Towards Scenario-Based Design and Verification of Resilient Cyber-Physical Systems

Extended Abstract

Rajeev Alur[1], Insup Lee[1], Rahul Mangharam[2], Mayur Naik[1], Oleg Sokolsky[1], James Weimer[1], Houssam Abbas[2]

I. Introduction

A cyber-physical system consists of computing devices communicating with one another and interacting with the physical world via sensors and actuators. Increasingly, such systems are everywhere, from smart buildings to autonomous vehicles to mission-critical military systems. Model-based design offers a promising approach for assisting developers to build cyber-physical systems in a systematic manner. In this methodology, a designer first constructs a model, with mathematically precise semantics, of the system under design, and performs extensive analysis with respect to correctness requirements before generating the implementation from the model [1], [2], [3]. However, as new vulnerabilities are discovered, requirements evolve aimed at ensuring resiliency. Current methodology demands an expensive, and at times infeasible, redesign and reimplementation of the system from scratch. The goal of the proposed methodology and the associated toolkit, which we call REAFFIRM, is to facilitate integration of evolving resiliency requirements in model-based design and verification.

Traditionally a model of a cyber-physical system consists of block diagrams describing the system architecture and a combination of state machines and differential equations describing the system dynamics [4]. Building a behavioral model at design time that offers resiliency for all kinds of failures is notoriously difficult. The REAFFIRM solution to design for resiliency is to allow a designer to specify *scenarios* as a separate part of the model description. A scenario describes a finite execution of the system corresponding to a specific situation, and consists of the sequences of actions by different agents including interaction among them. Such scenarios were first used in design of telecommunication software to specify different features separately, and were formalized. Our insight is that scenarios can be used naturally to describe how a system should respond when a particular sensor fails or a previously unanticipated attack is discovered. Furthermore, *negative* scenarios can express undesirable, or *shall not*, situations. The model synthesizer can automatically integrate scenarios with state-machine-based models thus allowing incremental design to support resiliency. The central research challenge is to come up with a formal and usable notion of scenarios for cyber-physical systems so that it allows specification of timing constraints and dynamic evolution of physical environment and is customizable to different system architectures.

II. The REAFFIRM Vision

The REAFFIRM solution aims to offer a fresh approach to integrate resiliency in model-based design of cyber-physical systems based on *scenarios*. The REAFFIRM vision consists of developing a toolkit that incorporates the following innovations.

- **Scenarios as an integral part of the model:** We advocate the idea that a model of the system under design consists of two parts: (1) a traditional model using block diagrams and state machines, and (2) a collection of positive and negative scenarios that correspond to resiliency requirements. As new vulnerabilities are discovered and requirements evolve, the scenarios get updated allowing iterative model-based design for resiliency in a convenient manner.
- **Model repair using resiliency patterns:** The task of the model synthesizer is to consistently integrate the original state-machine based model and scenarios to produce an updated resilient behavioral model. We propose a technique based on searching through the space of potential edits to the original model to solve this computational problem. This search will be guided by known patterns for ensuring resiliency. Examples of such patterns include switching to a safe mode of operation and adding redundant modes of operation.
- **Comprehensive verification:** Our methodology allows the use of formal verification with respect to requirements at multiple stages. The complete model generated by the model synthesizer can be verified using a static verification tool; the implementation can be monitored for violations using a runtime verification tool; and the scenarios themselves can be checked for mutual consistency and missing cases. The proposed scenario analyzer examines scenarios and counterexamples produced by verification tools to produce feedback to the designer.

*This work supported by the DARPA CASE program through SPAWAR Contract N66001-18-C-4007.

[1]Alur, Lee, Naik, Sokolsky, and Weimer are with the Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104, USA {alur, lee, mhnaik, sokolsky, weimerj}@cis.upenn.edu

[2]Mangharam and Abbas are with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104, USA rahulm@seas.upenn.edu

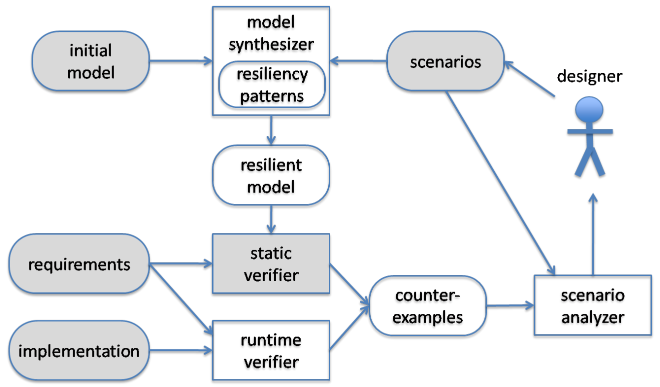

Fig. 1. Overview of Scenario-based Design for Resilient CPS

III. Towards a REAFFIRM Toolkit

The REAFFIRM toolkit will consist of three tools: (i) *Model Synthesizer*; (ii) *Runtime Verifier*; and (iii) *Scenario Analyzer*, shown in Figure 1. In the following we describe our vision for each tool and their integration.

