

Feature-Oriented Software Product Lines

Sven Apel · Don Batory
Christian Kästner · Gunter Saake

Feature-Oriented Software Product Lines

Concepts and Implementation

 Springer

Sven Apel
University of Passau
Passau
Germany

Christian Kästner
Carnegie Mellon University
Pittsburgh, PA
USA

Don Batory
The University of Texas
Austin, TX
USA

Gunter Saake
Otto von Guericke University
Magdeburg
Germany

ISBN 978-3-642-37520-0 ISBN 978-3-642-37521-7 (eBook)
DOI 10.1007/978-3-642-37521-7
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013935777

ACM Computing Classification (1998): D.2, K.6

© Springer-Verlag Berlin Heidelberg 2013

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)

Foreword

Features are a fundamental notion in modern software engineering. Defined once by Pamela Zave as “incremental units of functionality”, they are central to how software is developed now. Much of today’s software is developed using scenario-driven approaches. In essence, scenarios, also known as use cases or user stories, are specifications of features.

Feature-oriented software development shines in the context of software product lines. Virtually any successful software faces the need to cater different feature combinations to different customer segments. Product line engineering accelerates product development by leveraging the commonalities among the product line members, while managing the differences, also known as variabilities, among them. Features play a key role in modeling commonalities and variabilities and in managing the development of product lines. Major organizations, including General Motors and Danfoss, use feature-oriented approaches to successfully develop complex software-intensive product lines.

The message of this book is that much of the tremendous power of features is yet to be unlocked by *making features explicit throughout the entire systems and software lifecycle*. The explicit treatment of features in requirements, architecture, implementation, and verification and validation can greatly improve the management of software. Features are abstractions that can be made understandable to all stakeholders, both technical and nontechnical, enabling effective communication among the stakeholders and planning, implementation, and evolution of complex software product lines. Many of the ideas and tools presented in this book are applicable not only to traditional software product lines, but also to a wide range of variability-intensive systems, including highly-configurable applications, computing platforms, and software ecosystems.

The book provides a systematic introduction to feature-oriented software product lines, and leads the reader to more advanced topics in its second half. The authors distill the concepts and principles underlying the field with remarkable clarity, providing a much-needed foundation for the field. They also illustrate these

concepts and principles using concrete examples, showcasing languages, tools, and systems from both industrial practice and latest research. The advanced part of the book covers recent research results, many of which the authors have helped to advance. The reader can also enhance his or her learning experience by completing the provided exercises. The book will make an excellent upper-year undergraduate or introductory graduate text; but also practitioners will find it invaluable to enhance their software engineering toolbox with the powerful concepts and techniques of feature-oriented software product lines.

There is no better team than these four authors to write about feature-oriented software product lines. The authors have made fundamental scientific and engineering contributions to the field. Don has pioneered feature-oriented composition of software with his work on GENESIS, an extensible database management system, in late 1980s, and generalized the concepts and principles underlying it in early 1990s. He has continued on this path, advancing the theory and designing languages and tools, but also inspiring generations of researchers to join the effort. I started working in the field after attending Don's tutorial on "Software Systems Generators, Architecture, and Reuse" at the International Conference of Software Reuse in 1996. Shortly after, I shared my excitement for Don's ideas on software generation with Ulrich Eisenecker, which led to our work on automating component assembly based on feature models and, eventually, the book on Generative Programming in 2000. The subsequent decade has seen tremendous progress. New generations of young researchers have worked on techniques for feature-oriented modularization, variability-aware analyses, and empirical studies of systems with variability. Sven and Christian, enjoying the creative and fertile environment of Gunter's research group in Magdeburg and inspired by Don's work, have played leading roles in this progress. Their research results on feature-oriented software product lines have reached wide audiences at major software engineering conferences, such as the International Conference on Software Engineering and the Conference on Foundations of Software Engineering. Today, the four authors are central figures in the growing, vibrant community, known as Feature-Oriented Software Development (FOSD).

