides a way to justify the sufficiency of the resulting monitoring requirements (through creating explicit links to the safety claims made about the AS). The paper is structured as follows; Section 2 introduces our approach based on the use of dialectic arguments; Section 3 discusses the activities required pre-deployment of the AS in order to identify run-time monitoring requirements; Section 4 then discusses how the monitors are used post-deployment to ensure the validity of the AS safety case is maintained; We draw conclusions and describe further work in Section 5. We use an illustrative example throughout the paper to illustrate our approach.

2 Background

Safety cases can be created for systems prior to deployment in order to demonstrate why they are believed to be sufficiently safe to go into operation [10]. The safety case for any system is therefore inevitably based upon predictions and assumptions about the system and its operation. Despite the best efforts of engineers when developing the safety case to ensure all predictions and assumptions are valid, some of these may turn out to be incorrect during operation, or may be correct at the point of deployment, but later become incorrect due to changes during operation. For example, during operation, upredicted emergent properties of the system may become evident, unanticipated changes may occur in the external environment, or the operational performance of system components may start to diverge from predictions. Safety assurance requires that confidence can be provided in the continued validity of the safety case post-deployment once the AS is in operation. In order to do this we need to understand what are the correct things to monitor that will provide the required information about safety case validity. We also need to know at what point the information obtained from the monitors indicates that the validity of the safety case may be undermined. This requires the definition of triggers associated with each monitor. The triggers are used to indicate that corrective actions are required to restore the validity of the safety case. The analysis that we propose in order to define the safety monitoring requirements for an AS is based around the use of dialectic argument techniques. Below we introduce dialectic arguments and explain how we use them in defining safety monitors for AS.

2.1 Dialectic Safety Arguments

Dialectic arguments provide a way of explicitly representing not just the argument and evidence that support the truth of the claims that are made (as is done in a conventional safety argument), but also a way of representing argument and evidence that could undermine the truth of those claims. Dialectic arguments are created by identifying challenges to elements of the safety argument (claims evidence, context, assumptions etc.). The challenges can take the form of either claims that, if true, undermine the argument (challenge claims) or evidence that undermines the truth of the argument (counter-evidence). The Goal Structuring Notation (GSN) standard [1] defines a way of documenting counter evidence as part of a safety argument. Figure 1 shows a simple example of argument challenges and defeat (a key to the GSN symbology is provided in Figure 2). The diagram on the left hand side (a), shows a small GSN argument fragment regarding a claim that an algorithm is correctly implemented. The diagram on the right hand side (b), shows how various elements of that argument may be challenged. Challenges are represented by dotted arrows that link from the challenge claim to the element being challenged. In Figure 1 challenges are shown to three elements of the argument: strategy S12, evidence E121 and claim C12. Challenges are represented as claims (CC12.1, CC121.1 and CC2.1 respectively).

For challenges to be compelling they should be supported by some evidence (referred to as counter-evidence). In Figure 1, an example of counter-evidence is shown as EC2.1. If a challenge presented to the safety case is determined to be valid, then the challenged element in the safety argument is said to be defeated. Figure 1 shows an example of how a defeated claim, C12, can be represented.

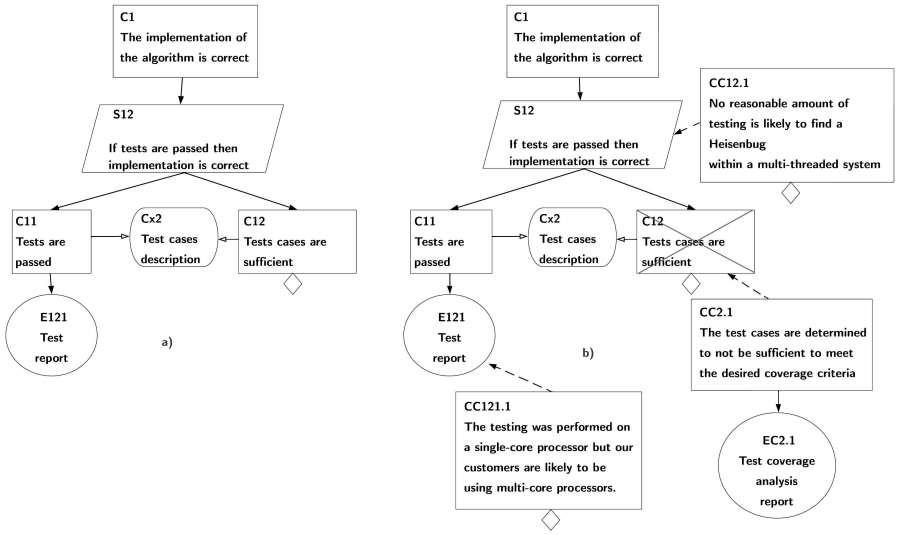

Fig. 1. A simple example GSN safety argument (a) and a dialectic argument showing how challenges and defeat of elements of the GSN structure can be represented (b).

