Scientific Writing

An Experience Report | Thomas Thüm | May 19, 2021

Software Engineering
Programming Languages

ulm university universität
uulm

Lecture Overview

1. Motivation
2. The Snapshot
3. A Master's Thesis
4. The Wristshot
5. Stay Home Messages

Motivation

Recap: Types of Scientific Contributions [youtube.com]

Scientific Contributions

- theory: mathematical or machine-checked proofs
- conceptual contribution: algorithms, applications of algorithms, methodologies
- artifact: prototypical tool support, data/benchmark, challenges
- empirical evaluation: experiments, hypothesis testing
- survey: literature overview, identification of gaps, research roadmap

Recap: Kinds of Scientific Literature [youtube.com]

Literature without Peer Review

- books
- bachelor's, master's, Ph.D. theses
- technical report, magazine articles, blog post, tweet

Peer-Reviewed Literature

- journal article (typically no presentation)
- conference/symposium/workshop paper
- poster, extended abstract

In Ulm: at least three papers/articles at major conferences/journals for cumulative dissertation

Criteria for Authorship [youtube.com]

Association for Computing Machinery:

"Anyone listed as author on an ACM manuscript submission must meet all the following criteria:"

- "they have made **substantial intellectual contributions** to some components of the original work described in the manuscript; and
- they have **participated** in drafting and/or revision of the manuscript and
- they are **aware** the manuscript has been submitted for publication; and
- they agree to be held **accountable** for any issues relating to correctness or integrity of the work"

https://www.acm.org/publications/policies/authorship

Scientific Peer Review [youtube.com]

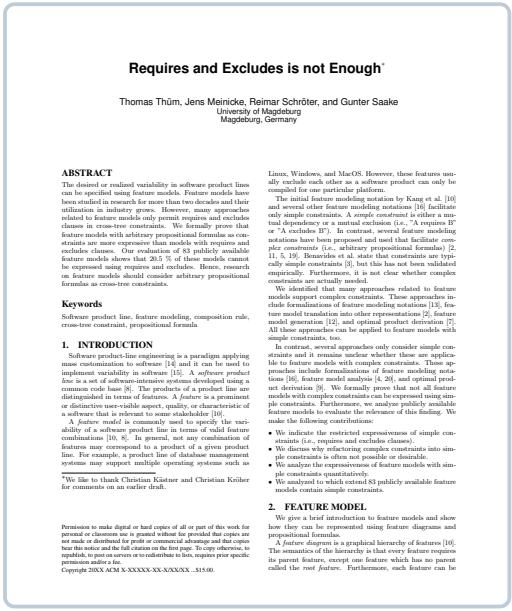
What is Peer Review?

- review: critical read and frank comments on your work
- peers: 3–4 external, independent researchers/experts (e.g., PhD students, PostDocs, professors)
- peer review: aim is objective evaluation and constructive feedback
- acceptance/rejection based on reviews
- central element in the process of research
- single-blind review: anonymous reviewers
- double-blind review: anonymous reviewers and authors

The Snapshot

The Snapshot (Der Schnellschuss)

The Snapshot



Requires and Excludes is not Enough*

Thomas Thüm, Jens Meinicke, Reimar Schröter, and Gunter Saake
University of Magdeburg
Magdeburg, Germany

ABSTRACT

The desired or realized variability in software product lines can be specified using feature models. Feature models have been studied in research for more than two decades and their utilization in industry grows. However, many approaches related to feature models only permit requires and excludes clauses in cross-tree constraints. We formally prove that feature models with arbitrary propositional formulas as constraints are more expressive than models with requires and excludes clauses. Our evaluation of 83 publicly available feature models shows that 20.5 % of these models cannot be expressed using requires and excludes. Hence, research on feature models should consider arbitrary propositional formulas as cross-tree constraints.

Keywords

Software product line, feature modeling, composition rule, cross-tree constraint, propositional formula

1. INTRODUCTION

Software product-line engineering is a paradigm applying mass customization to software [14] and it can be used to implement variability in software [15]. *A software product line* is a set of software-intensive systems developed using a common code base [8]. The products of a product line are distinguished in terms of features. A *feature* is a prominent or distinctive user-visible aspect, quality, or characteristic of a software that is relevant to some stakeholder [10].

