Ana Cavalcanti
Augusto Sampaio
Jim Woodcock (Eds.)

Refinement Techniques in Software Engineering

First Pernambuco Summer School
on Software Engineering, PSSE 2004
Recife, Brazil, November 23-December 5, 2004
Revised Lectures

Springer

Authors

Ana Cavalcanti
University of York
Department of Computer Science
Heslington, York YO10 5DD, UK
E-mail: Ana.Cavalcanti@cs.york.ac.uk

Augusto Sampaio
Federal University of Pernambuco
Centre for Informatics
CEP 50740-540, Recife-PE, Brazil
E-mail: acas@cin.ufpe.br

Jim Woodcock
University of York
Department of Computer Science
Heslington, York YO10 5DD, UK
E-mail: jim@cs.york.ac.uk

Library of Congress Control Number: 2006933059

CR Subject Classification (1998): D.2, D.1, F.3, K.6.3

LNCS Sublibrary: SL 2 – Programming and Software Engineering

ISSN 0302-9743
ISBN-10 3-540-46253-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-46253-8 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11889229 06/3142 5 4 3 2 1 0

Preface

The *Pernambuco School on Software Engineering (PSSE) 2004* was the first in a series of events devoted to the study of advanced computer science and to the promotion of international scientific collaboration. The main theme in 2004 was *refinement* (or *reification*). Refinement describes the verifiable relationship between a specification and its implementation; it also describes the process of discovering appropriate implementations, given a specification. Thus, in one way or another, refinement is at the heart of the programming process, and so is the major daily activity of every professional software engineer. The Summer School and its proceedings were intended to give a detailed tutorial introduction to the scientific basis of this activity.

These proceedings record the contributions from the invited lecturers. Each chapter is the result of a thorough revision of the initial notes provided to the participants of the school. The revision was inspired by the synergy generated by the opportunity for the lecturers to present and discuss their work among themselves, and with the school's attendees. The editors have tried to produce a coherent view of the topic by harmonizing these contributions, smoothing out differences in notation and approach, and providing links between the lectures. We apologize to the authors for any errors introduced by our extensive editing.

Although the chapters are linked in several ways, each one is sufficiently self-contained to be read in isolation. Nevertheless, Chap. 1 should be read first by those interested in an introduction to refinement.

Chapter 1. We begin by setting the scene, introducing ideas and notations that are taken as general background by the lecturers. We discuss program semantics, using Dijkstra's language of guarded commands as an illustration. We start with program assertions—perhaps the most widely used formal method in programming (assertions form about 1% of Microsoft Windows code)—and then continue with predicate transformers. We describe the basic notions of refinement, and discuss a simple refinement algebra. Then we relate this to program development by formalizing the process of stepwise refinement of specifications into programs. Finally, we describe a useful mathematical structure: the lattice of specifications ordered by refinement. Each of the following four chapters uses these ideas to illustrate refinement for a given paradigm: object orientation, concurrency, probabilistic programs, and real-time and fault-tolerant systems.

Chapter 2. Sampaio and Borba describe refinement in the object-oriented setting, illustrating the ideas using sequential Java, although their work is of general applicability. Their approach is algebraic: they give laws for reasoning about and refining object-oriented programs. They discuss soundness with respect to predicate transformer semantics, and demonstrate completeness by showing that their set of laws is comprehensive enough to be able to reduce any program to a normal form. Finally, they show how their laws can be used to refactor programs, in order to adapt their structure while preserving their semantics.

Chapter 3. Davies describes refinement of concurrent and distributed systems using the notations of CSP. He starts by introducing CSP as a process algebra, with a set of operators and a rich collection of laws for reasoning about the behavior of processes. The denotational semantics of the language is used to give a simple notion of refinement between processes: every behavior of an implementation must be a specified behavior. These ideas are explored through an example involving protocols and their service specifications.

Chapter 4. McIver and Morgan add probabilistic nondeterminism to Dijkstra's guarded command language. They make a corresponding change to the programming logic, replacing weakest preconditions by greatest pre-expectations. These are generalizations of predicates that can be used to express the probability that a program achieves a postcondition. They explain how we can extend standard reasoning concepts like invariants and variants to handle probabilistic programs. They give a series of examples and two longer case studies.

Chapter 5. Liu and Joseph deal with refinement in real-time and fault-tolerant systems. They use transition systems as their computational model and Lamport's "Temporal Logic of Actions (TLA)" as a specification language in reasoning about functional correctness, timing properties, fault-tolerance, and schedulability. Their work is explained through an example of the interface between a processor and a memory device.

Chapter 6. Cavalcanti and Woodcock introduce "Unifying Theories of Programming" as a uniform foundation for all these paradigms. They give a tutorial introduction to an alphabetized version of Tarksi's relational calculus. They show how this leads to a simple denotational semantics of a language of terminating programs, and show that they form a complete lattice. They extend this work to Hoare-He designs, a relational model of pre- and postcondition specifications, exploring the space of designs as a subtheory of relations characterized by certain healthiness conditions. Then they turn their attention to another relational subtheory—reactive processes—once again characterized by healthiness conditions. Finally, they show that the reactive image of the design lattice gives a suitable semantic model for CSP. They end by comparing this semantics with the model given by Davies in his chapter. After this survey of refinement and its different theories, the final two chapters are on mechanical or automated support for refinement.

