Collaborative Design for Embedded Systems

John Fitzgerald • Peter Gorm Larsen •
Marcel Verhoef
Editors

# Collaborative Design for Embedded Systems

## Co-modelling and Co-simulation

Springer

*Editors*
John Fitzgerald
Newcastle University
Newcastle upon Tyne
United Kingdom

Peter Gorm Larsen
Aarhus University
Aarhus
Denmark

Marcel Verhoef
Chess WISE B.V.
Haarlem
The Netherlands

ISBN 978-3-642-54117-9 ISBN 978-3-642-54118-6 (eBook)
DOI 10.1007/978-3-642-54118-6
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014936992

© Springer-Verlag Berlin Heidelberg 2014
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

# Foreword

Embedded systems permeate the world today, and they increase not only in numbers but also in complexity. One by now archetypical example, also quoted in this book, is the modern automobile, with its dozens of electronic control units (ECUs). The behaviour and functionality of the embedded control of a car is not the result of a single piece of software or silicon, but rather *emerges* from a very complex interplay of ECUs. These components communicate with each other in complex, hard-to-predict ways. Protocol mismatches are a common source of error, and agreeing on common interfaces among components is one of the most difficult and time-consuming tasks in managing large designs.

This very problem, of finding a common ground across collaboration interfaces, also applies to the designers themselves who create these complex embedded systems. Embedded, or *cyber-physical*, systems bring a multitude of disciplines together, including, but not limited to, computer science, electrical engineering, physics and psychology. Within each discipline, there is again a range of different fields, with experts that are mostly used to work among peers of their own domain. Vocabularies are different or, even worse, are the same but have different semantics.

In the realm of research projects, few attempt to bring together partners from very diverse fields, or if they do, they limit interaction to high-level, narrow interfaces. The European "Design Support and Tooling for Embedded Control Software" (DESTECS) project, which we had the pleasure to accompany as reviewers in 2010–2012, did have the ambitious goal to work closely together across domains. DESTECS combined discrete-event and continuous-time modelling, and this cross-cutting aspect would be a key part of the success or failure of the project. This did pose a non-trivial project risk, despite the fact that the participants were all already highly acknowledged experts in their domains and brought together much professional experience from earlier projects. Quoting from one of the project reports: "The project is quite multidisciplinary and as a consequence it turned out to be beneficial to produce exact definitions of terminology, as in the first half year of the project, understanding of relevant terms was quite different." This extra effort, not accounted for in the original resource planning, did pay off very well, as we could witness in subsequent project periods.

One outcome of DESTECS that lasts beyond the project is the *Crescendo approach*, which demonstrated its viability in several full-scale demonstrators developed jointly by academic and industrial partners. Another outcome that we expect to have lasting impact is this book, which makes Crescendo and its underlying concepts accessible in a way that transgresses any collection of separate papers and which has added significant value to the project even after it has formally ended. This book not only straddles the aforementioned boundaries between (mostly) separate fields, but also finds a perfect balance between theoretical underpinnings and practical advice. It should be equally valuable for researchers and practitioners in the field of embedded systems design.

Elancourt, France — Bernard Dion
Kiel, Germany — Reinhard von Hanxleden

# Preface

The embedded systems market is a lively—some would say volatile—place. There is a growing demand for products that make the best use of rapidly improving computer hardware to create everything from game consoles to flight controllers. In this setting, the developers of embedded systems have to form creative teams out of disparate engineering disciplines. For example, a product design team might encompass software, mechanical, electrical and control engineers. However, effective collaborative design is not simply a matter of sharing a whiteboard with each other—the bases of engineering disciplines are different, and perhaps the biggest gulf is between software and control engineering. Control engineers describe how phenomena evolve and flow over continuous time, but software is described using logic to relate discrete events.

The semantic gaps between engineering disciplines cost time and money because the results of misunderstandings are often only detected when the physical product is built and software fails to control it properly. Traditional product development involves specialist engineering groups working independently on aspects of the design which is passed between them, sometimes being misinterpreted and distorted as it goes. This is a particularly pressing problem when we attempt to design for resilience: dependability cannot be sprinkled over a completed design, but needs to be integrated from the outset. Indeed, our experience suggests that around 80 % of embedded software is related to complex supervisory control which includes switching between modes or dealing with error detection and recovery. It is essential, then, to have methods and tools to help manage the risk of early-stage design flaws.

