Safety Case Generation by Model-based Engineering: State of the Art and a Proposal

Fang Yan, Simon Foster, Ibrahim Habli
Department of Computer Science
University of York
York, UK
email: firstname.lastname@york.ac.uk

***Abstract*—The paper is a review to evaluate the current techniques for safety case generation using Model-based Engineering. Safety cases provide an explicit and structured means for assessing and assuring the safety of complex systems. For systems developed with Model-based Engineering, safety cases can be constructed with system models as input and should evolve hand-in-hand with system models when the system updates. Model-based Engineering can provide automatic means for the generation to improve efficiency. But there is not a full automation solution to cover the entire generation process. This paper investigates state-of-the-art of Model-based Engineering applications to safety case generation, explores the challenges and gaps, and proposes a solution framework to address the gaps through the model transformation within the Eclipse Modeling Framework.**

Keywords-safety case; assurance case; model-based engineering; generation; model.

I. Introduction

Safety Cases (SC) are defined as compelling arguments, supported by evidence, that systems operate as intended for defined applications in defined environments [1]. They provide a systematic way to argue the safety properties. SCs are important to the operation of safety-critical systems and recommended in some safety standards, such as ISO 26262 [2].

Many robotics and autonomous systems are safety-critical, such as autonomous cars, unmanned aerial vehicles, and medical robots. Their operational environments are relatively open and not sufficiently predictable during design. This may necessitate the system evolution, i.e., the redesign or replacement of system functions/components, during runtime at a higher frequency than traditional safety-critical systems.

SCs are constructed alongside the system development process. One of the issues for SC generation is the repeated workload from SC evolution due to system development iteration. Therefore, an automatic way for SC generation and co-evolution with system design is desired. SCs need to evolve when the systems are subjected to updates. This evolution is really an instance of the more general problem of generation, and so if we tackle the latter, we can more easily tackle the former. Therefore, We discuss the generation process in the paper.

In terms of the technical solution for generation automation, we explore Model-based Engineering (MBE). MBE has been well-adopted for system development thanks to its efficient tool support, and its applications have expanded into the surrounding aspects including SC generation. MBE techniques bring the capabilities of validation, model checking, simulation, model to model transformations, etc. From the published work, we can tell that the MBE applications on SC generation vary in terms of the techniques exploited, the generation phases applied to, and the extent of automation, etc. However, there is not an MBE solution to guide the whole engineering process of SC generation. The purpose of this paper is to understand the state-of-the-art of MBE applications on SC generation, to evaluate the automation degree of the solutions, and to point out the research gaps and the possible research directions. A new technical solution is proposed at the end to provide a framework to address the gaps. We only focus on the work that treats SCs as models, i.e., the whole set of SCs can be manipulated with MBE techniques.

Section II introduces the main background of the paper. Section III investigates the state-of-the-art of MBE based SC generation methods. Section IV evaluates these MBE solutions, identifies the open gaps, and proposes a new MBE solution. We conclude in Section V.

II. Background

A. Safety Case Notations

The widely used SC notations are the structured graphical forms, including Goal Structuring Notation (GSN) [1] and Claims-Arguments-Evidence (CAE) [3]. Many tools for SC generation and management are compliant with GSN. But most of them are not suitable for MBE applications due to the lack of a model-based foundation.

Structured Assurance Case Meta-Model (SACM) [4] is a standard for SC development and exchange released by Object Management Group (OMG). It specifies a metamodel composed of three concepts: argumentation, artifact, and terminology. SACM can support a variety of notations including GSN and CAE. SACM version 2.1 was published in 2020. As a new standard, SACM has little application in industry yet. However, it enables MBE techniques to be applied to SCs. We envisage future applications of SACM with possible toolchain support.

B. Model-based Engineering

To generate and manipulate SCs as models, metamodels of SC are indispensable. The most prominent modelling frameworks are the Eclipse Modeling Framework (EMF) [5], offering the metamodelling language Ecore, and the Meta Object Facility (MOF) [6], a standard metamodelling language defined by OMG.

