Teach Variability! A Modern University Course on Software Product Lines

Elias Kuiter
kuiter@ovgu.de
University of Magdeburg
Magdeburg, Germany

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

Timo Kehrer
timo.kehrer@inf.unibe.ch
University of Bern
Bern, Switzerland

Abstract

Teaching software product lines to university students is key in disseminating knowledge about software variability. In particular, education is needed to train new researchers and practitioners and, thus, sustain further research on software product lines. However, preparing appropriate teaching material is difficult and time-consuming, even when relying on existing literature. Thus, clone-and-own is a common practice among educators, with all its associated issues. Moreover, there is a lack of full-semester, open courses on software product lines. In this paper, we report on our experience of architecting and designing such a course from scratch, avoiding clone-and-own entirely. In addition, we perform a literature review of influential books on software product lines and which topics they cover. We position our course in terms of these topics, discuss how it compares to existing courses, and justify relevant design decisions. With our course, we aim to strengthen the positive interactions between research, industry, and education. So far, our course has already been held seven times across five universities. A preliminary evaluation of our course indicates that our course is mostly well-received by students.

CCS Concepts

• Social and professional topics → Software engineering education; **• Software and its engineering → Software product lines**.

Keywords

open educational resources, software product lines

ACM Reference Format:
Elias Kuiter, Thomas Thüm, and Timo Kehrer. 2025. Teach Variability! A Modern University Course on Software Product Lines. In *19th International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS 2025), February 04–06, 2025, Rennes, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3715340.3715441

1 Introduction

Software variability is ubiquitous, as many software systems are configurable [11, 12, 45, 65]. *Software product lines (SPLs)* [7, 20, 56, 67] are a well-known paradigm in research and practice [5, 10, 19, 26, 30, 54, 59] that apply systematic reuse to manage such variability.

VaMoS 2025, February 04–06, 2025, Rennes, France
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1441-2/25/02
https://doi.org/10.1145/3715340.3715441

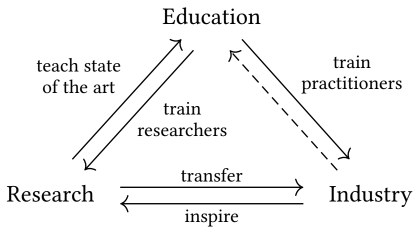


Figure 1: Interactions of research, industry, and education.

SPLs promise reduced costs for development and maintenance, faster time-to-market, and improved quality [7, 36, 40, 67].

Education at universities plays an important role in disseminating knowledge about SPL engineering [2]. First, it gives students the required tools to identify opportunities for beneficial software reuse in industry. Second, it is instrumental in teaching the next generation of SPL researchers. Third, education interacts positively with research, as students can participate in experiments and research projects, the results of which might then feed back into education. In Figure 1, we visualize these interactions between research, industry, and education as they can be observed in the context of SPLs. One arrow is less pronounced: This is because the industry typically has limited direct influence on education, besides occasional guest lectures. Thus, the responsibility to teach students appropriate skills mostly lies with educators (often researchers [2]), who need to balance the interests of both research and industry.

To achieve this balancing act, having appropriate teaching material is key. However, preparing such material is difficult and time-consuming, even when relying on existing books [2, 51]. Moreover, educators perceive that there is a lack of recognition for such efforts [60]. Thus, a common strategy among educators has been to use existing teaching material and only slightly adapt it [2] (e.g., by adding topics of their own). While such a clone-and-own [6, 24] strategy is tempting, over time it might lead to outdated or incorrect information, scope creep, and licensing issues. As long-time maintainers of clone-and-own teaching material, we can anecdotally confirm that such issues have repeatedly come up over the years.

Besides clone-and-own issues, creators of SPL teaching material seem to struggle with completeness and openness. To illustrate these problems, we collect SPL teaching material that is publicly available, and we show all complete English courses that we find in Table 1. In particular, this table is quite short with five entries, which is likely due to two reasons: First, much SPL teaching material only consists of one or two lectures in the context of a larger course (e.g., as a cross-cutting concern in a course on software engineering). While individual lectures can teach the basics [21, 52],

Table 1: Publicly available complete English courses on SPLs.

Authors	Year	University	Literature	Open?
Acher and Heymans[1]	2011	Namur	–	○
Kästner and Apel[2]	2015	Pittsburgh	[7]	○
Lopez-Herrejon and Rabiser[3]	2016	Linz	[20, 56, 67]	○
Donohoe and Northrop[4]	2020	Pittsburgh	[20]	◐
Gay and Berger[5]	2022	Gothenburg	[7, 67]	●
Thüm, Kehrer, and Kuiter[7]	2024	*(6 universities)*	[7, 50]	●

○ License unclear, no sources ◐ Open license ● Open license, sources available

We consider all material that is available in the online repository[6] of Acher et al. [3]. In addition, we consider material we found with the following Google search: `"software␣" ("product␣line" | "variability") ("␣course" | "␣slides")` We only include complete English courses that are mostly concerned with SPL topics. A more detailed version is available in our online appendix.[7]

we believe that full-semester courses on SPLs are necessary to positively contribute to the interactions depicted in Figure 1. Second, many courses on SPLs are never released publicly [18, 61] (e.g., due to clone-and-own issues). Even for publicly released courses, their license may be unclear or their sources not available, as we observe for some courses[1−3] in Table 1. This severely limits the material's potential for adaptation and reuse. While the next course[4] is openly licensed (CC-BY-4.0), its sources are not available, so no custom adaptations can be made. Indeed, we are only aware of one complete course on SPLs[5] with published and openly-licensed (CC-BY-SA-4.0) sources available. However, this course has only been released in PowerPoint format (originally Google Slides). While these tools allow for collaboration to some degree, they still invite problematic clone-and-own practices for performing adaptions, and therefore limits the course's reusability. We contacted an author of each course in Table 1 and asked them whether they are aware of their course being adapted to other universities (not considering affiliation changes of the course authors). We found that only one course[3] had been adapted to another university (i.e., Karlsruhe).

Evidently, the SPL community is missing a collaborative effort to create a new, modern full-semester university course on SPLs, which should be openly licensed and widely applicable. In this paper, we strive to fill this gap by reporting on our experience of architecting and designing such a course. Our aim is twofold: First, we want to avoid the above-mentioned problems of clone-and-own with existing courses. Second, we want to offer a modern SPL curriculum based on practical topics and recent research results. Thus, we decided to create a new course from scratch (i.e., proactively [41]). Since September 2022, our course has already been held seven times across five universities (i.e., in Bern, Ulm, Wernigerode, Magdeburg, and Paderborn). Currently, three new iterations of the course are ongoing, one of them at a new, sixth university (i.e., in Braunschweig) with more than one hundred course participants.

In particular, we contribute the following:

- We publish a course on SPLs with slides for 12 new 90-minute lectures, which we created and refined over the last three years.[7] Our slides are released under the permissive CC-BY-SA-4.0 license[8] and can be easily adapted to other universities with LaTeX. We also publish video recordings of each lecture under this license.[9]
- We describe and justify the scope and goals of our course (cf. Section 2), how we align its architecture (cf. Section 3) and lecture design (cf. Section 4) accordingly, and how we address several potential adoption challenges (cf. Section 5).
- We perform a literature review of topics covered in influential books on SPLs. Based on this review, we discuss how our course addresses each topic, and where we deviate from the books. Using our review, educators can choose suitable literature and avoid redundancies when they create new SPL teaching material. We also compare the contents of our course to existing courses (cf. Table 1), so educators can easily decide which material to use.
- We perform a preliminary evaluation of our course, which is based on feedback from 64 students across six teaching evaluations. We find that our course is mostly well-received by students and that it performs favorably in comparison to other courses.

With this, we build on a previous survey of SPL teaching practices by Acher et al. [2], who identified the need for such a curriculum.

