

SpringerBriefs in Computer Science

Series editors

Stan Zdonik, Brown University, Providence, USA
Shashi Shekhar, University of Minnesota, Minneapolis, USA
Jonathan Katz, University of Maryland, College Park, USA
Xindong Wu, University of Vermont, Burlington, USA
Lakhmi C. Jain, University of South Australia, Adelaide, Australia
David Padua, University of Illinois Urbana-Champaign, Urbana, USA
Xuemin (Sherman) Shen, University of Waterloo, Waterloo, Canada
Borko Furht, Florida Atlantic University, Boca Raton, USA
V.S. Subrahmanian, University of Maryland, College Park, USA
Martial Hebert, Carnegie Mellon University, Pittsburgh, USA
Katsushi Ikeuchi, University of Tokyo, Tokyo, Japan
Bruno Siciliano, Università di Napoli Federico II, Napoli, Italy
Sushil Jajodia, George Mason University, Fairfax, USA
Newton Lee, Newton Lee Laboratories, LLC, Tujunga, USA

More information about this series at <http://www.springer.com/series/10028>

Márcio Ribeiro · Paulo Borba
Claus Brabrand

Emergent Interfaces for Feature Modularization

 Springer

Márcio Ribeiro
Computing Institute
Federal University of Alagoas
Maceió, Alagoas
Brazil

Claus Brabrand
Department of Software and Systems
IT University of Copenhagen
Copenhagen
Denmark

Paulo Borba
Informatics Center
Federal University of Pernambuco
Recife
Brazil

ISSN 2191-5768

ISBN 978-3-319-11492-7

DOI 10.1007/978-3-319-11493-4

ISSN 2191-5776 (electronic)

ISBN 978-3-319-11493-4 (eBook)

Library of Congress Control Number: 2014951726

Springer Cham Heidelberg New York Dordrecht London

© The Author(s) 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)

To our families

Foreword

Software systems in a wide range of domains need to be increasingly configurable. Program families and software product lines are well-known forms of configurable software systems. Companies are often launching initiatives and new projects involving configurable software systems, as they enable organizations to achieve time-to-market gains and cost reductions. These systems are often decomposed into a set of features, which represent semantically cohesive units of behavior. Features describe the commonalities and variabilities of programs potentially derived from a configurable software system.

Unfortunately, developers still struggle to properly maintain the source code of software families and product lines. These systems typically have many features, and each feature may have from dozens to thousands of lines of code. The implementation of each feature is often scattered through the modules of a program. In other words, the boundaries of features are often not the same as the implementation modules in a program. Hence, a long standing challenge faced by the software maintainers is to identify, buried into hundreds of lines of code, relevant feature dependencies while realizing their changes. Existing techniques and tools lack proper modularity support and do not properly inform the boundaries and feature dependencies in the source code. Even after developers spent a huge effort to figure out how to implement their feature changes, they often miss feature dependencies and introduce bugs in their configurable systems.

Over the past two decades an extensive body of literature has emerged to cover a wide range of topics on software engineering for configurable systems. Researchers and practitioners have gathered a good collective understanding of how to design and implement high-quality configurable systems. However, until recently, our knowledge on how to support software developers on maintaining these systems was still very limited. It is stimulating to see a book that makes a significant improvement to modular maintenance of configurable systems. This book is clearly a distinguished asset in the history of software maintenance for product lines and program families.

During the past 5 years, I have had the honor to meet and closely follow the Doctoral research of Márcio Ribeiro, a talented young researcher, who came up with brilliant ideas on how to tame the complexity of maintaining configurable systems. The book characterizes the notion of emergent feature interface, an essential modularity mechanism to support daily tasks of configurable software maintainers. The on-demand computation of emergent feature interfaces is only possible through feature-sensitive data flow analysis, an innovative technique being described in this book. The description and evaluation of emergent feature interfaces are also inspiring, and treated with the rigor that is characteristic of a mature piece of research.

The several awards granted to Márcio's work over the last years are an evidence of the relevance and impact of his research. You should definitely read this book if you want a thorough coverage of an innovative approach that has the potential to shape next-generation tools for maintaining configurable systems.

Rio de Janeiro, December 2013

Alessandro Garcia

Contents

1 Introduction	1
References	4
2 Software Families, Software Products Lines, and Dataflow Analyses	7
2.1 Software Families and Software Product Lines	7
2.1.1 Conditional Compilation	11
2.1.2 Virtual Separation of Concerns	12
2.2 Dataflow Analysis	14
2.2.1 The Three Constituents	14
2.2.2 Reaching Definitions	18
References	20
3 Feature Dependencies	23
3.1 Maintaining Features	24
3.1.1 Scenario 1: Maintenance in a Feature Is Accomplished only for Some Products	24
3.1.2 Scenario 2: Maintenance in a Feature Breaks the Compilation of Another	25
3.1.3 Scenario 3: Maintenance in a Feature Breaks the Behavior of Another.	27
3.2 Problem’s Dimension	28
References	32
4 Emergent Feature Modularization	35
4.1 Maintenance Features (Revisited).	36
4.2 Definition	37
4.3 General Idea	37
4.4 Feature-Sensitive Analysis	39

4.4.1	Feature-Oblivious Analysis—Brute Force ($\mathcal{A}1$)	40
4.4.2	Consecutive Feature-Sensitive Analysis ($\mathcal{A}2$)	41
4.4.3	Simultaneous Feature-Sensitive Analysis ($\mathcal{A}3$)	43
4.5	From Feature-Sensitive Dataflow Analyses to Emergent Interfaces	45
4.5.1	Executing Feature-Sensitive Dataflow Analyses.	46
4.5.2	Generating Emergent Interfaces.	47
4.6	Supporting Developers with Emergo	50
4.6.1	Feature Model.	51
4.6.2	Architecture	51
4.7	Improving Expressiveness	52
4.8	Emergent Interfaces in Other Mechanisms.	53
	References	55
5	Evaluation	57
5.1	General Setting: Goal, Questions, and Metrics	57
5.2	Effort Study	59
5.2.1	Study Setting	59
5.2.2	Results and Discussion.	61
5.2.3	Answering Question 1	64
5.2.4	Threats to Validity.	65
5.3	Performance	66
5.3.1	Study Setting	66
5.3.2	Results and Discussion.	67
5.3.3	Answering Question 2	70
5.3.4	Threats to Validity.	70
	References	70
6	Comparison with Other Approaches	73
6.1	Interfaces for Non-annotative Approaches	73
6.2	Separation of Concerns.	74
6.3	Safe Composition	75
6.4	Analyses on Preprocessor-Based Systems	75
6.5	Dataflow Analysis for Maintainability	76
6.6	Checking and Verification.	77
	References	78
7	Concluding Remarks	81
7.1	Limitations	82
7.2	Future Work	83
	References	84

Acronyms

API	Application Programming Interfaces
AST	Abstract Syntax Tree
CFG	Control-Flow Graph
CIDE	Colored IDE
FODA	Feature-Oriented Domain Analysis
SLoC	Source Lines of Code
SPL	Software Product Line
VSoC	Virtual Separation of Concerns