The notion of features has already profoundly affected how software is engineered, and this is just the beginning. Features can substantially improve the communication among all stakeholders and will likely lead to new, more effective ways to modularize and develop software. Despite the tremendous progress so far, much potential and many more discoveries lie ahead. Future work topics include finding most effective ways to exploit features in software modularization, creating techniques for re-engineering legacy towards feature orientation and evolving feature-oriented software, and also supporting features more pervasively in tools and infrastructures, such as in configuration management. Years to come will bring new, unexplored topics, which none of us can possibly foresee.

It is a great joy to see the new generations of brilliant young researchers joining the thriving FOSD community. I invite you to join this exciting ride, too. This book is your ticket!

Waterloo, April 2013

Krzysztof Czarnecki

Preface

The idea for this book arose from a series of lectures on modern programming paradigms, feature-oriented programming, and software product lines that are continuously held at the Universities of Magdeburg, Marburg, Passau, Texas at Austin, and others. Our collaboration reaches back to 2006, when Sven and Christian visited Don's group in Austin. Don's lecture on feature-oriented programming was inspiration for the lecture series set up in 2007 at the Universities of Magdeburg and Passau, which is the basis for this book. In a joint effort, we developed and continuously refined the teaching material for the lectures since then, until the present day.

Our interest in this topic was always the developer's perspective of discussing implementation techniques that are suitable for constructing variable software. We would have preferred to use a textbook for our lectures from the shelf, but existing product-line textbooks said precious little on implementation techniques. Eventually, in 2011, the material was stable, such that the natural next step was to write a proper text on this topic, meant not only for our students, but for all students of computer science and related fields as well as researchers and practitioners interested in software product lines.

Writing a textbook is an enormous endeavor, and this book would not have been possible without the help of our colleagues, students, friends, and families. In particular, we thank Martin Kuhlemann, Jörg Liebig, Norbert Siegmund, and Thomas Thüm, who used and improved the teaching material that was the basis for this book. Furthermore, we thank David Broneske for his support in producing proper graphics for this book, Jörg Liebig, Sandro Schulze, Norbert Siegmund, and Thomas Thüm for feedback on selected chapters and the exercises proposed in the book, as well as pure-systems, Janet Siegmund, and Thomas Thüm for providing screenshots of the tools pure::variants, FeatureCommander, and FeatureIDE.

Besides feedback and support from colleagues, we acknowledge the financial support of the German Research Foundation and the National Science Foundation for a number of research projects related to the topics covered in this book (the NSF Science of Design projects: CCF-0438786 and CCF-0724979, the ERC

grant #203099, and the DFG projects: FAME-DBMS—Sa 465/32, Feature Foundation—AP 206/2, SafeSPL—AP 206/4, and Pythia—AP 206/5).

Finally, but not the least, we are grateful to our families and friends whose support was a necessary basis for the success of this endeavor.

Passau, February 2013
Austin
Pittsburgh
Magdeburg

Sven Apel
Don Batory
Christian Kästner
Gunter Saake

Contents

Part I Software Product Lines

1	Software Product Lines	3
1.1	From Individualism to Standardization and Back Again	3
1.2	Specialized and Standardized Software	6
1.3	Software Product Lines	7
1.4	Promises of Software Product Lines	9
1.5	Success Stories	10
1.6	A Feature-Oriented Approach	11
1.7	Running Examples	11
1.8	Intended Audience of the Book	13
1.9	How to Read this Book	14
1.10	Further Reading	15
2	A Development Process for Feature-Oriented Product Lines	17
2.1	Features and Products	17
2.2	A Process for Product-Line Development	19
2.2.1	Domain Analysis	22
2.2.2	Requirements Analysis	24
2.2.3	Domain Implementation	25
2.2.4	Product Derivation	26
2.3	Feature Modeling	26
2.3.1	Feature Models	27
2.3.2	Feature Diagrams	28
2.3.3	Formalization in Propositional Logic	31
2.3.4	The Feature Model for the Graph Library	32
2.3.5	Variations and Extensions of Feature Models	34
2.3.6	Feature Modeling in Practice	36
2.3.7	Tooling	39

