Model-Driven Safety of Autonomous Vehicles

N. Annable, A. Bayzat, Z. Diskin, M. Lawford, R. Paige, and A. Wassyng

Abstract We make the case that since model-based development of complex software-intensive systems has proven to be so effective, a model-based paradigm that encompasses assurance of the system makes excellent sense and will result in more rigorous, less ad hoc approaches to the development and maintenance of assurance cases. This will become especially clear in the manufacturing of autonomous motor vehicles. Adequate demonstration of the safety of autonomous vehicles is a huge challenge. Doing it once for a single vehicle is difficult. Doing it for multiple vehicles in a product family and coping with incremental changes in design from one model version to the next without redoing the complete safety analysis is even more difficult. We show that a comprehensive, rigorous model-driven approach to development and assurance holds the promise of more efficient and more effective assurance in general and also provides a mechanism for incremental assurance. We also briefly compare that with one of the current staples for documenting assurance cases – Goal Structuring Notation.

Keywords Model-based development · Model-based assurance · Assurance cases

1 Introduction

Model-based development (MBD) of complex automotive systems is well established and has proven to be the preferred approach. Analysis of models together with model management and correct-by-construction software generation are convincing reasons to use a model-based approach. The approach to the associated safety assurance has lagged somewhat, but it makes good sense to utilize model-based approaches for those same reasons. In addition, safety assurance needs to be planned

N. Annable (✉) · A. Bayzat · Z. Diskin · M. Lawford · R. Paige · A. Wassyng
McMaster Centre for Software Certification, Department of Computing and Software, McMaster University, Hamilton, ON, Canada
e-mail: annablnm@mcmaster.ca

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022

A. M. Madni et al. (eds.), *Recent Trends and Advances in Model Based Systems Engineering*, https://doi.org/10.1007/978-3-030-82083-1_34

ahead of development and tightly integrated with development as it proceeds. One of the huge assurance hurdles we have to overcome is the inability to perform incremental safety assurance. Incremental design is now the norm with manufacturers routinely modifying existing vehicle (or vehicle product-line) designs for their next model year. In doing this, manufacturers must be able to do the same with the associated safety assurance. In other words, they need to be able to produce safety assurance based on that of the previous model and changes in design as well as changes in regulations, operating conditions, etc., without redoing the assurance from scratch.

This is challenging right now with the current proliferation of Advanced Driving Systems and introductory autonomous features. It is going to be even more difficult with more autonomous features leading to full Level 5 autonomy (i.e., features that completely replace the driver) (SAE J3016 2018), exacerbated by the fact that the required levels of safety increase as the level of autonomy increases. A common mitigating factor for early autonomous features is that there is a human driver who can take over control of the vehicle. This will not be true for Level 5 autonomy, drastically increasing the level of safety required for these future vehicles.

We believe that existing notations and tools for safety assurance fall far short of what we need for achieving the necessary safety levels for (current and) future vehicles. Together with an automotive partner, we have developed an approach to safety assurance that better integrates the assurance processes and development processes; is much more rigorous than existing techniques; is less ad hoc; includes extremely comprehensive traceability; is compatible with current model management techniques; and facilitates incremental assurance.

The remainder of this paper briefly introduces one of the most popular assurance case notations used today, introduces our new methodology, and very briefly compares them. Readers can find excellent publications on various aspects of assurance/safety cases, for example, in (Rushby et al. 2015 and Rhinehart et al. 2015).

2 Goal Structuring Notation (GSN)

GSN is one of the most popular graphical notations for the presentation of assurance cases (Kelly 1998). It presents its "argument" by stating *Goals* (representing claims) and *Strategies* (reasons for decomposing goals into sub-goals) and supports terminal goals using *Solutions* (representing evidence). It has a rich supporting notation including *Assumptions*, *Contexts*, and *Justifications*.