A *feature model* is commonly used to specify the variability of a software product line in terms of valid feature combinations [10, 8]. In general, not any combination of features may correspond to a product of a given product line. For example, a product line of database management systems may support multiple operating systems such as Linux, Windows, and MacOS. However, these features usually exclude each other as a software product can only be compiled for one particular platform.

The initial feature modeling notation by Kang et al. [10] and several other feature modeling notations [16] facilitate only simple constraints. A *simple constraint* is either a mutual dependency or a mutual exclusion (i.e., "A requires B" or "A excludes B"). In contrast, several feature modeling notations have been proposed and used that facilitate *complex constraints* (i.e., arbitrary propositional formulas) [2, 11, 5, 19]. Benavides et al. state that constraints are typically simple constraints [3], but this has not been validated empirically. Furthermore, it is not clear whether complex constraints are actually needed.

We identified that many approaches related to feature models support complex constraints. These approaches include formalizations of feature modeling notations [13], feature model translation into other representations [2], feature model generation [12], and optimal product derivation [7]. All these approaches can be applied to feature models with simple constraints, too.

In contrast, several approaches only consider simple constraints and it remains unclear whether these are applicable to feature models with complex constraints. These approaches include formalizations of feature modeling notations [16], feature model analysis [4, 20], and optimal product derivation [9]. We formally prove that not all feature models with complex constraints can be expressed using simple constraints. Furthermore, we analyze publicly available feature models to evaluate the relevance of this finding. We make the following contributions:

- We indicate the restricted expressiveness of simple constraints (i.e., requires and excludes clauses).
- We discuss why refactoring complex constraints into simple constraints is often not possible or desirable.
- We analyze the expressiveness of feature models with simple constraints quantitatively.
- We analyzed to which extend 83 publicly available feature models contain simple constraints.

2. FEATURE MODEL

We give a brief introduction to feature models and show how they can be represented using feature diagrams and propositional formulas.

A *feature diagram* is a graphical hierarchy of features [10]. The semantics of the hierarchy is that every feature requires its parent feature, except one feature which has no parent called the *root feature*. Furthermore, each feature can be

*We like to thank Christian Kästner and Christian Kröher for comments on an earlier draft.

The Snapshot

Timeline for a Paper

- 2011-08-29 reviewers tell us to use FAMA for feature-model analysis in FeatureIDE
- 2011-11-29 we respond that the FAMA models are not expressive enough
- 2012-01-09 reviewers propose that we ask the FAMA developers to change their tool
- 2012-04-17 email discussion after conference meeting
- 2012-10-26 abstract (first draft)
- 2012-10-28 comments by co-author
- 2012-10-29 comments and data by co-author
- 2012-11-01 received weird and probably wrong formula by co-author

$$2^{2^{n-1}} - \sum_{i=1}^{n-1} \left(\frac{1}{4}\right)^i 2^{2^{n-1}} - 1$$

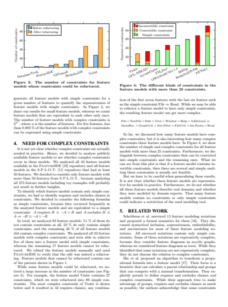
Continued Timeline for a Paper

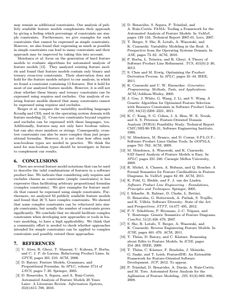
- 2012-11-02 algorithm by co-author (to avoid weird formula)
- 2012-11-05 comments by two co-authors
- 2012-11-06 paragraph and diagram by co-author, comments by peer
- 2012-11-07 comment by peer: "you are making a formal argument about expressiveness without formal support yet. i would have expected a proof. the empirical part is rather boring to me here."
- 2012-11-08 algorithm (second draft) by co-author, discussion via e-mail
- 2012-11-08 **submission to workshop**
- 2012-12-07 **workshop notification**: "We regret to inform you that your paper could not be accepted for publication"

The Snapshot

The Submitted Version

The Submitted Version





Recap: Related Work

Related Work (Verwandte Arbeiten)

- quality of research depends knowing the state-of-the-art
- what related work exists and what is the relation to your work?
- evidence for claims on novelty, contributions
- #NoClassificationOfRW: paragraph for each paper ⇒ paragraph for each topic classifying several papers
- #MissingRW: paper omits related work
- #NoComparisonWithRW: missing relation to your work ⇒ end each paragraph with 1–3 sentences describing pros, cons, trade-offs, synergies, and unique characteristics
- #NoveltyOverPriorWorkUnclear: own prior work not cited or not discussed

