Cultures of Programming

What defines a correct program? What education makes a good programmer? The answers to these questions depend on whether programs are seen as mathematical entities, engineered socio-technical systems or media for assisting human thought. Programmers have developed a wide range of concepts and methodologies to construct programs of increasing complexity. This book shows how those concepts and methodologies emerged and developed from the 1940s to the present. It follows several strands in the history of programming and interprets key historical moments as interactions between five different cultures of programming. Rooted in disciplines such as mathematics, electrical engineering, business management or psychology, the different cultures of programming have exchanged ideas and given rise to novel programming concepts and methodologies. They have also clashed about the nature of programming; those clashes remain at the core of many questions about programming today. This title is also available as Open Access on Cambridge Core.

Tomas Petricek is Assistant Professor in the Faculty of Mathematics and Physics at Charles University, Prague. He uses interdisciplinary research methods to understand programming and to design simple and expressive programming tools. He developed a theory of context-aware computations at the University of Cambridge, worked on the F# programming language at Microsoft Research and built data exploration tools for non-programmers at The Alan Turing Institute and the University of Kent.

‘Tomas Petricek’s engaging and very readable book argues that programming is a complex human activity that must be understood from multiple perspectives. Written with deep and sympathetic understanding, it provides a unique overview of the history of programming and programming languages and profound insights into current practice that will help demystify the “black art” of coding.’

— Mark Priestley, National Museum of Computing

‘Is programming a practice of hacking, engineering, mathematics, management or humanities? As a computer scientist with the insights and style of a humanist, Petricek leads the reader through the last 70 years of programming to show how these practices have both persisted independently and interacted to produce key technical and methodological innovations through their disputes and synergies.’

— Warren Sack, University of California, Santa Cruz, author of The Software Arts

‘For some, computer programming is an abstract activity like mathematics; for others, an act of craftsmanship; for others, an industrial tool of organisation and control. This book presents a history of programming that preserves all these different points of view; indeed, it shows that these different “cultures” (and others) have been crucial to its development and success.’

— Simone Martini, University of Bologna

Cultures of Programming

The Development of Programming Concepts and Methodologies

TOMAS PETRICEK
Charles University, Prague

CAMBRIDGE
UNIVERSITY PRESS

Shaftesbury Road, Cambridge CB2 8EA, United Kingdom

One Liberty Plaza, 20th Floor, New York, NY 10006, USA

477 Williamstown Road, Port Melbourne, VIC 3207, Australia

314–321, 3rd Floor, Plot 3, Splendor Forum, Jasola District Centre, New Delhi – 110025, India

103 Penang Road, #05-06/07, Visioncrest Commercial, Singapore 238467

Cambridge University Press is part of Cambridge University Press & Assessment, a department of the University of Cambridge.

We share the University's mission to contribute to society through the pursuit of education, learning and research at the highest international levels of excellence.

www.cambridge.org
Information on this title: www.cambridge.org/9781009492348
DOI: 10.1017/9781009492379

When citing this work, please include a reference to the DOI 10.1017/9781009492379

First published 2026

Cover image: Simon McGill/Moment via Getty Images

A catalogue record for this publication is available from the British Library

Library of Congress Cataloging-in-Publication Data
Names: Petricek, Tomas author.
Title: Cultures of programming : the development of programming concepts and methodologies / Tomas Petricek.
Description: Cambridge ; New York, NY : Cambridge University Press, [2026] | Includes bibliographical references and index.
Identifiers: LCCN 2025029700 (print) | LCCN 2025029701 (ebook) | ISBN 9781009492348 hardback | ISBN 9781009492386 paperback | ISBN 9781009492379 epub
Subjects: LCSH: Computer programming–History. | Programming languages (Electronic computers)
Classification: LCC QA76.6 .P485 2026 (print) | LCC QA76.6 (ebook)
LC record available at https://lccn.loc.gov/2025029700
LC ebook record available at https://lccn.loc.gov/2025029701

ISBN 978-1-009-49234-8 Hardback

Contents

Preface

The first spark that eventually led to this book was a conversation between two expert programmers that I witnessed at the NDC Oslo conference in 2014. They were advocates of two different programming languages, Erlang and Haskell, talking about how a programming language can help you write good software. The two experts, both knowledgeable and open-minded speakers at the conference, were simply not able to have a reasonable conversation. The Erlang advocate explained how robust error recovery in Erlang makes software reliable, whereas the Haskell advocate was explaining how type systems can be used to eliminate programming errors. One could not understand why bother with types when you will have failures in your software anyway, whereas the other could not understand how restarting a process could make software correct. The two programmers were talking about a similar kind of programming problem, but each of them was looking at the problem from a different perspective and hence saw very different issues.