III. Model-based SC Generation

A. A common SC generation practice

From a practical point of view, a common SC process is threefold, as shown in Fig. 1, including SC pattern design, instantiable data management, and pattern instantiation.

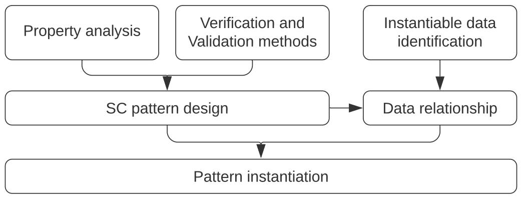

Fig. 1. A common process for SC generation

The concept of SC pattern is introduced by Kelly in [7]. It is an abstract structure containing placeholders that can be instantiated by concrete argument elements. For example, hazards with placeholders {Hazard} in patterns need to be replaced by the specific hazard content. It is a good practice for reusing SC structures.

To implement the instantiation, we need to manage the instantiable data. That is to identify the data required for SCs, and the relationships between the data elements and also between the data and SC elements. For example, a regulatory requirement may require that all hazards are mitigated. Thus, this requirement and all the hazards are instantiable data and the traceability between them shall be built. Further, the regulatory requirements may fit into the top claim of SCs, the hazards shall be used for lower claims to support the top claim.

The SC process progresses along with system development, and takes the system data as input, such as regulations, hazards, safety requirements, architecture, specification and design, validation and verification plan and results. To feed the system data into the pattern is the process of instantiation. This can be either manual or automatic depending on the data form and the tool support. If the intention is an automatic instantiation by tool, machine-readable instantiation data is required. Similar to the instantiation process, the data processing can be either manual or automatic through MBE as well. Based on the process above, different possibilities to use MBE for SC generation automation are discussed in the rest of this section.

B. SC generation by pattern instantiation

In this section, we discuss a common way to exploit MBE in SC generations following the process in Section III-A. The idea is to generate SC pattern models compliant with a metamodel, manually build the mapping between instantiable data and SC pattern nodes, then to instantiate the pattern automatically through MBE. The method includes following steps.

Step 1, to build a SC metamodel. This is usually done by building a GSN-based metamodel, or extending the SACM metamodel to be compliant with GSN.

Step 2, to design the SC pattern according to the system nature and create the pattern models using the SC metamodel.

Step 3, to identify and organize the instantiable data as a data table. The data may include hazards, causes, safety requirements, system requirements, tests, etc. The data types and the inter-relationships among data are defined in the table. The data can be either in a structured or unstructured manner.

Step 4, to manually establish the mapping between the nodes of SC pattern and the elements of the data tables.

Step 5, to instantiate the pattern models according to the mapping table of Step 4 and to output the SC models. The way to instantiate is first to identify the node in the pattern to be instantiated and the corresponding data for the node from the mapping table, then to fill the data into the pattern nodes.

The SC pattern is represented as models which allows the generation of SC models by automatic instantiation, and the subsequent model management capabilities, e.g., model validation, model query, and model comparison. However, since the data mapping table is generated manually, every time the source data in Step 3 is changed, e.g., a hazard is added or a safety requirement is deleted, the mapping needs a manual upgrade. This will bring high workload due to the frequent system design modifications.

Denney and Pai [8] have developed an automatic tool AdvoCATE based on this process for SC generation, management, and evaluation with the Eclipse EMF. The AC metamodel is created based GSN with a formal syntax. However, this is not a fully automated process as the logical mapping between SC nodes and system data are identified manually in the instantiable data processing phase.

The approach of Hawkins et al. [9] follows the same Step 1 and Step 2 as in [8], but exploits model weaving [10] to establish the mapping between instantiable data models and SC pattern models at the metamodel level. Model weaving is used to build relationships between elements of different metamodels, and can be realized manually or automatically by model transformation. The process differs from the method above in Step 3 to 5 as follows.