Example Paragraphs [Knüppel 2017]

Formal Semantics of Feature Models. The idea of defining a general formal semantics to catch a variety of feature modeling dialects, and, thus, enhancing applicability of algorithms and research in general, is not new. Czarnecki et al. [13] proposed a cardinality-based notation to support their introduced concept of staged configurations. They proposed a similar formal semantics to the one we defined in this paper. Schobbens et al. [33] surveyed 12 feature modeling languages and based upon them proposed a general formal semantics called *Free Feature Model* [33]. They discussed properties such as expressive power, embeddability, and succinctness. All surveyed languages contain only simple constraints and the role of complex constraints is not discussed. Our work extends their prior theory and highlights the differences in expressive power with real numbers between basic feature models and feature models with complex constraints.

Eliminating Cross-Tree Constraints. Only little emphasis in the product-line community was put on the elimination of cross-tree constraints. Broek and Galvao [43] discussed the elimination of simple constraints by translating a feature model to a generalized feature tree, a structure allowing features to occur multiple times in different places including a potential negation. Our approach uses abstract features and simple constraints. Therefore, it is applicable for most tools requiring basic feature models as input. Gil et al. [17] aim to prove that all cross-tree constraints can be eliminated for the price of introducing a new set of features. Their approach is, however, only addressed from a theoretical perspective. Both approaches are beyond the goals we intend to accomplish, since some sort of constraints are typically supported in product-line research.

Mistakes with the Snapshot

#NoClassificationOfRW

5. RELATED WORK

Schobbens et al. surveyed 12 feature modeling notations and proposed a formal semantics for them [16]. They discussed syntactical inclusion, expressiveness, embeddability, and succinctness for most of these feature modeling notations. All surveyed notations contain only simple constraints. Some of these notations are expressively complete, because they consider feature diagrams as acyclic graphs, whereas we considered feature diagrams as trees. While they identified that some notations are not expressively complete, they do not discuss the relation to complex constraints.

She et al. proposed an algorithm to transform a propositional formula into a feature model [17]. Their focus is a heuristic that can calculate a parent relationship and groups that can compete with a manual transformation. They explicitly permit to define requires and excludes clauses *and* complex constraints. While their approach tries to make advantage of groups, requires and excludes clauses as much as possible, the authors acknowledge that some constraints may remain as additional constraints. Our analysis of publicly available feature models complements their approach by giving a feeling which percentage of constraints are simple constraints. Furthermore, we give examples for such constraints that cannot be expressed as simple constraints. However, we also found that expressing as much as possible in simple constraints can lead to many constraints and their approach may be improved by taking this into account.

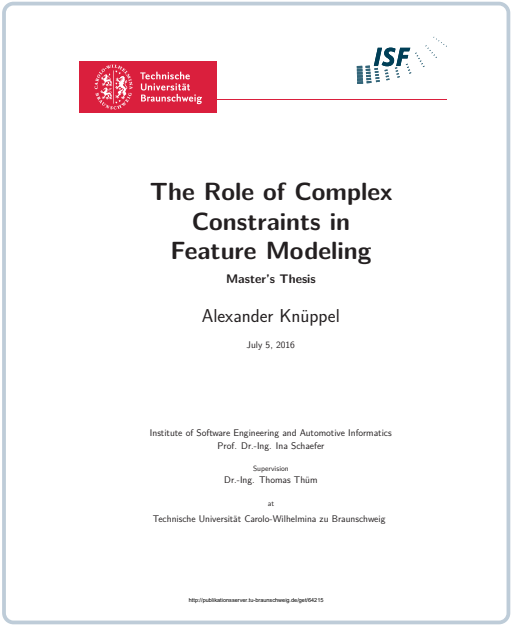
#MissingRW

Mendonca et al. focus on the generation of hard feature models to evaluate algorithms for automated analysis of feature models [12]. They analyzed existing feature models and found that feature models contain only binary and ternary cross-tree constraints. Their observation does not hold for the feature models subject to our analysis, in which we found a constraint containing 13 features. But it hold for most of our analyzed feature models. However, it is still not clear whether these binary and ternary constraints can be expressed using requires and excludes. Our analysis of existing feature models showed that many constraints cannot be expressed using requires and excludes.