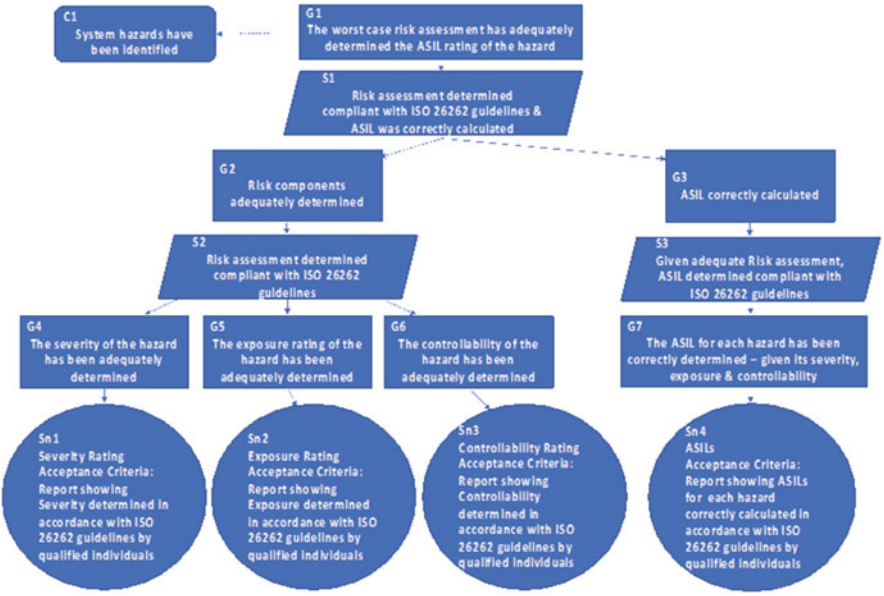

Fig. 1 GSN ASIL example

2.1 A GSN Example

Figure 1 provides an example of a GSN fragment representing the calculation of ASILs (Automotive Safety Integrity Levels) for all identified system-level hazards as described in ISO 26262 [ISO 26262, 2018]. Briefly, an ASIL captures the risk associated with a system-level hazard and guides the development of its mitigation. The context C1 is included to emphasize that this fragment would follow demonstration that system hazards were adequately determined. There is no space here to include other support nodes, such as assumptions.

The top-level claim G1 is decomposed into G2 and G3 as described in S1. G2 is further decomposed and supported by evidence that the severity, exposure, and controllability ratings have all been correctly determined in accordance with ISO 26262 guidelines. G3 is supported by a calculated ASIL rating that utilizes the correctly assigned severity, exposure, and controllability ratings.

2.2 GSN Benefits

GSN is intuitive. People understand it readily, and it seems to enable people to ask critical questions related to the (somewhat) implicit argument presented by GSN. It has motivated many developers and certifiers to consider seriously what must

be demonstrated to ensure safety. There is significant work on automating aspects of GSN, including safety case construction and formal approaches to dealing with evidence. An excellent source for this is https://ti.arc.nasa.gov/profile/edenney/.

2.3 GSN Challenges

GSN promotes an ad hoc approach to structuring an assurance case. There is nothing in GSN itself that helps us decide how to present a safety argument. Patterns and experience are the basis of good GSN assurance cases. The tree-like structure, while intuitive, results in cross-cutting concerns that make creating, understanding, and maintaining a GSN assurance case extremely challenging. The major challenges introduced by GSN are: (i) it leads to a false sense of confidence because the reasoning in GSN is about why/how claims are decomposed, not as to why premises (grounded in evidence) support parent claims, and (ii) rigorous safety impact analysis is extremely difficult, bordering on impossible. The inherent traceability in GSN is through arcs connecting nodes, and this will not detect the impact of changes in parts of the tree that are not explicitly connected.

3 Workflow$^+$

Workflow$^+$ (WF+) is a modelling framework which aims to provide a way for all information necessary for safety assurance to be captured in a single model. This model shows relationships between development processes, assurance processes, development outputs, assurance outputs, and the environment of the system(s) of interest. WF+ grew out of our work with an automotive partner; an overview of WF+ modelling and its core mechanisms can be found in (Diskin et al. 2019).