2 Course Scope and Goals

What constitutes a modern course on SPLs? In the following, we define four goals ($\mathbf{G}_{1-4}$) that narrow the format and scope of our course. We justify why we deem the chosen goals to be reasonable premises to build our course on. We begin with generic goals and get more specific, such that each goal is derived from previous ones.

$\mathbf{G}_1$ Connect Research, Industry, and Education As we show in Figure 1, education plays a key role in connecting research and industry. To support and strengthen this connection, we aim to:

- describe the state of the art, prioritizing recent sources [18]
- include insights from recent research, pointers for further reading, and references for claims
- discuss open challenges, which lay a foundation for lectures on research (to train researchers) or industry (to train practitioners)

As we are researchers, our perspective is bound to be rather research-oriented. While we largely embrace this perspective, we also aim to recount experiences from industry for motivation (e.g., based on our collaborations with industry partners).

$\mathbf{G}_2$ Invite Contributions We aim to release our course in form of *open educational resources.* UNESCO [66] defines these as "learning, teaching and research materials in any format and medium that reside in the public domain or are under copyright that have been released under an open license, that permit no-cost access, re-use, re-purpose, adaptation and redistribution by others." This promises several benefits over a non-distributed, closed-source course [51]:

- It enables students to participate who are not enrolled at a university or whose university does not offer a course on SPLs.

[1]http://teaching.variability.io/namur.html
[2]http://www.cs.cmu.edu/~ckaestne/17708
[3]http://teaching.variability.io/jku2016.html
[4]https://insights.sei.cmu.edu/library/introduction-to-software-product-lines-course
[5]https://greg4cr.github.io/courses/fall22tda594
[6]https://teaching.variability.io

[7]Artifact with online appendix: https://doi.org/10.5281/zenodo.14417094
Repository: https://github.com/SoftVarE-Group/Course-on-Software-Product-Lines
[8]https://creativecommons.org/licenses/by-sa/4.0/
[9]https://www.youtube.com/playlist?list=PL4hJhdKDPIxha8So7muX2zfNUU8NBoiu3

- It invites fellow educators and researchers to reuse and adapt our course or even make new contributions (e.g., new lectures).
- It allows practitioners to teach SPL concepts to industrial stakeholders (e.g., domain experts) without any licensing issues.
- It holds us and other contributors publicly accountable, creating an incentive to commit high-quality and up-to-date material.

Moreover, each of these benefits contributes to goal $\mathbf{G}_1$ by either making our course more attractive to some party in Figure 1 or establishing accountability and, thus, confidence in the material.

$\mathbf{G}_3$ Address a Broad Audience Preparing new teaching material is a difficult and time-consuming endeavor. Consequently, it makes sense for us to try to address a broad audience of students, so our course is widely applicable in practice. Thus, we aim to:

- choose a course format that fits a typical university schedule [2]
- apply modern teaching methods that attract students [18]
- architect an inductive course structure that emphasizes the natural discovery of concepts over their definitions [57]

We believe these efforts will help to attract educators, making it way easier to teach SPLs at more universities (as per $\mathbf{G}_2$).

$\mathbf{G}_4$ Focus on Practical Skills SPLs have many facets [2, 59], not all of which can be realistically taught in a single university course. Consequently, we must choose a particular subset of topics and skills that is both likely to be relevant for training new researchers and practitioners (as per $\mathbf{G}_1$) and to appeal to students (as per $\mathbf{G}_3$). For this reason, we opt for a practical, hands-on approach and focus mostly on topics like modeling, implementation, and analysis of variability. In particular, we put less emphasis on management and organizational topics. While these topics are no less important, they are also more abstract and difficult to grasp for students, which typically have limited industrial experience. Overall, we therefore aim to create a rather technical course that can be accompanied by a suitable exercise class with, for example, programming tasks.

Building on goal $\mathbf{G}_1$–$\mathbf{G}_4$, we describe and justify the high-level (cf. Section 3) and low-level (cf. Section 4) design of our course.

3 Course Architecture

Initially, the authors of this paper spent $\approx$ 11 months only discussing the course format and structure, and which SPL topics to cover in which depth. In the following, we briefly describe the relevant design choices we made, which concern our course as a whole.

3.1 Format

Our course consists of 12 English lectures ($\mathbf{L}_{1-12}$), with $\approx$ 40 slides each. This format fits a typical weekly (under-)graduate university course of 3–4 months with 90-minute lectures. We create an entire series of lectures to close the existing gap in complete, open courses on SPLs (cf. Table 1). We use the English language and an established course format for improved applicability ($\mathbf{G}_3$). The schedule with 12 lectures leaves enough room for guest lectures ($\mathbf{G}_1$). For example, we typically host one additional lecture with conference talks as well as an industrial guest lecture (e.g., by pure-systems GmbH).

3.2 Literature Review

Similar to other courses [2] (e.g., those shown in Table 1), we loosely base our lecture slides on existing books. We do this for several reasons: First, it saves time and effort, as several books already prepare a curriculum with a well-laid-out common thread. Second, given such a curriculum, we can decide more easily where to deviate from it (cf. Section 4), for example to include recent research or correct outdated information ($\mathbf{G}_1$). Third, referring to books gives interested students material for further reading ($\mathbf{G}_1$).

While searching for appropriate literature (as per $\mathbf{G}_4$), we became aware that a comprehensive overview of educational books on SPLs is missing. Moreover, we are not aware of any detailed review of topics that are relevant for teaching SPLs, and how books cover these topics. To fill this gap, we perform a literature review of influential (i.e., well-known and often-cited) books on SPLs. We also review relevant SPL topics and the degree to which each book covers them. We show both the methodology for this review and its results in Table 2. A more in-depth version is available in our online appendix,[7] which also includes justifications and a list of excluded books. Our literature review serves as a first reference to help educators (such as ourselves) choose suitable literature for new courses and pointers for further reading.

For our course, we choose Apel et al. [7] and Meinicke et al. [50] as accompanying literature, due to several reasons: First, both books are well-known and comparably up-to-date ($\mathbf{G}_1$) as well as mostly oriented towards practical topics ($\mathbf{G}_3$). Second, they include both theoretical [7] and practical [50] exercises ($\mathbf{G}_3$) and are likely accessible to university students (e.g., via SpringerLink). The practical exercises, in particular, rely on the tool FeatureIDE [38], which is free and open-source software ($\mathbf{G}_2$) and already used in teaching [2]. Third, our literature review in Table 2 shows that of all considered books, the chosen two focus most on our topics of interest, such as variability modeling, implementation, and analysis ($\mathbf{G}_4$).

3.3 Structure

We divide our course into three parts, which we show on the top in Figure 2. In Part I ($\mathbf{L}_{1-3}$), we begin with ad-hoc approaches for software variability, which many students will be intuitively familiar with. Then, in Part II ($\mathbf{L}_{4-8}$), we introduce feature modeling [7, 34], implementation techniques [7, 64], and the development process [7, 56]. Finally, we discuss measures for quality assurance and advanced topics in Part III ($\mathbf{L}_{9-12}$). This structure is inductive (i.e., it builds concepts up from basic principles, $\mathbf{G}_3$) and emphasizes practicality ($\mathbf{G}_4$). In particular, we introduce feature models (in $\mathbf{L}_4$) and the SPL development process (in $\mathbf{L}_8$) much later than Apel et al. [7], when students already know how to apply them.

4 Lecture Design

After settling on the course architecture (cf. Section 3), we spent four months creating initial versions of most lectures. Below, we discuss the detailed lecture design, relating it to our goals (cf. Section 2).