This book is a response to the challenge of delivering effective multi-disciplinary design. It builds on the premise that early analysis of design models could lead to early detection of errors and performance bottlenecks, and the models themselves can serve as common bases for communication and the exploration of design alternatives. But how can model-based methods work if the engineering disciplines describe aspects of the product and its environment in such different ways? We focus on the creation and analysis of collaborative models ("co-models"), composed of discrete-event and continuous-time models. Typically, these contain discrete-event

models of control elements to be realised on computers, coupled with continuous-time models of controlled plants and the physical environment. However, rather than forcing diverse disciplines into a single common modelling framework, we show how it is possible to link otherwise separate tools for discrete-event and continuous-time modelling through a harness that allows them to share data during simulations to create a unified "co-simulation". Co-simulation between tools supports collaboration between designers, allowing engineers from each discipline to continue work within familiar formalisms while being able readily to judge the effects of a design decision in one domain on the behaviour of the other. For example, we might choose to resolve a known hardware fault either by using different sensors or by more complex control software. With co-simulation, we can trade off these alternatives on such factors as performance, energy consumption and cost before a commitment is made.

The methods and tools for co-modelling described in this book were developed in a collaborative research project, "Design Support and Tooling for Embedded Control Software" (DESTECS).[1] We were fortunate in DESTECS to have the cooperation of several pioneering companies, who endured the instabilities of our nascent methods and tools, applying them in several domains. Their experience demonstrated the value of co-modelling in reducing design iterations, easing the development of sophisticated software control and supporting dependability. We hope that this book gives the reader a sense of the potential for innovation enabled by methods and tools that support technically well-founded collaboration across discipline boundaries.

## Structure of the Text

Our goal is to present methods and tools for co-modelling, co-simulation and design space exploration in a thoroughly practical way, with running examples. The book is structured in three parts:

1. Part I introduces the technical basis of co-modelling and co-simulation using one Continuous-Time (CT) and one Discrete-Event (DE) formalism. Chapter 1 describes the need for collaborative design and the challenges in providing methods and tools to support it. We then introduce core concepts that underpin the rest of the book (Chap. 2). In Chaps. 3 and 4, we present the specific CT and DE technologies that we propose to link through co-modelling: respectively, bond graphs, supported by 20-sim, and the Vienna Development Method (VDM). Both are comprehensive formalisms, so in these chapters, we aim to give the reader a sense of the main features of each. We then introduce the Crescendo technology for co-simulation (Chap. 5). Finally, Chap. 6 discusses the use of structuring mechanisms to promote the reuse of controller models.

[1]European Union Framework 7 project CNECT-ICT-248134, January 2010–December 2012 (see http://www.destecs.org/).

2. In Part II, we move from foundations to the application of co-modelling. Chapter 7 introduces two case studies (a line-following robot and a ChessWay personal transporter), which are used in this part of the book. The process by which a co-model is developed is discussed in Chap. 8. Chapter 9 introduces techniques for co-modelling faults and fault tolerance mechanisms, while Chap. 10 examines the support for exploring large design spaces in the search for optimal solutions. Chapter 11 brings these strands together, describing how the technology has been applied on other industrial applications.

3. Part III considers more advanced topics. Chapter 12 reports the experiences of three industry users following their experimental deployment of the Crescendo technology. Chapter 13 gives a technical discussion of the semantics of the co-simulation framework underlying the Crescendo technology and explains how it can be re-used with other CT and DE technologies. Finally, Chap. 14 positions our work in the broader setting of model-based collaborative design and sets out the challenges posed by the development of cyber-physical systems.

4. The appendices include summaries of VDM and 20-sim, a catalogue of design patterns for co-models, and an abstract VDM model of the ChessWay with a focus on its safety aspects. A list of acronyms and a glossary of the main terms used in this book are included.

