The Eleventh International Conference on Performance, Safety and Robustness in Complex Systems and Applications
PESARO 2021
April 18, 2021 to April 22, 2021 - Porto, Portugal

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

Fang Yan(fang.yan@york.ac.uk), Simon Foster, Ibrahim Habli
Department of Computer Science, University of York

This project has received funding from the European Union's EU Framework Programme for Research and Innovation Horizon 2020 under Grant Agreement No. 812.788

About

Presenter

Fang Yan (fang.yan@york.ac.uk)

PhD student, early stage researcher at University of York.

Field of interest: Safety engineering and Model-based engineering

This project has received funding from the European Union's EU Framework Programme for Research and Innovation Horizon 2020 under Grant Agreement No. 812.788

Purpose of paper

- To present the state-of-the-art of Model-based Engineering applications to safety cases (SC) generation;
- To explores the research challenges and gaps, and
- To proposes a solution framework to address the gaps through the model transformation within the Eclipse Modeling Framework.

Background

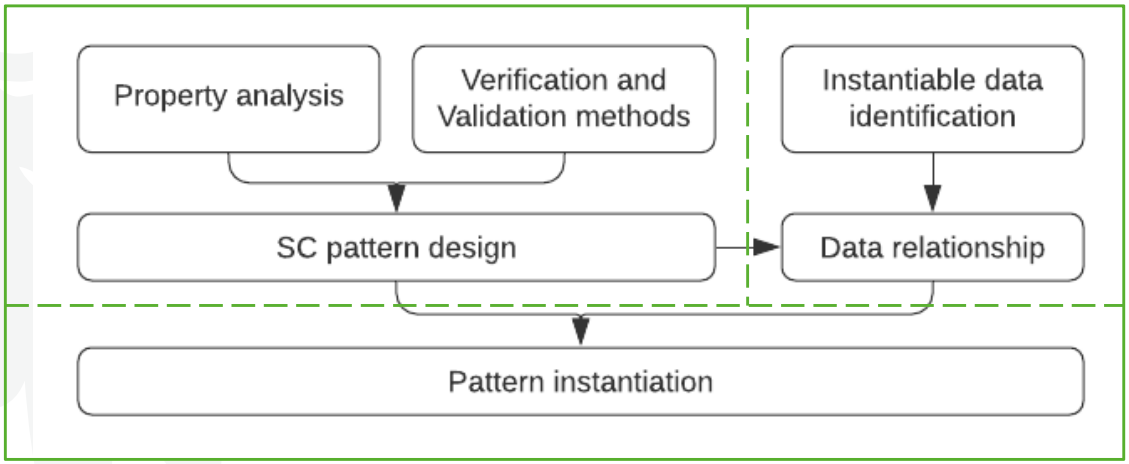
SCs are defined as compelling arguments, supported by evidence, that systems operate as intended for defined applications in defined environments [1],

- important to the safety-critical robotics and autonomous systems whose operational environments are relatively open and not sufficiently predictable during design.
- often bring repeated workload of SC evolution resulting from system development iteration due to its open environment

Model-based Engineering (MBE) as the technical solution for generation automation.

- applications varying in terms of the techniques exploited, the generation phases applied to, and the extent of automation, etc.
- No MBE solution for the whole engineering process of SC generation.

A common practice



A common SC generation process is threefold

- SC pattern design
  - an abstract structure containing placeholders to be instantiated by concrete argument elements.
- instantiable data management
  - to identify the data required for SCs
  - to identify the relationships between the data elements
  - to identify the relationships between the data and SC elements.
- pattern instantiation
  - To feed the system data into the in either a manual or automatic way

SC generation by pattern instantiation

The process 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.

Step 1, to build a SC metamodel.

Step 2, to design the SC pattern and create the pattern models.

Step 3, to identify and organize the instantiable data as a data table.

Step 4, to manually map the nodes of SC pattern with the data tables.

Step 5, to instantiate the pattern models according to the mapping.

Advantage : SC pattern models enable automatic instantiation, and the subsequent model management capabilities

Disadvantage: high workload due to manual updating of data mapping

Related work [2]

SC generation by pattern instantiation-variation

The process is similar to the above, but exploits model weaving [10] to establish the mapping between instantiable data models and SC pattern models at the metamodel level.

Step 1, same as above.

Step 2, same as above.

Step 3, to identify the instantiable models.

Step 4, to map the elements of SC pattern with the elements of the instantiable models at their metamodel level within a weaving model.

Step 5, to instantiate through the weaving model.

Advantage : automatic extraction of instantiable data from the system models

Limitation: the format of models are necessary for model weaving

Related work [3]

Integrated SC generation system model query

This method is to generate SC models by system model query.

Step 1, to design a Domain Specific Language (DSL)

Step 2, to formally define the top-level claim using DSL.

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

Advantage : the tight coupling of SCs with system models and ensures the automatic consistency of the two when design changes.

Limitation: DSL is specific and not applicable to other modelling languages; the claims do not involve the unstructured data including such as hazard and hazard causes, etc. Thus, the SCs generated are incomplete.

Related work [4]

SC generation by claim formalization and refinement

In this method, SC claims are formalized as a series of mathematical assertions about a system model, and inferred into the low level claims.

Step 1, to formalize the top claim as an assertion “M |= G under A”, M is the system model, A is an assumption, G is the guarantee on system model.

Step 2, to decompose the top claim by refining the system model through system development, weakening the assumptions, and decomposing or adding guarantees. The lower level claims are “M*j= G* under A*” , * means “refined”.