4.1 Structure

We divide each lecture into three blocks, which we show exemplary for the introduction ($\mathbf{L}_1$) on the bottom in Figure 2. Each block is designed to take 20–25 minutes of lecturing time, followed by an optional interaction with the audience of 5–10 minutes. In each block, we describe and discuss one distinct and cohesive topic related to SPLs. This discussion is then concluded by a summary of

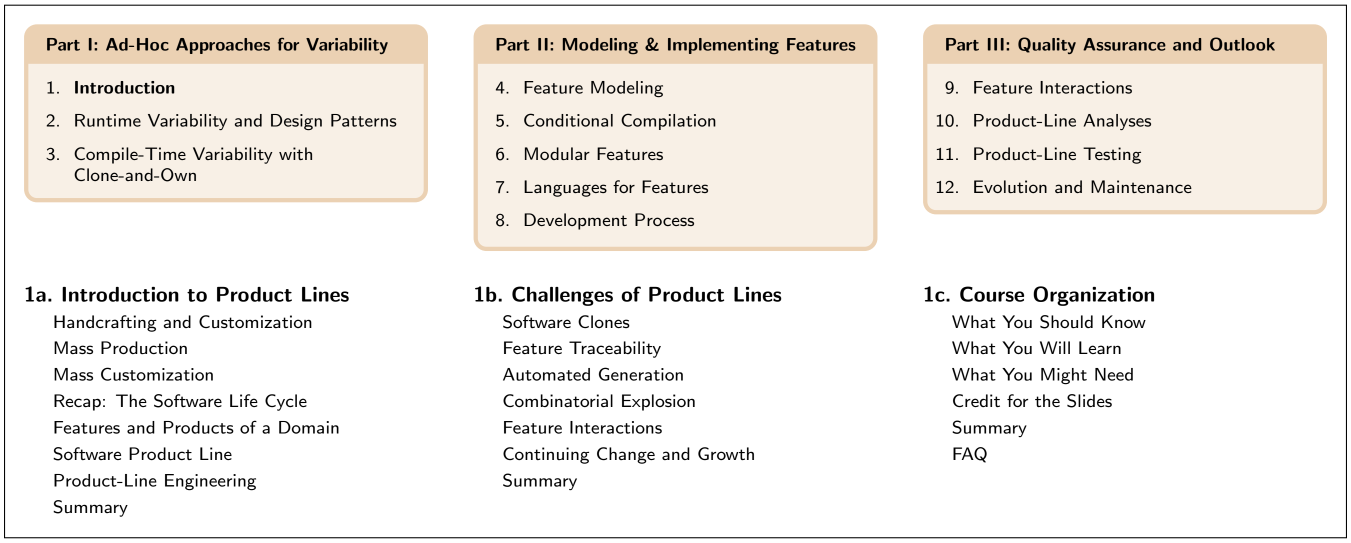
Table 2: Literature review of existing books and courses on SPLs, which topics they cover, and how they compare to our course.

Topics	**Books**								**Courses**					
This list of topics is based on the books' and courses' tables of contents, the session titles of recent conferences (SPLC [8], VaMoS [35], and FOSD), and also on related work [2, 59]. In line with our practical focus (**G**$_4$), we break down topics related to modeling, implementation, and analysis in more detail and put less emphasis on management issues.	Czarnecki and Eisenecker [23]	Bosch [15]	Clements and Northrop [20]	Gomaa [31]	Pohl, Böckle, and van der Linden [56]	van der Linden, Schmid and Rommes [67]	Apel, Batory, Kästner, and Saake [7]	Meinicke, Thüm, Schröter, Benduhn, Leich, and Saake [50]	Acher and Heymans[1]	Kästner and Apel[2]	Lopez-Herrejon and Rabiser[3]	Donohoe and Northrop[4]	Gay and Berger[5]	Thüm, Kehrer, and Kuiter[7]
Fundamentals														
Motivation, Goals, Context, History	●	●	●	●	●	●	●	●	●	●	●	●	●	●
SPL Definition, Delineation	◐	●	●	●	●	●	●	◐	●	●	●	●	●	●
SPL Engineering	○	●	●	●	●	●	●	◐	●	●	●	●	●	●
Modeling and Configuration														
Feature Modeling	●	○	◐	●	●	◐	●	●	●	◐	●	○	●	●
Decision Modeling	○	○	○	○	○	◐	○	○	○	◐	●	○	○	○
Product Configuration	○	●	●	●	◐	●	●	●	●	◐	●	●	●	●
Requirements Engineering	○	●	●	●	●	●	◐	○	○	●	◐	●	◐	◐
Scoping, Variability Reduction	●	●	●	●	●	●	◐	○	◐	●	●	●	◐	◐
Variability-Model Representations, Transformations	○	○	○	◐	○	○	●	◐	●	◐	◐	○	◐	●
Model-Driven Engineering, Domain-Specific Languages	●	○	○	◐	◐	◐	○	○	●	●	◐	○	○	○
Design and Implementation														
Product Derivation, Automation, Generative Programming	●	◐	◐	○	◐	◐	●	●	●	●	●	◐	●	●
Feature Mapping, Traceability, Location	●	○	○	○	◐	◐	◐	◐	◐	◐	◐	○	◐	◐
Runtime Variability, Design Patterns	●	◐	◐	●	◐	◐	●	●	◐	●	◐	◐	●	●
Clone-and-Own, Version Control Systems	○	◐	◐	○	○	○	◐	◐	○	●	◐	○	○	●
Preprocessors	◐	○	○	◐	◐	◐	●	●	◐	●	◐	◐	●	●
Build Systems	○	○	○	○	◐	◐	◐	○	○	◐	○	○	◐	●
Components, Services	◐	◐	◐	●	◐	◐	●	◐	○	◐	○	◐	●	●
Frameworks, Plug-ins	◐	●	○	○	●	◐	●	●	◐	●	◐	○	●	●
Feature-Oriented Programming	●	○	○	○	○	○	●	●	○	●	●	○	●	●
Aspect-Oriented Programming, Cross-Cutting Concerns	●	◐	◐	○	◐	○	●	●	○	●	●	◐	●	●
Quality Assurance														
Feature-Model Analysis, Satisfiability Solving, Model Counting	○	○	○	○	○	○	◐	◐	●	◐	◐	○	◐	●
Feature-Mapping Analysis, Presence Conditions	○	○	○	○	○	○	●	●	○	●	○	○	○	●
Solution-Space Analysis	○	○	○	○	○	○	●	◐	○	●	○	○	○	●
Feature Interactions	○	○	○	○	○	○	●	◐	○	●	○	○	◐	●
Testing, Sampling	◐	◐	◐	◐	◐	◐	◐	◐	○	●	●	◐	●	●
Formal Methods, Theory, Algebra	●	○	○	○	○	○	◐	○	◐	○	○	○	○	○
Validation, Certification, Specification	◐	◐	○	○	◐	○	○	◐	○	◐	○	○	○	○
Management														
Process Models, Development Life Cycle	●	●	●	●	●	●	◐	◐	◐	◐	◐	●	◐	◐
Organization, Roles, Business Cases	○	●	●	○	●	●	○	○	○	○	○	●	○	○
Financing, Economics, Cost Estimation	○	◐	●	○	●	●	○	○	○	○	○	●	○	○
Human Factors, Cognition, Knowledge	○	○	◐	○	○	○	○	○	○	○	○	◐	○	○
Transfer														
Adoption Strategies	○	●	◐	●	●	●	●	○	●	●	○	◐	○	●
Reengineering, Reverse Engineering, Refactoring	◐	○	○	◐	○	◐	◐	◐	◐	◐	●	○	○	◐
Evolution, Maintenance	○	●	○	●	◐	●	◐	◐	○	◐	●	○	◐	●
Release, Deployment, Operation	○	◐	◐	◐	○	◐	○	○	○	○	○	◐	○	○
Case Studies, Industrial Applications	◐	●	●	●	●	●	◐	◐	◐	◐	●	●	◐	◐
Tool Support	○	○	◐	○	○	○	●	●	◐	●	●	○	●	●
Exercises, Instructions, Training	○	○	●	○	◐	○	●	●	●	●	●	◐	●	●
Miscellaneous														
Multi SPLs, Software Ecosystems	○	○	○	○	◐	○	◐	○	○	◐	○	○	○	○
Dynamic SPLs, Adaptive Systems	◐	○	○	○	○	○	◐	○	○	○	○	○	○	◐
Computational Complexity, Limits, Combinatorial Explosion	○	○	○	○	◐	◐	○	○	◐	◐	◐	○	○	●

○ Not or barely mentioned ◐ Discussed partially or superficially ● Discussed in breadth or depth

Remark on Included Books We aim to include books that are concerned with SPLs, well-known in research or industry, and suitable for teaching (e.g., aimed at students or creators of educational resources). To this end, we consider SPL books in our inventory and our literature databases. In addition, we search Google, Amazon, eBay, and Google Scholar for the phrases: `"software␣" ("product␣line" | "variability") "␣book"`. We only include English books with available full-text. We exclude conference proceedings, dissertations, edited anthologies or collections, and books with less than 100 citations (according to Google Scholar). A more detailed version is available in our online appendix.[7]

Remark on Included Courses We include all publicly available complete English courses on SPLs that we identify in Table 1.


