

# Appendix A

## Tool Support

Throughout the book, we have seen several examples of tools for product-line development, from simple preprocessors to sophisticated composition tools and compilers. In this chapter, we introduce a number of tools for practical product-line development or useful for teaching. We do not intend to provide a comprehensive overview or discuss the pros and cons of specific commercial or academic tools—the field is too broad and continuously changing. Instead, we give brief recommendations of available tools, where we focus on freely available and stable tools that left the status of early academic prototypes.

### A.1 Overview

As introduced in [Chap. 2](#), product-line development consists of multiple phases, each of which shall be supported by proper tools.

- *Domain analysis.* During domain analysis, tools shall support the creation of and reasoning about feature models. They shall offer facilities to manage scoping decisions and document other concerns.
- *Domain implementation.* Depending on the modeling or implementation mechanism, editor support as known from modern IDEs like Eclipse shall support domain implementation. Furthermore, tools shall support different kinds of mappings between features and development artifacts.
- *Requirements analysis.* During requirements analysis, tools shall guide developers when selecting desired features, and verify the correctness or completeness of a selection with regard to the feature model.
- *Product derivation.* Again, depending on the implementation mechanisms, compilers, preprocessors, or composition engines shall automate the derivation process.

For each of these phases, proper tool support is desirable. However, there must also be tools that integrate multiple or all phases of product-line development. Integration allows tool support to cross phases, for example, to propagate the renaming of a feature from the feature model to the implementation mapping and

to all existing feature selections. Given the variety of different mechanisms in each phase, the mixture of mechanisms in practice, and the demand to process different artifacts uniformly with regard to variability (see [Sect. 3.2.6](#), p. 62), most product-line tools are extensible and try to connect the phases.

## A.2 Commercial Tools

At the time of writing there are two major players on the market for product-line tools *pure-systems GmbH* with *pure::variants* and *BigLever Software, Inc* with *Gears* (and several more regarding processes and consulting, which are not in our focus here). Both tools are used in industrial practice by many companies. Technically, both are extensible frameworks that cover all phases of the process of product-line development.

### A.2.1 *pure::variants*

*pure::variants* is a commercial tool developed and marketed by the German company *pure-systems GmbH* (<http://www.pure-systems.com/>). It integrates all phases of product-line development and can be used with different implementation mechanisms (they key concepts are independent of any specific language or tool). *pure::variants* integrates into Eclipse and can be extended with additional plug-ins.

For domain analysis, *pure::variants* uses feature models very close to the notation introduced in [Sect. 2.3](#). For large-scale models (with hundreds or thousands of features), *pure::variants* provides scalable tree-based editors and simple facilities to decompose feature models. Features can be enhanced with all kinds of annotations and parameters (only briefly discussed in [Sect. 2.3](#)) and logical rules can be used to express even complex non-Boolean constraints. There are connectors to various requirements-engineering tools. An editor for the requirements-engineering phase to select features is tightly integrated and supports various kinds of feature-model analyses (see [Sect. 10.1](#), p. 260).

For the mapping between problem and solution space, *pure::variants* provides a generic component model, the *component family model*. This model relates individual components to features. The term component is used in a broad sense and refers to a set of configurable functionalities, ranging from classes and aspects to compiler flags. In this context, *pure::variants* works like a sophisticated build system, in which developers can specify how and when to process artifacts and with which compilers, preprocessors, or composition tools. In addition to preconfigured scenarios for C/C++ and Java programs, *pure::variants* also ships with a customizable preprocessor for conditional compilation in arbitrary artifacts and can be extended with connectors, for example, for various modeling and testing tools.

Earlier in this book, in Fig. 2.10 (p. 38), we have shown a screenshot of the `pure::variant`'s workbench for our graph example, including editors for feature models and family models. A community edition of `pure::variants`, limited to comparably small models, is freely available for experimentation.

### A.2.2 Gears

Gears is a product-line tool of *BigLever Software, Inc.* (<http://www.biglever.com/>). Also, Gears aims at automating product derivation starting from a feature selection. Gears supports both feature models and simple lists of configuration parameters. Products are configured by selecting features or values for configuration parameters.

Given a configuration, Gears acts as a sophisticated build system that can run compilers, preprocessors, and composition tools, as specified by the developer. Next to calling external tools, standard examples are to use conditional compilation in various artifacts (a generic preprocessor is provided) or to inline the content of feature-specific files at marked locations in other files (named variation points). This way, building products from reusable artifacts can be automated entirely.

In addition, Gears is highly extensible. Connectors for integrated development environments (such as Eclipse), requirements engineering tools, modeling tools, word processors, and others exist. Gears provides an API so that tool builders can connect to a single central variability-management mechanism.

## A.3 FeatureIDE: An Open-Source Tool for Product-Line Implementation

While the commercial tools provide flexible production-quality solutions for industrial product-line development, researchers and educators might look for open-source solutions, easy for experimentation, extension, and class-room usage. FeatureIDE is an open-source development environment for product lines, targeted primarily at researchers, teachers, and students (<http://fosd.net/fide>). FeatureIDE integrates closely with several research tools, such as the AHEAD tool suite, FeatureC++, FeatureHouse, AspectJ, DeltaJ, and Java preprocessors. It is extensible using Eclipse's plug-in mechanism.

For domain analysis, FeatureIDE provides a graphical feature-model editor, as shown in Fig. A.1, which incorporates several analysis techniques (see Sect. 10.1, p. 248). The ability to produce high-quality graphics of feature models (for example, for teaching and research publications) was given precedence over scalability of the graphic editor. Also, the support for extra annotations and non-Boolean parameters is restricted compared to the commercial tools. Feature modeling is tightly integrated with the feature selection of the requirements-

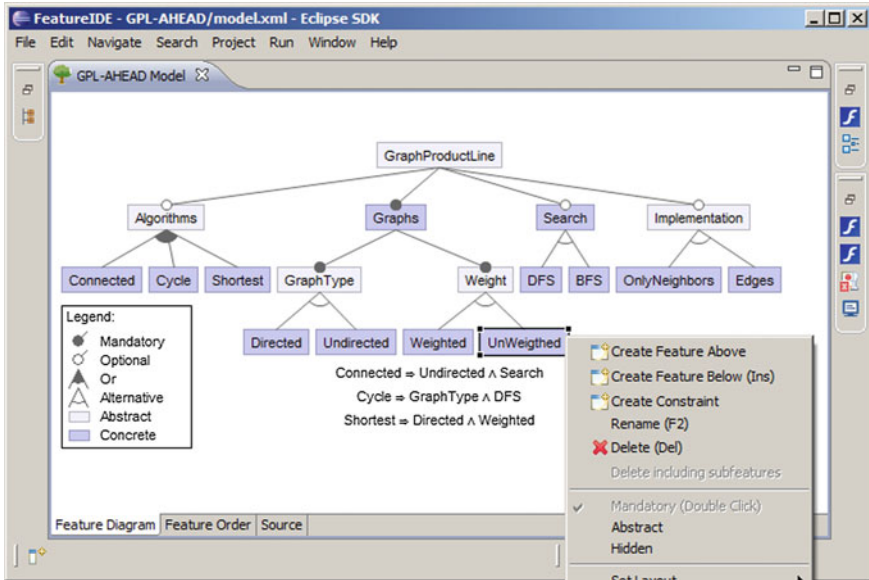


Fig. A.1 The feature-model editor of FeatureIDE

analysis phase (see Fig. 10.3 on p. 276; supporting reasoning about partial configurations, propagating feature renaming, updating feature selections when feature model constraints change, and so forth). The research nature is also visible by the fact that FeatureIDE provides several import and export mechanisms for feature models specified by several research tools.

For the solution space, FeatureIDE supports several specific tools, currently the AHEAD tool suite (Jak), FeatureC++, FeatureHouse, AspectJ, DeltaJ, and Java preprocessors. FeatureIDE is extensible by writing plug-ins, but, in contrast to the commercial tools, it does not provide general-purpose build-system mechanisms that would allow it to call arbitrary other tools. The tighter integration of specific implementation strategies (with plug-ins) provides better editor support and simplifies the learning curve (as most complexity from the build process is hidden).

For AHEAD, FeatureIDE provides an editor for Jak files as illustrated in Fig. A.3 (see also Sect. 6.1.3, p. 139). A tight integration allows several editor services for product lines implemented in Jak: syntax highlighting, automatic generation and compilation in the background, error reporting (traced back from Java errors on the composed files), and sophisticated visualizations such as the collaboration diagram shown in Fig. A.2 (see also Sect. 6.1.1, p. 136). We again see integration with other phases: Features are mapped to feature modules by name, and feature renaming propagates to the solution space, products are automatically recompiled when changing the feature selection, and so forth. The other mentioned languages are similarly deeply integrated; when languages come with

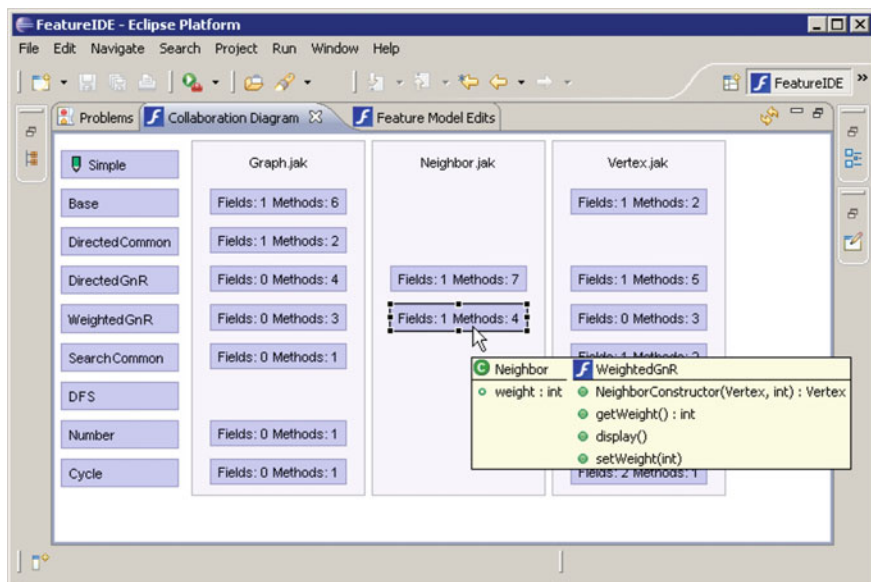


Fig. A.2 FeatureIDE: the collaboration diagram view

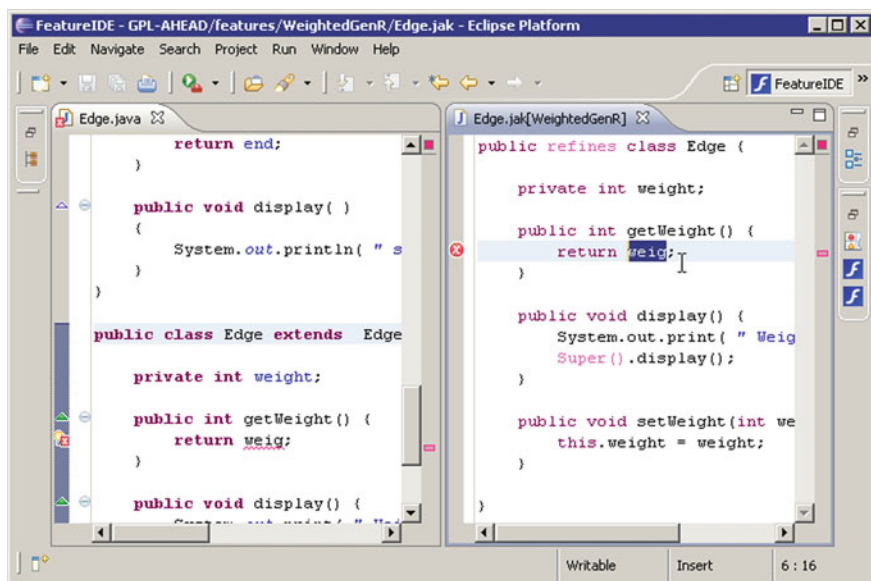


Fig. A.3 FeatureIDE: A syntax error in a class refinement in Jak

their own editor, such as C++ and AspectJ, these are reused.