WF+ uses metamodels that define workflows to be followed during the development of domain-specific safety-critical systems, complete with all process definitions, data definitions, control flow, data-to-data and data-to-process traceability, and constraints over processes and data. These core mechanisms allow all necessary validation, verification, checks, and reviews to be modelled and included. When the process defined in the metamodel is executed, an instance of this metamodel documents the development of the real-world system produced. This includes details of the system data, reports generated by tasks within the process, etc.

A metamodel can be based on prevailing standards such as ISO 26262 (ISO 26262, 2018), best practices, internal company procedures, etc. and can also be used to check compliance with the different types of guidelines mentioned. A metamodel can be checked to see that it is well-formed based on rules suggested by the mathematical foundations of these models. These checks on well-formedness result in assurance steps, and these assurance steps can be viewed in different ways – one

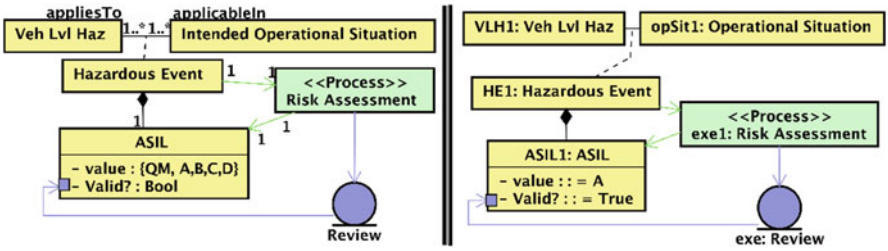

Fig. 2 A simple WF+ example (metamodel on the left, instance on the right)

important view is a GSN-like structure. Metamodels can be thought of as templates for development and/or assurance.

The WF+ metamodels presented in this paper are built using a profile of UML class diagrams with the following features: (i) two types of classes, process and data classes; (ii) two types of associations – dataflow associations (green) from data to process classes and back and static data associations (black) from data to data classes; (iii) a special type of process class to model reviewing; and (iv) several constraints on the interactions of the features mentioned above, the most important of which is that processes and their dataflow form a hierarchy, i.e., a directed acyclic graph.

Figure 2 shows a simple example of a WF+ metamodel of the risk assessment process described in ISO 26262 and an instantiation. In this example, the metamodel (left) specifies that when executed, the Risk Assessment process takes in a hazardous event, which is a pair of an operational situation and vehicle-level hazard, and outputs an ASIL classification for that hazardous event. The output data are connected to the input data, shown as a composition (black diamond) association from Hazardous Event (HE) to ASIL. The multiplicities dictate that this process is to be executed once for each HE and that each execution produces one ASIL. There is also a review process (purple), which evaluates the execution of Risk Assessment and the validity of its data. The instantiation (on the right) shows the documentation of an execution of Risk Assessment for a particular hazardous event HE1, which was determined to be ASIL A, and the output of a review process that validates this ASIL classification for HE1.

3.1 A WF+ Example

Figure 3 is a refined version of the example metamodel in Fig. 2. (The added argument elements will be explained later.) In this example, the input data definition has been refined to capture the types of Intended Operational Situations that can be part of a HE (operational situations are from (SAE J2980, 2015)) and the Consequence(s) of HEs. The definition of the Risk Assessment process itself has

