Review on Formal Methods for Software Engineering: Languages, Methods, Application Domains

MAURICE H. TER BEEK, CNR-ISTI, Pisa, Italy

CCS Concepts: • **Software and its engineering** → **Formal methods**; *Operating systems*; Software testing and debugging; • **Social and professional topics** → **Computer science education**; • **Theory of computation** → *Logic*; • **Computing methodologies** → *Symbolic and algebraic manipulation*; • **Human-centered computing** → Human computer interaction (HCI); • **Security and privacy** → Cryptography; Formal methods and theory of security;

Additional Key Words and Phrases: Computer science education, software engineering, formal methods, model checking, theorem proving, CSP, PAT, CASL, contracts, cryptography

ACM Reference Format:
Maurice H. ter Beek. 2025. Review on Formal Methods for Software Engineering: Languages, Methods, Application Domains. *Form. Asp. Comput.* 37, 4, Article 35 (October 2025), 3 pages. https://doi.org/10.1145/3746236

This book, authored by Markus Roggenbach, Antonio Cerone, Bernd-Holger Schlingloff, Gerardo Schneider, and Siraj Ahmed Shaikh, provides material for graduate and undergraduate courses in software engineering for teachers that are willing to address the effective adoption of formal methods and tools for software design and formal verification, as well as for beginning PhD students. The only prerequisite is some basic knowledge of mathematical notation as is typically taught in the beginning of any bachelor curriculum in computer science.

The core of the book is structured in three parts, each containing a few independent chapters, surrounded by an opening and a closing chapter. The opening Chapter 1 provides a general introduction to what constitutes a formal method and it explores their role in software engineering and industrial practice. The first part on *languages* is foundational and serves as a reference for later chapters. Chapter 2 defines several logics that are commonly used in computer science for specifying and analyzing systems and Chapter 3 presents the process algebra CSP tailored for the modeling and analysis of concurrent systems. The second part on *methods* showcases the general-purpose algebraic specification language CASL in Chapter 4 and discusses a formal (black box) testing approach based on specifications in Chapter 5. The third part on *application domains* discusses three concrete solutions based on formal methods to real-world problems in engineering software. Chapter 6 describes a formal logic-based approach to the specification and (runtime) analysis of (electronic, legal) contracts. Chapter 7 reviews formal logic and CSP-based approaches to the specification and (cognitive, failure) analysis of systems involving interaction with users

By Markus Roggenbach, Antonio Cerone, Bernd-Holger Schlingloff, Gerardo Schneider, and Siraj Ahmed Shaikh.
Published by Springer, ISBN: 978-3-030-38799-0, 524 pages, 2022. https://doi.org/10.1007/978-3-030-38800-3.
Author's Contact Information: Maurice H. ter Beek, CNR-ISTI, Pisa, Italy; e-mail: maurice.terbeek@isti.cnr.it.

ACM 1433-299X/2025/10-ART35
https://doi.org/10.1145/3746236

(human behaviour). Chapter 8 demonstrates the use of CSP for the design and verification of security protocols, in particular authentication properties. Finally, the book is wrapped up in Chapter 9 with a historical perspective on formal methods for software engineering, surveying formal methods and tools and the key players that shaped the field.

The book is written in a highly accessible format; it contains way more examples (247) than definitions (63).[1] The authors clearly are proud of this achievement, given that the book contains a list of examples (and not a much more common list of figures). The examples, displayed in dedicated environments and many of which fragmented throughout the chapters, concern applications of formal methods to current challenges in software engineering. In particular, they address three recent trends in engineering computer systems and software: pervasiveness, criticality, and user centricity. Moreover, the use of tools is emphasized and explicitly supported by a website that accompanies the book and which contains exercises and links to tools.[2] Each chapter of the core of the book contains an annotated bibliography, next to the usual scientific references, and a section dedicated to current research directions, sometimes indicating the major conference series and journals in the area. The choice of formal methods and tools used in the book is of course debatable, but the choice made by the authors is reasonable and consistent: CSP is anticipated in Chapters 1 and 2, introduced in detail in Chapter 3 and used in applications in Chapters 7 and 8; CASL is anticipated in Chapter 1, first introduced in Chapter 2, then in more detail in Chapter 4 and used in Chapter 5; PAT is anticipated in Chapter 1, introduced in Chapter 2 and used in Chapter 7.