FeatureIDE is not necessarily aimed at supporting industrial-scale product-line development. However, it provides a good base for teaching product-line engineering, with feature models (Chap. 2), preprocessors (Chap. 3), and advanced language-based mechanisms (Chap. 6). With its open-source nature based on Eclipse plug-ins it is highly extensible.

An overview on design considerations for FeatureIDE and learned lessons can be found in Thüm et al. (2013).

## A.4 Further Tools

In the corresponding parts of the book, we have mentioned and discussed several other tools for product-line development (mostly focused on one specific task). Here, we provide only a brief list of the stable research tools as well as publicly available open-source tools that can be used right away for implementing product lines. More experimental tools can be found in the corresponding chapters of the book. For each tool, we summarize for which task they were designed, where we discussed them in this book, and where to get them.

**AHEAD Tool Suite.** Collection of composition and supporting tools for feature-oriented programming in Jak and other languages. Includes a tool *SafeGen* for variability-aware type checking. Command-line research tools, partially integrated in *FeatureIDE*. See also Sect. 6.1.

<http://www.cs.utexas.edu/~schwartz/ATS.html>

**Antenna.** Lexical preprocessor designed for Java ME applications, with integrations in several IDEs, such as NetBeans, Eclipse, and FeatureIDE. See also Sect. 5.3.3.

<http://antenna.sourceforge.net>

**AspectJ.** Aspect-oriented programming language based on Java with corresponding compiler. A commercial-quality Eclipse-based IDE *AJDT* is available. See also Sect. 6.2.

<http://www.eclipse.org/aspectj/>

**CDL.** Textual variability modeling language and corresponding tool infrastructure. Originally designed for the *eCos* operating system. See also the description of Berger et al. (2010).

<http://ecos.sourceware.org/docs-2.0/cdl-guide/cdl-guide.html>

<https://code.google.com/p/variability/wiki/CDLTools>

**CIDE.** Eclipse-based research prototype for virtual separation of concerns. Has been partially reimplemented in several other projects. See also [Chap. 7](#).

<http://fosd.net/CIDE>

<http://fosd.net/fc>

<http://www.dcc.ufmg.br/~mtov/cideplus/>

**Clafer.** A lightweight yet expressive language for structural modeling: feature modeling and configuration, class and object modeling, and metamodeling. Clafer Tools is Integrated set of tools based on Clafer, supporting model analysis, configuration, and multi-objective optimization, exploration, and visualization.

<http://clafer.org>

**ConcernMapper.** Feature mapping and tracing tool, targeted at arbitrary concerns in general software development. See also [Sect. 7.1](#).

<http://www.cs.mcgill.ca/~martin/cm/>

**ContextJ.** Context-oriented extension of Java. Similar extensions with varying maturity exist for many other languages (*ContextL*, *ContextJS*, *ContextR*, and so forth). See also [Sect. 6.6.3](#).

<http://www.swa.hpi.uni-potsdam.de/cop/>

**cpp.** Lexical preprocessor part of the C language standard (ISO). Shipped with every C and C++ compiler. See also [Sect. 5.3.1](#).

<http://gcc.gnu.org/>

<http://clang.llvm.org/>, and others

**CVL.** Common variability language. Upcoming industry standard for variability modeling. Basic tool support is available in the form of Eclipse plug-ins.

[http://www.omgwiki.org/variability/doku.php/doku.php?id=cvl\\_tool\\_from\\_sintef](http://www.omgwiki.org/variability/doku.php/doku.php?id=cvl_tool_from_sintef)

**DeltaJ.** Delta-oriented programming language based on Java and corresponding composition engines. Command-line research tool and Eclipse-based IDE. See also [Sect. 6.6.1](#).

<http://deltaj.sourceforge.net/>

**FAMA.** Comprehensive research framework for feature-model analysis. See also [Chap. 10](#).

<http://www.isa.us.es/fama/>

**FeatureHouse.** Composition engine for feature-oriented programming in various languages. Declarative extension mechanisms to plug-in new languages. Command-line research tool, integrated also in *FeatureIDE*. See also [Sect. 6.1](#).

<http://fosd.net/fh>

**FeatureMapper.** Eclipse editor for product lines of ecore models; annotation-based; supports views and some analysis. See also [Sects. 5.3.3](#) and [7.5](#).

<http://featuremapper.org/>

**git.** State-of-the-art distributed version-control system, with advanced branching and merging capabilities. See also [Sect. 5.1](#).

<http://git-scm.com/>

**Kbuild.** A collection of build files and conventions that form the build system of the Linux kernel. Part of the Linux kernel, but also used by several other projects. Integrates with *Kconfig*. See also [Sect. 5.2.3](#).

<http://www.kernel.org>

<http://www.kernel.org/doc/Documentation/kbuild/makefiles.txt>

**Kconfig.** Textual variability-modeling language and corresponding configurator. Developed for and distributed with the Linux kernel, but also used by several other projects. See also [Sect. 2.3.6](#) and the description of Berger et al. (2010).

<http://www.kernel.org>

<http://www.kernel.org/doc/Documentation/kbuild/kconfig-language.txt>

<http://gsd.uwaterloo.ca/feature-models-in-the-wild>

**Koala.** Component infrastructure and composition mechanisms, originally developed by Philips Research for consumer electronics. An open-source implementation is available as well. See also [Sect. 4.4](#).

<http://www.program-transformation.org/Tools/KoalaCompiler>

**Munge.** Simple, open-source, lexical preprocessor for Java that does not break existing Java tooling (conditional-inclusion directives are inside comments). See also [Sect. 5.3.3](#).

<http://sonatype.github.com/munge-maven-plugin/>

**OSGi framework.** Module system for Java applications that can be used to develop framework and component-based solutions. Underlying technology of the Eclipse project. See also [Sects. 4.3](#) and [4.4](#).

<http://www.osgi.org>;

<http://www.eclipse.org>

**SNIP.** Variability-aware model-checking tool for product-line models written in the specification language Promela. See also [Chap. 10](#).

<http://www.info.fundp.ac.be/fts/>

**SPLIT.** Research tools for editing, collecting, and analyzing feature models. Entirely available as web-based online tools. See also [Chap. 10](#).

<http://www.splot-research.org/>

**SPLverifier.** Tool suite for variability-aware model checking of *FeatureHouse*-based product lines written in C and Java. See also [Chap. 10](#).

<http://fosd.net/FAV>



**TypeChef.** Research framework of variability-aware analysis of preprocessor-based product lines written in C, such as the Linux kernel; includes variability-aware type checking and data-flow analysis. See also [Chap. 10](#).

<https://github.com/ckaestne/TypeChef>

**Undertaker.** Tool that analyzes the mapping of preprocessor directives to configuration models, including dead-code detection and other analyses. See also [Chap. 10](#).

<http://vamos.informatik.uni-erlangen.de/trac/undertaker/>

**VMC.** Variability-aware model-checking tool for product-line models represented as transition systems. See also [Chap. 10](#).

<http://fmtlab.isti.cnr.it/vmc/>

# References

- Adams B, De Meuter W, Tromp H, Hassan AE (2009) Can we refactor conditional compilation into aspects? In: Proc. Int'l Conf. Aspect-Oriented Software Development (AOSD). ACM Press, pp 243–254
- Adams B, De Schutter K, Tromp H, De Meuter W (2007) Design recovery and maintenance of build systems. In: Proc. Int'l Conf. Software Maintenance (ICSM). IEEE Computer Society, pp 114–123
- Adams B, De Schutter K, Tromp H, De Meuter W (2008a) The evolution of the Linux build system. Electronic Communications of the EASST, 8
- Adams B, Van Rompaey B, Gibbs C, Coady Y (2008b) Aspect mining in the presence of the C preprocessor. In: Proc. AOSD Workshop on Linking Aspect Technology and Evolution (LATE). ACM Press, pp 1–6
- Adler C (2010) Optional composition: A solution to the optional feature problem? Master's thesis, School of Computer Science, University of Magdeburg
- Aldrich J (2005) Open modules: Modular reasoning about advice. In: Proc. Europ. Conf. Object-Oriented Programming (ECOOP). Lecture Notes in Computer Science, vol 3586. Springer, pp 144–168
- Allan C, Avgustinov P, Christensen A, Hendren L, Kuzins S, Lhotak O, de Moor O, Sereni D, Sittampalam G, Tibble J (2005) Adding trace matching with free variables to AspectJ. In: Proc. Int'l Conf. Object-Oriented Programming, systems, languages, and applications (OOPSLA), ACM Press, pp 345–364
- Alves V, Gheyi R, Massoni T, Kulesza U, Borba P, Lucena C (2006) Refactoring product lines. In: Proc. Int'l Conf. Generative Programming and Component Engineering (GPCE). ACM Press, pp 201–210
- Anastasopoulos M, Gacek C (2001) Implementing product line variabilities. In: Proc. Symposium on Software Reusability (SSR), ACM Press, pp 109–117
- Ancona D, Damiani F, Drossopoulou S, Zucca E (2005) Polymorphic bytecode: Compositional compilation for Java-like languages. In: Proc. Int'l Symp. Principles of Programming Languages (POPL). ACM Press, pp 26–37
- Ancona D, Zucca E (2001) True modules for Java-like languages. In: Proc. Europ. Conf. Object-Oriented Programming (ECOOP). Lecture Notes in Computer Science, vol 2072. Springer, pp 354–380
- Apel S (2007) The role of features and aspects in software development. Ph.D. thesis, School of Computer Science, University of Magdeburg
- Apel S (2010) How AspectJ is used: An analysis of eleven AspectJ programs. J Object Technol (JOT) 9(1):117–142
- Apel S, Hutchins D (2010) A calculus for uniform feature composition. ACM Trans Program Lang Syst (TOPLAS) 32(5):1–33
- Apel S, Kästner C (2009) An overview of feature-oriented software development. J Object Technol (JOT) 8(5):49–84

- Apel S, Kästner C, Batory D (2008) Program refactoring using functional aspects. In: Proc. Int'l Conf. Generative Programming and Component Engineering (GPCE). ACM Press, pp 161–170
- Apel S, Kästner C, Leich T, Saake G (2007) Aspect refinement—unifying AOP and stepwise refinement. *J Object Technol (JOT)—Special Issue: TOOLS, EUROPE 2007* 6(9):13–33
- Apel S, Kästner C, Lengauer C (2009) FeatureHouse: Language-independent, automated software composition. In: Proc. Int'l Conf. Software Engineering (ICSE), IEEE Computer Society, pp 221–231
- Apel S, Kästner C, Lengauer C (2013a) Language-independent and automated software composition: The FeatureHouse experience. *IEEE Trans Software Eng (TSE)* 39(1):63–79
- Apel S, Kolesnikov S, Liebig J, Kästner C, Kuhlemann M, Leich T (2012a) Access control in feature-oriented programming. *Sci Comput Program (SCP)* 77(3):174–187
- Apel S, Leich T, Rosenmüller M, Saake G (2005) FeatureC++: On the symbiosis of feature-oriented and aspect-oriented programming. In: Proc. Int'l Conf. Generative Programming and Component Engineering (GPCE). Lecture Notes in Computer Science, vol 3676. Springer, pp 125–140
- Apel S, Leich T, Saake G (2008b) Aspectual feature modules. *IEEE Trans Software Eng (TSE)* 34(2):162–180
- Apel S, Lengauer C, Möller B, Kästner C (2010) An algebraic foundation for automatic feature-based program synthesis. *Sci Comput Program* 75(11):1022–1047
- Apel S, Leßenich O, Lengauer C (2012). Structured merge with auto-tuning: Balancing precision and performance. In: Proc. Int'l Conf. Automated Software Engineering (ASE). ACM Press, pp 120–129
- Apel S, Liebig J, Brandl B, Lengauer C, Kästner C (2011) Semistructured merge: Rethinking merge in revision control systems. In: Proc. Europ. Software Engineering Conf. and Symp. Foundations of Software Engineering (ESEC/FSE). ACM Press, pp 190–200
- Apel S, von Rhein A, Wendler P, Größlinger A, Beyer D (2013b) Strategies for product-line verification: Case studies and experiments. In: Proceedings of the IEEE/ACM International Conference on Software Engineering (ICSE). IEEE Computer Society, pp 482–491
- Appeltauer M, Hirschfeld R, Haupt M, Masuhara H (2011) ContextJ: Context-oriented programming with Java. *Comput Softw* 28(1):272–292
- Aracic I, Gasiunas V, Mezini 1082 M, Ostermann K (2006) An overview of CaesarJ. *Trans aspect-orient Softw Dev (TAOSD)* 1(1):135–173
- Arendt T, Biermann E, Jurack S, Krause C, Taentzer G (2010) Henshin: Advanced concepts and tools for in-place EMF model transformations. In: Proc. Int'l Conf. Model Driven Engineering Languages and Systems (MODELS), lecture notes in computer science, vol. 6394. Springer, pp 121–135
- Arnoldus J, Bijpost J, van den Brand M (2007) Repleo: A syntax-safe template engine. In: Proc. Int'l Conf. Generative Programming and Component Engineering (GPCE). ACM Press, pp 25–32
- Atkins DL (1998) Version sensitive editing: Change history as a programming tool. In: Proc. ECOOP Symposium on System Configuration Management (SCM). Lecture Notes in Computer Science, vol 1439. Springer, pp 146–157
- Atkins DL, Ball T, Graves TL, Mockus A (2002) Using version control data to evaluate the impact of software tools: A case study of the version editor. *IEEE Trans Software Eng* 28(7):625–637
- Bass L, Clements P, Kazman R (1998) *Software architecture in practice*. Wesley
- Batory D (2005) Feature models, grammars, and propositional formulas. In: Proc. Int'l Software Product Line Conference (SPLC), Lecture Notes in Computer Science, vol 3714. Springer, pp 7–20
- Batory D, Höfner P, Kim J (2011) Feature interactions, products, and composition. In: Proc. Int'l Conf. Generative Programming and Component Engineering (GPCE). ACM Press, pp 13–22

