Book Review for the STVR Journal

Review of *Modeling and Verification Using UML Statecharts*
by Doron Drusinsky, Newnes Publishers, 2006.

Reactive controller programming is a difficult and significant task within embedded systems engineering, and is a multi-billion pound business. Despite its importance, surprisingly few books are available on this topic. So seeing this new book by Doron Drusinsky gave me a burst of excitement and hope, both for academics who are looking for a good textbook for their students, and for engineers who need to get to grips with modern tools for embedded software development.

Drusinsky has worked in industry for many years, mostly on designing statechart-based development tools. A few years ago, he took up an academic post and now teaches computer science and software engineering at the Naval Postgraduate School in California. This mix of industrial and academic experience should give Drusinsky just the right background for writing a book on reactive controller programming and verification.

However, first things first. The title of the book is somewhat misleading, showing foot stamps of the publisher's marketing department. Drusinsky's book is not about modelling and verification *using UML Statecharts*, but about programming and verification *using StateRover™*. StateRover is a graphical programming tool developed by Drusinsky at his Time-Rover company. It supports statecharts and flowcharts for the high-level programming of complex real-time software, automatic Java code generation and run-time verification of temporal assertions. As such, the book employs a new statechart dialect *based on* UML Statecharts, which has several unique syntactical and semantical features.

Drusinsky's book also advocates *Statechart Assertions*. These allow for writing programs and their desired temporal properties within the same language, although nondeterminism is only permitted in the latter. The concept of Statechart Assertions is close to the one of observer automata which is deployed in competing design tools, such as Esterel Technologies' SCADE™ or Reactive System's REACTIS® validator for The MathWorks' Simulink/Stateflow® tool. In contrast to these tools, however, StateRover does not support model checking of temporal properties at compile-time, but instead execution-based model checking at run-time or simulation-time.

The book is structured in six chapters:

- *Chapter 1* summarises selected topics of automata and formal language theory, including variants of finite automata and finite state machines, conversions between different kinds of automata, and succinctness. Drusinsky attempts to convey these basics to practitioners by adopting some of their language. But is "domain of discourse" really a more comprehensible term for "alphabet"?

- *Chapter 2* introduces the statechart dialect implemented in StateRover, illustrates Java code generation from such statecharts, and discusses some non-standard statechart features such as flowchart elements and critical regions. Particular attention is paid to practicality and code generation, rather than to proposing a statechart dialect with a clean semantics.

- *Chapter 3* discusses some "academic" languages for specifying reactive systems. The focus is on temporal logics and their abilities to express real-time constraints in general, and on Linear-time Temporal Logic (LTL) and Metric Temporal Logic (MTL) in particular. The chapter also

discusses run-time monitoring and glances at the TemporalRover™ and DBRover™ sister tools of StateRover.

- *Chapter 4* presents Statechart Assertions and illustrates how they can be used to specify a variety of temporal properties related to the traffic-light-controller example that is used throughout the book. Drusinsky also shows how Statechart Assertions can be simulated and tested using the JUnit testing framework.

- *Chapter 5* justifies Drusinsky's adoption of Statechart Assertions and run-time model checking for verification. It also gives advice on the process of devising, writing and applying assertions and, equally important, on how to reuse assertions via assertion libraries.

- *Chapter 6* describes a case study of applying StateRover in the context of the U.S. Ballistic Missile Defense Project. This chapter is contributed by Nick Sklavounos who is a key developer within this project. No details of the case study are presented, although there is a useful discussion on the wider project management issues when employing modern software tools such as StateRover.

Overall, this book is a brave attempt at addressing a difficult topic, adopting a fresh, pragmatic and illustrative approach. Nevertheless, I am disappointed. Firstly, the book's presentation could easily be much improved by eliminating the many forward and backward references. Secondly, a short introduction to the JUnit testing framework and a good discussion of the literature and related languages and tools would have rounded this book off. Thirdly, spelling out who the intended audience is, might have helped the author to write a more focused book.

On the one hand, this book is not what I am looking for in an academic textbook. Its technical content is too shallow, often imprecise and there are too many loose ends. This is because most aspects of reactive controller programming and verification are explained from a tool implementor's point of view. There is no formal treatment of StateRover's semantics, its code-generation facility and its run-time verification engine. All these aspects are introduced in a rather ad-hoc manner and refer to "*implementation semantics*." Without basic knowledge in statecharts and program verification, I feel that parts of the book are difficult to digest.

On the other hand, the book does also not give sufficient details for engineers who intend to deploy StateRover in their future projects. They would likely have wished for step-to-step guides on the use of StateRover's features and an in-depth case study. I also had expected a trial version of StateRover on the accompanying CD.

Despite its shortcomings, I can recommend this book to students as a *supplementary* textbook, so as to help them relate the theory taught in computing degree courses to software-engineering practise. Some engineers may also find this book a good source for gaining an understanding of run-time verification and of the philosophy behind the StateRover tool.

GERALD LÜTTGEN
Department of Computer Science
University of York
Heslington
York YO10 5DD, U.K.
E-mail: gerald.luettgen@cs.york.ac.uk