A. Model Synthesizer

The Model Synthesizer takes as input an original (partial) behavioral model and a set of positive and negative scenarios and outputs a complete behavioral model consistent with both the views. Communicating state machines and scenarios can be viewed as two dual views of the desired functionality. The task of the model synthesizer is to *consistently* integrate these two views to produce a complete state-machine-based behavioral model. We will leverage our past work in the domain of distributed protocols [5], [6] to design and implement such a synthesis tool for cyber-physical systems. Intuitively, compared to the partial state machines, the completed model can have additional modes of operation as well as new transitions between different modes, and as a result has resilient behaviors captured in scenarios. Realizing the model synthesizer requires developing a technique for searching through the space of potential edits to the original model to solve the model synthesis problem. This search will be guided by known patterns for ensuring resiliency. For example, a conservative way ensuring safety upon encountering an unexpected or hazardous situation is to hand over control to a baseline safety controller.

B. Runtime Verifier

The Runtime Verifier takes as input the implementation and requirements/scenarios and produces counterexamples. The model generated by the synthesizer can be verified for correctness with respect to requirements. As an illustrative case, the requirements are specified in a timed temporal logic, the generated model is a hybrid automaton, and in such a case, we will use off-the-shelf verifiers for hybrid systems such as DReach [7] and S-TaLiRo [8]. We have established a formal relationship between reachability and falsification as implemented in these two tools [9], and demonstrated an early application of this relationship in [10]. In addition, we will explore simulation-guided techniques for runtime verification based on dynamical models [11]. The counterexample generated by the verifier is then analyzed by the scenario analysis tool, described in the following subsection, to give feedback to the designer. To complement static verification, the tool-chain also includes a runtime verifier that analyzes the code generated from models. Building on the long-term technology for runtime monitoring of code with respect to temporal properties [12], the challenge is developing a runtime verifier suitable for checking timing and resiliency requirements, expressed as temporal logic formulas or positive and negative scenarios, of cyber-physical system implementations.

C. Scenario Analyzer

The final and key part of the proposed methodology is the scenario analyzer that analyzes the scenarios provided by the designer as well as the counterexamples generated by the verifiers to provide meaningful feedback to the designer. Note that the original scenarios can be checked for mutual consistencies and with respect to requirements before the model generation phase. Developing the scenario analyzer requires identifying designer feedback mechanisms for (1) identification of missing cases prompting the designer to provide additional scenarios, (2) identification of root causes for inconsistencies and violation of requirements, and (3) suggested patterns for scenarios for common ways of ensuring resiliency.

References

[1] E. A. Lee, "What's ahead for embedded software," *IEEE Computer*, pp. 18–26, 2000.
[2] T. Henzinger and J. Sifakis, "The embedded systems design challenge," in *FM 2006: 14th International Symposium on Formal Methods*, ser. LNCS 4085, 2006, pp. 1–15.
[3] R. Alur, *Principles of Cyber-Physical Systems*. MIT Press, 2015.
[4] R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoretical Computer Science*, vol. 138, pp. 3–34, 1995.
[5] A. Udupa, A. Raghavan, J. Deshmukh, S. Mador-Haim, M. Martin, and R. Alur, "TRANSIT: specifying protocols with concolic snippets," in *ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2013, pp. 287–296.
[6] R. Alur and S. Tripakis, "Automatic synthesis of distributed protocols," *SIGACT News*, vol. 48, no. 1, pp. 55–90, 2017.
[7] S. Kong, S. Gao, W. Chen, and E. M. Clarke, "dreach: delta-reachability analysis for hybrid systems," in *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, Proceedings*, ser. LNCS 9035. Springer, 2015, pp. 200–205.
[8] Y. Annpureddy, C. Liu, G. E. Fainekos, and S. Sankaranarayanan, "S-TaLiRo: A tool for temporal logic falsification for hybrid systems," in *Tools and Algorithms for the Construction and Analysis of Systems - 17th International Conference, Proceedings*, ser. LNCS 6605. Springer, 2011, pp. 254–257.
[9] H. Abbas, M. O'Kelly, and R. Mangharam, "Relaxed decidability and the robust semantics of metric temporal," in *Hybrid Systems: Computation and Control*, 2017.
[10] M. O'Kelly, H. Abbas, and R. Mangharam, "Computer-aided design for safe autonomous vehicles," in *Resilience Week*, 2017.
[11] S. Chen, O. Sokolsky, J. Weimer, and I. Lee, "Data-driven adaptive safety monitoring using virtual subjects in medical cyber-physical systems: a glucose control case study," *Journal of Computer Science and Engineering*, vol. 10, no. 3, p. 75, 2016.
[12] M. Kim, M. Viswanathan, S. Kannan, I. Lee, and O. Sokolsky, "Java-MaC: A run-time assurance approach for java programs," *Formal Methods in System Design*, vol. 24, no. 2, pp. 129–155, 2004.