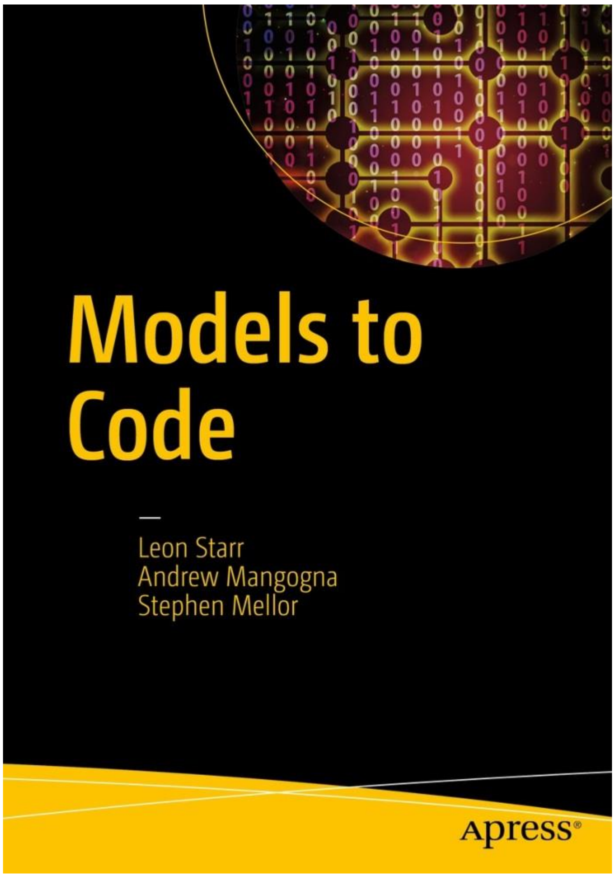
Models to Code



Authors:

Starr, Leon, **Mangogna**, Andrew, **Mellor**, Stephen

Highly pragmatic approach leaving models intact throughout the design process rather than destructing the model Techniques can be applied to any language or platform and adapted for your own implementation Written by three leading experts in the field

Teaches you how to translate an executable model of your application to efficient, running code on an embedded microcomputer platform without any mysterious gaps or hidden proprietary tools. There are many benefits to a model-oriented approach to software engineering but the path from models to code is not always clear.

Using a pragmatic approach, *Models to Code* uses annotated model and code examples to illustrate the key principles. You will start off with a brief overview of model based engineering concepts, then quickly dive into two case study Executable UML models that expose the key model elements. You will also understand the future of code translation and approaches you can take with other platforms and languages, as well as open source enhancements and alternative strategies for developers to try.

Although the techniques are shown using C and a microcomputer, they are not specific to that language or platform. The code generation strategy you learn can easily be adapted to your own implementation technology. Written by three industry experts, *Models to Code* is your number one resource for software modelling – add it to your library today.

What You Will Learn

- The purpose and benefits of model driven code generation
- The specific differences between application and implementation details
- What details are required in an Executable UML model prior to implementation
- How to specify an implementation without modifying the application models
- How to specify an Executable UML model in the **pycca** scripting language
- How to specify implementation choices in **pycca**
- See how a model can be repackaged as an efficient implementation automatically
- Learn diverse strategies for converting various model elements into code elements

Who This Book Is For

This book is for modelers and systems engineers on active MBSE projects (using Executable UML or not), projects using Simulink, Matlab, Dymola, MatrixX and other math modelling tools. Any developers with current or past model experience, professors and students, systems engineers, embedded systems developers, or anyone interested in learning more about software modelling.

Buy this book

- The eBook version of this title will be available soon
- ISBN 978-1-4842-2217-1
- Digitally watermarked, DRM-free
- Included format:
- ebooks can be used on all reading devices

Softcover ca. 39,21 €

Bibliographic Information

Book Title: Models to Code
Authors:

- Leon Starr
- Andrew Mangogna
- Stephen Mellor

Copyright: 2017

Publisher: Apress

Copyright Holder: Leon Starr, Andrew Mangogna, Stephen Mellor

eBook ISBN: 978-1-4842-2217-1

Softcover ISBN: 978-1-4842-2216-4

Edition Number: 1

About the authors

Leon Starr is a co-founder of Model Integration, LLC in San Francisco, California where he helps large projects develop model driven software successfully. Since 1984 he has been a prolific modeler of real-time, distributed and embedded systems. His models have been used in fighter jets, factory material transport control systems, ultrasound diagnostic and cardiac pacing systems, gas chromatography and semiconductor wafer inspection systems, video post-production systems and networked military battle simulators.

He has authored training and taught numerous modeling courses to systems engineers and software developers worldwide. Leon is the author of the books How to Build Shlaer-Mellor Object Models, How to Build Class Models, Executable UML: A Case Study, How to Build Executable UML Models and assorted papers at uml.org and modeling-languages.com. Leon speaks barely passable French and equally poor Swedish.

Stephen Mellor is a well-known technology consultant on methods for the construction of real-time and embedded systems, a signatory to the Agile Manifesto, and adjunct professor at the Australian National University in Canberra, ACT, Australia. Stephen is the author of *Structured Development for Real-Time Systems,Object Lifecycles, Executable UML*, and *MDA Distilled*.

He is presently the Chief Technical Officer for the Industrial Internet Consortium, where he directs the standards requirements and technology & security priorities for the industrial internet. In that role, he coordinates the activities of the several engineering, architecture, security and testbed working groups and teams. Before that, he was Chief Scientist of the Embedded Software Division at Mentor Graphics, and founder and past president of Project Technology, Inc., before its acquisition.

He participated in multiple UML/modeling-related activities at the Object Management Group (OMG), and was a member of the OMG Architecture Board, which is the final technical gateway for all OMG standards. Stephen was the Chairman of the Advisory Board to *IEEE Software* for ten years and a two-time Guest Editor of the magazine, most recently for an issue on Model-Driven Development.

For more than 30 years, **Andrew Mangogna** has been a hands-on builder of embedded software systems. He has worked in application areas ranging from laboratory instrumentation, remote data collection, and video special effects to implantable medical devices. Andrew has always had a special interest in applying more formal techniques to the challenge of engineering software to create systems in a cost effective manner with demonstrable quality.

Trained in the basics of object oriented analysis by Stephen Mellor himself, he has successfully applied executable modeling techniques and model translation to many projects and has written several tools to help automate the translation process. With a keen interest in technology and a practical realization of the benefits of modeling, he has a mastery of mapping models to appropriate implementation technology to obtain high quality software systems.