Berger et al. compare the variability modeling languages Kconfig and CDL used in the operating system domain with feature modeling [5]. Cross-tree constraints beyond requires and excludes can be expressed with these languages, too. Additionally, features may not only have boolean values, but can also store numbers or strings. Consequently, cross-tree constraints can also be more complex than just propositional formulas. However, it is not clear how often such non-boolean types are needed in practice. We think the need for non-boolean types should be investigate in future to complement our results.

A Master's Thesis

A Master's Thesis



Technische Universität Braunschweig

ISF

The Role of Complex Constraints in Feature Modeling

Master's Thesis

Alexander Knüppel

July 5, 2016

Institute of Software Engineering and Automotive Informatics
Prof. Dr.-Ing. Ina Schaefer

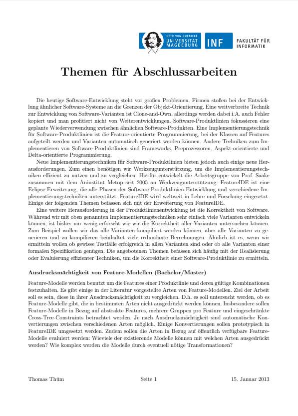
Supervision
Dr.-Ing. Thomas Thüm

at
Technische Universität Carolo-Wilhelmina zu Braunschweig

http://publikationsserver.tu-braunschweig.de/get/64215

A Master's Thesis

Published Thesis Topic in Magdeburg



Themen für Abschlussarbeiten

Die heutige Software-Entwicklung steht vor großen Problemen. Firmen stoßen bei der Entwicklung ähnlicher Software-Systeme an die Grenzen der Objekt-Orientierung. Eine weitverbreite Technik zur Entwicklung von Software-Varianten ist Clone-and-Own, allerdings werden dabei i.A. auch Fehler kopiert und man profitiert nicht von Weiterentwicklungen. Software-Produktlinien fokussieren eine geplante Wiederverwendung zwischen ähnlichen Software-Produkten. Eine Implementierungstechnik für Software-Produktlinien ist die Feature-orientierte Programmierung, bei der Klassen auf Features aufgeteilt werden und Varianten automatisch generiert werden können. Andere Techniken zum Implementieren von Software-Produktlinien sind Frameworks, Preprozessoren, Aspekt-orientierte und Delta-orientierte Programmierung.

Neue Implementierungstechniken für Software-Produktlinien bieten jedoch auch einige neue Herausforderungen. Zum einen benötigen wir Werkzeugunterstützung, um die Implementierungstechniken effizient zu nutzen und zu vergleichen. Hierfür entwickelt die Arbeitsgruppe von Prof. Saake zusammen mit dem Aninstitut Metop seit 2005 an Werkzeugunterstützung: FeatureIDE ist eine Eclipse-Erweiterung, die alle Phasen der Software-Produktlinien-Entwicklung und verschiedene Implementierungstechniken unterstützt. FeatureIDE wird weltweit in Lehre und Forschung eingesetzt. Einige der folgenden Themen befassen sich mit der Erweiterung von FeatureIDE.

Eine weitere Herausforderung in der Produktlinienentwicklung ist die Korrektheit von Software. Während wir mit oben genannten Implementierungstechniken sehr einfach viele Varianten entwickeln können, ist bisher nur wenig erforscht wie wir die Korrektheit aller Varianten untersuchen können. Zum Beispiel wollen wir das alle Varianten kompiliert werden können, aber alle Varianten zu generieren und zu kompilieren beinhaltet viele redundante Berechnungen. Ähnlich ist es, wenn wir ermitteln wollen ob gewisse Testfälle erfolgreich in allen Varianten sind oder ob alle Varianten einer formalen Spezifikation genügen. Die angebotenen Themen befassen sich häufig mit der Realisierung oder Evaluierung effizienter Techniken, um die Korrektheit einer Software-Produktlinie zu ermitteln.

