Mastering Software Variability with FeatureIDE

Jens Meinicke • Thomas Thüm •
Reimar Schröter • Fabian Benduhn •
Thomas Leich • Gunter Saake

Mastering Software Variability with FeatureIDE

Springer

Jens Meinicke
Carnegie Mellon University
Pittsburgh, PA, USA

Thomas Thüm
TU Braunschweig
Braunschweig, Germany

Reimar Schröter
Otto-von-Guericke Universität Magdeburg
Magdeburg, Germany

Fabian Benduhn
Otto-von-Guericke Universität Magdeburg
Magdeburg, Germany

Thomas Leich
Hochschule Harz
Wernigerode, Germany

Gunter Saake
Otto-von-Guericke Universität Magdeburg
Magdeburg, Germany

ISBN 978-3-319-61442-7 ISBN 978-3-319-61443-4 (eBook)
DOI 10.1007/978-3-319-61443-4

Library of Congress Control Number: 2017944731

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Foreword by Don Batory

Jay Misra once told me "The quality of a research area is measured by the quality of its teaching materials" (Misra 2004). A corollary is "and by the quality of its tools."

Although the term software product lines (SPLs) is about 20 years old, the concept of SPLs goes back 50 years, known back then as SYStem GENeration (SYSGEN) for the custom creation of operating systems (Wikipedia 2017). It was given a more visionary form by McIlroy in 1968, when he conceptualized software variability by the composition of components (McIlroy 1968). In the intervening 30 years grew the recognition that the custom production of software, not just operating systems, is a core activity of software engineering; the study of SPLs, unto themselves, was a worthy area of scientific and engineering research.

In the beginning, tools for SPL development were primitive—the use of C preprocessor dominated. Analyses that were specific to SPLs and their amenities were lacking. That was good enough for many in the trenches; the status quo was unacceptable to scientists. There should be Engineering (with a capital "E"), not hacking, behind SPLs. This embodied the thrust of SPL scientific/engineering research for the last 20 years. Scientific progress lagged far behind industrial tools prior to 2005. Now the situation has reversed—industrial tools for SPLs are far behind scientific advances. This is the normal state of scientifically-driven research areas.

FeatureIDE is an Eclipse plug-in that showcases many of the core conceptual advances in the last 10 years of SPL research. *FeatureIDE* presents the best tooling (in my opinion) of classical feature models—models that are devoid of numeric attributes, feature replication, and feature modularization,[1] a widely-accepted set of feature model analyses (tests for void models, dead features, false-optional features, generalizing/specializing feature models via edits), and amenities—such as visual prompts for feature-expression completion, next-generation tooling to distinguish the source code of different features with different colors, runtime-variability support, and *Javadoc* support for SPLs. Orthogonal to these analyses is the choice among several distinct ways to encode variability in source code (different preprocessors) and different ways to modularize features (*FeatureHouse*, *AHEAD*).

[1] Researchers—Take a hint at what to focus on next!

FeatureIDE is a much needed and necessary step forward. It presents the analyses and amenities that one might/should see in next-generation SPL tooling. I have used *FeatureIDE* for many years and am very pleased to see SPL research ideas come to life. So will you. I thank the *FeatureIDE* team for their tireless work and contribution to the SPL field.

University of Texas at Austin
Austin, TX, USA

Don Batory

Preface

In the era of mobile devices and the Internet of things, software systems are ubiquitous. A multitude of hardware specifics, fast development of applications, and the need of personalization foster the requirements for software reuse even more than in the past. Mastering *variability* is one of the key concerns of modern software engineering.

To master variability in industry-scale software development, tool support is absolutely essential. More than 10 years ago, we started the development of *FeatureIDE* as an Eclipse plug-in to address these needs. This software went through several severe redesigns over the years but is now a stable tool for feature-based implementation of variable software systems used in a large number of university courses as well as in industrial projects. *FeatureIDE* is the only open-source IDE for managing software variability with this stability and with support of several implementation techniques for mastering variability. Because of this central role of *FeatureIDE* for teaching and development, the core implementation team decided to write this book.