Despite the contemporary inspiration, this book is about the history of programming. Once you start looking for the kind of disagreements that I witnessed in Oslo, you find them in many places throughout the history of programming, ranging from the 1968 conference that popularised the term 'software engineering' to the debate in the 1980s about whether formal verification of software is logically possible and the rise of Agile development methods in the first decade of the twenty-first century. Fortunately, the interactions between different ways of thinking about programming do not lead just to disagreements. Once you start looking, you also find that many great advances in programming happened when different ways of thinking contributed to a single concept, starting with the very idea of a programming language.

What is surprising about those interactions is that the different ways of thinking about programming that have been shaping the discipline since the 1950s remain surprisingly stable over its 70-year history. In the 1950s, the different ways of thinking, which I refer to as *cultures of programming*, originated from the individual disciplines that contributed to programming including logic, mathematics, electrical engineering, but also business, psychology and military research. In the early days, this diversity may have been a sign of an immature field, but 70 years later, the existence of the same cultures of programming is instead a sign that programming is, and will remain, an inherently pluralistic discipline.

This book retells the history of programming through the perspective of interactions between five cultures of programming: the hacker; the engineering; the mathematical;

the managerial; the humanistic. While the five cultures are often linked to specific communities, the ways of thinking they represent also reappear independently in different contexts. As such, my use of the term culture is figurative rather than literal. I use it to highlight that different cultures of programming have different beliefs, assumptions and practices. Many of the cultures I identify in this book are, in some way, a part of computing folklore. This book exposes their more basic nature. To do so, I follow a number of historical strands from the 1950s to the present day, looking at how the different cultures of programming clash over the nature of programming, but also how they contribute to programming concepts ranging from programming languages and software engineering to structured programming, types, objects and tests.

Thinking about the different cultures of programming sheds a new light on the technical history of programming, but it also provides a new way of thinking about contemporary programming issues. The answer to what is a good program and how to create one differs dramatically when we see programs as mathematical entities, engineered socio-technical systems or media for assisting human thought. Similarly, the answer to how to best teach programming differs when you see programming as applied mathematics, a new form of literacy or a craft that can only be mastered through practice. To borrow a programming term, cultures of programming are a useful abstraction.

This book also points to broader socio-technical issues that are often discussed in the context of the history and philosophy of computing. The history of software engineering cannot be told without mentioning the struggle for managerial control and attempts to develop a more masculine notion of a computer professional. Similarly, the history of interactive programming cannot be told without a reference to the 1960s counterculture and the political ideology of free software. Although I focus on how different cultures of programming shape the way we program, the individual cultures can often be associated with particular social and political views. This book makes it possible to draw connections between the technical and the social history. In doing so, it is more technical than most history books and more historical than most computer science books.

My hopes for the readers of this book are twofold. On the one hand, I hope to provide computer scientists and programming practitioners with a useful perspective through which to look at their field, a much-needed context for understanding ‘how we got here’, but also to provide material that can inspire new ways of thinking. On the other hand, I hope to make the rich technical history of programming legible to those who are not programmers themselves, so that it can serve as a source of interesting episodes for further historical and philosophical reflection. Serving this dual purpose requires some careful balancing. In particular, I will sometimes use modern terminology to describe things from the past if it helps to make the text understandable. One specific challenge in writing this book has been the capitalisation of early programming language names where I generally follow the capitalisation used in recent historical reflections written by their creators.

To emphasise the importance of interactions between the five cultures of programming, each chapter starts with a dialogue between a teacher and students that represent

positions of the individual cultures. Although I sometimes adapt quotes from actual historical exchanges, the dialogues are a work of fiction.

In reality, individuals are much less explicit about their beliefs and rarely align with a single culture of programming completely. As we will see, the most interesting historical actors often bridge and contribute to multiple cultures over time. The purpose of the dialogues is to illustrate the clashes between the cultures, the ways in which they structure their arguments and how they can contribute to a single programming concept despite having different basic ideas about the nature of programming.