Step 3, to identify the instantiable models, such as system models, system error models.

Step 4, to establish the relationship between the elements of SC pattern and the elements of the instantiable models at their metamodel level within a weaving model.

Step 5, to instantiate through the weaving model execution and to output the SC models. The way to instantiate is first to identify the elements to be instantiated in the pattern, then to find the corresponding system model element through weaving model. For example, the “component goal” in SC pattern is the “process” element in Architecture Analysis and Design Language (AADL) models. So, the “process” models are extracted from the package of the whole system models and filled into the “component goal” in SCs.

Compared with [8], one of the advantages of the model weaving method is that the instantiable data can be extracted from the system models automatically. Secondly, the automatic co-evolution is enabled because the links between SC elements and system models are built between the metamodels instead of specific system data therefore can be updated automatically when the system design changes.

However, the method is limited to the systems developed with MBE because the data that has no metamodel supports cannot be processed by model weaving. We refer to this kind of data as ”unstructured” in the paper. Also, with the claims instantiated only by system models, the SC generated is incomplete. A SC structure usually starts from abstract property goal, goes down to the hazards and safety requirements, and then is related to system models representing functional requirements and design. Since the first three of data above are usually unstructured and cannot be processed by MBE directly, the corresponding claims are not covered by this method.

C. Integrated SC generation by system model query

This method is to generate SC models by system model query. The query language and the environment are both integrated with the system development environment. The query codes for SC generation are generated manually, but the codes can be reused as a library, thus the co-evolution of the SC models and system models can be automated. The method includes the following steps.

Step 1, to design a Domain Specific Language (DSL) specific to a certain system modelling language for SC claim generation in a formal manner and for system model query.

Step 2, to formally define the top-level claim using DSL within the system development environment.

Step 3, to design model query rules for the top-level claim using DSL, and return the query results as the SC evidence.

The claim formalization and the system query are implemented in the same environment of system modelling. This allows the tight coupling of SCs with system models and ensures the automatic consistency of the two when design changes. Resolute [11] is a DSL designed for creating SCs for AADL models following the steps above. The limitation is that the DSL is specific to AADL and not applicable to other modelling languages. Also, since SCs are highly integrated with system models, the claims do not involve the unstructured data including such as hazard and hazard causes, etc. Thus, the SCs generated are incomplete.

D. SC generation by claim formalization and refinement

In this method, SC claims are formalized as a series of mathematical assertions about a system model equipped with a formal semantics. While the system models are refined, the concrete low level claims are inferred from top level assertion in parallel. The main benefit of the method is that the inference from top level to lower level claims can be verified by rigorous mathematical refinement checking. The method includes three steps.

Step 1, to formalize the top claim as an assertion “$M \models G$ under *A*”, where *M* is the system model, *A* is an assumption on environment, *G* is the guarantee on system model. This assertion denotes that the system models satisfy the guarantee if the assumption is valid.

Step 2, to decompose the top claim by model refinement, i.e., by refining the system model through system development, weakening the assumptions, and decomposing or adding guarantees. Thus, the lower level claims are inferred as a set of “$M^* \models G^*$ under A^*” where * means “refined”.

Step 3, to verify the correctness and completeness of the refinement by Formal Method (FM) verification. This activity assures the completeness of the SC structure generated through model refinement in a rigorously mathematical way.

Besides the benefit of the rigorous verification, the integration of the system models with SC claims supports the automatic co-evolution of SC whenever system design changes. However, since the top level claims are usually abstract, engineering review is a more appropriate way for decomposition validation, and the formalization and refinement checking would add no extra value. Additionally, the tight coupling of SCs with system models requires that both the SC and system be modelled in a formal way, and this requires the expertise of formal methods.

Gleirscher et al. [12] proposes this solution and formalize the claims using differential dynamic logic. The refinement checking is demonstrated in Isabelle/HOL [13]. Diskin et al. [14] applied the similar concepts for SC construction using data refinement. To reduce the need for FM expertise, Block Diagrams (BD) are used to guide the system model refinement from the perspective of the system architecture. However, for the further detailed system implementation, FM expertise is still unavoidable.