This book is a self-contained practical introduction to the use of *FeatureIDE* to implement variable systems. Each presented technique can directly be tried out in a *FeatureIDE* installation on your own computer. All code examples are added to the standard distribution and can immediately be used for own modifications. The book contains three major thematic parts: modeling variability using features, implementing variability with conditional compilation, and implementing variability with feature-oriented programming.

The book is suited both for students using a tool for deepening the theoretical foundations of variability modeling and implementation, as well as a reference for practitioners needing a stable and scalable tool for industrial application. *FeatureIDE* is used in industrial settings with several thousand features for analyzing variability models and generating products.

More than a decade of developing an open-source project involved more students and developers than one can mention in a preface. During the production of the book, however, several persons were involved. We want to mention them explicitly: Sebastian Krieter, Christopher Sontag, Joshua Sprey, Marcus Pinnecke, Andy Kenner, Christopher Kruczek, Jacob Krüger, and Wolfram Fenske. Furthermore, we gratefully acknowledge fruitful discussions and contributions to the open-source

project by Christian Kästner, Sven Apel, Ina Schaefer, Stefan Krüger, Mustafa Al-Hajjaji, Juliana Alves Pereira, Sofia Ananieva, Timo Günther, Matthias Kowal, Alexander Knüppel, Klaus Birken, Hendrik Speidel, Frederik Kramer, Roman Popp, Roland Beckert, Jörg Liebig, Sandro Schulze, Janet Siegmund, and Norbert Siegmund.

The work on FeatureIDE was supported by several fundings, among them grants by German Federal Ministry of Education and Research (BMBF: 01IS14017A, 01IS14017B) and by German Research Foundation (DFG: SA 465/34-2, SA 465/49-1, LE 3382/2-1, SCHA 1635/4-2, and LO 2198/2-1). Last, but not least, we are grateful to our families and friends for their support, which was essential for the success of this endeavor.

Pittsburgh, PA, USA — Jens Meinicke
Braunschweig, Germany — Thomas Thüm
Magdeburg, Germany — Reimar Schröter
Magdeburg, Germany — Fabian Benduhn
Wernigerode, Germany — Thomas Leich
Magdeburg, Germany — Gunter Saake
May 2017

Contents

Part I Introduction

1 Software Variability 3
- 1.1 What Is Software Variability? 4
- 1.2 Variability Implementation Mechanisms 6
- 1.3 Mastering Variability with FeatureIDE 8
- 1.4 Structure of the Book 9

2 Getting Started 11
- 2.1 Download and Installation of Java 11
- 2.2 Download of FeatureIDE 12
- 2.3 Installation of FeatureIDE 12
- 2.4 Summary 17

3 FeatureIDE in a Nutshell 19
- 3.1 Opening the FeatureIDE Perspective 19
- 3.2 Loading FeatureIDE Examples 20
- 3.3 Structure of FeatureIDE Projects 22
- 3.4 Modeling Variability with Feature Models 24
- 3.5 Implementation of Software Variability 25
- 3.6 Creating Configurations 26
- 3.7 Product Generation and Execution 27
- 3.8 Summary and Further Reading 28

4 An Elevator as a Running Example 31
- 4.1 Creating the Elevator 31
- 4.2 Execution of the Running Example 37
- 4.3 Summary and Further Reading 38

Part II Tool Support for Feature Modeling and Configuration

5 Feature Modeling 43
- 5.1 Creation of Feature Models 43
- 5.2 Modeling Tree Constraints 47
- 5.3 Modeling Cross-Tree Constraints 49
- 5.4 Visualizing Large Feature Models 52

5.5 Importing and Exporting Feature Models 57
5.6 Further Pages of the Feature Model Editor 59
5.7 Summary and Further Reading 61

6 Product Configuration 63
6.1 Creating Configurations 64
6.2 Editing Configurations 65
6.3 Summary and Further Reading 70

7 Feature Traceability in Feature Models and Configurations 73
7.1 Tracing Features in the Feature Model 73
7.2 Tracing Features in Configurations 77
7.3 Summary and Further Reading 80