In order to identify how the validity of an AS safety case may be undermined during operation, we propose that dialectic argumentation can be used to explicitly identify, prior to deployment of the AS, how the safety argument or evidence may be challenged at run-time. This enables detailed challenge claims and counter-evidence specific to the operation of the AS under consideration to be identified. As this dialectic argument focuses on potential run-time challenges, we refer to it as an operational dialectic argument. Our approach says that if, at any point in the lifecycle of the AS, the counter-evidence or challenge-claim we have identified in the operational dialectic argument exists, then the safety case is undermined, since elements of that argument become defeated. Therefore by monitoring explicitly at run-time for occurrence of this counter-evidence, we can have confidence that the safety case will remain valid during operation.

Once the operational challenges have been identified using the operational dialectic argument, the mechanisms for successfully monitoring and responding to the occurrence of that counter-evidence must be determined and implemented.

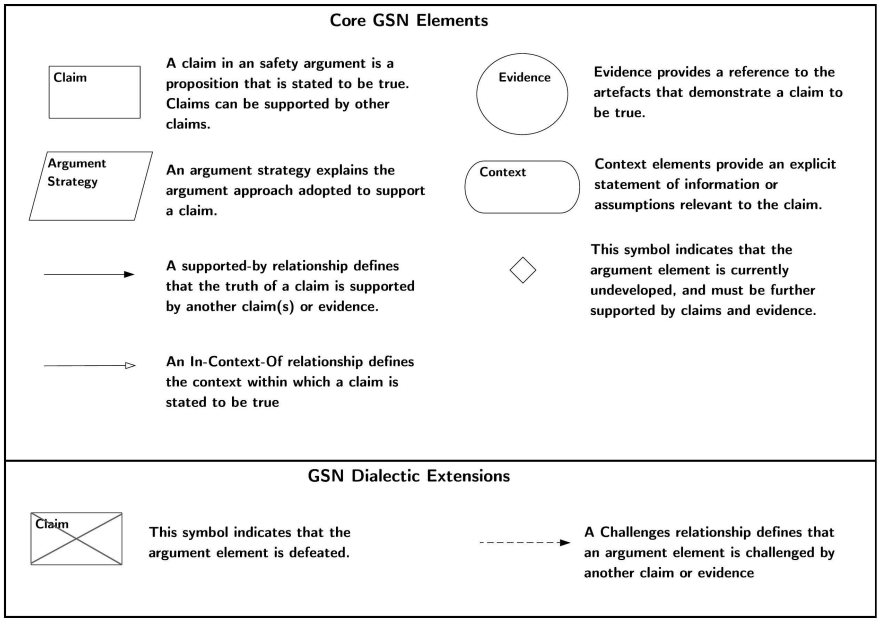

Fig. 2. A key to the GSN symbols used this paper

We discuss this in section 4. Firstly in the next section we look at how an operational dialectic argument can be created for an AS.

3 Pre-deployment

In this section we describe the activities required in our approach prior to deployment of the system in order to identify a sufficient set of run-time monitors and to be able to justify their sufficiency. This firstly requires the development of an operational dialectic argument, which is discussed in section 3.1.

3.1 Operational Dialectic Arguments

The starting point for creating a dialectic argument for an AS is the AS safety case itself. This safety case will have been created prior to deployment of the AS during the development of the system to demonstrate, through a safety argument supported by evidence, that the AS is sufficiently safe to operate in its defined context. There is existing guidance that can be used to help in the creation of AS safety cases [4]. Figure 3 shows a simplified example of part of the safety case for an autonomous vehicle that will operate on public roads. The safety case is represented using GSN.

