A Model-Driven Approach for Formally Verifying SysML-Based Dynamic Software Architectures

Camila Araújo[1,2]

[1]State University of Rio Grande do Norte, Natal, Brazil
[2]Federal University of Rio Grande do Norte, Natal, Brazil

Abstract

The critical nature of many complex software-intensive systems requires formal architecture descriptions towards better supporting automated architectural analysis regarding correctness properties. Due to the challenges of adopting formal approaches, many architects have preferred using notations such as UML, SysML, and their derivatives to describe the structure and behavior of software architectures. However, these semi-formal notations have limitations regarding the sought support for architectural analysis, particularly formal verification. This Ph.D. research investigates how to conciliate formal support and SysML-based architecture descriptions, to enable the formal verification of dynamic software architectures. The main contribution is proposing a model-driven approach that: (i) provides formal semantics to a SysML-based architectural language, SysADL, by transforming SysADL architecture descriptions in specifications expressed in π-ADL, a well-founded theoretically language based on the higher-order typed π-calculus, and (ii) enables the formal verification of properties for dynamic architectures. These facilities are integrated into an environment that facilitates modeling SysML architectures and formally verifying them.

Keywords

Dynamic software architecture, Architectural language, Formal verification, Model-driven development

1. Context

Dynamic software architectures focus on systems that operate on dynamic environments and are subjected to runtime changes. In this context, architectural analysis, i.e., the activity of identifying essential properties of the system using architectural models even before its implementation, is essential to avoid incorrectness, inconsistencies, and undesirable issues that can propagate until later phases of the development. This is even more essential when dealing with the critical nature of many complex systems whose architecture must be verified concerning their correctness and fulfillment of required behavior and properties of interest, as prescribed in security standards for critical software, such as RTCA/DO-178BC for avionics, IEC 62304 for medical systems, and IEC 62279 for railway systems, which strongly recommend the adoption of formal verification techniques as a means to ensure the security of these systems.

Considering the importance of formal verification in the context of dynamic software architectures as a way to assure quality in software systems, formal architectural descriptions are highly desirable as a means of better supporting automated architectural analysis, acknowledged as

ECSA'21: European Conference on Software Architecture - Doctoral Symposium, September 13–17, 2021, Vaxjo, Sweden
camilaaraujo@uern.br (C. Araújo)

ECSA2021 Doctoral Symposium Proceedings (ecsa2021.org)

an important activity in the software industry [1]. The main advantage of adopting a formal approach is precisely determining if a software system can satisfy properties and constraints related to requirements and check the accuracy and correctness of architectural designs.

For architectural modeling, the software industry is interested in easy-to-use languages with low learning curve [1]. To address this requirement, this work adopts SysADL [2], a SysML-based language that combines typical architectural language constructs using the popular diagrammatic notation based on the SysML Standard for modeling software-intensive systems. It complies with the ISO/IEC/IEEE 42010 International Standard [3] and has important features for describing dynamic software architectures [2]: (i) multi-view modeling (structural, behavioral, and executable); (ii) cross-view checking, validation, and execution of software dynamic architectures; and (iii) availability of a tool to enable software architects to describe software architectures either visually and/or textually [4]. However, SysADL does not include formal support. For this purpose, this work relies on π-ADL [5], a formal architectural language based on the π-calculus process algebra to describe software architectures, integrated with a set of architectural analysis tools. The choice of π-ADL as a supporting formalism for the proposal has two main advantages: (i) the semantic context is preserved with the use of a vocabulary known to users, e.g., components, connectors, ports, and connections; and (ii) the formal verification of software architecture properties through the π-ADL toolchain [6].

2. Problem Statement

In the literature, some studies focus on investigating the needs of the software industry concerning architectural languages [1]. Most users of architectural languages: (i) need to perform some analysis in architectural descriptions; (ii) prefer automatic analysis with tool support; (iii) point out a lack of interest due to the abstractness of architectural descriptions and complexity of their formal description. Nonetheless, the relevance of formal architecture descriptions is well known, mainly for supporting automated architectural analysis.

Software architects generally prefer to use semi-formal notations such as UML, SysML, or their derivatives to describe the structure and behavior of software architectures. This choice was primarily driven by the lower learning curve, visuality, and general-purpose scope. However, this type of notation has known limitations regarding automatically checking architectural properties, lack of well-defined semantics, and gaps in complex design decisions. Formal approaches fill these gaps and better support architectural analysis. Previous work on a systematic literature review [7] highlighted some significant lacks regarding the formal verification of software architecture descriptions. Some of these lacks identified in the literature and that are addressed in this doctoral research are described in the following.