- Batory D, O'Malley S (1992) The design and implementation of hierarchical software systems with reusable components. *ACM Trans Software Eng Methodol (TOSEM)* 1(4):355–398
- Batory D, Sarvela JN, Rauschmayer A (2004) Scaling step-wise refinement. *IEEE Trans Software Eng (TSE)* 30(6):355–371
- Baxter I, Mehlich M (2001) Preprocessor conditional removal by simple partial evaluation. In: *Proc. Working Conf. Reverse Engineering (WCRE)*. IEEE Computer Society, pp 281–290
- Baxter I, Yahin A, Moura L, Sant'Anna M, Bier L (1998) Clone detection using abstract syntaxtrees. In: *Proc. Int'l Conf. Software Maintenance (ICSM)*, IEEE Computer Society, pp 368–377
- Benavides D, Seguraa S, Ruiz-Cortés A (2010) Automated analysis of feature models 20 years later: A literature review. *Inf Syst* 35(6):615–636
- Benavides D, Trinidad P, Ruiz-Cortes A (2005) Automated reasoning on feature models. In: *Proc. Conf. Advanced Information Systems Engineering (CAiSE)*. Lecture notes in computer science, vol 3520. Springer, pp 491–503
- Bergel A, Ducasse S, Nierstrasz O (2005) Classbox/J: Controlling the scope of change in Java. In: *Proc. Int'l Conf. Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, ACM Press, pp 177–189
- Berger T, She S, Czarnecki K, Wąsowski A (2010a) Feature-to-code mapping in two large product lines. In: *Proc. Int'l Software Product Line Conference (SPLC)*. Lecture Notes in Computer Science, vol 6287. Springer, pp 498–499
- Berger T, She S, Lotufo R, Wąsowski A, Czarnecki K (2010b) Variability modeling in the real: A perspective from the operating systems domain. In: *Proc. Int'l Conf. Automated Software Engineering (ASE)*. ACM Press, pp 73–82
- Berre DL, Parrain A (2010) The Sat4j library, release 2.2. *J Satisf Boolean Model Comput (JSAT)* 7(2–3):59–64
- Beuche D, Papajewski H, Schröder-Preikschat W (2004) Variability management with featuremodels. *Sci Comput Program* 53(3):333–352
- Biggerstaff T (1994) The library scaling problem and the limits of concrete component reuse. In: *Proc. Int'l Conf. Software Reuse (ICSR)*, IEEE Computer Society, pp 102–109
- Biggerstaff T, Mitbander BG, Webster D (1993) The concept assignment problem in program understanding. In: *Proc. Int'l Conf. Software Engineering (ICSE)*. IEEE Computer Society, pp 482–498
- Binkley D et al (2006) Tool-supported refactoring of existing object-oriented code into aspects. *IEEE Trans Softw Eng (TSE)* 32(9):698–717
- Blom J, Jonsson B, Kempe L (1994) Using temporal logic for modular specification of telephone services. In: Bouma LG, Velthuisen H (eds) *Feature interactions in telecommunications systems*. IOS Press, pp 197–216
- Blume M, Appel AW (1999) Hierarchical modularity. *ACM Trans Program Lang Syst (TOPLAS)* 21(4):813–847
- Bodden E (2012) Inter-procedural data-flow analysis with IFDS/IDE and Soot. In: *Int'l Workshop on the State of the Art in Java Program Analysis (SOAP)*. ACM Press, pp 3–8
- Boehm BW (1988) A Spiral Model of Software Development and Enhancement. *Computer* 21(5):61–72. <http://dx.doi.org/10.1109/2.59>
- Borba P, Teixeira L, Gheyri R (2010) A theory of software product line refinement. In: *Proc. Int'l Colloquium Theoretical Aspects of Computing (ICTAC)*. Springer, pp 15–43
- Bosch J (2000) Design and use of software architectures: Adopting and evolving a product-line approach. *ACM Press/Addison-Wesley*
- Bosch J (2009) From software product lines to software ecosystems. In: *Proc. Int'l Software Product Line Conference (SPLC)*, ACM Press, pp 111–119
- Boucher Q, Classen A, Heymans P, Bourdoux A, Demonceau L (2010) Tag and prune: A pragmatic approach to software product line implementation. In: *Proc. Int'l Conf. Automated Software Engineering (ASE)*. ACM Press, pp 333–336

- Bowen T, Dworack F, Chow C, Griffith N, Lin GHY-J (1989) The feature interaction problem in telecommunications systems. In: Proc. Int'l Conf. Software Engineering for Telecommunications Switching Systems (SETSS), IEEE Computer Society, pp 59–62
- Boxleitner S, Apel S, Kästner C (2009) Language-independent quantification and weaving for feature composition. In: Proc. Int'l Symp. Software Composition (SC). Lecture Notes in Computer Science, vol 5634. Springer, pp 45–54
- Brabrand C, Ribeiro M, Tolêdo T, Borba P (2012) Intraprocedural dataflow analysis for software product lines. In: Proc. Int'l Conf. Aspect-Oriented Software Development (AOSD). ACM Press, pp 13–24
- Brabrand C, Schwartzbach MI (2002) Growing languages with metamorphic syntax macros. In: Proc. Int'l Symp. Partial Evaluation and Semantics-Based Program Manipulation (PEPM). ACM Press, pp 31–40
- Bracha G, Cook W (1990) Mixin-based inheritance. In: Proc. Int'l Conf. Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), ACM Press, pp 303–311
- Bruns G, Mataga P, Sutherland I (1998) Features as service transformers. In: Feature Interactions in Telecommunications Systems V. IOS Press, pp 85–97
- Calder M, Kolberg M, Magill EH, Reiff-Marganiec S (2003) Feature interaction: A critical review and considered forecast. *Comput Netw* 41(1):115–141
- Cardelli L (1997) Program fragments, linking, and modularization. In: Proc. Int'l Symp. Principles of Programming Languages (POPL). ACM Press, pp 266–277
- Chacon S (2009) Pro Git. Apress. <http://progit.org/>
- Chen K, Zhang W, Zhao H, Mei H (2005) An approach to constructing feature models based on requirements clustering. In: Proc. Int'l Conf. Requirements Engineering (RE). IEEE Computer Society, pp 31–40
- Cheng B, de Lemos R, Giese H, Inverardi P, Magee J et al (2009) Software engineering for self-adaptive systems: A research roadmap. In: Software Engineering for Self-Adaptive Systems, Lecture Notes in Computer Science, vol 5525. Springer, pp 1–26
- Chu-Carroll M, Wright J, Ying A (2003) Visual separation of concerns through multidimensional program storage. In: Proc. Int'l Conf. Aspect-Oriented Software Development (AOSD). ACM Press, pp 188–197
- Classen A, Heymans P, Schobbens P (2008) What's in a feature: A requirements engineering perspective. In: Proc. Int'l Conf. Fundamental Approaches to Software Engineering (FASE) Lecture notes in computer science, vol 4961. Springer, pp 16–30
- Classen A, Heymans P, Schobbens P-Y, Legay A, Raskin J-F (2010) Model checking lots of systems: Efficient verification of temporal properties in software product lines. In: Proc. Int'l Conf. Software Engineering (ICSE). ACM Press, pp 335–344
- Classen A, Cordy M, Heymans P, Legay A, Schobbens P-Y (2012) Model checking software product lines with SNIP. *Software Tools Technol Transfer (STTT)* 14(5):589–612
- Clements P, Krueger CW (2002) Point/counterpoint: Being proactive pays off/eliminating the adoption barrier. *IEEE Softw* 19(4):28–31
- Clements P, Northrop L (2001) Software product lines: Practices and patterns. Addison-Wesley
- Cohen MB, Dwyer MB, Shi J (2007) Interaction testing of highly-configurable systems in the presence of constraints. In: Proc. Int'l Symp. Software Testing and Analysis (ISSTA). ACM Press, pp 129–139
- Cole L, Borba P (2005) Deriving refactorings for AspectJ. In: Proc. Int'l Conf. Aspect-Oriented Software Development (AOSD). ACM Press, pp 123–134
- Colyer A, Clement A, Harley G, Webster M (2004a) Eclipse AspectJ: Aspect-oriented programming with AspectJ and the eclipse AspectJ development tools, 1st edn. Addison-Wesley Professional, Reading
- Colyer A, Greenfield J, Jacobson I, Kiczales G, Thomas D (2005) Aspects: Passing fad or new foundation? In: Companion Int'l Conf. Object-Oriented Programming, Systems, Languages and Applications (OOPSLA), ACM Press, pp 376–377

