Lecture Notes in Computer Science 4821

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Jens Bennedsen Michael E. Caspersen
Michael Kölling (Eds.)

Reflections on the Teaching of Programming

Methods and Implementations

Springer

Volume Editors

Jens Bennedsen
IT University West
Fuglsangs Allé 20, 8210 Aarhus V, Denmark
E-mail: jbb@it-vest.dk

Michael E. Caspersen
University of Aarhus, Department of Computer Science
Aabogade 34, 8200 Aarhus N, Denmark
E-mail: mec@daimi.au.dk

Michael Kölling
University of Kent, Computing Laboratory
Canterbury, Kent CT2 7NF, UK
E-mail: mik@kent.ac.uk

Library of Congress Control Number: 2008925133

CR Subject Classification (1998): K.3, K.4, D.2, J.1

LNCS Sublibrary: SL 2 – Programming and Software Engineering

ISSN 0302-9743
ISBN-10 3-540-77933-7 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-77933-9 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12224616 06/3180 5 4 3 2 1 0

Foreword

For 50 years, we have been teaching programming. In that time, we have seen momentous changes. From teaching a first course using an assembly language or Fortran I to using sophisticated functional and OO programming languages. From computers touched only by professional operators to computers that children play with. From input on paper tape and punch cards, with hour-long waits for output from computer runs, to instant keyboard input and instant compilation and execution. From debugging programs using pages-long octal dumps of memory to sophisticated debugging systems embedded in IDEs. From small, toy assignments to ones that inspire because of the ability to include GUIs and other supporting software. From little knowledge or few theories of the programming process to structured programming, stepwise refinement, formal development methodologies based on theories of correctness, and software engineering principles.

And yet, teaching programming still seems to be a black art. There is no consensus on what the programming process is, much less on how it should be taught. We do not do well on teaching testing and debugging. We have debates not only on whether to teach OO first but on whether it *can* be taught first. This muddled situation manifests itself in several ways. Retention is often a problem. Our colleagues in other disciplines expect students to be able to program almost anything after a course or two, and many complain that this does not happen. In some sense, we are still floundering, just as we were 50 years ago.

Part of the problem may be that we are not sure what we are teaching. Are we simply providing knowledge, or are we attempting to impart a skill? Many introductory texts are oriented at teaching *programs* rather than *programming*—they contain little material on the programming process and on problem solving. And, judging from introductory texts, there is little consensus on how and when to specify program parts, how to document variables, how to teach algorithmic development, etc.

Another part of the problem may be that programming is indeed a difficult mixture of art and science—difficult to do and more difficult to teach. Yet another part of the problem may be that we have not discovered enough about programming and about teaching it. We need more research, experimentation, assessment, discussion, and debate.

In this context, this book, *Reflections on the Teaching of Programming*, is a welcome—and much needed—addition to our knowledge of programming and its teaching. Written by Scandinavian researchers and practitioners in computer science education, this book brings together a nice collection of articles on programming education, providing food for thought for anyone involved in the field, with articles that are, for the most part, based on proven implementation and real experience.

Learn about pedagogical experiments in using online tutorials, apprentice-based learning, the programming process, and problem-based learning. Delve into some issues of teaching OO. Look at software engineering issues in later courses. And read two chapters that deal with how we assess the work of students—a topic that has just begun to be explored.

You may not like, agree with, or believe everything you read—if you did, the book would be too "safe" and not provocative enough. But the book is clearly written by professionals who have thought deeply about and care about teaching programming, and we can all learn from it.

Perhaps, even, this book will be a catalyst for a renewed effort by our field to debate and do research on the teaching of programming, with teachers and researchers and textbook authors working together. Perhaps it will inspire us to incorporate research on pedagogy into our work, to learn how to assess our own teaching methods, and to think more deeply about the skill of programming and how it can be taught. Goodness knows, we need all this.

So, members of the SPoP network—the Scandinavian Pedagogy of Programming Network—thanks for developing this wonderful book. I have benefited greatly from reading it in draft form, and now the computing community will get the chance to benefit too.

February 2008 David Gries

Preface

The book you are holding is the result of the cooperation of a number of computing educators passionate about programming and teaching and is aimed at programming education practitioners in secondary and tertiary education and at computing education researchers.

The book is written by a group of primarily Scandinavian researchers and educators with special interest and experience in programming education. There are contributions from 24 authors about practical experience gathered in the process of teaching programming —for most of the authors for the past 15–20 years.

The reports are practically oriented. While several experiences described are associated with computing education research work, the emphasis here is on practical advice and concrete suggestions. It is expected that readers can get ideas for the teaching of programming that are directly applicable to the implementation of their own programming courses.

Care has been taken to select work for inclusion in this book that is not speculative, but based on proven implementation and real experience. The topics span a wide range of problems and solutions associated with the teaching of programming.

Part I consists of five chapters addressing issues related to introductory programming courses. The primary issues covered in this part are exposition of the programming process, apprentice-based learning, functional programming first, problem-based learning, and the use of on-line tutorials.

Part II consists of four chapters that specifically address issues related to introductory courses on object-oriented programming —the currently most prevailing approach to introductory programming. The primary issues covered are transitioning to object-oriented programming and Java, the use of the BlueJ environment to introduce programming, the use of model-driven programming as opposed to the prevailing language-driven approach, and the particular challenge of how to organize the first couple of weeks of an introductory course.

Part III consists of three chapters that address the more general challenge of teaching software engineering. The primary issues covered in this part are testing, extreme programming, and frameworks. These are all issues that are typically covered in later courses.

Part IV, the last part of the book, consists of two chapters addressing innovative approaches to feedback and assessment. The primary issues covered are active learning, technology-based individual feedback, and mini project programming exams.

It is expected that the topic areas covered will be of interest to a wide range of programming educators.

The authors have substantial experience in teaching programming, as well as a substantial body of publications in the computer science education literature.

The authors are members of the *Scandinavian Pedagogy of Programming Network*, and bring together a diverse body of experiences from the Nordic European countries. The 14 chapters of the book broadly describe experiences from their varying

backgrounds, but are carefully written and edited to present coherent units, not just individual, independent papers. Although the experiences described were gathered largely in Scandinavian countries, the book should be of equal interest to programming educators throughout the world.

February 2008

Jens Bennedsen
Michael E. Caspersen
Michael Kölling

Table of Contents

I Issues in Introductory Programming Courses

II Introducing Object-Oriented Programming

III Teaching Software Engineering Issues

IV Assessment

V Appendix