First Steps Towards a Wise Development Environment for Behavioral Models

David Harel, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel
Guy Katz, Computer Science Department, Stanford University, Stanford, CA, USA
Rami Marelly, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel
Assaf Marron, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel

ABSTRACT

The authors present an initial wise development framework: a development environment that proactively and interactively assists the software engineer in modeling complex reactive systems. Their framework repeatedly analyzes models of the system under development at various levels of abstraction, and then reasons about these models in order to detect possible errors, to derive emergent properties of interest, and to assist in system testing and debugging. Upon request, the environment can instrument the system model in order to monitor or test the execution for certain behaviors, or even augment it in order to repair or avoid detected behavior that is undesired. The direction and prioritization of the analysis and related tasks is based on the relevance of the observed properties and the expected impact of actions to be taken, and is performed by specialized automated and human-assisted techniques that have been incorporated into the framework. The authors’ development environment is an initial step in the direction of their recent Wise Computing vision, which calls for turning the computer (namely, the development environment) into an equal member of the development team: knowledgeable, independent, concerned and proactively involved in the development process. They have implemented their tool within the context of behavioral programming (BP) – a scenario-based modeling approach, in which components are aligned with how humans often describe desired system behavior. The authors’ work thus further enhances the naturalness and incrementality of developing in BP.

KEYWORDS

Behavioral Models, Interactive Development, Proactive Analysis, Reactive Models, Wise Computing

1. INTRODUCTION

The development of large reactive software systems is an expensive and error-prone undertaking. Deliverables will often fail, resulting in unintended software behavior, exceeded budgets and breached time schedules. One of the key reasons for this difficulty is the growing complexity of many kinds of reactive systems, which increasingly prevents the human mind from managing a comprehensive picture of all their relevant elements and behaviors. Moreover, of course, the state-explosion problem typically prevents us from exhaustively analyzing all possible software behaviors. While major advances in modeling tools and methodologies have greatly improved our ability to develop reactive systems by allowing us to reason on abstract models thereof, specific solutions are quickly reaching their limits, and resolving the great difficulties in developing reliable reactive systems remains a major, and critical, moving target.
Over the years it has been proposed, in various contexts, e.g., (Cerf, 2014; Harel, Katz, Marelly, & Marron, 2015; Reubenstein & Waters, 1991; Rich & Waters, 1988), that a possible strategy for mitigating these difficulties could lay in changing the role of the computer in the development process. Instead of having the computer serve as a tool, used only to analyze or check specific aspects of the code as instructed by the developer, one could seek to actually transform it into a member of the development team — a proactive participant, analyzing the entire system and making informed observations and suggestions. This way, the idea goes, the computer’s superior capabilities of handling large amounts of code could be manifested. Combined with human insight and understanding of the system’s goals, this synergy could produce more reliable and error-free systems.

In this paper we follow this spirit, and present a methodology and an interactive framework for the modeling and development of complex reactive systems, in which the computer plays a proactive role. Following the terminology of (Harel, Katz, Marelly, & Marron, 2015), and constituting a very modest initial effort along the lines of the Wise Computing vision outlined there, we term this framework a wise framework. Intuitively, a truly wise framework should provide the developer with an interactive companion for all phases of system development, “understand” the system, draw attention to potential errors and suggest improvements and generalizations; and this should be done via two-way communication with the developer, which will be very high-level, using natural (perhaps natural-language-based) interfaces. The framework presented here is but a first step in that direction, and focuses solely on providing an interactive development assistant capable of discovering interesting properties and drawing attention to potential bugs; still, it can already handle non-trivial programs, as we later demonstrate through a case-study.

Various parts of this approach have been implemented by a variety of researchers in other forms, as described in the Related Work section. A main novel aspect of our approach, however, is in the coupling of the notion of a proactive and interactive framework with a modeling language called behavioral programming (Harel, Marron, & Weiss, 2012) — a scenario-based language, in which systems are modeled as sets of independent scenarios that are interleaved at runtime. This formalism makes it possible for our interactive development framework to repeatedly and quickly construct abstract executable models of the program, and then analyze them in order to reach meaningful conclusions. It is now widely accepted that a key aspect in the viability of analysis tools and environments is that they are sufficiently lightweight to be integrated into the developer’s workflow without significantly slowing it down (Cristiano et al., 2015; Sadowski, Gogh, Jaspan, Söederberg, & Winter, 2015). We attempt to achieve this by leveraging scenario-based modeling. As demonstrated in later sections, the proactiveness of our approach and its tight integration into the development cycle can lead to early detection of bugs during development, when they are still relatively easy and cheap to fix.

The rest of this paper is organized as follows. In the second section we introduce scenario-based programming — the modeling formalism on top of which our approach is implemented, and also discuss some analysis techniques for scenario-based programs that are used in subsequent sections. In the third section we introduce our development framework by means of a simple example. In the fourth section we discuss the various components of the framework in more detail, and in the fifth we describe a case-study that we conducted. We then provide a discussion of related work and a conclusion.

2. SCENARIO-BASED MODELING

Behavioral programming (BP) (Harel, Marron, & Weiss, 2012) is a modeling approach aimed at designing and incrementally developing reactive systems. BP emerged from the live sequence charts (LSCs) formalism (Damm & Harel, 2001; Harel & Marelly, 2003), and, like LSCs, its basic modeling objects are scenarios. A behavioral model consists of independent scenario objects, each encoding a single desired or undesired behavior of the system. These behavioral models are executable: when run, the behaviors encoded by their constituent objects are all interwoven together, in a way that yields cohesive system behavior.
Related Content

Investigation of ANFIS and FFBNN Recognition Methods Performance in Tamil Speech Word Recognition

Security Issues of CPS
www.igi-global.com/chapter/security-issues-of-cps/121254?camid=4v1a
Facilitating Biodefense Research with Mobile-Cloud Computing
www.igi-global.com/article/facilitating-biodefense-research-mobile-cloud/58510?camid=4v1a

The Impact of eXtreme Programming on Maintenance
www.igi-global.com/chapter/impact-extreme-programming-maintenance/4899?camid=4v1a