2-2016

Formalizing and verifying stochastic system architectures using Monterey Phoenix

Songzheng SONG

Jiexin ZHANG

Yang LIU

Mikhail AUGUSTON

Jun SUN
Singapore Management University, junsun@smu.edu.sg

See next page for additional authors

Citation

SONG, Songzheng; ZHANG, Jiexin; LIU, Yang; AUGUSTON, Mikhail; SUN, Jun; DONG, Jin Song; and CHEN, Tieming. Formalizing and verifying stochastic system architectures using Monterey Phoenix. (2016). *Software and Systems Modeling*. 15, (2), 453-471. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/4970

Author

Songzheng SONG, Jiexin ZHANG, Yang LIU, Mikhail AUGUSTON, Jun SUN, Jin Song DONG, and Tieming CHEN

Formalizing and Verifying Stochastic System Architectures Using Monterey Phoenix (SoSyM Abstract)

Songzheng Song*, Yang Liu*, Mikhail Auguston†, Jun Sun‡, Jin Song Dong§, and Tieming Chen¶

*Nanyang Technological Univerisity, Singapore
†Naval Postgraduate School, USA
‡Singapore University of Technology and Design, Singapore
§National University of Singapore, Singapore
¶Zhejiang University of Technology, China

***Abstract*—The analysis of software architecture plays an important role in understanding the system structures and facilitate proper implementation of user requirements. Despite its importance in the software engineering practice, the lack of formal description and verification support in this domain hinders the development of quality architectural models. To tackle this problem, in this work, we develop an approach for modeling and verifying software architectures specified using Monterey Phoenix (MP) architecture description language. MP is capable of modeling system and environment behaviors based on event traces, as well as supporting different architecture composition operations and views. First, we formalize the syntax and operational semantics for MP; therefore, formal verification of MP models is feasible. Second, we extend MP to support shared variables and stochastic characteristics, which not only increases the expressiveness of MP, but also widens the properties MP can check, such as quantitative requirements. Third, a dedicated model checker for MP has been implemented, so that automatic verification of MP models is supported. Finally, several experiments are conducted to evaluate the applicability and efficiency of our approach.**

I. Introduction

This is an extended abstract for the Models 2015 Conference of the journal paper of the same name [1]. Monterey Phoenix (MP) is a framework for software system architecture and related workflow modeling with the focus on behavior of software system and its environment [2], [3], [4]. When in traditional architecture models the main elements are components (representing the functionality), and connectors (representing the information flow between components), in MP the main concepts are behaviors and interactions between behaviors. Behavior is defined as a set of events (event trace) with two basic relations: precedence and inclusion, and the structure of possible event traces is specified using event grammar and composition operations organized into schemas. Events may have attributes, for example, timing.

The essential MP feature is event coordination abstraction for modeling interactions within the system. The generative event grammar concept provides a lightweight operational semantics definition for the computing system under development. The approach leads to executable architecture specifications with following features.

1) System developers are guided from the very beginning to think about system's behavior and interactions.
2) Environment's behavior is an integral part of architecture model. MP provides a uniform method for modeling behaviors of the software, hardware, business processes, and other actors. This emphasizes the role of architecture as a bridge between the system requirements and design.
3) Event grammar formalism supports exhaustive generation of system's behavior examples (Use Cases). Humans can understand and evaluate examples better then general descriptions, and most flaws in models could be demonstrated on relatively small counterexamples.
4) Executable architecture models may be used for assessment of non-functional requirements, such as performance, latency, and throughput based on systematic scenario generation (within a given scope) or on statistical simulation.
5) Multiple viewpoints (including diagrams) can be extracted from the same architecture model to facilitate the interaction with stakeholders.
6) MP framework is amenable to architecture reuse. Typical architecture styles and design patterns can be formalized and reused.
7) MP framework can supplement and enhance standard architecture frameworks, like UML, SysML, DoDAF.

This paper extends MP to support stochastic characteristics of behavior, and to support early system safety and security assessment. A dedicated model checker for MP has been implemented using PAT tool [5], [6]. Currently MP editor and trace generator is available at http://firebird.nps.edu/.

References

[1] S. Song, J. Zhang, Y. Liu, M. Auguston, J. Sun and J. S. Dong and T Chen, ”Formalizing and verifying stochastic system architectures using Monterey Phoenix”, Software & Systems Modeling, pp.1–19, April 2014.
[2] M. Auguston, “Software Architecture Built from Behavior Models”, ACM SIGSOFT Software Engineering Notes, vol. 34, no. 5, 2009.
[3] M. Auguston, “Monterey Phoenix, or How to Make Software Architecture Executable”, in OOPSLA, pp.1031–1038, 2009.
[4] M. Auguston, “Behavior Models for Software Architecture”, Technical Report NPS-CS-14-003, Naval Postgraduate School, http://calhoun.nps.edu/handle/10945/43851, 2014.
[5] J. Sun and Y. Liu and J. S. Dong and J. Pang, “PAT: Towards Flexible Verification under Fairness”, in CAV, pp. 709–714, 2009.
[6] Y. Liu, J. Sun, and J. S. Dong. “Pat 3: An Extensible Architecture for Building Multi-domain Model Checkers”. In ISSRE, pages 190C199, 2011.