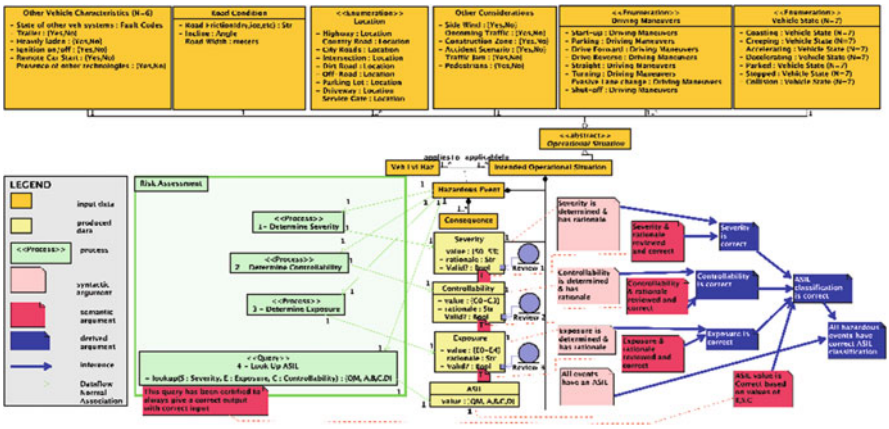

Fig. 3 Refined version of the WF+ example

been refined by decomposing it into four steps. When executed, steps 1 and 2 take in one HE, including Consequences, and produce values for the Severity and Controllability of that HE.

When executed, Step 3 takes in one HE and outputs Exposure as an attribute of the Intended Operational Situation of that HE. This HE, which will have its Severity, Controllability, and Exposure assigned, is then input to Step 4. Step 4 is a query (i.e., automatic process) that, when executed, assigns an ASIL classification to that HE. As steps 1, 2, and 3 define processes that are to be executed by humans, they have accompanying review processes to assess the validity of their output. As Step 4 is a Query, its output does not require validation (more on this in 4.6 Automation).

3.2 Building Arguments Over a WF+ Example

When a WF+ metamodel is built, its creators include constraints designed to ensure that its instances will be (i) syntactically correct (i.e., properly structured) and (ii) semantically correct (i.e., valid). Assurance-related arguments can be created based on these constraints almost mechanically. Light and dark pink elements in Fig. 3 are natural-language arguments based on syntactic and semantic constraints on the data definitions in the metamodel. Syntactic constraints (light pink), such as the multiplicity constraint that each HE must have exactly 1 Severity attribute, are put in place in the process definition to ensure that instances documenting execution of the metamodel are always properly structured. Violations of these syntactic constraints in the instantiation of the metamodel could be detected automatically by tooling. We can also set the tool to make such checks in the process of building the instance so that the instance is syntactically correct by construction.

Semantic constraints (dark pink), such as the constraint T on the attribute Valid? of Severity, are put in place to ensure that all (critical) data in an instance are validated by a review. In an instance documenting the execution of a process, if the value of a Valid? attribute is False or missing, that constraint is violated.

Since ASIL is produced by an automated Query, we want to certify that the Query itself will produce the correct result given correct inputs, rather than manually review the Query's output every time it is executed. This is represented by the pink node attached to Step 4 (see 4.6 Automation).

By composing syntactic and semantic constraints, we can derive higher-level constraints and add higher-level assurance-related arguments for those derived constraints (blue). For example, when combining the syntactic and semantic constraints on Severity, we add the argument "Severity is correct." That is, we claim that if a severity value is present *and* it has been reviewed, then it is correct, as shown by the blue arrows in Fig. 3. Derived constraints can be formed by combining syntactic, semantic, and/or derived constraints. In a GSN setting, a *strategy* must be included to explain the logic as to why sub-claims support their parent claim(s). In a WF+ setting, derived constraints (and their corresponding arguments) are based solely on the logical composition of constraints and thus do not require a strategy. The logical soundness of this approach enables us to ensure that any valid execution of the process definition will satisfy all constraints and thus all arguments in the process definition will hold for any valid instance (see (Diskin et al. 2019) for details). This is useful as it allows process definitions to be used as templates for the development of assurance cases (see 4.7 Templates). It is worth mentioning that while WF+ itself does not ensure that considerations for safety are included in some particular workflow, it does, however, provide a setting in which workflows can be planned and evaluated by experts to ensure that we are adequately confident they will result in safe systems. In safety-critical embedded systems, including AVs, this usually amounts to ensuring that we must (i) demonstrate that the requirements specification will result in a safe system; (ii) demonstrate that the system satisfies its requirements; and (iii) demonstrate the system does not implement any behavior not in the requirements specification.