Lack of approaches proposing a seamless formal enrichment of non-formal languages. A formal enrichment is required for a non-formal language to support an architectural analysis process. Most existing studies encourage using some formalisms to provide formal support and specify properties to be verified. However, formally describing software architectures is very complex, with a high learning curve for architects [1]. There is a gap concerning proposals with a seamless formal enrichment of non-formal languages to reduce the learning curve associated with formalisms, especially in describing dynamic architectures.

Lack of support for formally verifying properties in SysML-based languages enabling dynamic software architectures. The software industry has adopted SysML as a standard for describing software-intensive system architectures, especially for its simplicity and low learning curve. Architects working with architecture modeling for critical systems are interested in performing different types of analysis in the architecture, preferably in an automatic way [1]. However, non-formal languages are not directly suitable for formal architectural analysis. The literature reports some approaches on associating the ease of SysML-based descriptions with the possibility of formal property verification using some mathematical formalism [8] or a formal language [9], but they often do not consider essential features such as general-purpose architectural languages and tool support. In particular, there is also a lack of approaches with such a formal support for dynamic architectures.

Lack of tools for specifying and verifying properties in dynamic architectures. Tools provide an essential support to software architects as they contribute to advance the architecture design practice by supporting a syntactically correct model, reusing standard architectural elements, and executing simulation at design time. In the context of formal property verification in dynamic architectures, tools have an equally relevant supporting role since formal verification in manual settings is an impossible task, especially considering the size of current architectures that have thousands of components. Therefore, integrated environments and tools, support mechanisms, and technologies able to transparently dealing with architectural descriptions and verification of architectural properties is an essential gap to be filled.

3. Research Goals

This doctoral research aims to define an approach to add formal semantics to a SysML-based ADL and support the formal property verification in dynamic software architecture description. This goal can be achieved by answering the following set of research questions.

RQ1: Can a Model-Driven Development (MDD)-based approach enable a seamless formal enrichment of SysADL models? This RQ investigates whether an MDD-based approach can enable a seamless formal enrichment of SysADL. The idea is to establish a mapping process between SysADL and π-ADL to define the correspondences between the elements of the languages. Implementing this transformation based on the denotation semantics established between SysADL and π-ADL models ensures the semantic equivalence between the models. The main interest behind the idea of transforming an architecture description in SysADL to a corresponding one in π-ADL is allowing for its further formal verification, which is already available for π-ADL [6].

RQ2: Can property specifications associated with the SysADL model (translated to π-ADL) be formally verified with the π-ADL toolchain? Expressing properties concerning a dynamic software architecture needs to consider architectural elements that are created or removed at runtime, i.e., they may exist at a specific moment in time and no longer exist at another. DynBLTL [6] is a logic and notation that aims to express properties in dynamic software architectures and is designed to deal with the absence of an architectural element in a given limited formula that expresses a property. The PLASMA plug-in for π-ADL performs formal verification of defined properties using DynBLTL. This RQ aims to investigate whether the model transformation between SysADL and π-ADL models, associated with their set of properties specified in DynBLTL,

can be formally verified using the π-ADL toolset.

RQ3: Does the integration of SysADL Studio with the π-ADL formal analysis tools enable both specification and formal verification properties in dynamic architectures? This RQ investigates if the integration of SysADL Studio tool [4] and the PLASMA plug-in for π-ADL [6] enables both the specification and formal verification of properties in dynamic architectures. PLASMA embraces a toolchain for statistical model checking of DynBLTL properties, a strategy chosen to deal with the classic states explosion problem associated with model checking.

4. Proposed Approach

Aiming to answer the research questions, the main contribution of this work is to propose an MDD approach that conciliates a SysML-based architecture language, SysADL, with the formal support of π-ADL, towards the specification and formal verification of dynamic software architecture properties. The proposed approach is structured from the elements represented in Figure 1 and briefly described as follows.

To answer RQ1, a model-to-model (M2M) transformation between SysADL and π-ADL is defined. The M2M transformation consists of specifying rules to produce target models from source models by mapping their respective elements as defined in their metamodels. This work uses the SysADL and π-ADL metamodels as a source and target metamodels, respectively. The goal is to transform an input architecture in SysADL and produce an architecture in π-ADL. The transformation algorithms and implementation are built by following the definition of the denotational semantics of SysADL as a function of π-ADL.