This part of the safety case shows the argument that a particular safety requirement (SR1) has been addressed through the design and implementation of a particular system component (in this case the object detection component of the vehicle). The object detection component in this case is responsible for identifying objects that are present in the operating environment of the AS. This is demonstrated through supporting two claims. Firstly (G2.1.1) it is demonstrated through testing evidence that SR1 is satisfied. This argument is supported through the provision of the relevant test results, as well as claims regarding the sufficiency of both the test cases and the test platform that was used. Secondly, it is argued that the manner in which the object detection component was developed gives confidence that SR1 is met. This argument considers the machine learning (ML) model that is used as part of the object detection component (G2.1.2.1), the appropriateness of the design decisions that were taken (G2.1.2.2), and the rigour of the development process followed (G2.1.2.3). Each of these claims is developed further through argument and evidence that is not shown in Figure 3. A further claim (G2.1.2.4) shows that no hazardous errors were identified in the design of the object detection component based on evidence from a design review.

With the pre-deployment safety case established, the operational dialectic safety case is then created by using the GSN argument structure to systematically identify potential run-time challenges to elements of that safety case. These challenges, at this point, are hypothetical, in that the challenges to the safety case elements do not, prior to deployment of the AS, exist. If the counter-evidence that supports a challenge becomes present during the operation of the AS, then that challenge becomes valid, and the relevant element of the AS safety case may be defeated. For this reason it is important to be able to know, at run-time, if any of this counter-evidence exists. Unless sufficient monitoring is put in place prior to deployment of the AS, then the counter-evidence that can defeat elements of the safety case may exist without the knowledge of the system operator. By identifying the potential run-time challenges prior to deployment, sufficient monitors can be put in place to identify the presence of counter evidence.

Figure 4 shows a simplified version of the argument fragment from Figure 3 with the potential run-time challenges and counter-evidence identified (for clarity in the diagram only those argument elements with challenges are fully represented). This dialectic argument was created by systematically considering each element of the safety argument and considering whether any events that could foreseeably occur during operation of the AS could defeat that element. These challenges are captured as challenge claims, which are stated as propositions that would become true in the presence of particular counter-evidence arising during operation. For example, the claim G2.1 that the object detection component satisfies the requirement SR1 would be directly challenged at run-time if the performance that is actually being observed by the system in operation is seen not to meet that specified by the requirement. This has been captured by the challenge claim CC1 in Figure 4. CC1 would be become a valid challenge during operation of the AS if there was evidence to show that this