- Colyer A, Rashid A, Blair G (2004b) On the separation of concerns in program families. Technical Report COMP-001-2004, Computing Department, Lancaster University
- Colyer A, Clement A (2004) Large-scale AOSD for middleware. In: Proc. Int'l Conf. Aspect-Oriented Software Development (AOSD), ACM Press, pp 56–65
- Cornelissen B, Zaidman A, van Deursen A, Moonen L, Koschke R (2009) A systematic survey of program comprehension through dynamic analysis. *IEEE Trans Software Eng* 35(5):684–702
- Czarnecki K, Antkiewicz M (2005) Mapping features to models: A template approach based on superimposed variants. In: Proc. Int'l Conf. Generative Programming and Component Engineering (GPCE). Lecture notes in computer science, vol 3676. Springer, pp 422–437
- Czarnecki K, Grünbacher P, Rabiser R, Schmid K, Wąsowski A (2012) Cool features and tough decisions: A comparison of variability modeling approaches. In: Proc. Int'l Workshop on Variability Modelling of Software-intensive Systems (VaMoS). ACM Press, pp 173–182
- Czarnecki K, Helsen S, Eisenecker U (2005a) Formalizing cardinality-based feature models and their specialization. *Softw Process: Improv Pract* 10(1):7–29
- Czarnecki K, Helsen S, Eisenecker U (2005b) Staged configuration through specialization and multilevel configuration of feature models. *Softw Process: Improv Pract* 10(2):143–169
- Czarnecki K, Pietroszek K (2006) Verifying feature-based model templates against wellformedness OCL constraints. In: Proc. Int'l Conf. Generative Programming and Component Engineering (GPCE). ACM Press, pp 211–220
- Czarnecki K, Eisenecker U (2000) Generative programming: Methods, tools, and applications. ACM Press/Addison-Wesley
- da Mota Silveira Neto PA, do Carmo Machado I, McGregor JD, de Almeida ES, de Lemos Meira SR (2011) A systematic mapping study of software product lines testing. *Inf Softw Technol* 53(5):407–423
- Dantas D, Walker D (2006) Harmless advice. In: Proc. Int'l Symp. Principles of Programming Languages (POPL), ACM Press, pp 383–396
- Delaware B, Cook WR, Batory D (2009) Fitting the pieces together: A machine-checked model of safe composition. In: Proc. Europ. Software Engineering Conf. and Symp. Foundations of Software Engineering (ESEC/FSE). ACM Press, pp 243–252
- DeRemer F, Kron HH (1976) Programming-in-the-large versus programming-in-the-small. *IEEE Trans Softw Eng* 2:80–86
- Dessi M (2009) Spring 2.5 Aspect Oriented programming. Packt Publishing
- Dietrich C, Tartler R, Schröder-Preikschat W, Lohmann D (2012a) A robust approach for variability extraction from the Linux build system. In: Proc. Int'l Software Product Line Conference (SPLC). ACM Press, pp 21–30
- Dietrich C, Tartler R, Schröder-Preikschat W, Lohmann D (2012b) Understanding Linux feature distribution. In: Proc. AOSD Workshop on Modularity In Systems Software (MISS). ACM Press, pp 15–20
- Dijkstra EW (1976) A discipline of programming. Prentice-Hall
- Engström E, Runeson P (2011) Software product line testing—a systematic mapping study. *Inf Softw Technol (IST)* 53(1):2–13
- Erl T (2005) Service-oriented architecture: Concepts, technology, and design. Prentice Hall
- Ernst M, Badros G, Notkin D (2002) An empirical analysis of C preprocessor use. *IEEE Trans Softw Eng (TSE)* 28(12):1146–1170
- Estler H-C, Ruhroth T, Wehrheim H (2007) Model checking correctness of refactorings—some experiments. *Electron Notes Theor Comput Sci* 187:3–17
- Favre J-M (1995) The CPP paradox. In: Proc. European Workshop on Software Maintenance
- Favre J-M (1997) Understanding-in-the-large. In: Proc. Int'l Workshop on Program Comprehension. IEEE Computer Society, p 29
- Favre J-M (2003) CPP denotational semantics. In: Proc. Int'l Workshop Source Code Analysis and Manipulation (SCAM). IEEE Computer Society, pp 22–31

- Feigenspan J, Kästner C, Apel S, Liebig J, Schulze M, Dachsel R, Papendieck M, Leich T, Saake G (2012) Do background colors improve program comprehension in the #ifdef hell? *Empirical Software Eng*. Online first. doi:10.1007/s10664-012-9208-x
- Felty A, Namjoshi K (2003) Feature specification and automated conflict detection. *ACM Trans Software Eng Methodol (TOSEM)* 12(1):3–27
- Fernandez-Amoros D, Gil RH, Somolinos JC (2009) Inferring information from feature diagrams to product line economic models. In: *Proc. Int'l Software Product Line Conference (SPLC)*. ACM Press, pp 41–50
- Filman R, Friedman D (2005) Aspect-oriented programming is quantification and obliviousness. In: *Aspect-Oriented Software Development*, Addison-Wesley, pp 21–35
- Filman RE, Elrad T, Clarke S, Aksit M (eds) (2005a) *Aspect-oriented software development*. Addison-Wesley
- Flatt M, Krishnamurthi S, Felleisen M (1998) Classes and mixins. In: *Proc. Int'l Symp. Principles of Programming Languages (POPL)*, ACM Press, pp 171–183
- Ford H, Crowther S (1922) *My life and work (the autobiography of Henry ford)*. Doubleday
- Fowler M (1999) *Refactoring: Improving the design of existing code*. Addison-Wesley
- Gamma E, Beck K (2003) *Contributing to eclipse: Principles, patterns, and plugins*. Wesley
- Gamma E, Helm R, Johnson R, Vlissides J (1995) *Design patterns: Elements of reusable object oriented software*. Addison-Wesley, Reading
- Garlan D, Allen R, Ockerbloom J (1995) Architectural mismatch or why it's hard to build systems out of existing parts. In: *Proc. Int'l Conf. Software Engineering (ICSE)*, IEEE Computer Society, pp 179–185
- Garvin BJ, Cohen MB (2011) Feature interaction faults revisited: An exploratory study. In: *Proc. Int'l Symp. Software Reliability Engineering (ISSRE)*, IEEE Computer Society, pp 90–99
- Gazzillo P, Grimm R (2012) SuperC: Parsing all of C by taming the preprocessor. In: *Proc. Int'l Conf. Programming Language Design and Implementation (PLDI)*. ACM Press, pp 323–334
- Gosling J, Joy B, Steele G, Bracha G (2005) Java™ language specification. The Java™ series, 3rd edn. Addison-Wesley, Reading
- Gradecki JD, Lesiecki N (2003) *Mastering AspectJ: Aspect-oriented programming in Java*. John Wiley & Sons, Inc.
- Griffeth N, Velthuisen H (1994) The negotiating agents approach to runtime feature interaction resolution. In: Bouma LG, Velthuisen H (eds) *Feature interactions in telecommunications systems*. IOS Press, pp 217–235
- Griss M (2000) Implementing product-line features by composing aspects. In: *Proc. Int'l Software Product Line Conference (SPLC)*. Kluwer Academic Publishers, pp 271–288
- Griss ML, Favaro J, d' Alessandro M (1998) Integrating feature modeling with the RSEB. In: *Proc. Int'l Conf. Software Reuse (ICSR)*. IEEE Computer Society, p 76
- Haase S (2012) A program slicing approach to feature identification. Master's thesis, School of Computer Science, University of Magdeburg
- Hall RJ (2005) Fundamental nonmodularity in electronic mail. *Autom Software Eng* 12(1):41–79
- Hanenberg S, Oberschulte C, Unland R (2003) Refactoring of aspect-oriented software. In: *Proc. Int'l Conf. Object-Oriented and Internet-based Technologies, Concepts, and Applications for a Networked World (Net.ObjectDays)*, pp 19–35 (transIT GmbH)
- Harbulot B, Gurd J (2006) A join point for loops in AspectJ. In: *Proc. Int'l Conf. Aspect-Oriented Software Development (AOSD)*, ACM Press, pp 63–74
- Hay J, Atlee J (2000) Composing features and resolving interactions. In: *Proc. Int'l Symp. Foundations of Software Engineering (FSE)*, ACM Press, pp 110–119
- Heidenreich F, Kopceck J, Wende C (2008b) FeatureMapper: Mapping features to models. In: *Companion Int'l Conf. Software Engineering (ICSE)*. ACM Press, pp 943–944
- Heidenreich F, Şavga I, Wende C (2008a) On controlled visualisations in software product line engineering. In: *Proc. SPLCWorkshop on Visualization in Software Product Line Engineering (ViSPLE)* Lero, pp 303–313

- Heidenreich F, Sánchez P, Santos J a, Zschaler S, Alférez M, Araújo J. a, Fuentes L, Kulesza U, Moreira A, Rashid A (2010) Relating feature models to other models of a software product line: A comparative study of FeatureMapper and VML. In: Transactions on aspect-oriented software development VII. Springer, pp 69–114
- Herrmann S (2002) Object teams: Improving modularity for crosscutting collaborations. In: Proc. Int'l Conf. Object-Oriented and Internet-based Technologies, Concepts, and Applications for a Networked World (Net.ObjectDays). Lecture Notes in Computer Science, vol 2591. Springer, pp 248–264
- Heymans P (2012) Formal methods for the masses. In: Proc. Int'l Software Product Line Conference (SPLC), ACM Press, p 4
- Hirschfeld R, Costanza P, Nierstrasz O (2008) Context-oriented programming. *J Object Technol (JOT)* 7(3):125–151
- Hofer W, Elsner C, Blendinger F, Schröder-Preikschat W, Lohmann D (2011) Tool chain independent variant management with the Leviathan filesystem. In: Proc. GPCE Workshop on Feature-Oriented Software Development (FOSD) ACM Press, pp 18–24
- Hu Y, Merlo E, Dagenais M, Laguë B (2000) C/C++ conditional compilation analysis using symbolic execution. In: Proc. Int'l Conf. Software Maintenance (ICSM). IEEE Computer Society, pp 196–206
- Huang SS, Smaragdakis Y (2011) Morphing: Structurally shaping a class by reflecting on others. *ACM Trans Prog Lang Syst (TOPLAS)* 33(2), 6:1–6:44
- Huang SS, Zook D, Smaragdakis Y (2005) Statically safe program generation with SafeGen. In: Proc. Int'l Conf. Generative Programming and Component Engineering (GPCE). Lecture Notes in Computer Science, vol 3676. Springer, pp 309–326
- Hubaux A, Xiong Y, Czarniecki K (2012) A user survey of configuration challenges in Linux and eCos. In: Proc. Int'l Workshop on Variability Modelling of Software-intensive Systems (VaMoS). ACM Press, pp 149–155
- Hunleth F, Cytron RK (2002) Footprint and feature management using aspect-oriented programming techniques. In: Proc. Conf. Languages, Compilers and Tools For Embedded systems (LCTES), ACM Press, pp 38–45
- Hunleth F, Cytron RK (2002) Footprint and feature management using aspect-oriented programming techniques. In: Proc. Conf. Languages, Compilers and Tools For Embedded Systems (LCTES). ACM Press, pp 38–45
- Jackson M, Zave P (1998) Distributed feature composition: A virtual architecture for telecommunications services. *IEEE Trans Software Eng (TSE)* 24(10):831–847
- Janota M (2010) SAT solving in interactive configuration. Ph.D. thesis, Department of Computer Science, University College Dublin
- Janzen D, De Volder K. (2004). Programming with crosscutting effective views. In: Proc. Europ. Conf. Object-Oriented Programming (ECOOP). Lecture Notes in Computer Science, vol 3086. Springer, pp 195–218
- Jarzabek S, Bassett P, Zhang H, Zhang W (2003) XVCL: XML-based variant configuration language. In: Proc. Int'l Conf. Software Engineering (ICSE). IEEE Computer Society, pp 810–811
- Johnson RE, Foote B (1988) Designing reusable classes. *J Object-Oriented Program* 1(2):22–35
- Jones ND, Gomard CK, Sestoft P (1993) Partial evaluation and automatic program generation. Prentice-Hall
- Kang K, Cohen SG, Hess JA, Novak WE, Peterson AS (1990) Feature-oriented domain analysis (FODA) feasibility study. Tech Rep CMU/SEI-90-TR-21, SEI
- Kang K, Kim S, Lee J, Kim K, Kim G, Shin E (1998) FORM: A feature-oriented reuse method with domain-specific reference architectures. *Ann Softw Eng* 5(1):143–168
- Kang K, Lee J, Donohoe P (2002) Feature-oriented project line engineering. *IEEE Softw* 19:58–65
- Kang KC, Sugumaran V, Park S (eds) (2009) Applied software product line engineering. Auerbach Publications