Ausdrucksmächtigkeit von Feature-Modellen (Bachelor/Master)

Feature-Modelle werden benutzt um die Features einer Produktlinie und deren gültige Kombinationen festzuhalten. Es gibt einige in der Literatur vorgestellte Arten von Feature-Modellen. Ziel der Arbeit soll es sein, diese in ihrer Ausdrucksmächtigkeit zu vergleichen. D.h. es soll untersucht werden, ob es Feature-Modelle gibt, die in bestimmten Arten nicht ausgedrückt werden können. Insbesondere sollen Feature-Modelle in Bezug auf abstrakte Features, mehrere Gruppen pro Feature und eingeschränkte Cross-Tree-Constraints betrachtet werden. Je nach Ausdrucksmächtigkeit sind automatische Konvertierungen zwischen verschiedenen Arten möglich. Einige Konvertierungen sollen prototypisch in FeatureIDE umgesetzt werden. Zudem sollen die Arten in Bezug auf öffentlich verfügbare Feature-Modelle evaluiert werden: Wieviele der existierende Modelle können mit welchen Arten ausgedrückt werden? Wie komplex werden die Modelle durch eventuell nötige Transformationen?

Thomas Thüm Seite 1 15. Januar 2013

Timeline for a Thesis

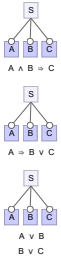
- 2014-06-16 student writes that he is interested in the thesis topic
- 2014-07-23 try to contact student again
- 2014-10-02 email discussion after conference meeting, proof by peer, counter-example found
- 2014-10-07 feedback on own proof by later co-author
- 2014-12-09 answer by student
- 2015-02-02 email discussion with peer
- 2015-02-10 presentation by student
- 2015-03-09 last response by student
- 2015-03-11 contacted student
- 2015-05-05 try to contact student again
- 2015-06-05 last try to contact student over student office

A Master's Thesis

Published Thesis Topic in Braunschweig

Ausdruckmächtigkeit von Feature-Modellen

- Problem: Feature-Modellierungssprachen unterstützen oft nur Requires ($A \Rightarrow B$) und Excludes ($\neg A \vee \neg B$) als Cross-Tree-Constraints
- Ziel: Algorithmus zum Überführen von beliebigen Constraints in FeatureIDE implementieren und evaluieren
- Beispiele rechts erfordern das Einführen von neuen Features
- Vorraussetzung: Aussagenlogik, Java
- Ansprechpartner: Thomas Thüm (`t.thuem@tu-bs.de`)
- Kooperation mit Jens Meinicke (Magdeburg)



Technische Universität Braunschweig | 1. April 2015 | ISF Promotion | Seite 1 | ISF

A Timeline for Knüppel 2016

- 2015-11-04 new student Alexander Knüppel agrees on topic
- 2015-11-17 proof
- 2015-11-23 proof (second draft)
- 2015-12-08 introduction
- 2016-01-04 background
- 2016-01-18 proposal — 13 comments
- 2016-01-18 slides
- 2016-01-31 slides (second draft)
- 2016-02-05 slides (third draft)
- 2016-02-08 background (second draft)
- 2016-02-11 background (third draft)
- 2016-03-01 background (fourth draft)
- 2016-03-22 theory (first draft)
- 2016-05-03 algorithms (first draft) — 159 comments
- 2016-06-01 algorithms (second draft)
- 2016-06-10 implementation (first draft)
- 2016-06-16 evaluation outline, proof
- 2016-06-21 evaluation (first draft)
- 2016-06-27 abstract, related work, conclusion, future work (first complete draft)
- 2016-06-29 second complete draft
- 2016-07-01 third complete draft
- 2016-07-04 **thesis submitted**
- 2016-07-07 slides (fourth draft)
- 2016-07-12 slides (fifth draft)
- 2016-08-22 thesis, minor editing for online version

A Master's Thesis

A Master's Thesis [Knüppel 2016]

A Master's Thesis [Knüppel 2016]

The Wristshot

The Wristshot (Der Schlenzer)

The Wristshot

Is There a Mismatch between Real-World Feature Models and Product-Line Research?

Alexander Knüppel
TU Braunschweig, Germany
a.knueppel@tu-bs.de

Thomas Thüm
TU Braunschweig, Germany
t.thuem@tu-bs.de

Stephan Mennicke
TU Braunschweig, Germany
mennicke@ips.cs.tu-bs.de

Jens Meinicke
University of Magdeburg, Germany
Carnegie Mellon University, USA
meinicke@ovgu.de

Ina Schaefer
TU Braunschweig, Germany
i.schaefer@tu-bs.de

ABSTRACT

Feature modeling has emerged as the de-facto standard to compactly capture the variability of a software product line. Multiple feature modeling languages have been proposed that evolved over the last decades to manage industrial-size product lines. However, less expressive languages, solely permitting require and exclude constraints, are permanently and carelessly used in product-line research. We address the problem whether those less expressive languages are sufficient for industrial product lines. We developed an algorithm to eliminate complex cross-tree constraints in a feature model, enabling the combination of tools and algorithms working with different feature model dialects in a plug-and-play manner. However, the scope of our algorithm is limited. Our evaluation on large feature models, including the Linux kernel, gives evidence that require and exclude constraints are not sufficient to express real-world feature models. Hence, we promote that research on feature models needs to consider arbitrary propositional formulas as cross-tree constraints prospectively.