8 Quality Assurance for Feature Models and Configurations 81
8.1 Quality Assurance for Feature Models Using the Editor 81
8.2 Quality Assurance Based on FeatureIDE's Statistics View 89
8.3 Quality Assurance for Configurations 91
8.4 Quality Assurance by an Automatic Generation of Configurations 92
8.5 Summary and Further Reading 94

Part III Tool Support for Conditional Compilation

9 Conditional Compilation with FeatureIDE 97
9.1 Introduction to Conditional Compilation 97
9.2 Preprocessor Munge 99
9.3 Preprocessor Antenna 101
9.4 Summary and Further Reading 102

10 Developing an Elevator with Conditional Compilation 105
10.1 Creating an Elevator Product Line Using Antenna 106
10.2 Adding the Feature "Service" to the Elevator Product Line 109
10.3 Adding Feature "FIFO" to the Elevator Product Line 113
10.4 Summary and Further Reading 120

11 Feature Traceability for Conditional Compilation 123
11.1 Tracing Features in Project Explorer 123
11.2 Tracing Features in Java Editor 125
11.3 Tracing Features in FeatureIDE Outline 127
11.4 Tracing Features in Collaboration Diagram 129
11.5 Summary and Further Reading 130

12 Quality Assurance for Conditional Compilation 131
12.1 Consistency Checking for Preprocessors 132
12.2 Product-Based Analyses for Preprocessors 135

12.3 Code Metrics for Preprocessors 137
12.4 Summary and Further Reading 138

Part IV Tool Support for Feature-Oriented Programming

13 Feature-Oriented Programming with FeatureIDE 143
13.1 Feature-Oriented Programming 143
13.2 Feature Modules with FeatureHouse 146
13.3 Feature Modules with AHEAD 148
13.4 Feature Modules with FeatureC++ 150
13.5 Feature-Oriented Programming with FeatureIDE 152
13.6 Summary and Further Reading 154

14 Developing an Elevator with Feature-Oriented Programming 155
14.1 Creating an Elevator Product Line 155
14.2 Adding Feature "Service" to the Elevator Product Line 159
14.3 Adding Feature "FIFO" to the Elevator Product Line 164
14.4 Summary and Further Reading 171

15 Feature Traceability for Feature-Oriented Programming 173
15.1 Tracing Features in Project Explorer 174
15.2 Tracing Features in Java Editor 177
15.3 Tracing Features in FeatureIDE Outline 178
15.4 Tracing Features in Collaboration Diagram 179
15.5 Summary and Further Reading 180

16 Quality Assurance for Feature-Oriented Programming 183
16.1 Consistency Checking for Feature Modules 184
16.2 Product-Based Analyses for Feature Modules 186
16.3 Code Metrics for Feature Modules 192
16.4 Summary and Further Reading 195

Part V Further Tool Support in FeatureIDE

17 Tool Support Beyond Preprocessors and Feature Modules 199
17.1 Product-Line Implementation with Runtime Variability 200
17.2 Product-Line Implementation with Black-Box Frameworks 203
17.3 Product-Line Implementation with Aspect-Oriented Programming 207
17.4 Summary and Further Reading 210

18 Tool Support for Product-Line Maintenance 211
18.1 Refactoring of Product Lines 212
18.2 Source-Code Documentation with Javadoc 215
18.3 Formal Specification with Method Contracts 222
18.4 Summary and Further Reading 225

19 Overview on FeatureIDE 227
19.1 Overview on Implementation Techniques 227
19.2 Editors and Views for Feature Models 228
19.3 Editors and Views for Configurations 230
19.4 Package Explorer and Project Explorer 231
19.5 Editors and Views for Source Code 232
19.6 Summary and Further Reading 234

References 235

Index 241

http://www.springer.com/978-3-319-61442-7

Mastering Software Variability with FeatureIDE

Meinicke, J.; Thüm, Th.; Schröter, R.; Benduhn, F.; Leich, Th.; Saake, G.

2017, XII, 243 p. 50 illus., 30 illus. in color., Hardcover
ISBN: 978-3-319-61442-7