The students in the dialogues are named after ancient Greek philosophers. This distances the dialogues from actual historical actors and debates, but it provides a mnemonic for remembering which character represents which culture. I am well aware of the many inaccuracies and limitations of the name choices, but I hope you can forgive me those and enjoy the dialogues. The five students we will encounter in the openings of each chapter are:

Pythagoras, representing the *mathematical culture*, is named so for his many mathematical discoveries, but also the mystic view that all things were made of numbers. According to Aristotle, he believed that the principles of mathematics were the principles of all things.

Diogenes, speaking for the *hacker culture*, is named so as a critic of a corrupt society who believed that virtue is better revealed in action than in theory. He is also known for his unconventional lifestyle that included sleeping in a ceramic jar.

Xenophon, a philosopher and a military leader, represents the *managerial culture*. He is named so as the leader who managed a military force of 10,000 Greek mercenaries and for his later writings on its organisation.

Archimedes, speaking for the *engineering culture*, is named so as one of the first thinkers applying mathematics to physical phenomena and for his many practical inventions such as a screw pump and a compound pulley.

Socrates represents the *humanistic culture* for his pursuit of truth and knowledge, for his critical Socratic method of inquiry and his influence on the later humanist movement. He famously believed that the unexamined life is not worth living.

Acknowledgements

Looking at the acknowledgements sections of books that inspired me at the start of this project, I was always surprised by how long and how well written they were. Did the authors really talk to that many people, obtain and go through such quantities of feedback and visit that many places? If so, I thought, it must be almost impossible to write a book! Well, my acknowledgements are going to be equally long. When writing a book, you do, indeed, end up talking to many people, receive and incorporate lots of feedback, visit many places, stay at different institutions or even move countries. Alas, my acknowledgements are not going to be equally well written, because this is the last thing I need to finish before handing the final manuscript to my editor.

As a result of the unfortunate compartmentalisation of modern academia, being a computer scientist studying history and philosophy of the discipline raises some eyebrows and leads to the wrong kind of 'research outputs'. I'm grateful to my advisors, group leads and department heads who saw the value in my work and supported me over time, even if it did not fit in the usual boxes. This includes Alan Mycroft at the University of Cambridge, Richard Jones, Simon Thompson, Sally Fincher and Mark Batty at the University of Kent, and Petr Tůma at Charles University. I'm also grateful to David Corfield from the Department of Philosophy at the University of Kent who has been keen to cross the disciplinary boundary from the other side. The fact that the university administration has since decided to close down this thriving and highly ranked department remains a sad postscript to our inspiring collaboration.

I was fortunate to discover that even among computer scientists, many believe in transcending narrow disciplinary boundaries. The informal Cambridge Revolutionary Beers group provided both a sounding board for many of the ideas in this book and moral support over the numerous rejections and revisions that this manuscript went through. Thanks in particular to Antranig Basman, Stephen Kell, Joel Jakubovic, J. Ryan Stinnett and Dominic Orchard. Luke Church, Sam Aaron and Jonathan Edwards may not have often been there in person, but have regularly been there in spirit. The much-needed institutional support for the group has been provided by Fort St George, the Salisbury Arms and The Parcel Yard.

Outside my local circles, the impetus to start writing came in 2017 when Richard Gabriel and Guy Steele encouraged me to submit a paper drawing on my earlier work on programming errors to the Association for Computing Machinery (ACM) History of Programming Languages conference. I'm grateful for the initial nudge, but more importantly for their continued support of the project and feedback on multiple drafts

of the manuscript. The work of Dick Gabriel repeatedly reminded me that computer science does not have to be mere paradigmatic puzzle solving and served as inspiration throughout the work on the book. I'm also grateful to James Noble, Gilad Bracha, Rebbeca Wirfs-Brock and Allen Wirfs-Brock who provided interesting additional historical information and feedback on parts of the book.

Other communities that were instrumental for the development of this book centre around a number of industry-oriented programming conferences. These provided the opportunity to encounter first-hand several of the culture clashes that I write about in this book and to talk to the participants in order to figure out what was going on. The long-term organiser of the functional programming track at the NDC Oslo conference, Bryan Hunter, deserves credit for facilitating the meeting that I wrote about in the Preface, but also for many late-night chats that encouraged me to work on things that I truly believe in. Andrea Magnorsky, who brought history and philosophy of programming to the CodeMesh conferences, not only invited me to share my early ideas, but also broadened my perspective in many ways and connected me to another group of like-minded people. I'm grateful to Alvaro Videla and Miles Sabin for their comments. Finally, the organiser of the NewCrafts conference in Paris, Rui Carvalho, managed to create a unique space for discussing the broader picture of programming. I loved coming to learn about intriguing philosophical and technical ideas and for the insightful discussions. The list of NewCrafts community members who gave me interesting ideas and references includes Romeu Moura and Arnaud Bailly, but is much longer. I may have initially come to NDC, CodeMesh and NewCrafts, as well as Lambda Days and other places to share my passion for F# and I appreciate them inviting me again as my interests shifted to history and philosophy.

