



Formal Methods

*Industrial Use
from Model to the Code*

**Edited by
Jean-Louis Boulanger**

ISTE

 WILEY

Formal Methods

Formal Methods

Industrial Use from Model to the Code

Edited by
Jean-Louis Boulanger

ISTE

 WILEY

First published 2012 in Great Britain and the United States by ISTE Ltd and John Wiley & Sons, Inc.

Apart from any fair dealing for the purposes of research or private study, or criticism or review, as permitted under the Copyright, Designs and Patents Act 1988, this publication may only be reproduced, stored or transmitted, in any form or by any means, with the prior permission in writing of the publishers, or in the case of reprographic reproduction in accordance with the terms and licenses issued by the CLA. Enquiries concerning reproduction outside these terms should be sent to the publishers at the undermentioned address:

ISTE Ltd
27-37 St George's Road
London SW19 4EU
UK

www.iste.co.uk

John Wiley & Sons, Inc.
111 River Street
Hoboken, NJ 07030
USA

www.wiley.com

© ISTE Ltd 2012

The rights of Jean-Louis Boulanger to be identified as the author of this work have been asserted by him in accordance with the Copyright, Designs and Patents Act 1988.

Library of Congress Cataloging-in-Publication Data

Formal methods : industrial use from model to the code / edited by Jean-Louis Boulanger.

p. cm. -- (Industrial implementation of formal methods series)

Includes bibliographical references and index.

ISBN 978-1-84821-362-3

1. Railroads--Management--Data processing. 2. Formal methods (Computer science) 3. Application software--Development. I. Boulanger, Jean-Louis.

TF507.F66 2012

385.0285'53--dc23

2012011496

British Library Cataloguing-in-Publication Data

A CIP record for this book is available from the British Library

ISBN: 978-1-84821-362-3

Printed and bound in Great Britain by CPI Group (UK) Ltd., Croydon, Surrey CR0 4YY



Table of Contents

| | |
|--|----|
| Introduction | xi |
| Jean-Louis BOULANGER | |
| Chapter 1. From Classic Languages to Formal Methods | 1 |
| Jean-Louis BOULANGER | |
| 1.1. Introduction | 1 |
| 1.2. Classic development | 2 |
| 1.2.1. Development process | 2 |
| 1.2.2. Coding | 6 |
| 1.2.3. Specification and architecture | 18 |
| 1.2.4. Verification and validation (V&V) | 27 |
| 1.2.5. Summary | 33 |
| 1.3. Structured, semi-formal and/or formal methods | 33 |
| 1.3.1. E/E/PE system | 33 |
| 1.3.2. Rail sector | 35 |
| 1.3.3. Taking into account techniques and formal methods | 36 |
| 1.4. Formal methods | 39 |
| 1.4.1. Principles | 39 |
| 1.4.2. Examples of formal methods | 39 |
| 1.5. Conclusion | 45 |
| 1.6. Bibliography | 49 |
| Chapter 2. Formal Method in the Railway Sector the First Complex Application: SAET-METEOR | 55 |
| Jean-Louis BOULANGER | |
| 2.1. Introduction | 55 |
| 2.2. About SAET-METEOR | 56 |
| 2.2.1. Decomposition of the SAET-METEOR | 57 |

| | |
|--|------------|
| 2.2.2. Anticollision functions | 61 |
| 2.2.3. Restrictions | 62 |
| 2.3. The supplier realization process | 62 |
| 2.3.1. Historical context | 62 |
| 2.3.2. The hardware aspect | 64 |
| 2.3.3. The software aspect | 67 |
| 2.3.4. Assessment of the processes | 78 |
| 2.4. Process of verification and validation set up by RATP | 78 |
| 2.4.1. Context | 78 |
| 2.4.2. RATP methodology | 79 |
| 2.4.3. Verification carried out by RATP | 79 |
| 2.4.4. Validation | 103 |
| 2.4.5. Assessment of RATP activity | 112 |
| 2.5. Assessment of the global approach | 114 |
| 2.6. Conclusion | 115 |
| 2.7. Appendix | 116 |
| 2.7.1. Object of the track | 116 |
| 2.7.2. Block logic | 120 |
| 2.8. Bibliography | 122 |
| Chapter 3. The B Method and B Tools | 127 |
| Jean-Louis BOULANGER | |
| 3.1. Introduction | 127 |
| 3.2. The B method | 128 |
| 3.2.1. The concept of abstract machines | 128 |
| 3.2.2. Machines with implementations | 133 |
| 3.3. Verification and validation (V&V) | 137 |
| 3.3.1. Internal verification | 137 |
| 3.4. B tools | 141 |
| 3.4.1. General principles | 141 |
| 3.4.2. Code generation | 142 |
| 3.4.3. Prover | 142 |
| 3.4.4. Atelier B | 144 |
| 3.5. Methodology | 146 |
| 3.5.1. Layered development | 146 |
| 3.5.2. Link between the project structure and the POs | 148 |
| 3.5.3. Cycle of development of a B project | 148 |
| 3.6. Feedback | 150 |
| 3.6.1. Some figures | 150 |
| 3.6.2. Some users | 151 |
| 3.7. Conclusion | 155 |
| 3.8. Bibliography | 155 |