Step 3, to verify the correctness and completeness of the refinement by FM.

Advantage : the rigorous mathematical refinement checking of the inference.

Limitation: 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.

Related work [5][6]

Evaluation

- In the SC generation process[2], the non-automated processing of instantiation data brings a high workload of SC update. And the system model is not well integrated with SCs.
- The solutions of integration of the system models into SCs [3] [4] [5] [6] only create the lower structure of SCs.
- The model query [4] 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 [5] [6] requires FM expertise which may block the way of the engineering practical application.

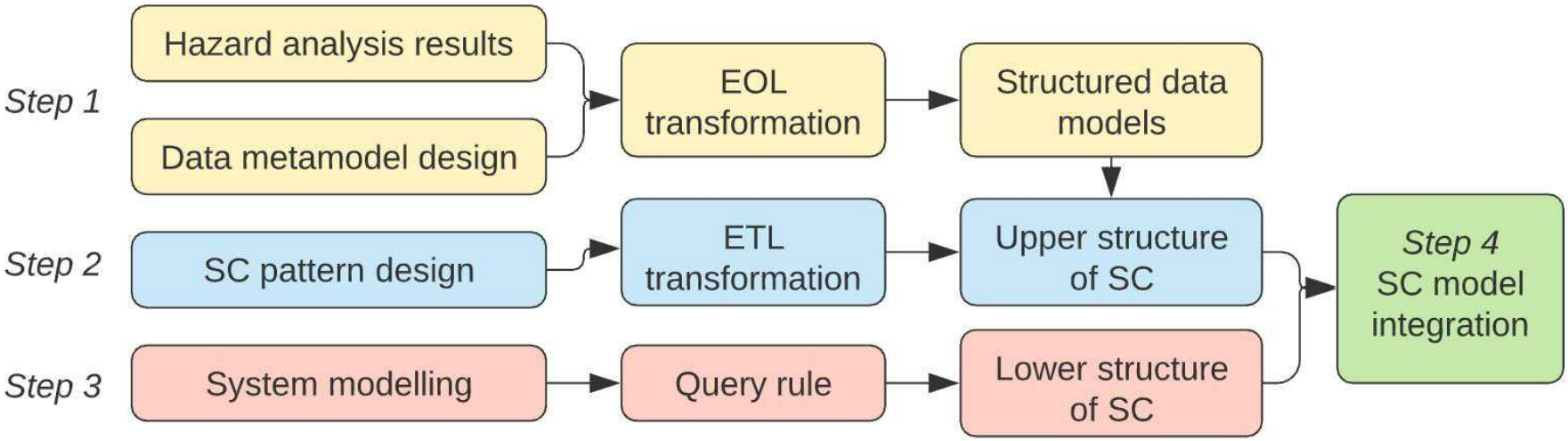
Evaluation

There is not an automatic solution fully covering the SC generation process with a wide application scope.

The gaps lie mainly in:

- a lack of an automatic way to process the unstructured instantiable data for MBE manipulation;
- the missing of integration of upper SC structure derived from hazard analysis and the lower SC structure from system models;
- a narrowed scope of applicability to the system development techniques.

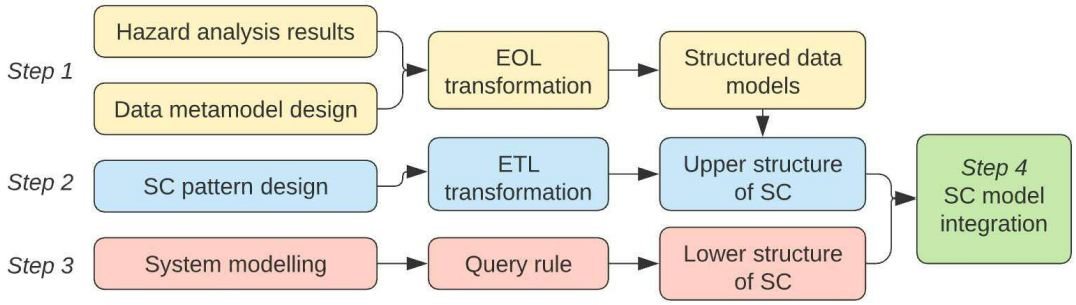
A proposal



An SACM compliant framework for SC generation combining the pattern instantiation based method and system model query based method.

- applied within the Eclipse EMF framework.
- instantiable data, system design, and SCs are all handled as EMF models.
- RoboChart as the system modelling language which is designed in Eclipse and can be exported as EMF models.

A proposal



Step 1 : Structured data model generation from hazard analysis result.
Step 2 : Upper structure generation of SCs by instantiation of EMF models of hazard analysis data.
Step 3 : Lower structure generation by querying system design models.
Step 4 : Integration of the SC structures.

The automatic solution covering the entire SC generation,

- automating the data processing,
- streamlining the process by removing the pattern modelling and the SC metamodel design
- integrating the SC structures generated from both structured and unstructured data. providing a wide scope of applicability for EMF models
- being compatible with SACM metamodel and the upcoming SACM based tools.

Conclusion

- 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.

References

[1] Assurance Case Working Group, “GSN Community Standard. Version 2,” 2018.

[2] E. Denney And G. Pai, “Tool Support for Assurance Case Development,” Automated Software Engineering, Vol. 25, No. 3, Pp. 435–499, 2018.

[3] 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.

[4] 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.

[5] 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.

[6] 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.