## Using the Book

The book is aimed at both researchers and practitioners in embedded systems development. Among researchers, the book should be of interest to those working in cyber-physical systems, embedded systems design and formal methods; in control engineering, the material should be of interest to those working on advanced control and modelling technology, especially for fault-tolerant and resilient systems. Among practitioners, we target the book at those in research and product development with an interest in improved design processes and tools. Among academics, we expect the text to be of value for those teaching embedded software development at all levels. We recommend that all readers make use of the Crescendo tool for hands-on experience and access the additional content, including tutorials and training material, on the accompanying web site (see below).

In keeping with the spirit of co-modelling, we assume that the reader has some experience in either software development or control, but we do not assume both. In Part I of the book, the introductions to CT modelling in Chap. 3 and DE (computing) modelling in Chap. 4 are written with readers from both backgrounds in mind.

Practitioners with an interest in the techniques of co-modelling and co-simulation are invited to approach the contents of Parts I and II in the order presented. Most of the chapters in the technical flow of the book assume that preceding chapters will have been covered. The advanced topics in Part III are relatively independent of one another. Among the advanced topics, Chap. 13 will be of technical interest mainly

to those working on the formal semantics of modelling languages and so may be omitted by others on a first reading.

Readers with a primary interest in engineering management may wish to cover the motivation and technical foundations in Chaps. 1 and 2, example case studies in Chap. 7 and elements of co-model creation methodology in Chap. 8, followed by industry applications and deployment experience in Chaps. 11 and 12 and future directions for the technology in Chap. 14.

## Accompanying Web Site

The accompanying web site, www.crescendotool.org, provides additional material, including tool support for co-simulation, as described in the book, additional example co-models that can be used with the tool and course material. We invite readers wishing to use the material for teaching or research to take the distribution only from this web site and contact the editors for further support.

## Acknowledgments

DESTECS was supported by the European Commission under the Seventh Framework programme. We are grateful to the expert reviewers, Mr. Bernard Dion (Esterel Technologies) and Prof. Reinhard von Hanxleden (Kiel University), for their constructive suggestions and recommendations throughout the project. It is a pleasure to thank the many contributors to the book, particularly Kenneth Pierce, Carl Gamble, Jan Broenink, Job van Amerongen, Christian Kleijn, Augusto Ribeiro, Kenneth Lausdahl, Bert Bos, Sune Wolff and Joey Coleman. We are very grateful to Koenraad Rombaut and Peter van Eijk, who kindly agreed to be interviewed about their industrial application of the technology that we developed, allowing us to recount their experiences in Chap. 12. We gladly acknowledge our other colleagues from DESTECS who contributed to the technology: Claire Ingram, Kim Bjerge, José Antonio Esparza Isasa, Claus Ballegard Nielsen, Xiaochen Zhang, Yunyun Ni, Angelika Mader, Jelena Marinčić, Stefan Groothuis, Peter Visser, Frank Groen, Marcel Groothuis, Dusko Jovanovic, Jan Remijnse, Eelke Visser, Michiel De Paepe, Yoni De Witte, Roeland Van Lembergen, Wouter Vleugels, Kim Visser and Jeffrey Simons. We are grateful to Nick Battle, Stefan Hallerstede, Hiroshi Sako and all those who provided feedback to us on draft material. We would also like to thank Martin Peter Christiansen for providing the T1X tractor example and simulation results. At a personal level, we are deeply grateful to our families and friends for their patience and support since the genesis of this book, especially John Hudson, Margit Sandvang Larsen and Natalie Ree.

Newcastle upon Tyne, UK — John Fitzgerald
Aarhus, Denmark — Peter Gorm Larsen
Haarlem, The Netherlands — Marcel Verhoef