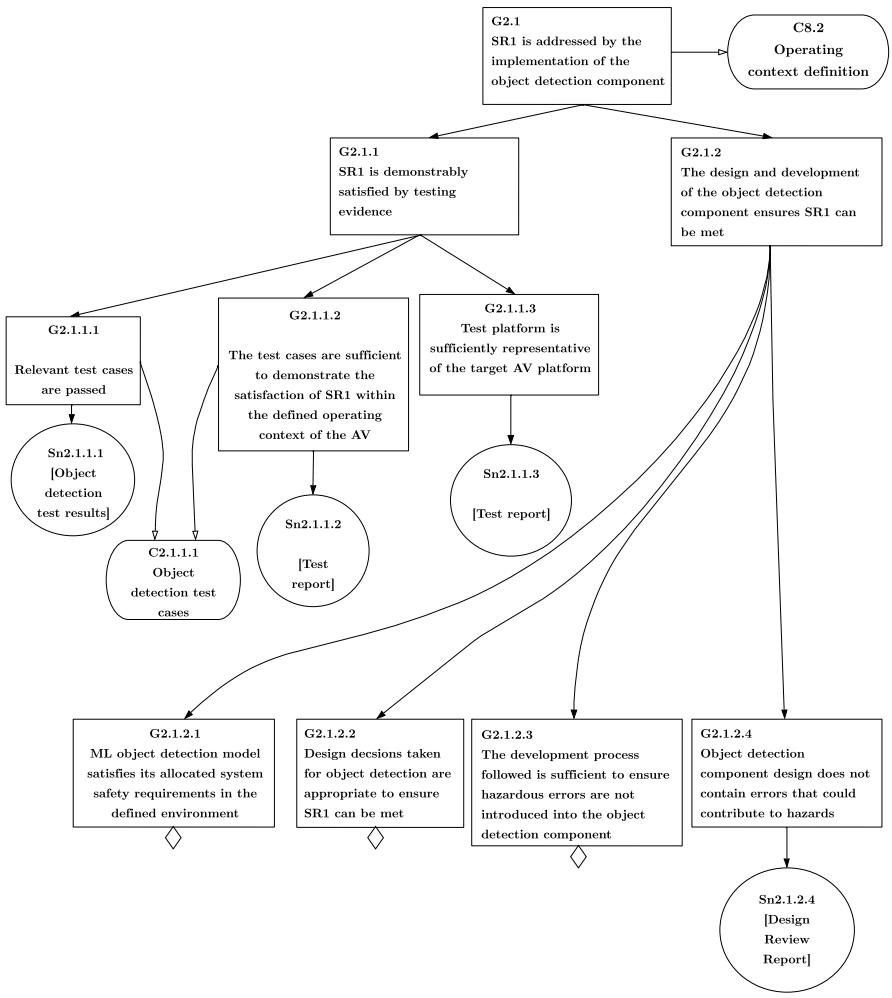

Fig. 3. GSN safety argument extract from a safety case for an autonomous vehicle.

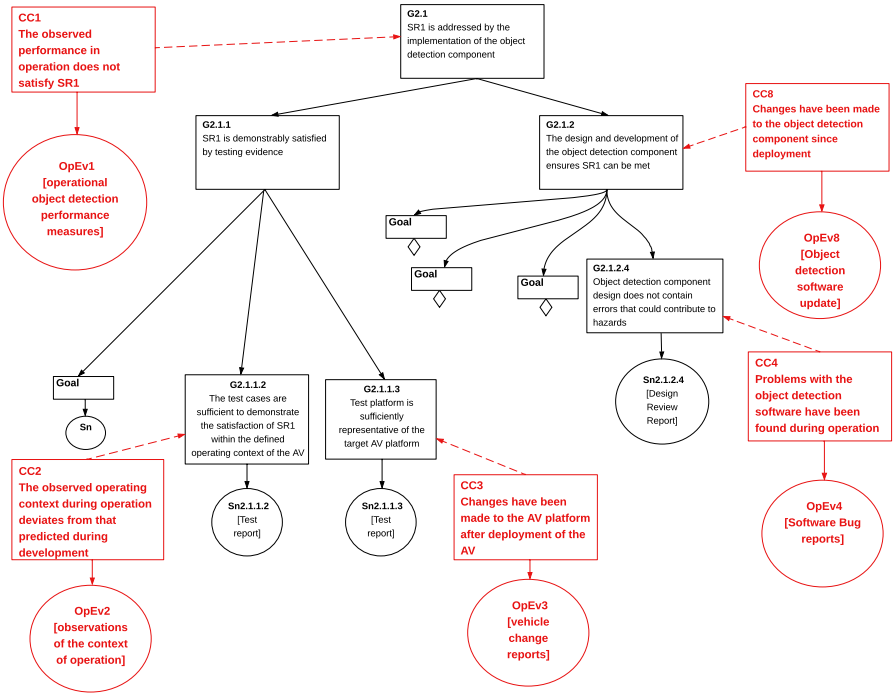

Fig. 4. Extract of an operation dialectic argument for an AV.

challenge claim was true. OpEv1 states the nature of the operation counter evidence that would support this. OpEv1 will therefore be used as the basis for a monitoring requirement in Section 3.2 to ensure that the presence of the counter evidence is known.

Another example provided in Figure 4 relates to claim G2.1.1.3, that the test platform represents the actual AV platform on which the component is operating. This claim could be challenged if the AV platform is changed during operation such that the test platform is no longer indicative of the AV. Such changes could mean that the testing results are no longer valid. This has been captured by the challenge claim CC3 in Figure 4. CC3 would be become a valid challenge during operation of the AS if there was evidence to show that such a change had indeed occurred to the vehicle in operation. OpEv3 identifies change reports as the mechanism by which evidence of this could be identified. Once again OpEv3 can be used as the basis for a monitoring requirement in Section 3.2.

Each of the items of operational counter-evidence from Figure 4, along with other examples from related parts of the wider safety case are listed in Table 1. It is only by identifying the presence of this counter-evidence at run-time that the existence of challenges to the validity of the safety case can be known. If such challenges were to occur during operation and the operator of the AS

was unaware that this was the case (due to insufficient monitoring), then the continued operation of the AS is potentially unsafe. It is therefore important that monitoring requirements are defined for each item of operational counter-evidence to ensure the necessary information is available during run-time. Then we can use this to know if a safety argument challenge has occurred and hence an appropriate response can be enacted to ensure the safety of the AS is maintained. The next section discusses these monitoring requirements.