IV. Evaluation and proposal

From Section III, we can see that the SC generation by automatic pattern instantiation [8] provides a solution for construction of a complete SC. But the instantiable data process is not automated. This will bring a high workload of SC update when system data change. Also, the system model is not well integrated with SCs. On the other hand, the integration of the system models into SCs [9] [11] [12] [14] brings the benefit of automatic co-evolution of the SCs and system models. However, these system model-based solutions only create the lower structure of SCs because the upper structure of SC does not involve the concrete system design but the unstructured hazard analysis data. Moreover, the model query [11] provides an automatic traceability from system model to SCs, but the application is constrained to a certain system modelling language. The method of claim formalization and refinement [12] [14] requires FM expertise which may block the way of the engineering practical application.

To summarize, there is not an automatic solution fully covering the SC generation process with a wide application scope. The gaps lie mainly in: (1) a lack of an automatic way to process the unstructured instantiable data for MBE manipulation; (2) the missing of integration of upper SC structure derived from hazard analysis and the lower SC structure from system models; (3) a narrowed scope of applicability to the system development techniques.

To close the gaps, we propose an SACM compliant framework for SC generation combining the pattern instantiation based method and system model query based method. The method is to be applied within the Eclipse EMF framework. The instantiable data, the system design, and SCs are all handled as EMF models. For a use case study, RoboChart [15] is chosen as the system modelling language which is designed in Eclipse and can be exported as EMF models. The framework includes following steps as shown in Fig. 2.

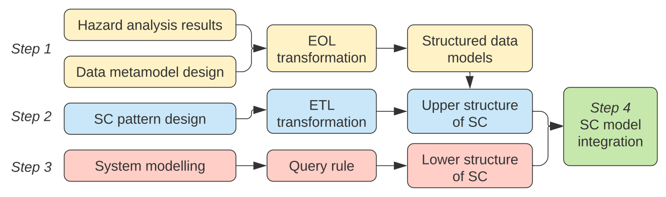

Fig. 2. A common process for SC generation

Step 1, the structured data model generation from hazard analysis result. In order to manipulate the unstructured data with MBE, we need first to design the unified metamodel in Ecore for the hazard analysis data in different format, then convert automatically the unstructured data to the EMF models through Epsilon Object Language (EOL) [16].

Step 2, to generate upper structure of SCs by instantiation of EMF models of hazard analysis data. We need to design the SC pattern according to the system property, and then design the instantiation rule with Epsilon Transformation Language (ETL) [16] to link the elements in the SC pattern with the instantiable EMF models. Here, we refer to the model weaving method [9]. But we do not need to create a standalone pattern model as the pattern has been integrated into the instantiation rule. Also, there is no need to design a specific SC metamodel as we use SACM as the SC metamodel.

Step 3, to generate the lower structure of SCs by querying system design models. We refer the model query concept in [11] in this step. The query rule is designed based on the property to be argued, and needs to obey the metamodels of RoboChart and SC, and the SC pattern. The difference from [11] is that we execute the query in Eclipse instead of a specific system development environment that is only applicable to certain system modelling language such as the Open Source AADL Tool Environment (OSATE) for AADL. This independence from the specific system modelling environment will allow the wider scope of the applicability.

Step 4, the integration of the SC structures. We create and insert an identifier keyword in the raw data of hazard analysis, instantiation rule of Step 2, and the query rule of Step 3. Through this identifier, the position in SC structure where system model query is required can be automatically identified and used to link the two parts of the SC structures as a whole.

Our framework can provide an automatic solution covering the entire SC generation. Compared with Section 3, our framework may automate the data processing, streamline the process by removing the pattern modelling and the SC metamodel design. It also closes the gap by integrating the SC structures generated from both structured and unstructured data. The proposal may have a wide scope of applicability as it can be applied to any system as long as the models can be converted into EMF models. Moreover, the utilisation of SACM metamodel instead of GSN-based metamodel may make our solution compatible with the upcoming SACM based tools in future.