# List of Acronyms

| | |
|---|---|
| ACA | Automated co-model analysis |
| AD | Analog to digital |
| BDD | Block definition diagram |
| BLDC | Brushless direct current |
| CoDES | Collaborative design for embedded systems |
| CPS | Cyber-physical system |
| CPU | Central processing unit |
| CoDES | Collaborative design for embedded systems |
| CRC | Cyclic-redundancy check |
| CSL | CoDES scripting language |
| CT | Continuous-time |
| DA | Digital to analog |
| DAL | Development assurance level |
| DATE | Design automation and test in Europe |
| DE | Discrete event |
| DESTECS | Design support and tooling for embedded control software |
| DSE | Design space exploration |
| DT | Discrete time |
| EDA | Electronic design automation |
| EMF | Electro motive force |
| FCFS | First come first served |
| FDIV | Floating-point DIVision |
| FMEA | Failure mode and effects analysis |
| FMI | Functional mockup interface |
| FMU | Functional mockup unit |
| ForTIA | Formal Techniques Industry Association |
| FP | Fixed priority |

| | |
|---|---|
| FPGA | Field-programmable gate array |
| FSM | Finite state machine |
| GUI | Graphical user interface |
| HAZOP | Hazard and operability |
| HiL | Hardware-in-the-loop |
| IBD | Internal block diagram |
| IDE | Integrated design environment |
| IEEE | Institute of electrical and electronics engineers |
| IFG | Industry follow group |
| INCOSE | International Council on Systems Engineering |
| iOS | Internet operating system |
| IoT | Internet of things |
| IPM | Ideal physical model |
| ISO | International Standards Organisation |
| MIL | Model in the loop |
| NFC | Near field communication |
| NTP | Network time protocol |
| OMG | Object Management Group |
| PC | Personal computer |
| PID | Proportional integral derivative |
| PWM | Pulse width modulated |
| RC | Resistor-capacitor |
| RPC | Remote procedure call |
| RUP | Rational unified process |
| SDP | Shared design parameter |
| SHARD | Software hazard analysis and resolution in design |
| SI | Système Internationale d‘unité |
| SiL | Software-in-the loop |
| SIL | Safety integrity level |
| SoS | System of systems |
| SOS | Structural operational semantics |
| SRI | French abbreviation for Inertial Reference System |
| SSS | Single-state simulation semantics |
| SUT | System under test |
| SysML | Systems Modelling Language |
| TSS | Transactional simulation semantics |
| TTM | Time to market |
| UML | Unified modeling language |
| VDM | Vienna development method |
| VDM-SL | Vienna Development Method Specification Language |
| VDM-RT | Vienna development method real time |

| | |
|---|---|
| WAM | Weighted additive method |
| XBMC | Xbox Media Center |
| XML | Extensible Markup Language |
| XTE | Cross track error |

# Contents

**Part I Co-modelling and Co-simulation: The Technical Basis**

**1 Collaborative Development of Embedded Systems** ..... 3
Marcel Verhoef, Kenneth Pierce, Carl Gamble, and Jan Broenink
1.1 Introduction ..... 3
1.2 Setting the Scene ..... 4
1.3 The Embedded Systems Design Challenge ..... 8
1.4 Embedded Systems Design: An Illustrative Story ..... 10
1.5 A Solution: The Crescendo Approach ..... 13
1.6 Conclusion ..... 14

**2 Co-modelling and Co-simulation in Embedded Systems Design** ..... 15
John Fitzgerald and Kenneth Pierce
2.1 Introduction ..... 15
2.2 Systems and System Boundaries ..... 15
2.3 Models ..... 16
2.4 Co-models ..... 17
2.5 Co-simulation ..... 19
2.6 DSE and Automated Co-model Analysis ..... 21
2.7 Co-simulation in Practice ..... 22
2.8 Conclusion ..... 25

**3 Continuous-Time Modelling in 20-sim** ..... 27
Job van Amerongen, Christian Kleijn, and Carl Gamble
3.1 Introduction ..... 27
3.2 Physical Systems ..... 29
3.3 Icons and Iconic Diagrams ..... 34
3.4 A Domain-Independent Description: Bond Graphs ..... 36
3.5 Simulating Physical Systems with 20-sim ..... 46
3.6 Control Systems ..... 49