- Kästner C (2007) Aspect-oriented refactoring of Berkeley DB. Master's thesis, School of Computer Science, University of Magdeburg
- Kästner C (2010) Virtual separation of concerns. Ph.D. thesis, School of Computer Science, University of Magdeburg
- Kästner C, Apel S (2009) Virtual separation of concerns—a second chance for preprocessors. *J Object Technol* 8(6):59–78
- Kästner C, Apel S, Batory D (2007) A case study implementing features using AspectJ. In: Proc. Int'l Software Product Line Conference (SPLC). IEEE Computer Society, pp 223–232
- Kästner C, Apel S, Kuhlemann M (2008) Granularity in software product lines. In: Proc. Int'l Conf. Software Engineering (ICSE). ACM Press, pp 311–320
- Kästner C, Apel S, Kuhlemann M (2009) A model of refactoring physically and virtually separated features. In: Proc. Int'l Conf. Generative Programming and Component Engineering (GPCE). ACM Press, pp 157–166
- Kästner C, Apel S, Ostermann K (2011) The road to feature modularity? In: Proc. SPLC Workshop on Feature-Oriented Software Development (FOSD), ACM Press, pp 5:1–5:8
- Kästner C, Apel S, Thüm T, Saake G (2012a) Type checking annotation-based product lines. *ACM Trans Softw Eng Methodol (TOSEM)* 21(3):14:1–14:39
- Kästner C, Apel S, Trujillo S, Kuhlemann M, Batory D (2009b) Guaranteeing syntactic correctness for all product line variants: A language-independent approach. In: Proc. Int'l Conf. Objects, Models, Components, Patterns (TOOLS EUROPE). Lecture Notes in Business Information Processing, vol 33. Springer, pp 175–194
- Kästner C, Apel S, ur Rahman SS, Rosenmüller M, Batory D, Saake G (2009) On the impact of the optional feature problem: Analysis and case studies. In: Proc. Int'l Software Product Line Conference (SPLC), ACM Press, pp 181–190
- Kästner C, Giarrusso PG, Ostermann K (2011) Partial preprocessing of C code for variability analysis. In: Proc. Int'l Workshop on Variability Modelling of Software-intensive Systems (VaMoS). ACM Press, pp 137–140
- Kästner C, Giarrusso PG, Rendel T, Erdweg S, Ostermann K, Berger T (2011) Variability-aware parsing in the presence of lexicalmacros and conditional compilation. In: Proc. Int'l Conf. Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA). ACM Press, pp 805–824
- Kästner C, Trujillo S, 432 Apel S (2008b) Visualizing software product line variabilities in source code. In: Proc. SPLC Workshop on Visualization in Software Product Line Engineering (ViSPL). Lero, University of Limerick, pp 303–313
- Kästner C, von Rhein A, Erdweg S, Pusch J, Apel S, Rendel T, Ostermann K (2012c) Toward variability-aware testing. In: Proc. GPCE Workshop on Feature-Oriented Software Development (FOSD). ACM Press, pp 1–8
- Kästner C, Ostermann K, Erdweg S (2012b). Avariability-awaremodule system. In: Proc. Int'l Conf. Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA). ACM Press, pp 773–792
- Kiczales G, Hilsdale E, Hugunin J, Kersten M, Palm J, Griswold W (2001) An overview of AspectJ. In: Proc. Europ. Conf. Object-Oriented Programming (ECOOP). Lecture Notes in Computer Science, vol 2072. Springer, pp 327–353
- Kiczales G, Lamping J, Menhdhekar A, Maeda C, Lopes C, Loingtier J-M, Irwin J (1997) Aspect oriented programming. In: Proc. Europ. Conf. Object-Oriented Programming (ECOOP). Lecture Notes in Computer Science, vol 1241. Springer, pp 220–242
- Kim CHP, Khurshid S, Batory D (2012) Shared execution for efficiently testing product lines. In: Proc. Int'l Symp. Software Reliability Engineering (ISSRE). IEEE Computer Society pp 221–230
- Kniessel G, Koch H (2004) Static composition of refactorings. *Sci Comput Progr (SCP)* 52(1–3):9–51

- Kolberg M, Magill E, Marples D, Reiff S (2000) Results of the second feature interaction contest. In: Calder M, Magill E (eds) *Feature interactions in telecommunication systems*, vol VI. IOS Press, pp 311–325
- Krone M, Snelting G (1994) On the inference of configuration structures from source code. In: *Proc. Int'l Conf. Software Engineering (ICSE)*. IEEE Computer Society, pp 49–57
- Krueger CW (2002) Easing the transition to software mass customization. In: *Proc. Int'l Workshop on Software Product-Family Engineering (PFE) Lecture Notes in Computer Science*, vol 2290. Springer, pp 282–293
- Krueger CW (2006) New methods in software product line practice. *Commun ACM* 49:37–40
- Kuhlemann M, Batory D, Apel S (2009a) Refactoring feature modules. In: *Proceedings of the International Conference on Software Reuse (ICSR)*, Springer, pp 106–115
- Kuhlemann M, Batory D, Kästner C (2009b) Safe composition of non-monotonic features. In: *Proc. Int'l Conf. Generative Programming and Component Engineering (GPCE)*, ACM Press, pp 177–185
- Kuhn DR, Wallace DR, Gallo AM (2004) Software fault interactions and implications for software testing. *IEEE Trans Software Eng (TSE)* 30:418–421
- Kühne T (1999) A functional pattern system for object-oriented design. Ph.D. thesis, Department of Computer Science, Darmstadt University of Technology
- Laddad R (2003) *AspectJ in action: Practical aspect-oriented programming*. Manning Publications
- Laddad R (2003b) *AspectJ in Action: Practical Aspect-Oriented Programming*. Manning Publications
- Lafferty D, Cahill V (2003) Language-independent aspect-oriented programming. In: *Proc. Int'l Conf. Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, ACM Press, pp 1–12
- Lämmel R (1999) Declarative aspect-oriented programming. In: *Proc. Int'l Symp. Partial Evaluation and Semantics-Based Program Manipulation (PEPM)*, ACM Press, pp 131–146
- Latendresse M (2003). Fast symbolic evaluation of C/C++ preprocessing using conditional values. In: *Proc. Europ. Conf. on Software Maintenance and Reengineering (CSMR)*. IEEE Computer Society, pp 170–179
- Latendresse M (2004) Rewrite systems for symbolic evaluation of C-like preprocessing. In: *Proc. Int'l Conf. Automated Software Engineering (CSMR)*. IEEE Computer Society, pp 165–173
- Lauenroth K, Pohl K, Toehning S (2009) Model checking of domain artifacts in product line engineering. In: *Proc. Int'l Conf. Automated Software Engineering (ASE)*, IEEE Computer Society, pp 269–280
- Le D, Walkingshaw E, Erwig M (2011) #ifdef confirmed harmful: Promoting understandable software variation. In: *Proc. Int'l Symp. Visual Languages and Human-Centric Computing (VLHCC)*. IEEE Computer Society, pp 143–150
- Lee K et al (2006) Combining feature-oriented analysis and aspect-oriented programming for product line asset development. In: *Proc. Int'l Software Product Line Conference (SPLC)*, IEEE Computer Society, pp 103–112
- Leich T (2012) Variables Nanodatenmanagement für eingebettete Systeme. Ph.D. thesis, School of Computer Science, University of Magdeburg
- Leich T, Apel S, Rosenmüller M, Saake G (2005) Handling optional features in software product lines. In: *OOPSLA workshop on managing variabilities consistently in design and code*. <http://www.kircher-schwanninger.de/workshops/MV CDC/>
- Leroy X (1994) Manifest types, modules, and separate compilation. In: *Proc. Int'l Symp. Principles of Programming Languages (POPL)*. ACM Press, pp 109–122
- Li HC, Krishnamurthi S, Fisler K (2005) Modular verification of open features using three-valued model checking. *Autom Softw Eng* 12(3):349–382
- Lieberherr KJ, Lorenz DH, Ovlinger J (2003) Aspectual collaborations—Combining modules and aspects. *Comput J* 46(5):542–565

- Liebig J, Apel S, Lengauer C, Kästner C, Schulze M (2010) An analysis of the variability in forty preprocessor-based software product lines. In Proc. Int'l Conf. Software Engineering (ICSE). ACM Press, pp 105–114
- Liebig J, Kästner C, Apel S (2011) Analyzing the discipline of preprocessor annotations in 30million lines of C code. In: Proc. Int'l Conf. Aspect-Oriented Software Development (AOSD). ACM Press, pp 191–202
- Liebig J, von Rhein A, Kästner C, Apel S, Dörre J, Lengauer C (2013) Scalable analysis of variable software. In: Proc. Europ. Software Engineering Conf. and Symp. Foundations of Software Engineering (ESEC/FSE). ACM Press, to appear
- Lin F, Lin Y-J (1994) A building block approach to detecting and resolving feature interactions. In: Bouma LG, Velthuijsen H (eds) Feature interactions in telecommunications systems. IOS Press, pp 86–119
- Liu J, Batory D, Lengauer C (2006) Feature oriented refactoring of legacy applications. In: Proc. Int'l Conf. Software Engineering (ICSE), ACM Press, pp 112–121
- Lohmann D, Hofer W, Schröder-Preikschat W, Spinczyk O (2011) Aspect-aware operating system development. In: Proc. Int'l Conf. Aspect-Oriented Software Development (AOSD), ACM Press, pp 69–80
- Lohmann D, Scheler F, Tartler R, Spinczyk O, Schröder-Preikschat W (2006a) A quantitative analysis of aspects in the eCos kernel. In: Proc. Int'l EuroSys Conference (EuroSys). ACM Press, pp 191–204
- Lohmann D, Spinczyk O, Schröder-Preikschat W (2006b) Lean and efficient system software product lines: Where aspects beat objects. *Trans Aspect-Orient Softw Dev (TAOSD)* 2(1):227–255
- Lopez-Herrejon R (2006) Understanding feature modularity. Ph.D. thesis, Department of Computer Sciences, The University of Texas at Austin
- Lopez-Herrejon R, Batory D, Cook W (2005) Evaluating support for features in advanced modularization technologies. In: Proc. Int'l Conf. Generative and Component-Based Software Engineering (ECOOP). Lecture notes in computer science, vol 3586. Springer, pp 169–194
- Lotufo R, She S, Berger T, Czarnecki K, Wąsowski A (2010) Evolution of the Linux kernel variability model. In: Proc. Int'l Software Product Line Conference (SPLC). Springer, pp 136–150
- Madsen OL, Moller-Pedersen B (1989) Virtual classes: A powerful mechanism in object-oriented programming. In: Proc. Int'l Conf. Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), ACM Press, pp 397–406
- Masuhara H, Kawachi K (2003) Dataflow pointcut in aspect-oriented programming. In: Proc. Asian Symp. Programming Languages and Systems (APLAS), Lecture Notes in Computer Science, vol 2895. Springer, pp 105–121
- McCloskey B, Brewer E (2005) ASTEC: A new approach to refactoring C. In: Proc. Europ. Software Engineering Conf. and Symp. Foundations of Software Engineering (ESEC/FSE). ACM Press, pp 21–30
- McIlroy MD (1969) Mass produced software components. In: Software engineering: Report of a conference sponsored by the NATO science committee, Garmisch, Germany, 7–11 Oct. 1968, Scientific Affairs Division, NATO, pp 138–155
- Mehner K, Rashid A (2003) Towards a generic model for AOP (GEMA). Technical report CSEG/1/03, Computing Department, Lancaster University
- Mendonça M, Wąsowski A, Czarnecki K (2009) SAT-based analysis of feature models is easy. In: Proc. Int'l Software Product Line Conference (SPLC). ACM Press, pp 231–240
- Mens T (2002) A state-of-the-art survey on software merging. *IEEE Trans Softw Eng (TSE)* 28(5):449–462
- Mens T, Tourwé T (2004) A survey of software refactoring. *IEEE Trans Softw Eng (TSE)* 30(2): 126–139