4 Advantages of WF+ for Model-Based Assurance

4.1 Making Assurance Less Ad Hoc

Assurance of AVs using GSN can be ad hoc and reliant upon how individual safety experts interpret the safety requirements, the certification standards, and the GSN syntax itself. More specifically, there is no precisely defined methodology for assurance when using GSN; it relies on individual expertise. There are benefits to this, as engineers can optimize the process of constructing an assurance case based on their experience with AVs, but this comes at a price in terms of repeatability and

learnability. An important benefit of using WF+ is the clear methodology that is to be followed when building assurance arguments; this will enable engineers to more readily learn the techniques and for the steps to be repeatable.

By comparison with GSN, the semantics for each element of WF+, as well as the role each element plays in the assurance process, are precisely defined. This should lead to fewer opportunities for misinterpretation. Also, WF + 's structure makes managing large assurance cases more systematic, repeatable, and inexpensive.

4.2 Improved Traceability

Detailed traceability is essential for assurance cases of complex systems such as AVs, because it improves understandability and facilitates following all pertinent links to an argument. In particular, change impact analysis is completely dependent on accurate and complete traceability. Current approaches lack mechanisms to include direct traceability and often rely on implicit (i.e., assumed) traceability between arguments. For example, in Fig. 1, the traceability between hazards and their respective severity, controllability, and exposure is left implicit through the wording of the diagrammatic elements. This implicit traceability is easy to identify and follow in this simple example, but in large-scale industrial safety cases it is often much more difficult to identify and understand, especially when cross-cutting concerns branch over multiple argument legs. This places an undue burden on independent reviewers to discover this implicit structure on their own and, when compounded with the ad hoc structure typical of GSN-style safety cases, can lead to significant misunderstanding of the intended argument. This prevents independent reviewers from being able to review an assurance case with sufficient confidence and may result in potentially dangerous flaws in arguments.

WF+ was developed from the ground-up with this in mind and enables detailed traceability. All traceability necessary between data and processes underlying arguments is maintained explicitly and allows for cross-cutting concerns to be accurately and explicitly represented. WF+ facilitates improved understandability and independent reviewing and provides an excellent basis for change impact analysis.

4.3 Change Impact Analysis

When dealing with highly complex embedded systems such as AVs, it can be difficult to determine the impact of incremental design changes on the system's assurance case. As AV systems continue to increase in complexity, even experienced engineers have trouble keeping up with the thousands of connections between design and their respective elements of an assurance case. The model-based nature of WF+ provides the necessary foundations for change impact analysis to be

automated as much as is possible. The detailed granularity and traceability possible in WF+ metamodels allow for tools to be built that can automatically follow traceability links to all related data and their associated arguments, directing engineers to areas of assurance that are affected by changes in design. On top of this, the well-defined semantics of WF+ metamodels and assurance cases allows for an explicit ontology of change propagation that enables a well-defined approach to assurance of incremental changes to systems.

4.4 Integrating Assurance with Development

The model-based approach of WF+ opens up the opportunity for WF+ models to be directly integrated with model-based development or V&V tools. This allows assurance to be built directly over data from development, rather than having an assurance case as a separate document with references to development documentation. With direct access to artifacts from development, some aspects of assurance cases can be generated automatically and validated based on the content of those design artifacts (see 4.6 Automation). While it is possible to integrate GSN approaches with development (Hawkins et al. 2015), integrating WF+ with development will allow for more scalable solutions that are better suited to change impact analysis. Also, its traceability into the environment facilitates dealing with feature interactions that stretch into the environment.

4.5 Automation

As AV systems continue to increase in complexity, it is desirable to automate as much of the assurance case development as possible to reduce development costs. Building WF+ on well-established MBD principles allows tool developers to leverage a wide range of pre-existing techniques for managing assurance cases, including automated querying to search assurance cases, and transformations for applying templates.

The model-based approach of WF+ allows for static syntactic correctness to be checked automatically. As more granularity is added to the WF+ metamodel, some semantically significant properties can be encoded in the structure of the metamodel through the use of constraints. For example, if there are certain structural properties of design-related elements desirable for safety, then the corresponding constraints can be placed on the metamodel to allow these properties to be checked automatically. Table 1 in ISO 26262-6 (ISO 26262, 2018) outlines properties of software architectural design that are desirable for avoiding systematic faults. Many of these properties such as restricted size of interfaces, restricted size of complexity of software components, and loose coupling between software components are all good candidates for automatic checking through constraints over detailed models.