3.2 Identifying Run-time Monitoring Requirements

For each item of operational counter-evidence identified, it is necessary to define:

- what needs to be monitored at run-time in order to be aware of the presence of the counter-evidence?
- what the criteria are that will be used for judging whether what has been monitored represents counter-evidence?
- what will be used as the trigger for determining the presence of counter-evidence?

Table 1, shows this information captured in the form of a table for the counter-evidence identified for the example system. This is captured as 'Monitor', 'Criteria' and 'Trigger' respectively. It can be seen in these examples that a diverse set of information must be monitored for the AV to ensure that all the relevant counter evidence will be identified. This includes things such as the number of missed detections across the fleet of AVs, information about changes to the vehicle platform, and detection of software errors. Each of these requires some mechanism to be put in place to obtain that information at run-time. This may include, for some of the information, the installation of physical sensors on the AS, but may also include more procedural mechanisms to check and record events. In this paper we consider all such mechanisms to be run-time monitors.

We can illustrate the monitoring requirements by considering examples from Table 1. The operational counter-evidence OpEv1 has been identified as operational object detection performance measures. It can be seen in Figure 4 that this was identified as counter-evidence because it would support a claim that the observed performance of the component in operation does not satisfy the defined safety requirement. In order to know if this evidence has arisen, we determined that it would be necessary to monitor the number of missed detections that are seen to occur across the fleet of vehicles that are operating. The criteria that would be used to determine if counter-evidence is observed is the number of missed detection that are observed in a set period (in this case 1000 miles of operation). This can then be compared against the defined trigger to determine if this represents counter-evidence. In setting the trigger we must consider the context of the safety argument, and the specific element to which the counter-evidence would relate. In this case we must consider the number of misses per 1000 miles that would represent a deviation from the performance claimed in the safety case. Here, it would be when the number of misses exceeds what was

Table 1. Identifying requirements for run-time monitoring from counter-evidence

Op. Evidence	**Monitor**	**Criteria**	**Trigger**
OpEv1 - [operational object detection performance measures]	Number of missed pedestrian detections across the vehicle fleet	Missed detections observed per 1000 miles of operation	#misses/1000 miles exceeds rate reporting in test results by 10%
OpEv2 - [observations of the context of operation]	Input images arising from the camera for operation within defined ODD	Measurement of key parameters within images (e.g., light levels, surfaces, colours etc.)	Operational images outside of test distribution
OpEv3 - [vehicle change reports]	Physical changes to vehicle platform (such as updates to sensors, processors etc.)	Changes that may impact software performance	Notification of AV platform modification
OpEv4 - [Software bug report]	Software errors discovered during operation	Errors identified in object detection during operation	Notification of error found in object detection
OpEv5 - [AV incident reports]	Reports raised by operators of the vehicle	Incidents that relate to object detection	Notification of object detection incidents that may be hazardous
OpEv6 - [Camera maintenance records]	Calibration of camera	Time since last calibration	Greater than 6 months since last calibration
OpEv7 - [Camera drift measurements]	Drift measurement of camera images	Rate of drift in operation	Rate of drifting exceeds design assumption
OpEv8 - [Object detection software update]	Software version	Change to object detection software	Non-approved version of software running
OpEv9 - [Lidar error status]	Lidar health monitoring	Lidar availability	Lidar fails to provide output to object detection component

demonstrated in the pre-deployment testing of the vehicle. Since in this case the performance seen in testing comfortably exceeded that required to satisfy the safety requirement (SR1), we do not need to consider every small deviation from this as a trigger for counter-evidence. Instead we have set a 10% threshold, which is still within the safety requirement, but indicates a threat to the validity of that aspect of the safety case that warrants assessment and potential action (see Section 4).

As a second example we can consider OpEv8, which has been identified as an update to the object detection software. It can be seen in Figure 4 that this was identified as counter-evidence because it would support a claim that the object detection software that is running on the operational vehicle is not the software for which the claim about the soundness of the development was made. In order to know if this has occurred we need to monitor the version of the object detection software that is running on the operational system. The criteria is if the version number of the executing software indicates that there has been a change to the software, and this will trigger counter-evidence if the version that is running is not approved (since this could mean that we are no longer able to claim that the development of the object detection software is sufficient). Further investigation would again be required at this point to determine the nature and impact of the changes to the software.