- Metzger A, Pohl K, Heymans P, Schobbens P-Y, Saval G (2007) Disambiguating the documentation of variability in software product lines: A separation of concerns, formalization and automated analysis. In: Proc. Int'l Conf. Requirements Engineering (RE). IEEE Computer Society, pp 243–253
- Meyer B (1997) Object-oriented software construction, 2nd edn. Prentice-Hall
- Mezini M, Ostermann K (2004) Variability management with feature-oriented programming and aspects. In: Proc. Int'l Symp. Foundations of Software Engineering (FSE), ACM Press, pp 127–136
- Michel R, Classen A, Hubaux A, Boucher Q (2011) A formal semantics for feature cardinalities in feature diagrams. In: Proc. Int'l Workshop on Variability Modelling of Software-intensive Systems (VaMoS). ACM Press, pp 82–89
- Monteiro MP, Fernandes JM (2005) Towards a catalog of aspect-oriented refactorings. In: Proc. Int'l Conf. Aspect-Oriented Software Development (AOSD). ACM Press, pp 111–122
- Murphy GC, Lai A, Walker R, Robillard M (2001) Separating features in source code: An exploratory study. In: Proc. Int'l Conf. Software Engineering (ICSE). IEEE Computer Society, pp 275–284
- Murphy-Hill E, Black AP (2008) Breaking the barriers to successful refactoring: Observations and tools for extract method. In: Proc. Int'l Conf. Software Engineering (ICSE). ACM Press, pp 421–430
- Muthig D, Patzke T (2002) Generic implementation of product line components. In: Proc. Int'l Conf. Object-Oriented and Internet-based Technologies, Concepts, and Applications for a Networked World (Net.ObjectDays), Lecture Notes in Computer Science, vol 2591. Springer, pp 313–329
- Nadi S, Holt RC (2012) Mining Kbuild to detect variability anomalies in Linux. In: Proc. Europ. Conf. on Software Maintenance and Reengineering (CSMR). IEEE Computer Society, pp 107–116
- Neves L, Teixeira L, Sena D, Alves V, Kulezsa U, Borba P (2011) Investigating the safe evolution of software product lines. In: Proc. Int'l Conf. Generative Programming and Component Engineering (GPCE). ACM Press, pp 33–42
- Nhlabatsi A, Laney R, Nuseibeh B (2008) Feature interaction: The security threat from within software systems. *Prog inform* 5:75–89
- Opdyke WF (1992) Refactoring object-oriented frameworks. Ph.D. thesis, University of Illinois at Urbana-Champaign
- Oster S, Markert 1235 F, Ritter P (2010) Automated incremental pairwise testing of software product lines. In: Proc. Int'l Software Product Line Conference (SPLC). Lecture Notes in Computer Science, vol 6287. Springer, pp 196–210
- Ostermann K, Giarrusso PG, Kästner C, Rendel T (2011) Revisiting information hiding: Reflections on classical and nonclassical modularity. In: Proc. Europ. Conf. Object-Oriented Programming (ECOOP). Lecture Notes in Computer Science, vol 6813. Springer, pp 155–178
- Ostermann K, Mezini M, Bockisch C (2005) Expressive pointcuts for increased modularity. In: Proc. Europ. Conf. Object-Oriented Programming (ECOOP). Lecture Notes in Computer Science, vol 3586. Springer, pp 214–240
- Ouellet M, Merlo E, Sozen N, Gagnon M (2012) Locating features in dynamically configured avionics software. In: Proc. Int'l Conf. Software Engineering (ICSE). IEEE Computer Society, pp 1453–1454
- Parnas DL (1972) On the criteria to be used in decomposing systems into modules. *Commun ACM* 15(12):1053–1058
- Parnas DL (1979) Designing software for ease of extension and contraction. *IEEE Trans Software Eng (TSE)*, SE-5(2):128–138
- Parnas, DL (1976) On the design and development of program families. *IEEE Trans Software Eng (TSE)*, 2(1):1–9
- Parr TJ (2004) Enforcing strict model-view separation in template engines. In: Proc. Int'l Conference on World Wide Web. ACM Press, pp 224–233

- Perrouin G, Sen S, Klein J, Baudry B, le Traon Y (2010) Automated and scalable t-wise test case generation strategies for software product lines. In: Proc. Int'l Conf. Software Testing, Verification, and Validation. IEEE Computer Society, pp 459–468
- Pierce BC (2002) Types and programming languages. MIT Press
- Pohl K, Böckle G, van der Linden FJ (2005) Software product Line engineering: Foundations, principles and techniques. Springer
- Pomakis K, Atlee J (1996) Reachability analysis of feature interactions: A progress report. In: Proc. Int'l Symp. Software Testing and Analysis (ISSTA), ACM Press, pp 216–223
- Popovici A, Alonso G, Gross T (2003) Just-in-time aspects: Efficient dynamic weaving for Java. In: Proc. Int'l Conf. Aspect-Oriented Software Development (AOSD), ACM Press, pp 100–109
- Poshyvanyk D, Guéhéneuc Y-G, Marcus A, Antoniol G, Rajlich V (2007) Feature location using probabilistic ranking of methods based on execution scenarios and information retrieval. *IEEE Trans Software Eng* 33(6):420–432
- Post H, Sinz C (2008) Configuration lifting: Verification meets software configuration. In: Proc. Int'l Conf. Automated Software Engineering (ASE). IEEE Computer Society, pp 347–350
- Prehofer C (1997) Feature-oriented programming: A fresh look at objects. In: Proc. Europ. Conf. Object-Oriented Programming (ECOOP). Lecture Notes in Computer Science, vol 1241. Springer, pp 419–443
- Rabkin A, Katz R (2011) Static extraction of program configuration options. In: Proc. Int'l Conf. Software Engineering (ICSE), IEEE Computer Society, pp 131–140
- Rashid A, Royer J-C, Rummeler A (2011) Aspect-oriented, model-driven software product lines: The AMPLE way. Cambridge University Press
- Reenskaug T, Andersen E, Berre A, Hurlen A, Landmark A, Lehne O, Nordhagen E, Ness-Ulseth E, Oftedal G, Skaar A, Stenslet P (1992) OORASS: Seamless support for the creation and maintenance of object-oriented systems. *J Object-Oriented Program (JOOP)* 5(6):27–41
- Refstrup JG (2009) Adapting to change: Architecture, processes and tools: A closer look at HP's experience in evolving the Owen software product line. In: Proc. Int'l Software Product Line Conference (SPLC). Keynote presentation
- Reisner E, Song C, Ma K-K, Foster JS, Porter A (2010) Using symbolic evaluation to understand behavior in configurable software systems. In: Proc. Int'l Conf. Software Engineering (ICSE), ACM Press, pp 445–454
- Ribeiro M, Pacheco H, Teixeira L, Borba P (2010) Emergent feature modularization. In: Companion Int'l Conf. Object-Oriented Programming, Systems, Languages and Applications (OOPSLA). ACM Press, pp 11–18
- Robillard M, Weigand-Warr F (2005) ConcernMapper: Simple view-based separation of scattered concerns. In: Proc. OOPSLA Workshop on Eclipse Technology eXchange (ETX). ACM Press, pp 65–69
- Robillard M, Murphy GC (2002) Concern graphs: Finding and describing concerns using structural program dependencies. In: Proc. Int'l Conf. Software Engineering (ICSE). ACM Press, pp 406–416
- Rosenmüller M, Apel S, Leich T, Saake G (2009a) Tailor-made data management for embedded systems: A case study on Berkeley DB. *Data Knowl Eng (DKE)* 68(12):1493–1512
- Rosenmüller M, Kästner C, Siegmund N, Sunkle S, Apel S, Leich T, Saake G (2009b) SQL à la carte—toward tailor-made data management. In: Proc. GI-Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW) Lecture Notes in Informatics. Gesellschaft für Informatik (GI), vol P-144, pp 117–136
- Rosenmüller M (2011) Towards flexible feature composition: Static and dynamic binding in software product lines. Ph.D. thesis, School of Computer Science, University of Magdeburg
- Rosenmüller M, Siegmund N, Apel S, Saake G (2011) Flexible feature binding in software product lines. *Autom Software Eng* 18(2):163–197

- Rosenmüller M, Siegmund N, Schirmeier H, Sincero J, Apel S, Leich T, Spinczyk O, Saake G (2008) FAME-DBMS: Tailor-made data management solutions for embedded systems. In: Proc. EDBT Workshop on Software Engineering for Tailor-made Data Management. ACM Press, pp 1–6
- Saake G, Rosenmüller M, Siegmund N, Kästner C, Leich T (2009) Downsizing data management for embedded systems. *Egyptian Comput Sci J* 31(1):1–13
- Sato Y, Chiba S, Tatsubori M (2003) A selective, just-in-time aspect weaver. In: Proc. Int'l Conf. Generative Programming and Component Engineering (GPCE). Lecture Notes in Computer Science, vol 2830. Springer, pp 189–208
- Savolainen J, Bosch J, Kuusela J, Männistö T (2009) Default values for improved product line management. In: Proc. Int'l Software Product Line Conference (SPLC). Carnegie Mellon University, pp 51–60
- Schaefer I, Bettini L, Damiani F (2011) Compositional type-checking for delta-oriented programming. In: Proc. Int'l Conf. Aspect-Oriented Software Development (AOSD). ACM Press, pp 43–56
- Schaefer I, Bettini L, Damiani F, Tanzarella N (2010) Delta-oriented programming of software product lines. In: Proc. Int'l Software Product Line Conference (SPLC), Springer, pp 77–91
- Schmid K, Rabiser R, Grünbacher P (2011) A comparison of decision modeling approaches in product lines. In: Proc. Int'l Workshop on Variability Modelling of Software-intensive Systems (VaMoS). ACM Press, pp 119–126
- Schobbens P-Y, Heymans P, Trigaux J-C, Bontemps Y (2007) Generic semantics of feature diagrams. *Comput Netw* 51(2):456–479
- Schulze S (2013) Analysis and removal of code clones in software product lines. Ph.D. thesis, School of Computer Science, University of Magdeburg
- Schulze S, Thüm T, Kuhlemann M, Saake G (2012) Variant-preserving refactoring in feature oriented software product lines. In: Proc. Int'l Workshop on Variability Modelling of Software-intensive Systems (VaMoS). ACM Press, pp 73–81
- She S, Lotufo R, Berger T, Wasowski A, Czarnecki K (2011) Reverse engineering feature models. In: Proc. Int'l Conf. Software Engineering (ICSE). ACM Press, pp 461–470
- She S, Lotufo R, Berger T, Wasowski A, Czarnecki K (2010) The variability model of the Linux kernel. In: Proc. Int'l Workshop on Variability Modelling of Software-intensive Systems (VaMoS). University of Duisburg-Essen, pp 45–51
- Siegmund N, Kästner C, Rosenmüller M, Heidenreich F, Apel S, Saake G (2009a) Bridging the gap between variability in client application and database schema. In: Proc. GI-Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW). Lecture Notes in Informatics, vol. P-144. Gesellschaft für Informatik (GI), pp 297–306
- Siegmund N, Rosenmüller M, Kuhlemann M, Kästner C, Apel S, Saake G (2011) SPL Conqueror: Toward optimization of non-functional properties in software product lines. *Softw Qual J Spec Issue Qual Eng Softw Prod Lines* 3:487–517
- Siegmund N, Rosenmüller M, Moritz G, Saake G, Timmermann D (2009b) Towards robust data storage in wireless sensor networks. *IETE Tech Rev* 26(5):335–340
- Simos MA (1995) Organization domain modeling (ODM): Formalizing the core domain modeling life cycle. In: Proc. Symp. Software Reusability (SSR). ACM Press, pp 196–205
- Sincero J, Schirmeier H, Schröder-Preikschat W, Spinczyk O (2007) Is the Linux kernel a software product line? In: Proc. Int'l Workshop Open Source Software and Product Lines (SPLC-OSSPL)
- Sincero J, Schröder-Preikschat W, Spinczyk O (2010) Approaching non-functional properties of software product lines: Learning from products. In: Proc. Asia-Pacific Software Engineering Conf. (APSEC). IEEE Computer Society, pp 147–155
- Singh N, Gibbs C, Coady Y (2007) C-CLR: A tool for navigating highly configurable system software. In: Proc. AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software (ACP4IS). ACM Press, p 9