An MBD approach also allows for some processing to be automated, such as Look Up ASIL in Fig. 3. It is possible for these automated tasks to be certified, i.e., have trustworthy outputs given correct inputs. Time is saved by automating the process, and time is saved by not requiring their outputs to be reviewed.

4.6 Templates

As with any safety-critical system, it is necessary to plan for the safety of AVs ahead of development. Assurance case templates aim to specify a nearly-complete assurance case for a particular type of system before development begins (see Wassyng et al. 2015). A template includes sufficiently prescriptive limitations on systems (as determined collectively by experts in the field) but still allows enough flexibility so as to not unduly interfere with the creative design of a system. Assurance case templates specify higher-level argumentation and the overall structure of an assurance case, as well as acceptance criteria for required evidence.

WF+ is well suited for implementing assurance case templates using high-level WF+ metamodels that must be conformed to. For a particular system, this WF+ template can be refined to fit the needs of the system of interest and can then be executed. The modular nature of WF+ allows for assurance case templates to be created hierarchically to produce different versions of the templates to fit different use cases. Benefits of this include repeatability, ease of audit, and potentially increased productivity as tools can be used to carry out refinements and instantiation. Importantly, it also facilitates incremental assurance when changes that eventually occur were already taken into account as options in the metamodel.

5 Conclusion

The modelling effort required by WF+ is substantial. However, the vast majority of the required modelling is of the form “model once – use many times.” What do we get from this effort? One of the most important attributes is that it facilitates effective incremental assurance! Traceability links in WF+ are comprehensive and cover planning, development, processes, work products, and the environment. The fact that we start with metamodels that act as templates means we reduce confirmation bias in the assurance process. We also conform to the dictum – design safety into the vehicle, do not add it afterwards. Compared with existing notations/methods, WF+ is extremely rigorous, less ad hoc, facilitates automation of many aspects of safety assurance, and reduces the difficulties associated with cross-cutting concerns.?

Acknowledgments The authors want to thank our industry collaborators for the in-depth discussion and suggestions over the years we have been working on this topic. Input from Joseph D'Ambrosio, Lucian Patcas, Galen Ressler, Ramesh S, and Sigrid Wagner has been invaluable to our research.

References

Diskin, Z., N. Annable, A. Wassyng, and M. Lawford. 2019. *Assurance via Workflow+ Modelling and Conformance (an extended version)*. https://www.mcscert.ca/wp-content/uploads/2019/11/McSCert-Technical-Report-32.pdf.

Hawkins, R., I. Habli, D. Kolovos, R. Paige, and T. Kelly. 2015. Weaving an Assurance Case from Design: A Model-Based Approach. In *IEEE 16th International Symposium on High Assurance Systems Engineering*, vol. 2015, 110–117. Daytona Beach Shores, FL.

International Organization for Standardization. 2018. *ISO 26262: Road Vehicles – Functional Safety*. 2nd ed.

Kelly, T. 1998. *Arguing safety – A systematic approach to managing safety cases*. Ph.D. dissertation, Univ. York, York UK.

Rinehart, D., J.C. Knight, and J. Rowanhill. 2015. *Current practices in constructing and evaluating assurance cases with applications to aviation*. NASA Report NASA/CR–2015-218678.

Rushby, J., X. Xu, M. Rangarajan, and T.L. Weaver. 2015. *Understanding and evaluating assurance cases*. NASA/CR–2015-218802.

SAE International. 2015. SAE J2980. Considerations for ISO 26262 ASIL Hazard Classification.

———. 2018. SAE J3016. Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles.

Wassyng, A., N. Singh, M. Geven, N. Proscia, H. Wang, M. Lawford, and T. Maibaum. 2015. Can Product-Specific Assurance Case Templates Be Used as Medical Device Standards? *IEEE Design & Test* 32 (5): 45–55.