- 2.4 Adoption Paths of the Product-Line Approach 39
 - 2.4.1 Proactive Approach 40
 - 2.4.2 Extractive Approach 40
 - 2.4.3 Reactive Approach 41
- 2.5 Further Reading 42
- Exercises 42

Part II Variability Implementation

- 3 Basic Concepts, Classification and Quality Criteria 47**
 - 3.1 Dimensions of Variability Implementation 48
 - 3.1.1 Binding Time 48
 - 3.1.2 Technology: Language-Based Versus Tool-Based . . . 49
 - 3.1.3 Representation: Annotation Versus Composition . . . 50
 - 3.2 Quality Criteria 52
 - 3.2.1 Preplanning Effort 52
 - 3.2.2 Feature Traceability 54
 - 3.2.3 Separation of Concerns 55
 - 3.2.4 Information Hiding 56
 - 3.2.5 Granularity 59
 - 3.2.6 Uniformity 59
 - 3.3 Structure of Subsequent Chapters 60
 - 3.4 Further Reading 61
 - Exercises 62
- 4 Classic, Language-Based Variability Mechanisms 65**
 - 4.1 Parameters 66
 - 4.1.1 Discussion 66
 - 4.2 Design Patterns 69
 - 4.2.1 Observer Pattern 70
 - 4.2.2 Template-Method Pattern 71
 - 4.2.3 Strategy Pattern 73
 - 4.2.4 Decorator Pattern 75
 - 4.2.5 Discussion 77
 - 4.3 Frameworks 79
 - 4.3.1 White-Box Frameworks 80
 - 4.3.2 Black-Box Frameworks 81
 - 4.3.3 An Implementation Example for Frameworks 82
 - 4.3.4 Loading Plug-Ins 85
 - 4.3.5 Discussion 86
 - 4.4 Components and Services 89
 - 4.4.1 Sizing Components 91
 - 4.4.2 Composing Components 92

- 4.4.3 Components Versus Plug-Ins 93
- 4.5 Further Reading 95
- Exercises 96

- 5 Classic, Tool-Driven Variability Mechanisms 99**
- 5.1 Version-Control Systems 99
 - 5.1.1 Terminology 100
 - 5.1.2 Building Product Lines with Version-Control Systems 101
 - 5.1.3 Discussion 103
- 5.2 Build Systems 105
 - 5.2.1 Variability in Build Scripts 105
 - 5.2.2 Custom Build Scripts 106
 - 5.2.3 Case Study: Build-System Variability in Linux 107
 - 5.2.4 Discussion 108
- 5.3 Preprocessors 110
 - 5.3.1 The C Preprocessor `cpp` 110
 - 5.3.2 Implementing Variability with Preprocessors 111
 - 5.3.3 Further Preprocessors 113
 - 5.3.4 Disciplined Annotations 116
 - 5.3.5 Preprocessors in Practice 118
 - 5.3.6 Discussion 120
- 5.4 Further Reading 124
- Exercises 125

- 6 Advanced, Language-Based Variability Mechanisms 129**
- 6.1 Feature-Oriented Programming 130
 - 6.1.1 Collaboration-Based Design 130
 - 6.1.2 Feature Modules 132
 - 6.1.3 The Jak Language 133
 - 6.1.4 Models of Feature-Oriented Programming 135
 - 6.1.5 Discussion 138
- 6.2 Aspect-Oriented Programming 141
 - 6.2.1 Aspects: Separating Crosscutting Concerns 142
 - 6.2.2 The AspectJ Language 145
 - 6.2.3 Aspects for Product Lines 147
 - 6.2.4 Discussion 149
- 6.3 Aspects and Feature Modules in Concert 152
 - 6.3.1 Homogeneous and Heterogeneous Crosscutting Concerns 153
 - 6.3.2 Static and Dynamic Crosscutting Concerns 157
 - 6.3.3 Summary of Comparison 161
 - 6.3.4 Combining Aspects and Feature Modules 161

