A Lightweight Approach to the Concurrent Use and Integration of SysML and Formal Methods in Systems Design

Robert Thorburn
Vladimiro Sassone
University of Southampton
robert.thorburn@soton.ac.uk

Asieh Salehi Fathabadi
Leonardo Aniello
Michael Butler
University of Southampton

Dana Dghaym
Thai Son Hoang
University of Southampton

ABSTRACT

Increased systems complexity and ubiquitous computing drive the need for improved systems design. Model-based systems engineering using general purpose languages such as SysML, is a well-established response to this challenge. However, for systems where correctness-by-construction is critical, formal methods are often also deployed. This is a significant undertaking often involving complete model translation. We address this problem by developing a novel requirements interchange system, presented as a SysML model library, to guide the concurrent use of SysML and formal models without requiring complete model translation.

CCS CONCEPTS

• **Computing methodologies → Model verification and validation**.

KEYWORDS

SysML, formal methods, model library, requirements engineering

ACM Reference Format:
Robert Thorburn, Vladimiro Sassone, Asieh Salehi Fathabadi, Leonardo Aniello, Michael Butler, Dana Dghaym, and Thai Son Hoang. 2022. A Lightweight Approach to the Concurrent Use and Integration of SysML and Formal Methods in Systems Design. In *ACM/IEEE 25th International Conference on Model Driven Engineering Languages and Systems (MODELS '22 Companion), October 23–28, 2022, Montreal, QC, Canada.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3550356.3559577

1 INTRODUCTION

The use of different modelling languages within a single project is an increasingly common occurrence, with each approach targeting a set application and level of abstraction [4]. Since no current modelling language can be all things to all users, a hybridised approach is a good alternative ensuring that all needed capabilities are provided to the larger project, albeit by more than one model. One prominent option, especially in systems design, is the Systems Modelling Language (SysML). While SysML models are intended to act as a single version of the "truth", that is a single point of reference for the entire system model [2], SysML cannot attend to matters such as guaranteeing component correctness, which is the domain of formal modelling. SysML can, however, integrate output from other models and systems. Accordingly, there is a clear case to make for the use of both SysML and formal modelling for the design and lifecycle management of a single system. Doing so though requires an explicit rule set for governing the interaction between the SysML and formal models [5].

We have experienced this need directly through work on a case study involving a smart ballot box (SBB), which stores and verifies paper ballots. The SBB contains a number of structural components, off-the-shelf components, and a Morello board with CheriBSD as operating system. Verifying the relationship between application-level security requirements and secure software implementations on this hardware is the main case study aim. The Morello board developed by ARM, deploys prototype architecture adapting the Capability Hardware Enhanced RISC Instructions (CHERI) architecture developed by the University of Cambridge. Cambridge also developed the CheriBSD operating system as a fork of FreeBSD. For our work, a SysML model can attend to the design and specification, including requirements derived from additional sources such as threat modelling, but it can not directly speak to correctness-by-construction for which formal methods must be employed. Full semantic translation of the model would, however, not only be a large undertaking but could also not yield any benefit beyond applying formal methods to only the needed critical subsystems.

Our contribution, therefore, is a novel and reusable system for linking SysML and formal models. This system is a lighter weight alternative to full semantic translation[1], includes the process of selecting formal modelling languages to use and is made available as a stand-alone model library.

2 DEVELOPING THE INTERCHANGE SYSTEM

The "*requirements interchange system*" (RIS) connects a SysML model to a formal model via bi-directional information flow, but does not include any assumptions about either model, making it reusable. As shown below, the RIS forms an operational loop with continual interaction between the two models which allows for iterative development across the system lifecycle and also governs synchronisation. A specifically problematic challenge to synchronisation is the prospect of parallel updating where either model can initiate changes the other is oblivious to, thereby bringing the models out of sync[3]. Although formalised translation can be deployed and may be a good option in certain circumstances, a simpler approach is to impose a fixed sequence for model changes. If the SysML model attends to the system as whole, while the formal model is focused on a subset of the system model it is possible to

MODELS '22 Companion, October 23–28, 2022, Montreal, QC, Canada
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9467-3/22/10.
https://doi.org/10.1145/3550356.3559577

[1]Which is traditionally employed for such work.

impose such a sequence. Here any changes needed in the subset are determined via requirements in the SysML model, then passed to the formal model for modelling and/or code generation before the results are passed back to the SysML model. Since the actual modelling for the subset is initiated in the SysML model but conducted in the formal model, parallel updating is prevented.