Chapter 7. Clayton and O'Halloran describe the practice of refinement in industry. They have designed the "Compliance Notation" for demonstrating the refinement relation between software and its specification. They have built a tool to support this demonstration, an essential item for industrial-scale application of refinement. They describe an extended example of a correctness argument for programs written in the Spark Ada subset. They present an application involving the correct implementation of control laws that govern control systems.

Chapter 8. Déharbe presents a very successful approach to verification, whose high level of automation has made it very attractive to industry. This chapter presents the main temporal logics used for specification of properties, and the main structures and algorithms used in tools. Widely used tools like SPIN and SMV are discussed. The approach is briefly compared with that adopted for model checking of CSP processes.

We are grateful to the members of the Organizing Committee, who worked very hard to provide an enjoyable experience for all of us. Without the support of our sponsors, *PSSE 2004* could not have been a reality. Their recognition of the importance of this event for the Software Engineering community in Latin America is greatly appreciated. We would also like to thank all the lecturers for their invaluable technical and scientific contribution, and for their commitment to the event; the effort of all authors is greatly appreciated. Finally, we are grateful to all the participants of the school. They are the main focus of the whole event.

April 2006

Ana Cavalcanti
Augusto Sampaio
Jim Woodcock

Organization

PSSE 2004 was organized by the Centro de Informática, Universidade Federal de Pernambuco (CIn/UFPE), Brazil, in cooperation with the United Nations University, International Institute for Software Technology (UNU/IIST), and the University of York, UK.

Executive Committee

Sponsoring Institutions

Formal Methods Europe
Sociedade Brasileira de Computação,Brazil
United Nations University, Macau
Universidade Federal de Pernambuco (CIn/UFPE), Brazil
University of York, UK

Acknowledgements

Paulo Borba and Augusto Sampaio. Several parts of their chapter were extracted and adapted from their previous joint work with Ana Cavalcanti and Márcio Cornélio: “Algebraic reasoning for object-oriented programming”, *Science of Computer Programming*, 52:53–100, 2004. They thank their collaborator David Naumann for many discussions that significantly contributed to the work they report here, which was partially carried out when the authors were visiting the Stevens Institute of Technology at New Jersey, USA. They also thank Tiago Massoni and Rohit Gheyi for several important comments on an earlier version of their chapter, and Leila Silva for discussions concerning the impact of reference semantics in the proposed laws for their language. They are partially supported by the Brazilian Research Agency, CNPq, grants 521994/96–9 (Paulo Borba), 521039/95–9 (Augusto Sampaio), and 680032/99-1 (DARE CO-OP project, jointly funded by CNPq PROTEM-CC and the National Science Foundation).

Annabelle McIver and Carroll Morgan. Their chapter reports work carried out with Jeff Sanders, Thai Son Hoang and Karen Seidel. It was supported in the UK by the *EPSRC* and in Australia by the *ARC.* An earlier version of Sect. 1–5 of their chapter first appeared in the *South African Computer Journal* [42],who have graciously allowed it to be reprinted. Section 6 is new; it is based on (and extends) one of the lectures of the Summer School. The *SACJ* article was in turn a "transliteration" of a still earlier work [194] concerning probabilistic *Generalized Substitutions* [3]Chapters 1,2 and Sect. 3.1 of a text [181] by the same authors provides further insight into the material of Sect. 1–5.

Zhiming Liu and Mathai Joseph. Zhiming Liu's work has been supported by the UNU-IIST Research Project on Formal Methods of Object and Component Systems and the project HighQSoftD funded by Macao Science and Technology Fund, and the Chinese NSF project 60573085.

Ana Cavalcanti and Jim Woodcock. Their contribution is partially funded by the Royal Society of London and by QinetiQ Malvern, but their greatest debt is to Tony Hoare and He Jifeng for their inspirational work in unifying theories. The authors were the *Formal Methods Europe Lecturers* at the *PSS 2004*, and are grateful to FME for their support of this event, where their chapter was first presented. Earlier versions of the material were presented at the Universities of Oxford, Kent, and York, at QinetiQ Malvern, at the Danish Technical University, and as an invited tutorial at *Integrating Formal Methods, IFM 2004.* The authors have benefited from discussions with Chen Yifeng about closure; their proof of ***R2-L0*** is based on his original idea. Augusto Sampaio made a large number of detailed and useful comments on the technical material in the chapter.

Phil Clayton and Colin O'Halloran. Simulink is a registered trademark.

David Déharbe. His chapter subsumes and improves on his work in *Logic for Concurrency and Synchronisation*, volume 18 of Trends in Logic.

Table of Contents