Part I: Ad-Hoc Approaches for Variability
1. **Introduction**
2. Runtime Variability and Design Patterns
3. Compile-Time Variability with Clone-and-Own

Part II: Modeling & Implementing Features
4. Feature Modeling
5. Conditional Compilation
6. Modular Features
7. Languages for Features
8. Development Process

Part III: Quality Assurance and Outlook
9. Feature Interactions
10. Product-Line Analyses
11. Product-Line Testing
12. Evolution and Maintenance

1a. Introduction to Product Lines
Handcrafting and Customization
Mass Production
Mass Customization
Recap: The Software Life Cycle
Features and Products of a Domain
Software Product Line
Product-Line Engineering
Summary

1b. Challenges of Product Lines
Software Clones
Feature Traceability
Automated Generation
Combinatorial Explosion
Feature Interactions
Continuing Change and Growth
Summary

1c. Course Organization
What You Should Know
What You Will Learn
What You Might Need
Credit for the Slides
Summary
FAQ

Figure 2: The first handout slide of our course on software product lines. We show the overall structure of our course on the top (including all 12 lectures) and the structure of the individual lecture on the bottom (in this case, the introductory lecture).

learned lessons, selected opportunities for further reading ($\mathbf{G}_1$), and an optional exercise that students can discuss in groups ($\mathbf{G}_3$). This three-block structure has several advantages: First, it allows educators and students to easily navigate the lecture slides. Second, it encourages students to pay attention, so they are able to participate in the interaction. This interaction can then immediately reinforce the learned concepts ($\mathbf{G}_3$). Third, it implements the *sandwich principle* [33], which mandates alternating periods of active listening and audience participation [16]. Recent evidence suggests that this principle may improve students' critical thinking and self-learning ability as well as their satisfaction with the course [13, 17].

Inside each block, we typically follow an inductive structure ($\mathbf{G}_3$), just as with the course at large. For instance, in $\mathbf{L}_{1a}$ (cf. Figure 2), we begin with examples of customization and slowly build up towards the concept of an SPL, finally asking the students to name their own examples of SPLs as part of the interaction. In addition, we occasionally use recaps to recapitulate a previous lecture, and even memes (e.g., XKCD comics)[10] to keep students engaged ($\mathbf{G}_3$). Where applicable, we also refer to relevant research publications in order to substantiate our claims and animate students to make themselves familiar with the literature ($\mathbf{G}_1$). Finally, we conclude each lecture with a list of frequently asked questions (FAQs), which students can use to check their grasp on each block's topic.

4.2 Topics

In Table 2 (in the column *our course* on the far right), we give an overview of the concrete topics that we cover in our course. To facilitate a comparison with existing courses (cf. Table 1), we also extend the literature review in Table 2 with an overview of the topics covered in these five courses. For instance, we can use this overview for a deeper comparison of our course to the only other complete, open course created by Gay and Berger.[5] This comparison shows that our course covers both more topics (e.g., clone-and-own) and several topics in more depth or breadth (e.g., feature-model analysis). Our extended review also shows how each course (including our own) relates to the topics covered in influential books on SPLs (cf. Section 3). As our course is loosely based on Apel et al. [7] and Meinicke et al. [50], it mostly covers similar topics as these books. However, we sometimes deviate from both books, for instance by reordering, omitting, or adding topics. In the following, we discuss some notable changes we make compared to these books, focusing on practical challenges, running examples, added topics, and original contributions.

Challenges In the first lecture ($\mathbf{L}_{1a}$), we name some promises of SPLs (e.g., cost reduction). However, we also want to ensure that students are immediately confronted with some potential drawbacks of SPLs ($\mathbf{G}_1$). Thus, in $\mathbf{L}_{1b}$, we identify six practical challenges of SPLs that relate to the topics of our course ($\mathbf{G}_4$). These challenges include (cf. Figure 2): software clones ($\mathbf{L}_{2b,3,6a}$), feature traceability ($\mathbf{L}_{2b,5c,6,7}$), automated generation ($\mathbf{L}_{2,5,6c,7}$), combinatorial explosion ($\mathbf{L}_{2c,3a,4,10,11}$), feature interactions ($\mathbf{L}_{9,10a,11b}$), and continuing change and growth ($\mathbf{L}_{8,12}$). Besides raising awareness, having these challenges serves as a common thread throughout the course ($\mathbf{G}_3$).

Examples Analogously to these challenges, we use several running examples throughout the course ($\mathbf{G}_3$). Some of these examples include: the graph product line (GPL) [47, 69] ($\mathbf{L}_{2,3,5-10}$), the Linux kernel [65] ($\mathbf{L}_{1,5,8,10-12}$), and the PigNap case study [43] ($\mathbf{L}_5$). Besides these examples from research and industry, we also contribute our own examples for configuring databases ($\mathbf{L}_{4,10,11}$), ordering waffles ($\mathbf{L}_4$), and assembling Lego minifigures ($\mathbf{L}_{1,10}$). Having such recurring examples, students can easily recognize and use them, for example to compare implementation techniques ($\mathbf{G}_3$).

Additions To account for recent research insights ($\mathbf{G}_1$), we include several practical topics in our course ($\mathbf{G}_4$), none of which is covered by any book in Table 2. Some of these topics include: the

[10]https://xkcd.com/

universal variability language (UVL) [27, 62] ($\mathbf{L}_{4b}$), model counting and enumeration [29, 63] ($\mathbf{L}_{4c}$), the KConfig language and tooling [22, 25, 53] ($\mathbf{L}_{5a}$), microservices [9, 55] ($\mathbf{L}_{6b}$), the promote-pl development process [41] ($\mathbf{L}_{8c}$), combinatorial interaction testing [4, 48] ($\mathbf{L}_{11b}$), and solution-space sampling [39, 68] ($\mathbf{L}_{11c}$).

Contributions Furthermore, we even make some original contributions in the preparation of this course. For example, we compute recent statistics regarding the evolution of the Linux kernel ($\mathbf{L}_{1b,8a}$), we distinguish build systems for clone-and-own and conditional compilation ($\mathbf{L}_{3c,5a}$), and we critically reflect on the complexity of SPLs and feature models ($\mathbf{L}_{4c,10c}$) [42]. These contributions demonstrate how education can positively interact with research ($\mathbf{G}_1$).

5 Adoption Challenges

In the following, we discuss several challenges related to the practical adoption of our course and how we address them.

Version Control We publish our course and its entire history [18] in a public Git repository[7] with > 600 commits by 7 contributors. Besides easing backup and distribution, this ensures transparency and, thus, accountability for any contributions made by us and others ($\mathbf{G}_2$). Potential concerns can be raised in our issue tracker.

No Clone-and-Own We proactively create a new course on SPLs to avoid typical issues of clone-and-own (cf. Section 1). To prevent such issues in the future as well, we discourage diverging forks by inviting other educators to integrate their contributions into the main Git repository, for example with pull requests ($\mathbf{G}_2$). This is possible because we create all our slides with LaTeX, which allows for textual diffs that are easy to review and merge (e.g., compared to PowerPoint or similar visually oriented tools).