If the length of the mention in the Acknowledgements was to be proportional to the influence that a person had on my work and thinking, the next couple of pages would have to be dedicated to Liesbeth De Mol. Her thorough knowledge of the early history of programming significantly influenced my writing and her uncompromising personal commitment to honest and rigorous scholarship has been a continuous inspiration. Liesbeth also established the History and Philosophy of Computing (HaPoC) community and the collaborative Agence Nationale de la Recherche (ANR) PROGRAMme project that have provided unique venues for thinking about the history and philosophy of computing. There is so much in this book that has been influenced by the many discussions at a HaPoC conference, History and Philosophy of Programming (HaPoP) symposium or the PROGRAMme retreats. The list of people from those communities with whom I was able to discuss the ideas leading to this book is, again, too long. Those whose work had the most direct effect on this book include Troy Astarte, Pierre Depaz, Simone Martini, David Nofre, Mark Priestley and Julian Rohrhuber. The book *A Science of Operations* by Mark Priestley gave me the first ideas about what research on the history of programming can be. The in-depth debates on programming that took place at the CEUB Bertinoro were equally influential, even if they are not visible in the References. In addition to those mentioned already, the list of those who shaped my thinking and work include Selmer Bringsjord, Maarten Bullynck, Felice Cardone, Martin Carlé, Edgar Daylight,

Marie-José Durand-Richard, Cliff Jones, Baptiste Mélès, Elisabetta Mori, Pierre Mounier-Kuhn, Alberto Naibo, Maël Pégny, Giuseppe Primiero, David Schmudde, Henri Stephanou, Matte Szabo, Ray Turner and Nick Wiggershaus. The staff at CEUB Bertinoro and the many bars where those discussions took place deserve gratitude for making them possible, often well past the advertised closing time.

In more practical terms, I'm grateful to Karen Merikangas Darling for initial support of the project and to Adrian Johns for feedback on early drafts. The book would never have been completed without the help of my editorial team at Cambridge University Press, Lauren Cowles and Arman Chowdhury, who helped me navigate the tricky details of book production. I'm grateful to the designer of the beautiful book cover, which playfully illustrates how the different cultures of programming fit together and to Fiona Cole, who copyedited the manuscript on an impossibly tight schedule and not only brought consistency to the text but also saved me from countless embarrassing mistakes. I'm indebted to everyone who kindly replied to my email enquiries about image permissions.

I'm also grateful to the many people who work hard to preserve the historical material on which this book draws. This includes Al Kossow, the creator of bitsavers.org, the team behind the Computer History Museum, the Internet Archive, as well as the, now sadly defunct, Living Computers Museum. Petra Hoffmannová from the Library of the Faculty of Mathematics and Physics was able to find a curious article in a long-lost journal. I also greatly benefited from the resources collected by multiple shadow libraries. Their operators and supporters do more than anyone to preserve academic knowledge at great personal cost.

On a more personal note, I would like to thank my parents, who probably caused some of this by surrounding me with mountains of books from my early childhood in the most unexpected locations including the goat barn. I could probably estimate how many banana boxes you need for a bookshelf before I learned how to read. I'm grateful to my family, Evelina, Juliana and Alfred for the joys they bring to the part of life not spent in front of the laptop screen. They made the lonely task of writing a book bearable. I might be tempted to blame the latter two for the late timestamps of most of my git commits, but being a night owl anyway, this would be unfair. I am, however, immensely grateful to the former for making that way of working possible.

The work on this book has been financially supported by the University of Kent and Charles University where I did some of the work on the project. I received invaluable financial support through an ACM History Committee Fellowship, the ANR PROGRAMme project 'What is a computer program?' (ANR-17-CE38-0003-01), from the Charles University-funded research centre 'Language, Image, and Gesture: Forms of Discursivity' (UNCE24/SSH/026) and from the Charles University-funded grant 'Type systems for data-centric programming' (PRIMUS/24 /SCI/021). I am also extremely grateful to the Faculty of Mathematics and Physics and the Department of Distributed and Dependable Systems, for jointly providing financial support that enabled an open access publication of this book.