- Smaragdakis Y, Batory D (2002a) Mixin layers: An object-oriented implementation technique for refinements and collaboration-based designs. *ACM Trans Softw Eng Methodol* 11(2):215–255
- Smaragdakis Y, Batory D (2002) Mixin layers: An Object-oriented implementation technique for refinements and collaboration-based designs. *ACM Trans Software Eng Methodol (TOSEM)* 11(2):215–255
- Snyder A (1986) Encapsulation and inheritance in object-oriented programming languages. In: *Proc. Int'l Conf. Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, ACM Press, pp 38–45
- Sommerville I (2010) *Software engineering*, 9th edn. Pearson Addison Wesley
- Spencer H, Collyer G (1992) #ifdef considered harmful or portability experience with C news. In: *Proc. USENIX Conf.*, USENIX Association, pp 185–198
- Spinczyk O, Lohmann D, Urban M (2005) AspectC++: An AOP extension for C++. *Softw Dev J* 14:68–74
- Spinczyk O (2002) *Aspektororientierung und Programmfamilien im Betriebssystembau*. Ph.D. thesis, School of Computer Science, University of Magdeburg
- Staples M, Hill D (2004) Experiences adopting software product line development without a product line architecture. In: *Proc. Asia-Pacific Software Engineering Conf. (APSEC)*. IEEE Computer Society, pp 176–183
- Steimann F (2005) Domain models are aspect free. In: *Proc. Int'l Conf. Model Driven Engineering Languages and Systems (MoDELS)*. Lecture Notes in Computer Science, vol 3713. Springer, pp 171–185
- Steimann F (2006) The paradoxical success of aspect-oriented programming. In: *Proc. Int'l Conf. Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, ACM Press, pp 481–497
- Steimann F, Pawlitzki T, Apel S, Kästner C (2010) Types and modularity for implicit invocation with implicit announcement. *ACM Trans Software Eng Methodol (TOSEM)* 20(1):1:1–1:43
- Störzer M, Koppen C (2004) PCDiff: Attacking the fragile pointcut problem, abstract. In: *European Interactive Workshop on Aspects in Software*
- Streitferdt D, Riebisch M, Philippow I (2003) Details of formalized relations in feature models using OCL. In: *Proc. Int'l Conf. and Workshop Engineering of Computer-Based Systems (ECBS)*. IEEE Computer Society, pp 297–304
- Strniša R, Sewell P, Parkinson M (2007) The Java module system: Core design and semantic definition. In: *Proc. Int'l Conf. Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*. ACM Press, pp 499–514
- Sullivan K, Griswold W, Song Y, Cai Y, Shonle M, Tewari N, Rajan H (2005) Information hiding interfaces for aspect-oriented design. In: *Proc. Int'l Symp. Foundations of Software Engineering (FSE)*, ACM Press, pp 166–175
- Sultana N, Thompson S (2008) Mechanical verification of refactorings. In: *Proc. Int'l Symp. Partial Evaluation and Semantics-Based Program Manipulation (PEPM)*. ACM Press, pp 51–60
- Sunyé G, Pollet D, Traon YL, Jézéquel J-M (2001) Refactoring UML models. In: *Proc. Int'l Conf. UML. Modeling Languages, Concepts, and Tools*, of Lecture Notes in Computer Science, vol. 2185. Springer, pp 134–148
- Swahnberg M, van Gurp J, Bosch J (2005) A taxonomy of variability realization techniques. *SoftwPract Exp* 35(8):705–754
- Szewczyk R et al (2004) Habitat monitoring with sensor networks. *Commun ACM* 47(6):34–40
- Szyperski C (1997) *Component software: Beyond object-oriented programming*. Wesley
- Tamrawi A, Nguyen HA, Nguyen HV, Nguyen TN (2012) Build code analysis with symbolic evaluation. In: *Proc. Int'l Conf. Software Engineering (ICSE)*. IEEE Computer Society, pp 650–660

- Tarr P, Ossher H, Harrison W, Sutton S Jr (1999) N degrees of separation: Multi-dimensionalseparation of concerns. In: Proc. Int'l Conf. Software Engineering (ICSE), IEEE Computer Society, pp 107–119
- Tartler R, Lohmann D, Sincero J, Schröder-Preikschat W (2011) Feature consistency in compiletime-configurable system software: Facing the linux 10,000 feature problem. In: Proc. Int'l EuroSys Conference (EuroSys). ACM Press, pp 47–60
- Tešanovic' A, Sheng K, Hansson J (2004) Application-tailored database systems: A case of aspects in an embedded database. In: Proc. Int'l Database Engineering and Applications Symposium. IEEE Computer Society, pp 291–301
- Thaker S, Batory D, Kitchin D, Cook W (2007) Safe composition of product lines. In: Proc. Int'l Conf. Generative Programming and Component Engineering (GPCE). ACM Press, pp 95–104
- Thüm T, Apel S, Kästner C, Kuhlemann M, Schaefer I, Saake G (2012a) Analysis strategies for software product lines. Technical Report FIN-004-2012, School of Computer Science, University of Magdeburg
- Thüm T, Batory D, Kästner C (2009) Reasoning about edits to feature models. In: Proc. Int'l Conf. Software Engineering (ICSE). IEEE Computer Society, pp 254–264
- Thüm T, Kästner C, Erdweg S, Siegmund N (2011a) Abstract features in feature modeling. In: Proc. Int'l Software Product Line Conference (SPLC). IEEE Computer Society, pp 191–200
- Thüm T, Schaefer I, Apel S, Hentschel M (2012b) Family-based theorem proving for deductive verification of software product lines. In: Proc. Int'l Conf. Generative Programming and Component Engineering (GPCE). ACM Press, pp 11–20
- Thüm T, Schaefer I, KuhlemannM, Apel S (2011b) Proof composition for deductive verification of software product lines. In: Proc. Int'l Workshop on Variability-Intensive Systems Testing, Validation & Verification (VAST). IEEE Computer Society, pp 270–277
- Tsang S, Magill E (1998) Learning to detect and avoid run-time feature interactions in intelligent networks. IEEE Trans Software Eng (TSE) 24(10):818–830
- Utas G (1998) A pattern language of feature interaction. In: Kimbler K, Bouma LG (eds) Feature interactions in telecommunications systems V, IOS Press, pp 98–114
- van der Linden FJ, Schmid K, Rommes E (2007) Software product lines in action: The best industrial practice in product line engineering. Springer
- van der Linden R (1994) Using an architecture to help beat feature interaction. In: Bouma W, Velthuisen H (eds) Feature interactions in telecommunications systems. IOS Press, pp 24–35
- van der Storm T (2004). Variability and component composition. In: Proc. Int'l Conf. Software Reuse (ICSR) Lecture notes in computer science, vol 3107. Springer, pp 157–166
- van Ommering R (2002) Building product populations with software components. In: Proc. Int'l Conf. Software Engineering (ICSE), ACM Press, pp 255–265
- VanHilst M, Notkin D (1996) Using role components in implement collaboration-based designs. In: Proc. Int'l Conf. Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), ACM Press, pp 359–369
- Voelter M, Groher I (2007) Product line implementation using Aspect-oriented and model-driven software development. In: Proc. Int'l Software Product Line Conference (SPLC), IEEE Computer Society, pp 233–242
- Wand M, Kiczales G, Dutchyn C (2004) A semantics for advice and dynamic join points in aspect oriented programming. ACM Trans Program Lang Syst (TOPLAS) 26(5):890–910
- Weise D, Crew R (1993) Programmable syntax macros. In: Proc. Int'l Conf. Programming Language Design and Implementation (PLDI). ACM Press, pp 156–165
- Zave P (2003) An experiment in feature engineering. In: Programming Methodology. Springer, pp 353–377



- Zave P (2010) Modularity in distributed feature composition. In: Nuseibeh B, Zave P (eds) *Software requirements and design: The work of Michael Jackson*. Good Friends Publishing, pp 267–290
- Zhang C, Jacobsen H-A (2003) Quantifying aspects in middleware platforms. In: *Proc. Int'l Conf. Aspect-Oriented Software Development (AOSD)*. ACM Press, pp 130–139

# Index

## Symbols

- composition operator, 137
- #define, 110
- #else, 111
- #ifdef, 110, 181
- #ifdef hell, 120
- #ifndef, 111
- #include, 110
- #undef, 113
- n-way interaction
- 150-percent analysis, 244
- 2-way interaction, 216
- 3-way interaction, 216

## A

- abc, 81
- Abstract feature
  - detection, 257
  - extension, 35
- Abstract method, 71, 72
- ACC, 167
- ACDT, 167
- Activated feature, 250
- Adapter pattern, 93
- Adoption path, 40
- Advanced dynamic crosscutting concern, 158
- Advice, 143
- AFM, 161
- AHEAD, 137, 138, 166, 167, 172, 275
- AHEAD Tool Suite
- AJDT, 167, 171
- Algebraic Hierarchical Equations for Application Design, 137
- Alternative, 28, 31, 32
- Alternative macros, 113
- Analysis
  - approximations, 269
  - brute force, 261

- family-based, 264, 269
- feature-based, 272
- of product lines, 260
- sampling, 263
- variability aware, 260
- variability-aware, 243, 261
- annotation, 111
- Annotation-based approach, 50
- ANT, 125
- ant, 105
- Antenna, 111, 114, 126
- AOP, 141
- Application engineering, 20, 21
  - component composition, 92
- Aspect, 142, 143
- Aspect weaver, 143
- Aspect weaving, 143
- Aspect-oriented programming, 141
- AspectC++, 167
- AspectC#, 165
- AspectJ, 145, 146, 167
- Aspectual Collaborations, 172
- Aspectual feature modules, 162
- Atoms in the universe, 243
- Automatic product derivation, 14
- Automation
  - component composition and plug-in selection, 92
- Automation of product derivation, 182

## B

- Basic dynamic crosscutting concern, 158
- BDD, 254
- Berkeley DB, 111, 207, 234
- Best-of-breed approach, 89
- BigLever Software, Inc., 304
- Binary decision diagram, 254
- Binding time, 48

- Black box
  - component, 89
- Black box, 80
- Black-box framework, 81
- Boolean satisfiability problem, 247
- Branch
  - version control, 100
- Bridge pattern, 93
- Brute-force analysis, 261
- Build system, 105
  - analysis, 109, 259
  - presence conditions, 259
- Busybox, 275
  
- C**
- CaesarJ, 172
- Calculator example, 82
- Call forwarding and call waiting interaction, 214
- Callback, 74
- Callback function, 73
- Captain Feature, 39
- Case study
  - variability-aware analysis, 275
- CDL, 39, 309
- Cflow, 146, 158
- Cflowbelow, 160
- Change feature model, 224
- Checking validity, 182
- Choice, 31
- CIDE, 177, 180, 186, 277
- CIDE+, 277
- Class refinement, 133
- Clone-and-own approach, 41
- Closed world, 273
- Coarse-grained, 59
- Code fragment
  - analysis, 254
- Code scattering, 56
- Code smell, 194
  - duplicate code, 194
  - large class, 194
  - long method, 194
  - shotgun surgery, 194
- Code tangling, 56
- Cohesion, 55, 56
- Collaboration, 130, 131
- Collaboration-based design, 131
- Combinatorial explosion, 243
- Command-line parameter, 66
- Comparing feature models, 252
- Compile-time, 120
- Compile-time binding, 48
- Compile-time variability, 49
- Completeness, 263
- Component, 76, 89
  - definition, 89
  - sizing, 89
  - versus plug-in, 93
- Component family model, 304
- Component interface, 76
- Composition operator, 137
- Composition-based approach, 50
- ConcernMapper, 176, 186
- Concrete observer, 70
- Conditional compilation, 110, 227
  - in Java, native, 113
  - on models, 115
  - presence conditions, 258
  - with Antenna, 114
  - with cpp, 111
  - with Munge, 111
- Conditional statement, 66
- Configuration file, 66
- Configuration knowledge, 121
- Configuration lifting, 272
- Configuration object, 68
- Configuration parameter, 66
- Consistency
  - of tracing, 177
- Consistent feature model, 247
- Constant, 135
- Constant function, 135
- Constraint satisfaction problem, 254
- Containment hierarchy, 135
- Context-oriented programming, 170
- ContextJ, 170
- ContextJS, 309
- ContextL, 309
- ContextR, 309
- Control flow
  - with preprocessors, 121
- Control-flow analysis, 255
  - variability-aware, 273
- cpp, 110, 111, 122
- Cross-tree, 32
- Cross-tree constraint, 29, 252
- Crosscutting concern, 55
- CSP solver, 254
- CVL, 99
- CVS, 99
  