| | |
|---|-----|
| Chapter 4. Model-Based Design Using Simulink – Modeling, Code Generation, Verification, and Validation | 159 |
| Mirko CONRAD and Pieter J. MOSTERMAN | |
| 4.1. Introduction | 159 |
| 4.2. Embedded software development using Model-Based Design | 162 |
| 4.3. Case study – an electronic throttle control system | 164 |
| 4.3.1. System overview | 164 |
| 4.3.2. Simulink [®] model | 164 |
| 4.3.3. Automatic code generation | 169 |
| 4.3.4. Code optimization | 170 |
| 4.3.5. Fixed-point code | 170 |
| 4.3.6. Including legacy code | 172 |
| 4.3.7. Importing interface definitions | 172 |
| 4.3.8. Importing algorithms | 173 |
| 4.4. Verification and validation of models and generated code | 173 |
| 4.4.1. Integrating verification and validation with Model-Based Design | 173 |
| 4.4.2. Design verification | 175 |
| 4.4.3. Reviews and static analyses at the model level | 175 |
| 4.4.4. Module and integration testing at the model level | 175 |
| 4.4.5. Code verification | 176 |
| 4.4.6. Back-to-back comparison testing between model and code | 176 |
| 4.4.7. Measures to prevent unintended functionality | 176 |
| 4.5. Compliance with safety standards | 177 |
| 4.6. Conclusion | 178 |
| 4.7. Bibliography | 178 |
| Chapter 5. Proving Global Properties with the Aid of the SIMULINK DESIGN VERIFIER Proof Tool | 183 |
| Véronique DELEBARRE and Jean-Frédéric ETIENNE | |
| 5.1. Introduction | 183 |
| 5.2. Formal proof or verification method | 184 |
| 5.2.1. Model verification | 186 |
| 5.2.2. Formal methods and proof of correction | 189 |
| 5.2.3. Combining models and proof tools | 192 |
| 5.3. Implementation of the SIMULINK DESIGN VERIFIER tool. | 193 |
| 5.3.1. Reminders of the MATLAB modeling and verification environment | 194 |
| 5.3.2. Case study | 201 |
| 5.3.3. Modeling | 204 |
| 5.3.4. Modeling | 211 |
| 5.4. Experience feedback and methodological aspects | 211 |

| | |
|--|------------|
| 5.4.1. Modeling rules and convergence control | 211 |
| 5.4.2. Modular proof phase | 213 |
| 5.4.3. Proof of global properties | 214 |
| 5.4.4. Detection of counterexamples | 217 |
| 5.5. Study case feedback and conclusions | 218 |
| 5.5.1. SIMULINK model | 218 |
| 5.5.2. Proofs achieved | 218 |
| 5.5.3. Incremental verification approach | 220 |
| 5.6. Contributions of the methodology compared with the EN50128 normative referential | 220 |
| 5.7. Bibliography | 222 |
| Chapter 6. SCADE: Implementation and Applications | 225 |
| Jean-Louis CAMUS | |
| 6.1. Introduction | 225 |
| 6.2. Issues of embedded safety-critical software | 225 |
| 6.2.1. Characteristics of embedded safety-critical software | 225 |
| 6.2.2. Architecture of an embedded safety-critical application | 226 |
| 6.2.3. Criticality and normative requirements for embedded safety-critical applications | 226 |
| 6.2.4. Complexity, cost and delays | 227 |
| 6.3. Origins of SCADE | 228 |
| 6.3.1. Introduction | 228 |
| 6.3.2. Initial industrial approaches | 228 |
| 6.3.3. “Real-time” extensions of current languages | 230 |
| 6.3.4. Synchronous formal languages dedicated to “real-time” created in laboratories | 230 |
| 6.3.5. Birth of SCADE | 231 |
| 6.4. The SCADE data-flow language | 231 |
| 6.4.1. Introduction | 231 |
| 6.4.2. Synchronous language | 232 |
| 6.4.3. “Data-flow” functional language | 233 |
| 6.4.4. Scalar data types | 234 |
| 6.4.5. Structured data types | 235 |
| 6.4.6. Clocks, temporal operators, and causality | 235 |
| 6.4.7. Selectors | 237 |
| 6.4.8. Imperative structures | 238 |
| 6.4.9. Rigor and functional safety | 239 |
| 6.5. Conclusion: extensions of languages for controllers and iterative processing | 240 |
| 6.5.1. Objectives | 240 |
| 6.5.2. Control flow | 241 |
| 6.5.3. Iterative processing | 243 |