Customization By relying on LaTeX, we can easily adapt our slides to new universities ($\mathbf{G}_2$) and satisfy needs for further customization (e.g., handout and dark-mode slides). To this end, we currently use an annotative technique (i.e., the `\ifuniversity{}` command, a LaTeX equivalent of C's `#ifdef` [44]). However, we almost exclusively use this technique for customizing appearance in order to avoid unneeded variability [1]. In particular, we deliberately avoid creating an "SPL of SPL courses" [3], as this would introduce a significant amount of additional complexity, which might be a source for inconsistencies and obstacle for future extensions. Using this technique, we adapted our course to six universities (cf. Table 3).

Underrepresented Topics With our course, we attempt to flesh out the SPL baseline curriculum proposed by Acher et al. [2]. However, we are aware that some particular SPL topics (cf. Table 2) are currently underrepresented in our curriculum. Management topics, in particular, are covered much more extensively by Donohoe and Northrop.[4] We envision two ways to close such gaps: First, educators with the respective expertise may contribute additional guest lectures on these topics to our course. Second, we encourage and look forward to other educators creating completely different courses on SPLs, in which these topics may be covered in more depth. To this end, the results of our literature review (cf. Table 2) can be used to identify gaps for future books and courses on SPLs.

Exercise Sheets We currently accompany the lecture with an exercise class, in which students solve and discuss in-depth exercises on each lecture's topic. Because our primary focus is on the lecture slides, we currently use work-in-progress exercise sheets based on previous courses. In the near future, we aim to revise these exercise sheets to match well with the lecture slides, so we can release them.

6 Preliminary Evaluation

Creating new lecture slides from scratch is not an easy task, but getting them actually adopted in practice is yet another challenge. In order to investigate whether our course is worth adopting, we aim to evaluate how it is received by our students with regard to several quality criteria. Thus, we seek to complement the discussion of our intentions (cf. Section 2) and their implementation (cf. Section 3 and 4) with real feedback from students. In particular, we aim to address the following research questions:

$\mathbf{RQ_1}$ How well do students receive our course in general?
$\mathbf{RQ_2}$ How well do students receive our course compared to …
  $\mathbf{RQ_{2.1}}$ … other courses at the same faculty?
  $\mathbf{RQ_{2.2}}$ … a previous course on SPLs at the same faculty?

Performing comprehensive, meaningful evaluations of university courses is known to be methodologically challenging [28, 37, 49]. Nonetheless, we collect feedback from several *student evaluations of teaching (SETs)* [32], which are conducted at many universities. We believe this feedback is a valuable first step towards assessing the quality of our course and whether it is suitable for adoption.

6.1 $\mathbf{RQ_1}$: General Reception

First, we aim to determine how students receive our course in general (e.g., aspects they like or dislike, and room for improvement).

Methodology Over the last two years, we have conducted six SETs at five universities, in which we collected anonymized feedback from 64 students. The feedback is both quantitative and qualitative, summarized in Table 3 and 4, respectively. For quantitative feedback, all universities use some kind of Likert scale to indicate (dis-)agreement. Still, we need to unify the questionnaires used by different universities. Thus, we aggregate relevant questions into a set of recurring quality criteria, and we normalize all scores to the same scale (details indicated in Table 3). We leave gaps in Table 3 when a SET does not cover a given quality criterion at all.

Results The results we show for RQ_1 in Table 3 indicate scores from 78.3 to 97.0 points for course-related quality criteria. In the mean (computed over all universities), students rate the structure of our course with 88.5, its quality with 88.9, and its difficulty and pacing with 91.7 out of 100 points. As for the self-assessment of our students, they rate their own motivation for the course with 69.3 points, their gain in knowledge with 83.7, and their overall satisfaction with 85.4 points in the mean. To put these numbers into context, we also show aggregated qualitative student feedback in Table 4, which reveals several points of praise and criticism.

Discussion The feedback suggests that our course was well-received by the majority of students. In particular, students seem to appreciate the structure of our course (i.e., as outlined in Section 3 and 4), its material (i.e., the lecture slides and exercise sheets), and its difficulty and pacing (i.e., it is not too hard/fast and not too easy/slow). However, it is of course not possible to satisfy everyone—for instance, there will always be students who are challenged too

Table 3: Student feedback from Likert-scale questions (RQ_1, $RQ_{2.1}$, $RQ_{2.2}$).

	University of Bern				**Ulm University**			**Harz**	**Magdeburg**		**Paderborn**	**Aggregated**	
	WT 2022/23		**WT 2023/24**		**ST 2023**			**ST 2023**	**WT 2023/24**		**ST 2024**	Total or Mean	
Quality Criteria	RQ_1	$RQ_{2.1}$	RQ_1	$RQ_{2.1}$	RQ_1	$RQ_{2.1}$	$RQ_{2.2}$	RQ_1	RQ_1	$RQ_{2.1}$	RQ_1	RQ_1	$RQ_{2.1}$
# Participants	8	4110	8	2608	13	856	9	8	11	471	16	64	8045
Course													
Structure	90	76 +19% .00∗	90	80 +13% .03∗	95	81 +17% .00∗	91 +4% .37		88	80 +10% .24	80	89	79 +15%
Material	88	73 +20% .03∗	90	80 +13% .10	93	80 +17% .00∗	85 +9% .25				85	89	77 +17%
Difficulty, Pacing	97	82 +18% .11	97	91 +6% .01∗	91	78 +17% .01∗	87 +5% .46		78	75 +4% .67	95	92	82 +11%
Self-Assessment													
Motivation	75	68 +11% .43	79	70 +12% .08	86	77 +12% .05	86 +0% .99	44	63	73 −14% .19		69	72 +5%
Gain in Knowledge	86	74 +16% .05∗	89	77 +16% .02∗	91	74 +23% .00∗	90 +2% .80		73	70 +5% .65	79	84	74 +15%
Satisfaction	94	78 +21% .00∗	94	82 +14% .02∗	96	80 +20% .00∗	90 +7% .27	75	73	70 +5% .60	80	85	77 +15%

Harz University of Applied Studies (Wernigerode) University of **Magdeburg** **Paderborn** University TU Braunschweig (not yet evaluated)

WT Winter term **ST** Summer term **Score Scale** 0 = worst, 100 = best $+x\%$ $-x\%$ Comparison relative to RQ_1 p ∗ Statistically significant ($p < 0.05$)

Quantitative feedback aggregated from six questionnaires across five universities with 64 student participants in total. We exclude one unevaluated and three ongoing iterations of our course. We identify six key categories of quality criteria that most questionnaires cover. We calculate each score as the mean of the constituent questions, normalized to a scale from 0 to 100 for easy comparison. We omit questions that only relate to the exercises and the performance of lecturers. A more detailed version is available in our online appendix.[7]

Table 4: Student feedback from open questions (RQ_1).

Praise
Detailed course structure with a clear overview and a common thread
Many images with good visualizations, appropriate information depth
Many practical, recurring examples
Interactions, FAQs, opportunities for asking questions and discussions
Video recordings, dark-mode slides, guest lectures
Criticism
Having English slides was challenging for some non-native speakers
Too theoretical for some students, too practical and coding-focused for others
Some lectures are too long (e.g., feature modeling) and too fast towards the end

Qualitative feedback aggregated from the same responses that we consider in Table 3.

much or too little by the material. Notably, the student's own motivation seems comparably low in the mean, which is due to the low rating at the Harz University of Applied Studies. This is a vocational university with less emphasis on research, which might explain the lower self-assessment at this university. Another point of criticism we aim to address is that, despite the good overall pacing, some individual lectures have too much content and could be tightened.

6.2 $RQ_{2.1}$: Comparison to Faculty

While RQ_1 indicates generally positive feedback, the results for our course become more meaningful when we compare them to a suitable baseline. For $RQ_{2.1}$, we aim to compare our course to other courses that have been held at the same university.