To answer RQ1, RQ2, and RQ3, this work proposes a process illustrated in Figure 1. The architectural description in SySADL is provided as input (1) to a mechanism that automates their transformation into π-ADL models (2), based on denotational semantics. The architectural description is transformed into π-ADL (3), together with the DynBLTL (3) specification of architectural properties to be checked by the π-ADL toolchain.

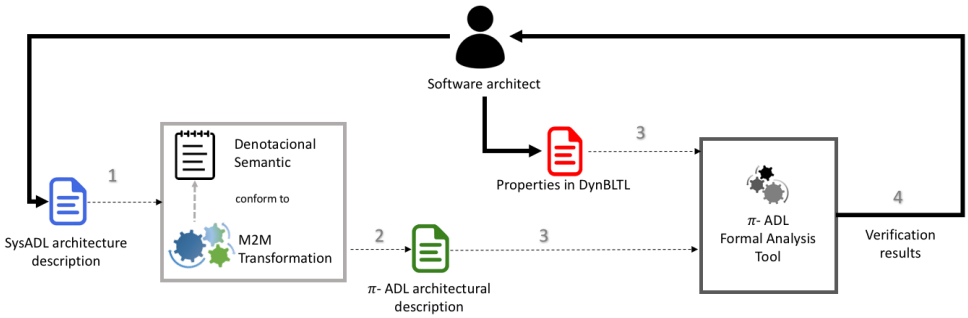

Figure 1: Process for generating a π-ADL architecture description from a SysADL model.

5. Research Activities

Ongoing Activities. An ATL implementation of M2M transformation between SysADL and π-ADL models has already been completed. However, it is necessary to complete the denota-

tional semantics of SysADL in the function of π-ADL as a formal proof that this transformation is possible. The definition of the denotational semantics is currently an ongoing activity. Another ongoing activity related to the M2M transformation between SysADL and π-ADL is the integration of the ATL transformation with the SysADL Studio tool.

Next Activities. The next activities towards the accomplishment of the goals of this research encompass: (i) the integration the π-ADL toolchain to SysADL Studio; (ii) the definition of classes of architectural properties that will be used to evaluate the approach; (iii) the development of proofs-of-concept to validate the approach; and (iv) quantitative and qualitative evaluations of the main elements of the proposed approach.

6. Expected Contributions

The main contribution of this Ph.D. research is to define an MDD approach that provides a seamless transformation of SysADL architecture descriptions to corresponding formal specifications in π-ADL, which can be formally verified. Associated contributions are:(i) a denotational semantics to SysADL as a function of π-ADL; (ii) the definition of algorithms to transform SysADL models into π-ADL models; and (iii) an integrated environment for dynamic architecture modeling in SysADL that supports formal property verification.

References

[1] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, A. Tang, What industry needs from architectural languages: A survey, IEEE Trans. on Software Engineering 39 (2013) 869–891.

[2] F. Oquendo, J. Leite, T. Batista, Software Architecture in Action: Designing and executing architectural models with SysADL grounded on the OMG SysML Standard, Springer International Publishing, Switzerland, 2016.

[3] ISO/IEC/IEEE 42010(E), Systems and software engineering – Architecture description, ISO, Switzerland, 2011.

[4] J. Leite, et al., Designing and executing software architectures models using SysADL Studio, in: 2018 IEEE Int. Conf. on Software Architecture Companion, IEEE, USA, 2018, pp. 81–84.

[5] F. Oquendo, π-ADL: An architecture description language based on the higher-order typed π-calculus for specifying dynamic and mobile software architectures, ACM SIGSOFT Software Engineering Notes 29 (2004) 1–14.

[6] E. Cavalcante, et al., Statistical model checking of dynamic software architectures, in: B. Tekinerdogan, U. Zdun, A. Babar (Eds.), ECSA 2016, volume 9839 of *LNCS*, Springer, Switzerland, 2016, pp. 185–200.

[7] C. Araujo, et al., A research landscape on formal verification of software architecture description, IEEE Access 7 (2019).

[8] L. Lima, et al., An integrated semantics for reasoning about sysml design models using refinement, Softw. Syst. Model. 16 (2017) 875–902.

[9] S. R. Taoufik, B. M. Tahar, K. Mourad, Behavioral Verification of UML2.0 Software Architecture, Proc. 12th Int. Conf. on Semantics, Knowledge and Grids, SKG (2017).