Having identified the run-time monitoring requirements prior to deployment of the AS, those requirements then have to be managed post-deployment to ensure their effectiveness. This is discussed in Section 4.

4 Post-deployment

In this section we discuss in more detail the post-deployment monitoring mechanisms and how they might be implemented in practice. First we need to identify responsible organisations and create processes to keep track of the specified monitors and triggers (the effectiveness of these also needs to be justified in the safety case). Second we need to perform those processes when required.

4.1 Organisation and continual monitoring processes

Once the AS is in-service, safety management processes will need to be put in place to review each of the identified criteria. Whilst the triggers provide clear thresholds beyond which the safety case can be considered no-longer valid, and some may be considered higher priority than others, we would expect all of the criteria to be reviewed regularly as part of planned safety case review. Additionally, we would expect a periodic review of the safety case in case additional run-time monitors have been identified, or could be improved over time, e.g., if there is new technology for monitoring or a changed legal requirement. Further, there may be other changes in the operating environment which impact on the safety case. We emphasise that the run-time safety monitoring is a continual process which evolves during the life of the project.

To be effective, a responsible organisation must be identified, who have the role of collating the specified run-time monitored information. The precise organisation, or organisations, required to support this may be dependent on the nature of the run-time monitoring requirements, the type of AS, and the regulatory regime. For our autonomous vehicle example this will include fleet owners, independent servicing centres and/or original manufacturers as well as individuals who own the vehicle. A further complicating factor would be the need to consider international boundaries, and national requirements.

4.2 Impact assessment process

Assuming that data indicates the trigger threshold has been reached, or a planned review shows concern about a particular trigger, there will need to be an impact assessment. This will first need to identify areas of potential impact on the safety case. This will differ depending on the type of issue. At a minimum, all branches which refer to a particular trigger would need to be reviewed, but the impact on the case could be more wide ranging, e.g., if a new hazard was discovered.

For example, if the trigger is *Op Ev1 - #misses/1000 miles exceeds rate reporting in test results by 10%*, the potential impact of this could include one or more of the following:

- the validity of the goals below G2.1 in Figure 4 are challenges, as if the observed performance relating to SR1 is not as expected which implies a problem with the test and design evidence
- more specifically, the deployment environment may differ to that anticipated - implying a further issue for *Op Ev2 - observations of the operating context*
- the risk of collision with a pedestrian was higher than that anticipated due to one or more causal subsystems having lower performance than anticipated and each of these subsystems should be investigated for shortfall
- the *Op Ev1* trigger criteria was defined incorrectly, essentially a false positive problem
- the validity of one or more higher branches of the safety argument above G2.1 are undermined
- the scope of the impact could be an entire fleet of vehicles, or a subset of the fleet (for example operating in a particular region)

Alternatively, if we consider a low level trigger such as *Op Ev 7 - Rate of drifting exceeds design assumption*, the potential safety impact could extend to the performance of any subsystems using the camera. We should investigate any claims in the safety argument about the performance of the camera, which might be in different branches of the case.

Having assessed the impact of the operational counter-evidence on the safety case and the AS itself, it is necessary to either provide a rebuttal to this, or propose actions to address the impact. We should do this considering the impacted claims and their context, rather than considering the trigger in isolation. In practice a shortfall in measured performance may still be within tolerable safety

limits and be localised to different parts of the case and this should be investigated. To continue our examples, if the trigger for *Op Ev1 - #misses/1000 miles exceeds rate reporting in test results by 10%* is breached but there have been no significant incidents or other subsystems are still compensating for shortfall in one systems performance (e.g. where there are diverse means of detecting pedestrians) then it may be possible to argue that the AS is still acting well within tolerable levels of risk. Alternatively, it could be that the trigger has only been observed in an isolated case with unusual environmental conditions which it could be argued are unlikely to be seen again. Such rebuttals should be explicitly document and could be added to the dialectic argument during operation to document the resolution of the challenge claim.

If the trigger *Op Ev 7 - Rate of drifting exceeds design assumption* is breached, it may be that this results in no significant performance alteration in any of the systems which are using the camera and/or that a more regular re-calibration process can be automatically performed without impacting on the AS user. Again, such rebuttals could be added explicitly to the dialectic argument.