Methodology Four of the six SETs we conducted (i.e., in Bern, Ulm, and Magdeburg) also include data on many other courses that have been evaluated at the same faculty. By aggregating and normalizing the other courses' scores analogously to RQ_1, we can compare them to our course and assess where it performs better or worse. We show these results in Table 3 in the column for $RQ_{2.1}$, including the differences to our course's results as percentages. To calculate these percentages, we only consider the universities for which data on other courses is available. This way, the all-university percentage in the last column of Table 3 is computed correctly, although the absolute mean values for RQ_1 and $RQ_{2.1}$ are not directly comparable. To test the differences' statistical significance, we perform a two-tailed Welch's t-test with a significance level of $\alpha = 0.05$. This test is more robust than Student's t-test for unequal variances and unequal sample sizes, which apply here. We indicate p-values and significant results ($*$) in Table 3 and omit variances for brevity.

Results The results for $RQ_{2.1}$ show that, in the mean, students rate our course as better than other courses at the same faculty in almost all quality criteria. In particular, our course scores 11% to 17% more points than other courses in course-related quality criteria in the mean. Regarding self-assessment, we find that, in the mean, students award our course 5% to 15% more points than other courses. Only at the University of Magdeburg do we find a mean reduction in motivation by 14% compared to other courses. Regarding statistical significance, we find that for almost all quality criteria, the differences are significant for at least two distinct universities. Again, the students' motivation is the single exception, having no statistically significant differences ($0.05 \leq p \leq 0.43$).

Discussion These results suggest that students generally regard our course as better than a typical course at the same faculty. While we consider this a very positive result, we must interpret it carefully. In particular, not all differences are statistically significant, and for these we cannot completely rule out random effects. However, for most quality criteria, we find significant differences for at least two distinct universities, which is promising. The single exception is student motivation, which does not seem to significantly differ from other courses at the same faculty.

6.3 $RQ_{2.2}$: Comparison to Previous Course

Finally, as an alternative baseline, we aim to compare our course to a previous course on SPLs that has been held at the same faculty.

Methodology At two of the five universities considered in Table 3 (i.e., Ulm and Magdeburg), another course on SPLs (C_{old}) used to be taught before our course (C_{new}) was introduced. The course C_{old} had completely different, unpublished slides, which were adapted and slightly updated from year to year with clone-and-own. To reduce the influence of confounding factors, we aim to only compare an SET of C_{new} to an SET of C_{old} if both courses were held at the same university, by the same lecturer, and in a similar timeframe (i.e., within two years). Unfortunately, these criteria exclude most SETs we have for C_{old}, as they are simply too old to allow for meaningful comparison. Thus, we can only compare C_{new} (held in 2023) to one iteration of C_{old} (held in 2022) at Ulm University. We show the results for this comparison in Table 3 in the column for $RQ_{2.2}$. We test for statistical significance analogous to $RQ_{2.1}$.

Results The results for $RQ_{2.2}$ show that, in the mean, students at Ulm University rate C_{new} slightly better than C_{old}. In particular, C_{new} scores 4% to 9% more points than C_{old} in course-related quality criteria in the mean. Regarding self-assessment, these differences range from 0% to 7% more points for C_{new}. We find no statistically significant differences between C_{old} and C_{new} for any quality criterion ($0.25 \leq p \leq 0.99$).

Discussion Unfortunately, the results for $RQ_{2.2}$ remain inconclusive due to a lack of statistical significance, which maybe could have been reached with more regular SETs before introducing our course. Nonetheless, they do not contradict the favorable results we found for RQ_1 and $RQ_{2.1}$, which is reassuring.

6.4 Threats to Validity

Evaluations of teaching material can be challenging due to a variety of biases and confounding factors [28, 37, 49]. Thus, we aim to openly discuss potential threats to the validity of our conclusions.

Internal Validity Several biases and confounding factors might distort the results of our SETs compared to reality, some of which we discuss in the following. First, only students with very positive or negative experiences may choose to fill out questionnaires. To avoid this issue, we compare our course to other courses at the same faculty ($RQ_{2.1}$), which would suffer from the same bias. Second, students might simply like the topic and would also rate any other SPL course better than faculty average. We partially mitigate this by comparing our course to a previous course on SPLs ($RQ_{2.2}$). Third, students might be impacted in their answers by the lecturer's performance (e.g., are they experienced/enthusiastic?), exercise class (e.g., is it too labor-intensive/strict?) and other external factors not related to the course (e.g., organizational structures). We reduce this threat by collecting SETs over several distinct lecturers at multiple universities. To improve comparability, we unify the questionnaires from different universities and aggregate their questions into relevant quality criteria. Also, we ignore questions that only relate to teaching personnel and exercise classes. This way, we can focus on the actual course material and the students' self-assessment.

External Validity The external validity of our conclusions might be limited by the size of our dataset (e.g., regarding the number of universities, SETs, and participants). While more evidence is always welcome, our dataset already demonstrates how our course can be successfully introduced to new universities, which is the main goal of our evaluation. For the comparison to other courses at the same faculty ($RQ_{2.1}$), we only have data for four universities. However, at these universities, a total of 8045 students participated in SETs, which makes this a suitable baseline for comparison nonetheless.

7 Related Work

To the best of our knowledge, we are the first to create a new course on SPLs from scratch, describe our approach in detail, and publish it as open educational resources. We are also not aware of another attempt at evaluating a course on SPLs by collecting both quantitative and qualitative feedback from SETs. However, there are several case studies and experience reports on teaching SPLs [18, 21, 46, 52, 58, 61] published either at CSEE&T [14] or the SPLTEA workshop [3]. Some of these publications only discuss course goals superficially [21, 46, 52, 58], do not discuss a complete course [21, 52, 58], or target other audiences than computer science students [52, 61]. Other publications have a different focus from ours, for example on course evolution [18, 21] or specific pedagogical approaches [18, 61]. In particular, none of these publications review the literature on SPLs or perform a quantitative evaluation, as we do.

Acher et al. [2] performed two surveys and a workshop to capture a snapshot of current practices and challenges in teaching SPLs. In contrast to our work, they do not focus on a specific course. Our work can be regarded as a continuation of their work, which identified the need for a baseline curriculum, such as ours.

8 Conclusion

In this paper, we shared our experiences in architecting and designing a new university course on SPLs from scratch. We publish our course in form of open educational resources. Thus, we hope to ignite a collaborative effort to create and continuously improve SPL teaching material. Ideally, this will contribute to SPL education growing more mature along with research and industry.

In the future, we aim to add new lectures on underrepresented topics, release exercise sheets, and introduce an open call for scientific talks to attract guest lecturers and, thus, potentially extend the network of universities participating in this project. We also intend to regularly reevaluate our course and consider new feedback.

Acknowledgments

This project would not have been possible without the work and support of several of our colleagues: First, we want to thank Sebastian Krieter, Paul Maximilian Bittner, Chico Sundermann, Benno Hölz, and Florian Sihler for their contributions to the lecture slides and slide template. Second, we want to thank Sandra Greiner, Chico, Paul, and Sebastian for their contributions to the exercise sheets, which we hope to publicly release in the future. Third, we want to thank Gunter Saake, Thomas Leich, Sandy Grawe, and Roman Macháček for their organizational support throughout this project. Fourth, we want to thank our students for participating in the course and giving valuable feedback. Finally, special thanks go out to Sandra, who was very involved in discussing and developing new and improved exercises for our course.

Student Testimonials "I have not heard the term SPL before the course at all, and I think I have acquired a pretty good understanding of it now." — "The slides were of good quality, the lecture always interactive and never boring." — "It's really a pity that not more students have attended this high quality lecture, but I would recommend it to any other computer science student."

References

[1] Mathieu Acher, Luc Lesoil, Georges Aaron Randrianaina, Xhevahire Tërnava, and Olivier Zendra. 2023. A Call for Removing Variability. In *Proc. Int'l Working Conf. on Variability Modelling of Software-Intensive Systems (VaMoS)*. ACM, 82–84.