3.7 A Small Note on Notation ..... 58
3.8 Conclusion ..... 59

**4 Discrete-Event Modelling in VDM** ..... 61
Peter Gorm Larsen, John Fitzgerald, Marcel Verhoef, and Kenneth Pierce
4.1 Introduction ..... 61
4.2 Basic Elements: Data and Functionality ..... 63
4.3 Example: A Basic Controller Model ..... 72
4.4 Modelling with Structured Data ..... 73
4.5 Example: Supervisory Control ..... 81
4.6 Example: Controlling for Safety ..... 84
4.7 Object-Oriented Structuring ..... 87
4.8 Concurrency ..... 91
4.9 Modelling Systems ..... 93
4.10 Conclusion ..... 95

**5 Support for Co-modelling and Co-simulation: The Crescendo Tool** ..... 97
Peter Gorm Larsen, Carl Gamble, Kenneth Pierce, Augusto Ribeiro, and Kenneth Lausdahl
5.1 Introduction ..... 97
5.2 Importing the Torsion Bar Co-model ..... 98
5.3 Crescendo Contracts ..... 98
5.4 Starting a Co-simulation ..... 104
5.5 Using Scripts and SDPs ..... 108
5.6 Changing the Torsion Bar Model ..... 109
5.7 Conclusion ..... 114

**6 Co-model Structuring and Design Patterns** ..... 115
Kenneth Pierce, Peter Gorm Larsen, and John Fitzgerald
6.1 Introduction ..... 115
6.2 Object-Orientation and Inheritance ..... 117
6.3 Interfaces for Sensors and Actuators ..... 119
6.4 Design Patterns ..... 121
6.5 Using Inheritance for Threads ..... 126
6.6 Structuring Constituent Models for Flexible Simulation ..... 131
6.7 Conclusion ..... 137

**Part II Methods and Applications: The Pragmatics of Co-modelling and Co-simulation**

**7 Case Studies in Co-modelling and Co-simulation** ..... 141
Marcel Verhoef, Bert Bos, Kenneth Pierce, Carl Gamble, and Job van Amerongen
7.1 Introduction ..... 141
7.2 The R2-G2P Line-Following Robot ..... 142

7.3 The ChessWay Self-balancing Scooter ..... 145
7.4 Conclusion ..... 151

**8 Methods for Creating Co-models of Embedded Systems** ..... 153
Kenneth Pierce, Sune Wolff, and Marcel Verhoef
8.1 Introduction ..... 153
8.2 Paths to Co-models ..... 154
8.3 Using SysML Initially ..... 157
8.4 The CT-first Approach ..... 164
8.5 The DE-first Approach ..... 171
8.6 The Contract-first Approach ..... 182
8.7 Conclusion ..... 183

**9 Co-modelling of Faults and Fault Tolerance Mechanisms** ..... 185
Carl Gamble, Kenneth Pierce, John Fitzgerald, and Bert Bos
9.1 Introduction ..... 185
9.2 Fault Identification ..... 186
9.3 Fault Selection ..... 188
9.4 Fault Modelling ..... 188
9.5 Fault Tolerance Coverage ..... 190
9.6 Fault Tolerance Modelling ..... 190
9.7 An Example Using the Line-Following Robot ..... 190
9.8 An Example Using the ChessWay ..... 194
9.9 Conclusion ..... 197

**10 Design Space Exploration for Embedded Systems Using Co-simulation** ..... 199
Carl Gamble and Kenneth Pierce
10.1 Introduction ..... 199
10.2 Using ACA ..... 200
10.3 An Example Using the Line-Following Robot ..... 202
10.4 Candidate Parameters ..... 205
10.5 Experimental Design ..... 207
10.6 Using Folder Launch Configuration ..... 210
10.7 An Example Using the T1X Tractor ..... 211
10.8 Ranking of Results ..... 216
10.9 An Example Using the Line-Measuring Robot ..... 218
10.10 Conclusion ..... 222

**11 Industrial Application of Co-modelling and Co-simulation Technology** ..... 223
Marcel Verhoef and Peter Gorm Larsen
11.1 Introduction ..... 223
11.2 A Dredging Excavator ..... 225
11.3 A Document Handling System ..... 237
11.4 The ChessWay Self-balancing Scooter ..... 253
11.5 Conclusion ..... 258