Where the monitors indicate that there is impact on the validity of the safety case but no rebuttals for the counter-evidence are identified, action must be taken at run-time to address this. Typical actions could include limitations on use, for example, avoiding using the AS in a particular environment until an issue is fully investigated and fixed, or limiting the operating speed. The safety case would need to be updated for this interim period, with the limitations made explicit at the top level of the case, and monitoring processes would continue. As an extreme example, action may require grounding of a fleet of AS. Once the safety issue was resolved, the safety case would again be updated to reflect this.

5 Conclusions and Further Work

The importance of effective monitoring in ensuring systems remain safe throughout their operational life is well understood. Due to the nature of AS, it becomes particularly important that the sufficiency of the monitoring that is in place at run-time can be justified. It is not sufficient therefore that the monitoring requirements for an AS be defined in an ad-hoc manner, or relying solely on engineering judgement. Instead it is important that a systematic and defensible method for deriving the run-time monitoring requirements is established. In this paper we have discussed how the continued validity of the AV safety case should be the focus of the run-time monitoring, with monitors used to identify if counter-evidence to the safety case occurs during operation, such that mitigations can be enacted. The approach we have described uses the concept of creating dialectic arguments as a way to systematically anticipate and identify operational counter-evidence, and thus to derive effective run-time monitoring requirements.

The work presented in this paper will lead to further related work. We have demonstrated how our approach can be applied in practice using an example from a self-driving vehicle and have provided simplified examples in this paper.

Further evaluation of this approach will be undertaken through additional case studies undertaken by independent engineers. We will consider the application of the approach to different types of AS in other domains in order to show the generalisabilty of the approach. We will also seek to evaluate, through observation during operation, the effectiveness of the monitoring requirements arising from following our approach. The approach we have described in this paper can be used to enhance the safety case for the operation of the AV, by enabling a compelling argument to be made about the sufficiency of the run-time monitoring. Further work will develop and demonstrate the structure for such arguments.

We will use the results obtained from case studies in order to investigate further the nature of the run-time monitoring requirements that are derived from applying our approach. It was seen in this paper that the nature of the monitoring requirements can be diverse in nature, and consequently the monitoring and mitigation mechanisms are also diverse. We will look to characterise the run-time requirements for AVs as the basis for providing further guidance on their effective management.

References

1. ACWG: Goal Structuring Notation Community Standard. Tech. Rep. SCSC-141C v3.0, Safety Critical Systems Club (2021), https://scsc.uk/scsc-141C
2. Asaadi, E., Denney, E., Menzies, J., Pai, G.J., Petroff, D.: Dynamic assurance cases: a pathway to trusted autonomy. Computer **53**(12), 35–46 (2020)
3. Haupt, N.B., Liggesmeyer, P.: A runtime safety monitoring approach for adaptable autonomous systems. In: Computer Safety, Reliability, and Security: SAFECOMP 2019 Workshops, Turku, Finland, September 10, 2019. pp. 166–177. Springer (2019)
4. Hawkins, R., Osborne, M., Parsons, M., Nicholson, M., McDermid, J., Habli, I.: Guidance on the Safety Assurance of Autonomous Systems in Complex Environments (SACE). arXiv preprint arXiv:2208.00853 (2022)
5. Hörwick, M., Siedersberger, K.H.: Strategy and architecture of a safety concept for fully automatic and autonomous driving assistance systems. In: 2010 IEEE Intelligent Vehicles Symposium. pp. 955–960 (2010)
6. Laboratories, U.: UL 4600: Standard for Evaluation of Autonomous Products (2020). Standard for safety, Underwriters Laboratories (2020)
7. Machin, M., Guiochet, J., Waeselynck, H., Blanquart, J.P., Roy, M., Masson, L.: SMOF: A safety monitoring framework for autonomous systems. IEEE Transactions on Systems, Man, and Cybernetics: Systems **48**(5), 702–715 (2016)
8. Reich, J., Sorokos, I., Papadopoulos, Y., Kelly, T., Wei, R., Armengaud, E.: Engineering of runtime safety monitors for cyber-physical systems with digital dependability identities. In: Computer Safety, Reliability, and Security. pp. 3–17 (2020)
9. Reich, J., Trapp, M.: Sinadra: towards a framework for assurable situation-aware dynamic risk assessment of autonomous vehicles. In: 2020 16th European dependable computing conference (EDCC). pp. 47–50. IEEE (2020)
10. Sujan, M.A., Habli, I., Kelly, T.P., Pozzi, S., Johnson, C.W.: Should healthcare providers do safety cases? lessons from a cross-industry review of safety case practices. Safety science **84**, 181–189 (2016)