[2] Mathieu Acher, Roberto E. Lopez-Herrejon, and Rick Rabiser. 2017. Teaching Software Product Lines: A Snapshot of Current Practices and Challenges. *ACM Trans. on Computing Education (TOCE)* 18, 1, Article 2 (2017), 2:1–2:31 pages.

[3] Mathieu Acher, Rick Rabiser, and Roberto E. Lopez-Herrejon (Eds.). 2019. *Fourth International Workshop on Software Product Line Teaching (SPLTea 2019)*. ACM.

[4] Mustafa Al-Hajjaji, Sebastian Krieter, Thomas Thüm, Malte Lochau, and Gunter Saake. 2016. IncLing: Efficient Product-line Testing Using Incremental Pairwise Sampling. In *Proc. Int'l Conf. on Generative Programming: Concepts & Experiences (GPCE)*. ACM, 144–155.

[5] Vander Alves, Nan Niu, Carina Alves, and George Valença. 2010. Requirements Engineering for Software Product Lines: A Systematic Literature Review. *J. Information and Software Technology (IST)* 52, 8 (2010), 806–820.

[6] Michał Antkiewicz, Wenbin Ji, Thorsten Berger, Krzysztof Czarnecki, Thomas Schmorleiz, Ralf Lämmel, Stefan Stănciulescu, Andrzej Wąsowski, and Ina Schaefer. 2014. Flexible Product Line Engineering With a Virtual Platform. In *Proc. Int'l Conf. on Software Engineering (ICSE)*. ACM, 532–535.

[7] Sven Apel, Don Batory, Christian Kästner, and Gunter Saake. 2013. *Feature-Oriented Software Product Lines*. Springer.

[8] Paolo Arcaini, Maurice H. ter Beek, Gilles Perrouin, Iris Reinhartz-Berger, Miguel R. Luaces, Christa Schwanninger, Shaukat Ali, Mahsa Varshosaz, Angelo Gargantini, Stefania Gnesi, Malte Lochau, Laura Semini, and Hironori Washizaki (Eds.). 2023. *SPLC '23: Proceedings of the 27th ACM International Systems and Software Product Line Conference*. ACM.

[9] Wesley K. G. Assunção, Jacob Krüger, and Willian D. F. Mendonça. 2020. Variability Management Meets Microservices: Six Challenges of Re-Engineering Microservice-Based Webshops. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)*. ACM, Article 22.

[10] Thorsten Berger, Ralf Rublack, Divya Nair, Joanne M. Atlee, Martin Becker, Krzysztof Czarnecki, and Andrzej Wąsowski. 2013. A Survey of Variability Modeling in Industrial Practice. In *Proc. Int'l Workshop on Variability Modelling of Software-Intensive Systems (VaMoS)*. ACM, 7:1–7:8.

[11] Thorsten Berger, Steven She, Rafael Lotufo, Andrzej Wąsowski, and Krzysztof Czarnecki. 2013. A Study of Variability Models and Languages in the Systems Software Domain. *IEEE Trans. on Software Engineering (TSE)* 39, 12 (2013), 1611–1640.

[12] Thorsten Berger, Steven She, Rafael Lotufo, Andrzej Wąsowski, and Krzysztof Czarnecki. 2010. Variability Modeling in the Real: A Perspective From the Operating Systems Domain. In *Proc. Int'l Conf. on Automated Software Engineering (ASE)*. ACM, 73–82.

[13] Anna Bock, Bianca Idzko-Siekermann, Martin Lemos, Kristian Kniha, Stephan Christian Möhlhenrich, Florian Peters, Frank Hölzle, and Ali Modabber. 2020. The Sandwich Principle: Assessing the Didactic Effect in Lectures on "Cleft Lips and Palates". *BMC medical education* 20 (2020), 1–7.

[14] Andreas Bollin, Ivana Bosnić, Jennifer Brings, Marian Daun, and Meenakshi Manjunath (Eds.). 2024. *36th International Conference on Software Engineering Education and Training (CSEE&T)*. IEEE. https://doi.org/10.1109/CSEET62301.2024.10663010

[15] Jan Bosch. 2000. *Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach*. Pearson Education.

[16] Diane M. Bunce, Elizabeth A. Flens, and Kelly Y. Neiles. 2010. How Long Can Students Pay Attention in Class? A Study of Student Attention Decline Using Clickers. *Journal of Chemical Education* 87, 12 (2010), 1438–1443. https://doi.org/10.1021/ed100409p

[17] Xiaoyan Cai, Mingmei Peng, Jieying Qin, Kebing Zhou, Zhiying Li, Shuai Yang, and Fengxia Yan. 2022. Sandwich Teaching Improved Students' Critical Thinking, Self-Learning Ability, And Course Experience in the Community Nursing Course: A Quasi-Experimental Study. *Frontiers in Psychology* 13 (2022), 11 pages. https://doi.org/10.3389/fpsyg.2022.957652

[18] Jaime Chavarriaga, Rubby Casallas, Carlos Parra, Martha Cecilia Henao-Mejía, and Carlos Ricardo Calle-Archila. 2019. Nine Years of Courses on Software Product Lines at Universidad de los Andes, Colombia. In *Proc. Int'l Workshop on Software Product Line Teaching (SPLTea)*. ACM, 130–133.

[19] Lianping Chen and Muhammad Ali Babar. 2011. A Systematic Review of Evaluation of Variability Management Approaches in Software Product Lines. *J. Information and Software Technology (IST)* 53, 4 (2011), 344–362.

[20] Paul Clements and Linda Northrop. 2001. *Software Product Lines: Practices and Patterns*. Addison-Wesley.

[21] Philippe Collet, Sébastien Mosser, Simon Urli, Mireille Blay-Fornarino, and Philippe Lahire. 2014. Experiences in Teaching Variability Modeling and Model-Driven Generative Techniques. In *Proc. Int'l Workshop on Software Product Line Teaching (SPLTea)*. ACM, 26—-29.

[22] The Kernel Development Community. 2018. KConfig Language. Website: https://www.kernel.org/doc/html/latest/kbuild/kconfig-language.html. Accessed: 2024-01-30.

[23] Krzysztof Czarnecki and Ulrich Eisenecker. 2000. *Generative Programming: Methods, Tools, and Applications*. ACM/Addison-Wesley.

[24] Yael Dubinsky, Julia Rubin, Thorsten Berger, Slawomir Duszynski, Martin Becker, and Krzysztof Czarnecki. 2013. An Exploratory Study of Cloning in Industrial Software Product Lines. In *Proc. Europ. Conf. on Software Maintenance and Reengineering (CSMR)*. IEEE, 25–34.

[25] Sascha El-Sharkawy, Adam Krafczyk, and Klaus Schmid. 2015. Analysing the KConfig Semantics and its Analysis Tools. In *Proc. Int'l Conf. on Generative Programming: Concepts & Experiences (GPCE)*. ACM, 45–54.

[26] Emelie Engström and Per Runeson. 2011. Software Product Line Testing - A Systematic Mapping Study. *J. Information and Software Technology (IST)* 53 (2011), 2–13. Issue 1.

[27] Kevin Feichtinger, Chico Sundermann, Thomas Thüm, and Rick Rabiser. 2022. It's Your Loss: Classifying Information Loss During Variability Model Roundtrip Transformations. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)*. ACM, 67–78.

[28] Daniela Feistauer and Tobias Richter. 2017. How Reliable Are Students' Evaluations of Teaching Quality? A Variance Components Approach. *Assessment & Evaluation in Higher Education* 42, 8 (2017), 1263–1279.

[29] José A Galindo, Mathieu Acher, Juan Manuel Tirado, Cristian Vidal, Benoit Baudry, and David Benavides. 2016. Exploiting the Enumeration of All Feature Model Configurations: A New Perspective With Distributed Computing. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)*. ACM, 74–78.

[30] Matthias Galster, Danny Weyns, Dan Tofan, Bartosz Michalik, and Paris Avgeriou. 2013. Variability in Software Systems-a Systematic Literature Review. *IEEE Trans. on Software Engineering (TSE)* 40, 3 (2013), 282–306.