## Part III Advanced Topics

**12 Deploying Co-modelling in Commercial Practice** ..... 263
Sune Wolff, Peter Gorm Larsen, and Marcel Verhoef
12.1 Introduction ..... 263
12.2 Company Introductions ..... 264
12.3 Traditional Development ..... 264
12.4 Integrating Co-modelling and Co-simulation with Existing Processes ..... 265
12.5 Resources ..... 266
12.6 Challenges Encountered ..... 266
12.7 Key Benefits ..... 267
12.8 The Future of Co-modelling ..... 269
12.9 Conclusion ..... 270

**13 Semantics of Co-simulation** ..... 273
Joey W. Coleman, Kenneth Lausdahl, and Peter Gorm Larsen
13.1 Introduction ..... 273
13.2 Structure of Co-simulation ..... 274
13.3 Co-simulation Semantics ..... 278
13.4 Adding Fault Injection Semantics to the Co-simulation ..... 283
13.5 Semantics of the CSL ..... 285
13.6 Conclusion ..... 291

**14 From Embedded to Cyber-Physical Systems: Challenges and Future Directions** ..... 293
John Fitzgerald, Peter Gorm Larsen, and Marcel Verhoef
14.1 Introduction ..... 293
14.2 The Co-modelling and Co-simulation Landscape ..... 295
14.3 Co-modelling in the CPS Design Flow ..... 297
14.4 Enabling Collections of DE and CT Models to Be Combined ... 298
14.5 Open Co-simulation ..... 298
14.6 Ubiquitous and Distributed Computing ..... 299
14.7 An Open and Lively Research Field ..... 301
14.8 Conclusion ..... 302

**A 20-sim Summary** ..... 305
Christian Kleijn
A.1 Introduction ..... 305
A.2 Overview ..... 306
A.3 Graphical Models ..... 307
A.4 Equation Models ..... 307
A.5 Modelling Tools ..... 307
A.6 Simulation ..... 308
A.7 Analysis ..... 309
A.8 Scripting ..... 310

A.9 Co-simulation ..... 310
A.10 Code Generation ..... 310

**B VDM-RT Language Summary** ..... 313
Peter Gorm Larsen
B.1 Operators for Basic Types ..... 313
B.2 Operators for Set Types ..... 313
B.3 Operators for Sequence Types ..... 313
B.4 Operators for Mapping Types ..... 315
B.5 Record Types and Values in VDM ..... 315
B.6 Small VDM-RT Examples ..... 316
B.7 Threads and Synchronisation in VDM ..... 319
B.8 The System Class Concept in VDM-RT ..... 319
B.9 Example of Classes ..... 320
B.10 UML Diagrams ..... 321

**C Design Patterns for Use in Co-modelling** ..... 323
Carl Gamble, Kenneth Pierce, John Fitzgerald, Bert Bos, and Marcel Verhoef
C.1 Introduction ..... 323
C.2 Controller Patterns ..... 324
C.3 Fault Patterns ..... 341
C.4 Fault Tolerance Patterns ..... 347

**D Abstract Modelling of ChessWay Safety** ..... 359
Marcel Verhoef and Bert Bos

**References** ..... 371

**Glossary** ..... 377

**Index** ..... 381

# List of Contributors

**Bert Bos,** Chess iX, Haarlem, The Netherlands

**Jan Broenink,** University of Twente, Enschede, The Netherlands

**Joey Coleman,** Aarhus University, Aarhus, Denmark

**John Fitzgerald,** Newcastle University, Newcastle upon Tyne, UK

**Carl Gamble,** Newcastle University, Newcastle upon Tyne, UK

**Christian Kleijn,** Controllab Products, Enschede, The Netherlands

**Peter Gorm Larsen,** Aarhus University, Aarhus, Denmark

**Kenneth Lausdahl,** Aarhus University, Aarhus, Denmark

**Kenneth Pierce,** Newcastle University, Newcastle upon Tyne, UK

**Augusto Ribeiro,** d60, Aabyhoej, Denmark

**Job van Amerongen,** University of Twente, Enschede, The Netherlands

**Marcel Verhoef,** Chess WISE, Haarlem, The Netherlands

**Sune Wolff,** Aarhus University, Aarhus, Denmark