The model interchange requirements (MIR) directing the operation of the RIS are captured in a single SysML package and are:

- `MIR01` Formal Modelling Method Determination: The SysML model shall include documentation describing and justifying the formal modelling approach opted for.
- `MIR02` Formal Modelling Requirements Setup: The SysML model shall include dedicated packages for collecting, deriving and managing requirements passed to a formal model.
- `MIR03` Return Rationale: The SysML model shall, by way of either a «Rationale» note or preferably a linked document, described the expected revised requirements and source code returned from the formal model and how this is to be implemented into the SysML model.
- `MIR04` Model Synchronisation: Any relevant model changes shall be reflected in the MIR02 requirements, initiating the formal modelling process and updating the MIR03 process.
- `MIR05` Return Check: The return from the formal model shall be checked against the expected results of MIR03, with any variance triggering MIR04.
- `MIR06` No Parallel Updating: Updating the formal model shall proceed once requirements are passed to it via MIR04 and cease once revised requirements or source code is returned. («deriveReqt» from MIR04)
- `MIR07` Consistency Assessment: Consistency between the formal and SysML models shall be assessed, any variance addressed, and MIR04 triggered. («deriveReqt» from MIR04)

3 INTERCHANGE SEQUENCE

The RIS directs the iterative updating of both the SysML and formal models with the sequence for this process described in an activity diagram as presented in Figure 1. This figure lists all possible actions including those for starting the process and eventual code generation, which would of course not execute in every run.

The full sequence in a single run is as follows:

(1) Determine the formal modelling method to use and document this decision.
(2) Derive requirements to pass to the formal model.
(3) Develop rationale for feedback from the formal model, including how to act on said feedback.
(4) Ensure model synchronisation, including passing new requirements to the formal model and returning to step two, based on return data received from the formal model.
(5) Construct or update the formal model.
(6) Assure that the formal model is fit for purpose.
(7) Conduct model checking, assessing not only newly passed requirements but also the model as a whole.
(8) Determine if source code should be generated.
(9) Pass model checking results and source code (if generated) back to the SysML model.
(10) Implement revised requirements and/or source code.
(11) Check formal model returns against the expected return and move to step four if needed.

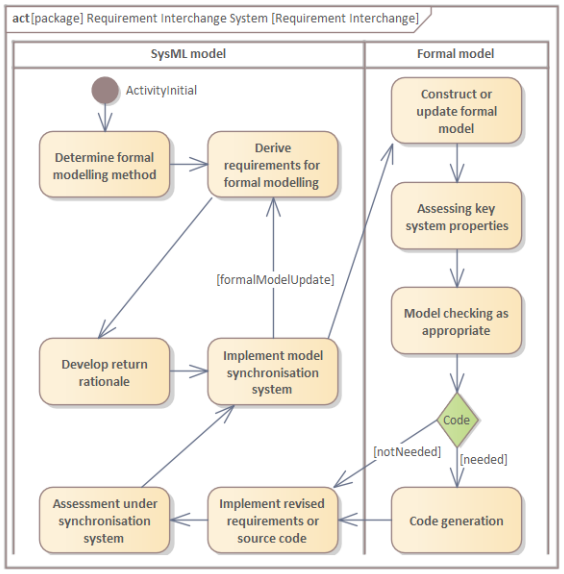
act [package] Requirement Interchange System [Requirement Interchange]
SysML model
Formal model
ActivityInitial
Determine formal modelling method
Derive requirements for formal modelling
[formalModelUpdate]
Develop return rationale
Implement model synchronisation system
Assessment under synchronisation system
Implement revised requirements or source code
Construct or update formal model
Assessing key system properties
Model checking as appropriate
Code
[notNeeded]
[needed]
Code generation

Figure 1: Activity diagram for the Requirement Interchange System.

4 CONCLUSIONS AND FUTURE WORK

In this paper we presented a lightweight requirements-based alternative to existing model translation approaches. By implementing the RIS, a SysML model and formal model can be linked without the need for extensive additional effort. Discussing inter-model interaction, [1] specifically credits the Object Management Group (OMG) [2] with generating interest in such interactions. The need addressed by the RIS is therefore both relevant and timely. The RIS is being extensively used by the authors and is under active development. The current version is available for download.[3]

ACKNOWLEDGMENTS

As part of the HD-Sec project, this work is funded by the Digital Security by Design (DSbD) Programme delivered by UKRI.

REFERENCES

[1] Krzysztof Czarnecki and Simon Helsen. 2006. Feature-based Survey of Model Transformation Approaches. *IBM systems journal* 45, 3 (2006), 621–645.
[2] Sanford Friedenthal, Alan Moore, and Rick Steiner. 2014. *A Practical Guide to SysML: the Systems Modeling Language* (3rd ed.). Morgan Kaufmann, Waltham.
[3] Lars Fritsche, Jens Kosiol, Adrian Möller, Andy Schürr, and Gabriele Taentzer. 2020. A Precedence-driven Approach for Concurrent Model Synchronization Scenarios Using Triple Graph Grammars. In *Proceedings of the 13th ACM SIGPLAN International Conference on Software Language Engineering*. 39–55.
[4] Holger Giese, Stephan Hildebrandt, and Stefan Neumann. 2010. Model Synchronization at Work: Keeping SysML and AUTOSAR Models Consistent. In *Graph transformations and model-driven engineering*. Springer, 555–579.
[5] Satoko Kinoshita, Hidekazu Nishimura, Hiroki Takamura, and Daichi Mizuguchi. 2014. Describing Software Specification by Combining SysML with the B Method. In *2014 IEEE International Symposium on Software Reliability Engineering Workshops*. IEEE, 146–151.

[2] The OMG maintains SysML and a number of other languages and systems.
[3] https://hd-sec.github.io/publications/