CCS CONCEPTS

• **Software and its engineering** → **Feature interaction**; **Software product lines**;

KEYWORDS

Software product lines, feature modeling, cross-tree constraints, model transformation, expressiveness, require constraints, exclude constraints

ACM Reference Format:

Alexander Knüppel, Thomas Thüm, Stephan Mennicke, Jens Meinicke, and Ina Schaefer. 2017. Is There a Mismatch between Real-World Feature Models and Product-Line Research?. In *Proceedings of ESEC/FSE'17, Paderborn, Germany, September 04-08, 2017,* 12 pages.
https://doi.org/10.1145/3106237.3106252

ESEC/FSE'17, September 04-08, 2017, Paderborn, Germany
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5105-8/17/09…$15.00
https://doi.org/10.1145/3106237.3106252

1 INTRODUCTION

Software product-line engineering is a paradigm enabling mass customization of software [30]. Instead of developing a monolithic software product, the goal is to develop reusable software artifacts for a specific domain in a process called *domain engineering*. Multiple software artifacts composed together eventually result in a software product. A *software product line* is a family of similar software products sharing common artifacts. We distinguish between common and varying characteristics of products in terms of *features*. Features are user-visible aspects or characteristics of a software [22], being of interest for some stakeholders. Later, in a process called *application engineering*, a set of features is selected based on the requirements of stakeholders and a software product is derived.

The standard technique in research and industry to manage variability of a product line is *feature modeling* [12, 22]. Feature models offer an easy-to-understand formalism and unambiguously describe dependencies among features. In the context of product-line engineering, feature modeling is a valuable asset in several areas such as domain scoping [12, 22], feature-oriented software development [22, 24, 42], product-line analysis [18], and configuration management [40]. Our ten year experience with developing the open-source tool FeatureIDE [26] and integrating product-line tools is that a typical obstacle is the expressive power of different feature modeling dialects. Varying expressiveness in feature modeling languages prevents tool reuse and, thus, hinders efficient application of existing algorithms and concepts.

Over the last decades, several feature modeling languages, extending the initially proposed language by Kang et al. [22], have been suggested, either graphical [6, 12, 14, 18, 20, 23, 24, 31] or textual [2, 4, 5, 8, 10, 24, 28, 32, 44]. Ideally, given a set of features, a feature modeling language should be able to represent exactly the set of all valid feature combinations with respect to the requirements aquired during the domain engineering phase. A considerable portion of such languages, however, is not *expressively complete* (i.e., in theory, certain product lines cannot be represented). Although the restricted expressiveness was mentioned elsewhere [14, 17, 35, 37], an in-depth analysis of the problem for real-world feature models and a practical solution to overcome this limitation are still missing.

In particular, we identified several proposed methods dealing with feature models that are still based on expressively incomplete languages due to their simplicity and dominance in the product-line community. To name a few, the affected research areas include *automated analysis of feature models* [36], *synthesis of feature*

The Wristshot

A Timeline for Knüppel 2017

- 2016-09-02 first incomplete draft
- 2016-10-13 contact to other student on **industrial models**
- 2016-12-22 email discussion with peer on industrial models
- 2017-01-16 skype with peer about industrial models
- 2017-01-20 second incomplete draft
- 2017-01-30 update on external tool by peer
- 2017-02-01 extension of external tool by Alex
- 2017-02-13 **first complete draft**, 0.5 pages over page limit
- 2017-02-15 comments by co-author
- 2017-02-17 second complete draft, **blinded version**
- 2017-02-21 third complete draft
- 2017-02-22 comments by co-author
- 2017-02-24 comments by co-author
- 2017-02-26 **submission to A*-level conference**
- 2017-05-01 preliminary reviews
- 2017-05-03 submission of rebuttal (answer to reviews)
- 2017-06-02 **conference notification**: "We are pleased to inform you that your submission [...] was accepted"

The Mismatch Paper [Knüppel 2017]

Is There a Mismatch between Real-World Feature Models and Product-Line Research?