Actually, there are three more authors. Liam O'Reilly co-authored Chapter 4 and Hoang Nga Nguyen co-authored Chapter 8, while John V. Tucker authored the final Chapter 9.

The authors set the bar high by expressing the "hope that their advocated approach of utilising Formal Methods in software engineering will prove to be of advantage and become the new standard." The capitalization in this quote gives away the authors' expertise. They are formal methods experts rather than software engineers. This focus on formal methods is confirmed by the nice Foreword provided by Manfred Broy, recipient of the 2018 FME Fellowship award[3] conferred by Formal Methods Europe who—like me and the authors [2]—believes that "every computer scientist needs to know formal methods." [4] "Even more, software developers not being aware of the various benefits of formal methods cannot be called computer scientists or software engineers." [1] And who hopes that "this book will also find interest by practical engineers to give them some clue how formal foundations and rigorous methods could be combined to formal methods to help them in their everyday development tasks."

The authors conclude the book by advocating the use of formal methods in industry for engineering dependable quality software, not limited to safety-critical applications, and by underlining the fact that education in formal methods (thinking) is the key to achieve this. This is in line with recent literature on the topic [1, 3, 5]. Finally, the authors express the hope that this book may help advance the use of formal methods in software engineering practice through improved education. I believe this is indeed the case. Teaching formal methods is often considered a challenge, not in the least due to choice of formal methods and tools and accompanying examples to expose the students to. This book offers a mathematical rigorous approach driven by concrete and appealing examples.

References

[1] Manfred Broy, Achim D. Brucker, Alessandro Fantechi, Mario Gleirscher, Klaus Havelund, Markus Alexander Kuppe, Alexandra Mendes, André Platzer, Jan Oliver Ringert, and Allison Sullivan. 2025. Does every computer scientist need to know formal methods? *Form. Asp. Comput.* 37, 1 (2025), 6:1–6:17. DOI: https://doi.org/10.1145/3670795

[1] The definitions of various logics, CSP and CASL are included in two Appendices.
[2] https://sefm-book.github.io
[3] https://www.fmeurope.org/awards/

[2] Antonio Cerone, Markus Roggenbach, James Davenport, Casey Denner, Marie Farrell, Magne Haveraaen, Faron Moller, Philipp Körner, Sebastian Krings, Peter Csaba Ölveczky, Bernd-Holger Schlingloff, Nikolay Shilov, and Rustam Zhumagambetov. 2021. Rooting formal methods within higher education curricula for computer science and software engineering – a white paper. In *Proceedings of the Revised Selected Papers of the 1st International Workshop on Formal Methods – Fun for Everybody (FMFun'19) (CCIS, Vol. 1301)*, Antonio Cerone and Markus Roggenbach (Eds.). Springer, Germany, 1–26. DOI: https://doi.org/10.1007/978-3-030-71374-4_1

[3] Brijesh Dongol, Catherine Dubois, Stefan Hallerstede, Eric Hehner, Carroll Morgan, Peter Müller, Leila Ribeiro, Alexandra Silva, Graeme Smith, and Erik de Vink. 2025. On formal methods thinking in computer science education. *Form. Asp. Comput.* 37, 1 (2025), 8:1–8:23. DOI: https://doi.org/10.1145/3670419

[4] Maurice ter Beek, Manfred Broy, and Brijesh Dongol. 2024. The role of formal methods in computer science education. *ACM Inroads* 15, 4 (2024), 58–66. DOI: https://doi.org/10.1145/3702231

[5] Maurice H. ter Beek, Rod Chapman, Rance Cleaveland, Hubert Garavel, Rong Gu, Ivo ter Horst, Jeroen J. A. Keiren, Thierry Lecomte, Michael Leuschel, Kristin Yvonne Rozier, Augusto Sampaio, Cristina Seceleanu, Martyn Thomas, Tim A. C. Willemse, and Lijun Zhang. 2025. Formal methods in industry. *Form. Asp. Comput.* 37, 1 (2025), 7:1–7:38. DOI: https://doi.org/10.1145/3689374

Received 5 June 2025; revised 5 June 2025; accepted 23 June 2025