V. Conclusion and Future Work

SCs are generated and evolve along with the system development. The automation of SC process reduces the workload and chances of errors. We believe MBE is a solution for this purpose. The paper discusses different MBE methods of SC generation and the automation capability of each method. The research gaps are identified as lacking of automatic processing of raw instantiable data, and of a solution for generating a complete SC from both structured and unstructured system data. We propose an SACM compliant framework for SC generation to close the gap. In future, we will apply our approach to an autonomous underwater vehicle, and revise the framework based on the implementation results.

Acknowledgment

This research is funded by the EU Horizon 2020 European Training Network “Safer Autonomous Systems” (Grant Agreement 812788) and the EPSRC-UKRI fellowship project CyPhyAssure (Grant EP/S001190/1).

References

[1] Assurance Case Working Group, “GSN Community Standard. Version 2,” 2018.
[2] ISO, “ISO 26262 Road vehicles–Functional Safety, Version 1,” 2011.
[3] “Claims-Argument-Evidence-Adelard LLP,” https://www.adelard.com/asce/choosing-asce/cae.html, online; accessed 6th March, 2021.
[4] Object Management Group (OMG), “Structured Assurance Case Metamodel (SACM), Version 2.1 beta,” 2020.
[5] D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro, *EMF: eclipse modeling framework*. Pearson Education, 2008.
[6] OMG, “Meta object facility (mof) core specification,” 2019.
[7] T. P. Kelly and J. A. McDermid, “Safety case construction and reuse using patterns,” in *Safe Comp 97*. Springer, 1997, pp. 55–69.
[8] E. Denney and G. Pai, “Tool support for assurance case development,” *Automated Software Engineering*, vol. 25, no. 3, pp. 435–499, 2018.
[9] R. Hawkins, I. Habli, D. Kolovos, R. Paige, and T. Kelly, “Weaving an Assurance Case from Design: A Model-Based Approach,” in *2015 IEEE 16th International Symposium on High Assurance Systems Engineering*. IEEE, 2015, pp. 110–117.
[10] M. D. Del Fabro, J. Bézivin, and P. Valduriez, “Weaving models with the eclipse amw plugin,” in *Eclipse Modeling Symposium, Eclipse Summit Europe*, vol. 2006, 2006, pp. 37–44.
[11] A. Gacek, J. Backes, D. Cofer, K. Slind, and M. Whalen, “Resolute: an assurance case language for architecture models,” in *ACM SIGAda Ada Letters*, vol. 34, no. 3. ACM, 2014, pp. 19–28.
[12] M. Gleirscher, S. Foster, and Y. Nemouchi, “Evolution of Formal Model-Based Assurance Cases for Autonomous Robots,” *Lecture Notes in Computer Science*, vol. 11724 LNCS, pp. 87–104, 2019.
[13] T. Nipkow, L. C. Paulson, and M. Wenzel, *Isabelle/HOL: a proof assistant for higher-order logic*. Springer Science & Business Media, 2002, vol. 2283.
[14] Z. Diskin, T. Maibaum, A. Wassyng, S. Wynn-Williams, and M. Lawford, “Assurance via model transformations and their hierarchical refinement,” in *Proc. the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, 2018, pp. 426–436.
[15] A. Miyazawa, P. Ribeiro, W. Li, A. Cavalcanti, J. Timmis, and J. Woodcock, “Robochart: modelling and verification of the functional behaviour of robotic applications,” *Software & Systems Modeling*, vol. 18, no. 5, pp. 3097–3149, 2019.
[16] D. S. Kolovos, R. F. Paige, and F. A. Polack, “The epsilon transformation language,” in *International Conference on Theory and Practice of Model Transformations*. Springer, 2008, pp. 46–60.