Alexander Knüppel
TU Braunschweig, Germany
a.knueppel@tu-bs.de

Thomas Thüm
TU Braunschweig, Germany
t.thuem@tu-bs.de

Stephan Mennicke
TU Braunschweig, Germany
mennicke@ips.cs.tu-bs.de

Jens Meinicke
University of Magdeburg, Germany
Carnegie Mellon University, USA
meinicke@ovgu.de

Ina Schaefer
TU Braunschweig, Germany
i.schaefer@tu-bs.de

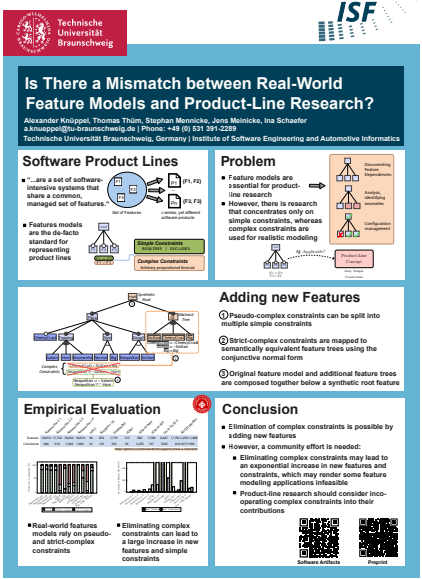
A Timeline for Knüppel 2017

- 2017-06-06 artifact description
- 2017-06-07 artifact description (second draft)
- 2017-06-10 **artifact submission**
- 2017-06-15 problems with question mark in title
- 2017-06-19 camera-ready copy (CRC, first draft), comments by co-author
- 2017-06-24 preliminary reviews on artifacts
- 2017-06-27 CRC (second draft)
- 2017-06-27 submission of rebuttal (answer to artifact reviews)
- 2017-07-03 **artifact notification**: "We are happy to accept your artifact [...] and award it an "Artifacts Evaluated – Reusable" badge!"

The Wristshot

A Timeline for Knüppel 2017

- 2017-07-03 CRC (third draft)
- 2017-07-05 CRC (fourth draft)
- 2017-07-08 **CRC submitted**
- 2017-07-13 CRC submitted (second time)
- 2017-07-17 CRC submitted (third time)
- 2017-07-26 **CRC approval**
- 2017-08-01 discussion on journal extension
- 2017-08-02 preprint published
- 2017-08-11 slides (first draft)
- 2017-08-21 instructions for talk
- 2017-08-28 **poster** (first and second draft)
- 2017-08-31 **practice presentation**
- 2017-09-04 slides (second draft)
- 2017-09-06 Alex gives **presentation in Paderborn** and presents the poster



The Wristshot

Talk at German Software Engineering Conference

(Hrsg.): Software Engineering und Software Management 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 1

Is There a Mismatch between Real-World Feature Models and Product-Line Research?

Alexander Knüppel[1] Thomas Thüm[2] Stephan Mennicke[3] Jens Meinicke[4] Ina Schaefer[5]

Abstract: This work has been presented at the joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering in Paderborn, Germany [Kn17].

Feature modeling has emerged as the de-facto standard to capture variability of a software product line in a compact and understandable fashion. Multiple feature modeling languages that evolved over the last decades to manage industrial-size product lines have been proposed. However, less expressive languages, solely permitting require and exclude constraints, are permanently and carelessly used in product-line research. We address the problem whether those less expressive languages are sufficient for industrial product lines. We developed an algorithm to eliminate complex cross-tree constraints in a feature model, enabling the combined usage of tools and algorithms working with different feature model dialects in a plug-and-play manner. However, the scope of our algorithm is limited. Our evaluation on large feature models, including the Linux kernel, gives evidence that require and exclude constraints are not sufficient to express real-world feature models. Hence, we promote that research on feature models needs to consider arbitrary propositional formulas as cross-tree constraints prospectively.