- D**
- Data-flow analysis
  - variability-aware, 273
- Deactivated feature, 250

- Dead feature
    - detection, 249
  - Dead feature code, 255
  - Dead-code elimination, 255
  - Decision model, 26, 42
  - Decorator class, 76
  - Decorator pattern, 75
  - Delegation, 73
  - Delta-oriented programming, 168
  - DeltaJ, 309
  - Derivative modules
    - presence conditions, 257
  - Design for change, 58, 87, 89, 92
  - Design pattern, 70
    - adapter, 93
    - bridge, 93
    - decorator, 75
    - definition, 70
    - facade, 91
    - factory method, 109
    - observer, 70
    - singleton, 91
    - strategy, 73
    - template method, 71
  - Diamond problem, 82
  - Disciplined annotation, 116
  - Distinct module for coordination
    - code, 230
  - Domain, 19
  - Domain analysis, 21, 22
    - for sizing components, 92
  - Domain engineering, 20, 21
  - Domain implementation, 21, 25
  - Domain modeling, 22, 23
  - Domain scoping, 22
  - Duplicate-code code smell, 194
  - Dynamic binding, 48
  - Dynamic crosscutting concern, 157
  - Dynamic variability, 48
- E**
- Early binding, 48
  - Eclipse, 80, 87
  - eCos
  - Edits of feature models, 252
  - Embedded data management, 12
  - Encapsulation, 56
  - endif, 110
  - Equivalent feature models, 252
  - Exponential explosion, 243
  - Extract Method, 208
  - Extract-method refactoring, 195
  - Extractive product line adoption, 120, 184
- F**
- Facade pattern, 91
  - Factory-method pattern, 109
  - False optional code fragments, 257
  - False optional feature
    - detection, 249
  - FAMA, 277
  - FAME-DBMS, 236
  - Family-based analysis, 264
  - Family-based type checking, 264, 269
  - FEAT, 186
  - Feature, 17, 18
  - Feature attribute, 38
  - Feature dependency, 18
  - Feature diagram, 26, 28, 32
  - Feature expression, 136
  - Feature interaction
  - Feature model, 26, 27
    - analysis, 244
    - consistent, 247
    - generalization, 252
    - Linux kernel, 36
    - normalization, 35
    - refactoring, 252
    - specialization, 252
    - testing, 248
    - validity, 247
  - Feature modeling, 26
  - Feature Modeling Plug-in, 39
  - Feature modularity, 130
  - Feature module, 132
  - Feature orientation
  - Feature selection, 18
    - validity analysis, 245
  - Feature traceability, 54
  - Feature-aware analysis, 244
  - Feature-based type checking, 272
  - Feature-code mapping
    - analysis, 254
  - Feature-interaction problem, 214
  - Feature-model refactoring, 252
  - Feature-oriented programming, 130
    - presence conditions, 257
  - FeatureC++, 167
  - FeatureCommander, 178, 180, 186
  - FeatureHouse, 138, 166, 172
  - FeatureIDE, 39, 126, 135, 166, 167, 172, 183, 186, 276
  - FeatureMapper, 115, 116, 177, 186, 277
  - Femto OS, 121, 123
  - File-level variability, 108
  - Fine-grained, 59
  - Fire and flood control interaction, 214
  - First-order interaction, 216

fmp2rsm, 115, 177, 186  
 FOP, 130  
 Fragile-pointcut problem, 151  
 Framework, 79, 80  
   black-box, 81  
   definition, 80, 81  
   versus preprocessor, 120  
   white-box, 80  
 Framework evolution, 87  
 Framework example, 82  
 Function parameter, 66

## G

Gears, 39, 114, 115, 186  
 Generalization, 252, 253  
   feature model, 252  
 GenVoca, 135, 137  
 GenVoca model, 135  
 git, 99  
 Global variable, 66  
 Glue code, 92  
 GNU M4, 114  
 GPP, 114  
 Granularity, 59  
   with preprocessors, 120

## H

Handcrafting, 3  
 Hello-world example, 264  
 Heterogeneous crosscutting concern, 153  
 Higher-order interaction, 216  
 Homogeneous crosscutting  
   concern, 153  
 Hook method, 217  
 Hot spot, 80

## I

IDE, 80  
 ifdef, 110  
 ifdef hell, 120  
 ifndef, 111  
 Implementation strategy, 25  
 Inadvertent feature interaction, 214  
 Include directive, 111  
 Inconsistent feature model, 247  
 Inflexible inheritance hierarchies, 82  
 Information hiding, 56–58  
 Inheritance  
   limitations, 82  
 Inlining functions, 113  
 Integrated development environment, 80

Inter-type declaration, 144  
 Interface, 56, 57  
 Interface compatibility, 272  
 Interface inference, 272  
 Invasive extension, 54  
 Inversion of control, 80, 85

## J

Jak, 133  
 JBoss AOP, 167  
 Join point, 143  
 Join-point shadow, 143

## K

Kbuild, 107–109, 275  
 Kconfig, 36, 39  
 Koala, 310

## L

Language-based approach, 50  
 Large-class code smell, 194  
 Late binding, 48  
 Level of granularity, 59  
 Lifting  
   analysis, 261  
 Linux kernel, 310  
   dead-code analysis, 275  
   feature modeling, 36  
 Load-time binding, 48  
 Load-time variability, 49  
 Long-method code smell, 194

## M

M4, 114  
 Macro, 111  
 make, 105, 107  
 Mandatory, 31, 32, 257  
 Mandatory feature, 249  
   detection, 249  
 Mapping  
   analysis, 254  
 Market  
   for components, 89  
   for services, 90  
 Mass customization  
 Mass production  
 maven, 105  
 Mentor-student collaboration, 130  
 Mercurial, 99  
 Metaprogramming

- lightweight, with preprocessors, 110
- Method argument, 66
- MFC, 81
- Mixin composition, 82
- Mixin-based inheritance, 134
- Mobile Media, 275
- Model checking, 276
  - variability-aware, 273
- Modular analysis, 272
- Modular reasoning, 57, 58
- Modularity, 58, 86
- Module, 56, 58
- Module interconnection language, 92, 95
- Move-method refactoring, 195
- Moving code, 226
- Multiple implementations, 225
- Multiple inheritance
- Munge, 111–114, 126
- Mutually exclusive, 28

## N

- n-way interaction, 216
- Negative variability, 51
- Noncode artifacts, 60
- Noninvasive extension, 54
- Number of valid feature selections, 251

## O

- Obfuscation
  - code, with preprocessors, 121
- Object Teams, 172
- Obliviousness, 54
- Observer interface, 70
- Observer pattern, 70
- One-size-fits-all, 7
- Open world, 273
- Open-world development, 87
- Optional, 31, 32
- Optional weaving, 228
- Optional-feature problem, 219
- or, 31, 32
- Orthogonal variability model, 26, 42
- OSGi framework, 310

## P

- Parameter, 66
  - static analysis, 260
- Parameter object, 68
- Parsing
  - variability-aware, 273

- Partial evaluation, 68
- Partial feature selection
  - analysis, 250
  - valid, 250
- Patch, 100, 104, 109
- Perforce, 99
- Physical separation of concerns, 52, 175
- Plug-in, 80, 81
  - versus component, 93
- Pointcut, 143
- Polyglot, 81
- Positive variability, 51
- Preference dialog, 66
- Preplanned, 81
- Preplanning, 52, 71, 79
  - for design patterns, 70
  - with preprocessors, 121
- Preprocessor, 109, 110
  - lexical vs syntactic, 114
- Presence condition, 265
  - extraction, 257
  - on structures, 265
- Presence conditions, 265
  - symbolic execution, 258
- Principle of uniformity, 60
- Problem space, 20, 21
  - analysis, 244
- Product, 19
- Product assembly, 21
- Product configuration, 21
- Product derivation, 21, 26
  - with components, 92
- Product generation, 21
- Product line
  - manufacturing, 4
  - software, 7
- Product line-aware analysis, 244
- Product-line refactoring, 201
- Product-line refinement, 223
- Product-preserving refactoring, 200, 201
- Programming-in-the-large, 95
- Propositional formula
  - feature selection, 245
- Propositional logic, 28
- ProVeLines, 277
- Publish/subscribe pattern, 70
- Pure-systems GmbH, 304
- pure::variants, 38, 39, 114, 115, 186, 276
- Push-button approach, 26

## Q

- Quantification, 145

**R**

Refactoring, 193–195  
 extract method, 195  
 feature model, 252  
 move method, 195  
 product-preserving, 201  
 rename method, 195  
 variability-enhancing, 201  
 variability-preserving, 201  
 Refactoring catalog, 195  
 Refactoring feature modules, 169  
 Refactoring product lines, 169  
 Refactoring step, 195  
 Release, 100, 101  
 Rename-method refactoring, 195  
 Requirements analysis, 21, 24  
 Reuse, 20  
 Reuse framework, 25  
 Reuse-versus-use dilemma, 91  
 Revision, 100, 101  
 definition, 100  
 Revision control, 99  
 Role, 131  
 Run-time binding, 48  
 Run-time variability, 49

**S**

Safe evolution, 223  
 SafeGen, 277  
 Sampling strategy, 263  
 SAT solver, 247  
 Scattered code, 56  
 Scope, 21  
 Second-order interaction, 216  
 Separate compilation, 58  
 Separate concerns, 55  
 Separation of concerns, 58, 121  
 with preprocessors, 121  
 Service, 89, 90  
 Service orchestration, 90, 92  
 Service-oriented architecture, 90  
 Service-oriented architectures, 95  
 Shotgun-surgery code smell, 194  
 Singleton, 91  
 Singleton pattern, 91  
 Size of a feature model, 251  
 SNIP, 277  
 Software component, 89  
 definition, 89  
 Software configuration management, 99, 105  
 Software ecosystem, 87  
 Software factory, 186  
 Software product line, 7

Solution space, 20, 21  
 Soundness, 263  
 Specialization, 252, 253  
 feature model, 252  
 Specification  
 global, 263  
 SPL2go, 172  
 SPLOT, 276  
 SPLverifier, 277  
 Spring, 167  
 Staged configuration, 25  
 Standard software, 7  
 Static analysis, 243, 276  
 of build systems, 109, 259  
 Static binding, 48  
 Static crosscutting concern, 157  
 Static variability, 48  
 Store example, 269  
 Strategy pattern, 73  
 StringTemplate, 116  
 Structuring, 20  
 Subject, 70  
 Subversion, 99, 125  
 Swing, 81  
 Syntactic preprocessor, 114

**T**

Tag-and-prune, 114, 115  
 Tangled code, 56  
 Template processor, 115  
 Template-method design pattern, 81  
 Template-method pattern, 71  
 Testing  
 variability-aware, 273  
 Theorem proving  
 variability-aware, 273  
 Tool-based approach, 50  
 Tracing link, 175  
 Type checking  
 family-based, 264, 269  
 feature-based, 272  
 TypeChef, 277  
 Tyranny of the dominant decomposition, 55

**U**

Undead code fragments  
 detection, 257  
 undef, 113  
 Undertaker, 277  
 Uniformity, 60  
 in version-control systems, 103  
 with preprocessors, 120

Uniformly, 103, 120  
Unreachable-code detection, 255

**V**

Valid feature model, 247  
Valid feature selection, 27, 245  
    counting, 251  
Valid product, 27  
Variability, 20, 48  
Variability encoding, 272  
Variability model, 26  
    analysis, 244  
Variability modeling, 26  
Variability smell, 197  
Variability-aware analysis, 243, 244, 260, 261  
    patterns, 274  
Variability-enhancing refactoring, 200, 201  
Variability-preserving refactoring, 200, 201  
Variant, 100, 101  
Version  
    definition, 100  
Version control, 99

Version Editor, 181, 182  
View, 179  
View Infinity, 186  
Vim, 117  
Virtual product, 49  
Virtual separation of concerns, 52, 175, 180,  
    184  
Visual SourceSafe, 99  
Visual Studio, 87  
Visualization  
    of tracing, 178  
VMC, 277

**W**

Web service, 89, 90  
White-box, 80  
White-box framework, 80  
Whole-product-line analysis, 244, 260

**X**

XVCL, 116