- 6.3.5 A Study on Advanced Crosscutting Mechanisms 163
- 6.3.6 Discussion 164
- 6.4 Tooling 166
- 6.5 Practical Relevance. 167
- 6.6 Further Approaches. 168
 - 6.6.1 Delta-Oriented Programming 168
 - 6.6.2 Refactoring Feature Modules 169
 - 6.6.3 Context-Oriented Programming 170
- 6.7 Further Reading 171
- Exercises 172

- 7 Advanced, Tool-Driven Variability Mechanisms 175**
 - 7.1 Exploiting Feature Tracing 175
 - 7.1.1 Consistency Checking 177
 - 7.1.2 Visualizing Tracing Information. 178
 - 7.2 Views on Code. 180
 - 7.3 Integrated Product Derivation. 182
 - 7.4 Discussion: Virtual Separation of Concerns 184
 - 7.5 Tooling 186
 - 7.6 Further Reading 186
 - Exercises 187

Part III Advanced Topics

- 8 Refactoring of Software Product Lines 193**
 - 8.1 Refactoring in General 194
 - 8.2 Refactoring in Software Product Lines 197
 - 8.2.1 Variability Smells in Software Product Lines. 197
 - 8.2.2 Defining Product-Line Refactorings 200
 - 8.2.3 Examples of Product-Line Refactorings. 201
 - 8.3 Refactoring as Path Toward a Product Line. 203
 - 8.3.1 Example: Extraction of Feature Colored
of the Graph Library. 203
 - 8.3.2 Case Study: Refactoring of Berkeley DB
with AspectJ 207
 - 8.4 Further Reading 210
 - Exercises 212

- 9 Feature Interactions 213**
 - 9.1 The Feature-Interaction Problem. 214
 - 9.1.1 Higher Order Interactions 216
 - 9.2 Detecting Feature Interactions 217
 - 9.3 The Optional-Feature Problem 219
 - 9.4 Implementing Feature Interactions 222

- 9.4.1 Implementation Strategies: Overview and Goals. 223
- 9.4.2 Change Feature Model 224
- 9.4.3 Multiple Implementations 225
- 9.4.4 Moving Code. 226
- 9.4.5 Conditional Compilation 227
- 9.4.6 Optional Weaving. 228
- 9.4.7 Distinct Module for Coordination Code. 230
- 9.4.8 Comparison of Solutions 232
- 9.5 Experience. 233
 - 9.5.1 Decomposition of Berkeley DB 234
 - 9.5.2 Design and Implementation of FAME-DBMS 236
- 9.6 Further Reading 239
- Exercises 240

- 10 Analysis of Software Product Lines 243**
 - 10.1 Analysis of Feature Models 244
 - 10.1.1 Valid Feature Selection 245
 - 10.1.2 Consistent Feature Models. 247
 - 10.1.3 Testing Facts about Feature Models 248
 - 10.1.4 Dead Features and Mandatory Features. 249
 - 10.1.5 Constraint Propagation 250
 - 10.1.6 Number of Valid Feature Selections 251
 - 10.1.7 Comparing Feature Models 252
 - 10.1.8 Other Feature-Model Analyses 254
 - 10.2 Analysis of Feature-to-Code Mappings 254
 - 10.2.1 Dead Code. 255
 - 10.2.2 Abstract Features 257
 - 10.2.3 Determining Presence Conditions 257
 - 10.3 Analysis of Domain Implementations 260
 - 10.3.1 Design Space 262
 - 10.3.2 Sampling Strategies 263
 - 10.3.3 Family-Based Type Checking
of Preprocessor-Based Implementations. 264
 - 10.3.4 Family-Based Type Checking
for Feature-Oriented Programming 269
 - 10.3.5 Family-Based Analysis with Variability Encoding 271
 - 10.3.6 Feature-Based Analysis Strategies 272
 - 10.3.7 Beyond Type Checking. 273
 - 10.4 Case Studies and Experience 275
 - 10.5 Tooling 276
 - 10.6 Further Reading 277
 - Exercises 278

Appendix A: Tool Support 283

References 293

Index 309