Keywords: Software product lines, feature modeling, cross-tree constraints, model transformation, expressiveness, require constraints, exclude constraints

Overview

In this talk, we discuss the role of arbitrary propositional formulas as cross-tree constraints in feature modeling. We argue that the two most prominent kinds of cross-tree constraints, namely *require* and *exclude constraints*, are not enough to capture product lines created for real-world projects. Unfortunately, even nowadays, part of product-line research gives the impression that those simple constraints suffice to describe industrial software product lines.

The main results on this mismatch between newly proposed product-line research on the one hand and feature models of industrial product lines on the other hand have been

[1] TU Braunschweig, Germany
[2] TU Braunschweig, Germany
[3] TU Braunschweig, Germany
[4] University of Magdeburg, Germany; Carnegie Mellon University, USA
[5] TU Braunschweig, Germany

2 Alexander Knüppel, Thomas Thüm, Stephan Mennicke, Jens Meinicke, Ina Schaefer

presented at the joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering in Paderborn, Germany [Kn17]. We showed that, to this day, a significant portion of novel contributions on feature models (e.g., algorithms or empirical evaluations) do not discuss complex constraints. The consequence is that applicability of these contributions on large-scale feature models remains questionable. Our evaluation on real-world feature models of considerable size portraits that simple constraints are not enough.

Although Schobbens et al. [Sc07] were the first to indicate that tree-based feature modeling languages with simple constraints are expressively incomplete, our work extends their prior theory and highlights the differences in expressive power with real numbers. To help the community to overcome this limitation, we developed a semantics-preserving transformation between languages with complex constraints and languages with simple constraints and gave proof of its correctness. The algorithm is based on the notion of abstract features [Th11]. Moreover, we successfully evaluated the benefits of this algorithm on a total of 127 real-world feature models: four monthly snapshots of an obfuscated automotive product line from our industrial partner with up to 18,616 features and 1,369 cross-tree constraints, eight models translated from the KConfig variability language (including the linux kernel), and 116 models translated from the component definition language (CDL).

On the upside, our algorithm is able to eliminate complex constraints in a feature model while preserving the represented set of products. The downside of our algorithm is that, for large feature models, our algorithm may render feature model applications (e.g., SAT analysis) infeasible due to a potential blow-up in the numbers of new features and constraints. However, the elimination of complex constraints is indispensable for practical product-line engineering. We thus advocate that product-line research should consider complex constraints as default in the future. We further think that a community effort is needed to evaluate which and how approaches tailored to basic feature models can be applied to complex constraints.

References

[Kn17] Knüppel, Alexander; Thüm, Thomas; Mennicke, Stephan; Meinicke, Jens; Schaefer, Ina: Is there a mismatch between real-world feature models and product-line research? In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. ACM, pp. 291–302, 2017.

[Sc07] Schobbens, Pierre-Yves; Heymans, Patrick; Trigaux, Jean-Christophe; Bontemps, Yves: Generic semantics of feature diagrams. Computer Networks, 51(2):456–479, 2007.

[Th11] Thum, Thomas; Kastner, Christian; Erdweg, Sebastian; Siegmund, Norbert: Abstract features in feature modeling. In: Software Product Line Conference (SPLC), 2011 15th International. IEEE, pp. 191–200, 2011.

A Timeline for Knüppel 2017

- 2017-09-20 idea to advertise paper in talk at German conference in Ulm
- 2017-09-22 **talk submission**
- 2017-11-16 **talk notification**: “we are delighted to let you know that your paper [...] has been selected for presentation”
- 2017-11-28 CRC draft
- 2017-12-13 CRC submission
- 2018-03-06 Thomas travels to Ulm
- 2018-03-07 Thomas gives **presentation in Ulm** including live demo on FeatureIDE
- 2019-03-08 (Thomas sends application to University of Ulm)

Stay Home Messages

Stay Home Messages

Research Takes Time

- August 2011: idea for a paper
- December 2012: first reject (+1.25y)
- June 2014: begin master's thesis (+1.5y)
- November 2015: new student (+1.5y)
- August 2016: thesis published (+0.75y)
- June 2017: paper accepted (+0.75y)
- September 2017: paper presented (+0.25y)
- March 2018: paper presented again (+0.5y)
- 6.5 years of research

Lessons Learned

- research takes time
- (scientific) writing requires reading
  #NoveltyUnclear #MissingRW
- related work matters
  #NoClassificationOfRW
- reproducibility matters
  #NotReproducible
- writing requires feedback and revisions
- welcome feedback, do revisions immediately
- writing requires role models