| | |
|---|------------|
| 6.6. The SCADE system | 246 |
| 6.6.1. Outline of the SCADE workbench | 246 |
| 6.6.2. Model verification | 247 |
| 6.6.3. Performance prediction | 252 |
| 6.6.4. The qualified code generator | 253 |
| 6.7. Application of SCADE in the aeronautical industry | 256 |
| 6.7.1. History: Aérospatiale and Thales Avionique | 256 |
| 6.7.2. Control/command type applications | 257 |
| 6.7.3. Monitoring/alarm type applications | 260 |
| 6.7.4. Navigation systems | 261 |
| 6.8 Application of SCADE in the rail industry | 261 |
| 6.8.1. First applications | 262 |
| 6.8.2. Applications developed for the RATP and other French metros | 262 |
| 6.8.3. Generic PAI-NG applications | 263 |
| 6.8.4. Example of automated door control | 264 |
| 6.9. Application of SCADE in the nuclear and other industries | 265 |
| 6.9.1. Applications in the nuclear industry | 265 |
| 6.9.2. Deployment of SCADE in the civil nuclear industry | 268 |
| 6.10. Conclusion | 269 |
| 6.11. Bibliography | 270 |
| Chapter 7. GATeL: A V&V Platform for SCADE Models | 273 |
| Bruno MARRE, Benjamin BIANC, Patricia MOUY and Christophe JUNKE | |
| 7.1. Introduction | 273 |
| 7.2. SCADE language | 275 |
| 7.3. GATeL prerequisites | 276 |
| 7.3.1. GATeL kernel | 277 |
| 7.3.2. Example | 278 |
| 7.4. Assistance in the design of test selection strategies | 279 |
| 7.4.1. Unfolding of SCADE operators | 279 |
| 7.4.2. Functional scenarios | 281 |
| 7.5. Performances | 283 |
| 7.6. Conclusion | 284 |
| 7.7. Bibliography | 285 |
| Chapter 8. ControlBuild, a Development Framework for Control Engineering | 287 |
| Franck CORBIER | |
| 8.1. Introduction | 287 |
| 8.2. Development of the control system | 289 |
| 8.2.1. ERTMS | 290 |
| 8.2.2. Development process equipment | 291 |

| | |
|---|------------|
| 8.2.3. A component-based approach | 293 |
| 8.2.4. Development methodology | 294 |
| 8.3. Formalisms used | 300 |
| 8.3.1. Assembly editor | 301 |
| 8.3.2. IEC1131-3 languages for embedded control | 302 |
| 8.3.3. Electrical schematics for conventional control | 309 |
| 8.3.4. Electromechanical and physical environment | 311 |
| 8.4. Safety arrangements | 311 |
| 8.4.1. Metrics | 313 |
| 8.4.2. Assertions | 314 |
| 8.4.3. Automatic test procedure execution | 314 |
| 8.4.4. Functional tests | 315 |
| 8.4.5. Code coverage | 315 |
| 8.4.6. SSIL2 code generation | 315 |
| 8.4.7. Management of the project documentation | 316 |
| 8.4.8. Traceability of requirements | 317 |
| 8.5. Examples of railway use cases | 318 |
| 8.5.1. Specification validation | 318 |
| 8.5.2. TCMS development | 319 |
| 8.5.3. Progressive integration bench – HiL | 320 |
| 8.6. Conclusion | 323 |
| 8.7. Bibliography | 323 |
| Chapter 9. Conclusion | 325 |
| Jean-Louis BOULANGER | |
| 9.1. Introduction | 325 |
| 9.2. Problems | 326 |
| 9.3. Summary | 327 |
| 9.3.1. Model verification | 327 |
| 9.3.2. Properties and requirements | 328 |
| 9.3.3. Implementation of formal methods | 330 |
| 9.4. Implementing formal methods | 332 |
| 9.4.1. Conventional process | 332 |
| 9.4.2. Process accounting for formal methods | 333 |
| 9.4.3. Problems | 335 |
| 9.5. Realization of a software application | 337 |
| 9.6. Conclusion | 339 |
| 9.7. Bibliography | 340 |
| Glossary | 345 |
| List of Authors | 351 |
| Index | 353 |