[31] Hassan Gomaa. 2004. *Designing Software Product Lines With UML: From Use Cases to Pattern-Based Software Architectures*. Addison-Wesley.

[32] Robert L Isaacson, Wilbert J McKeachie, John E Milholland, Yi G Lin, Margaret Hofeller, and Karl L Zinn. 1964. Dimensions of Student Evaluations of Teaching. *Journal of Educational Psychology* 55, 6 (1964), 344.

[33] Martina Kadmon, Veronika Strittmatter-Haubold, Rainer Greifeneder, Fadja Ehlail, and Maria Lammerding-Köppel. 2008. The Sandwich Principle – Introduction to Learner-centred Teaching/Learning Methods in Medicine. *Zeitschrift für Evidenz, Fortbildung und Qualität im Gesundheitswesen* 102, 10 (2008), 628–633. https://doi.org/10.1016/j.zefq.2008.11.018 Professionalisierung der medizinischen Ausbildung.

[34] Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, and A. Spencer Peterson. 1990. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Technical Report CMU/SEI-90-TR-21. Software Engineering Institute.

[35] Timo Kehrer, Marianne Huchard, Leopoldo Teixeira, and Christian Birchler (Eds.). 2024. *VaMoS '24: Proceedings of the 18th International Working Conference on Variability Modelling of Software-Intensive Systems*. ACM.

[36] Peter Knauber, Jesús Bermejo Muñoz, Günter Böckle, Julio Cesar Sampaio do Prado Leite, Frank van der Linden, Linda Northrop, Michael Stark, and David M. Weiss. 2001. Quantifying Product Line Benefits. In *Proc. Int'l Workshop on Software Product-Family Engineering (PFE)*. Springer, 155–163.

[37] Rebecca J. Kreitzer and Jennie Sweet-Cushman. 2021. Evaluating Student Evaluations of Teaching: A Review of Measurement and Equity Bias in Sets and Recommendations for Ethical Reform. *Journal of Academic Ethics* 20, 1 (2021), 73–84. https://doi.org/10.1007/s10805-021-09400-w

[38] Sebastian Krieter, Marcus Pinnecke, Jacob Krüger, Joshua Sprey, Christopher Sontag, Thomas Thüm, Thomas Leich, and Gunter Saake. 2017. FeatureIDE: Empowering Third-Party Developers. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)*. ACM, 42–45.

[39] Sebastian Krieter, Thomas Thüm, Sandro Schulze, Sebastian Ruland, Malte Lochau, Gunter Saake, and Thomas Leich. 2022. *T-Wise Presence Condition Coverage and Sampling for Configurable Systems*. Technical Report arXiv:2205.15180. Cornell University Library.

[40] Jacob Krüger and Thorsten Berger. 2020. An Empirical Analysis of the Costs of Clone- and Platform-Oriented Software Reuse. In *Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE)*. ACM, 432–444.

[41] Jacob Krüger, Wardah Mahmood, and Thorsten Berger. 2020. Promote-pl: A Round-Trip Engineering Process Model for Adopting and Evolving Product Lines. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)*. ACM, 1–12.

[42] Elias Kuiter, Tobias Heß, Chico Sundermann, Sebastian Krieter, Thomas Thüm, and Gunter Saake. 2024. How Easy Is SAT-Based Analysis of a Feature Model?. In *Proc. Int'l Working Conf. on Variability Modelling of Software-Intensive Systems (VaMoS)*. ACM, 149–151.

[43] Elias Kuiter, Jacob Krüger, and Gunter Saake. 2021. Iterative Development and Changing Requirements: Drivers of Variability in an Industrial System for Veterinary Anesthesia. In *Proc. Int'l Workshop on Variability and Evolution of Software-Intensive Systems (VariVolution)*. ACM, 113–122.

[44] Duc Le, Eric Walkingshaw, and Martin Erwig. 2011. #ifdef Confirmed Harmful: Promoting Understandable Software Variation. In *Proc. Int'l Symposium on Visual*

Languages and Human-Centric Computing (VL/HCC). IEEE, 143–150.

[45] Jörg Liebig, Sven Apel, Christian Lengauer, Christian Kästner, and Michael Schulze. 2010. An Analysis of the Variability in Forty Preprocessor-Based Software Product Lines. In *Proc. Int'l Conf. on Software Engineering (ICSE)*. IEEE, 105–114.

[46] Liana Barachisio Lisboa, Leandro Marques Nascimento, Eduardo Santana de Almeida, and Silvio Romero de Lemos Meira. 2008. A Case Study in Software Product Lines: An Educational Experience. In *Proc. IEEE Conf. on Software Engineering Education and Training (CSEE&T)*. IEEE, 155–162.

[47] Roberto E. Lopez-Herrejon and Don Batory. 2001. A Standard Problem for Evaluating Product-Line Methodologies. In *Proc. Int'l Conf. on Generative and Component-Based Software Engineering (GCSE)*. Springer, 10–24.

[48] Roberto E. Lopez-Herrejon, Stefan Fischer, Rudolf Ramler, and Aalexander Egyed. 2015. A First Systematic Mapping Study on Combinatorial Interaction Testing for Software Product Lines. In *Proc. Int'l Workshop on Combinatorial Testing (IWCT)*. IEEE, 1–10.

[49] Mark Davies Mark Shevlin, Philip Banyard and Mark Griffiths. 2000. The Validity of Student Evaluation of Teaching in Higher Education: Love Me, Love My Lectures? *Assessment & Evaluation in Higher Education* 25, 4 (2000), 397–405. https://doi.org/10.1080/713611436

[50] Jens Meinicke, Thomas Thüm, Reimar Schröter, Fabian Benduhn, Thomas Leich, and Gunter Saake. 2017. *Mastering Software Variability With FeatureIDE*. Springer.

[51] Tomohiro Nagashima and Susan Harch. 2021. Motivating Factors Among University Faculty for Adopting Open Educational Resources: Incentives Matter. *Journal of Interactive Media in Education* 2021, 1 (2021), 10 pages.

[52] Tsuneo Nakanishi, Kenji Hisazumi, and Akira Fukuda. 2018. Teaching Software Product Lines as a Paradigm to Engineers: An Experience Report in Education Programs and Seminars for Senior Engineers in Japan. In *Proc. Int'l Workshop on Software Product Line Teaching (SPLTea)*. ACM, 46–47.

[53] Jeho Oh, Necip Fazıl Yıldıran, Julian Braha, and Paul Gazzillo. 2021. Finding Broken Linux Configuration Specifications by Statically Analyzing the Kconfig Language. In *Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE)*. ACM, 893–905.

[54] Juliana Alves Pereira, Kattiana Constantino, and Eduardo Figueiredo. 2015. A Systematic Literature Review of Software Product Line Management Tools. In *Proc. Int'l Conf. on Software Reuse (ICSR)*. Springer, 73–89.

[55] Marcus Pinnecke. 2021. Product-Lining the Elinvar Wealthtech Microservice Platform. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)*. ACM, 60–68.

[56] Klaus Pohl, Günter Böckle, and Frank J. van der Linden. 2005. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer.

[57] Michael J. Prince and Richard M. Felder. 2006. Inductive Teaching and Learning Methods: Definitions, Comparisons, And Research Bases. *Journal of Engineering Education* 95, 2 (2006), 123–138. https://doi.org/10.1002/j.2168-9830.2006.tb00884.x

[58] Clément Quinton. 2018. Giving Students a Glimpse of the SPL Lifecycle in Six Hours: Challenge Accepted!. In *Proc. Int'l Workshop on Software Product Line Teaching (SPLTea)*. ACM, 42–43.

[59] Rick Rabiser, Klaus Schmid, Martin Becker, Goetz Botterweck, Matthias Galster, Iris Groher, and Danny Weyns. 2018. A Study and Comparison of Industrial vs. Academic Software Product Line Research Published at SPLC. (2018), 14–24.

[60] Zaynab Sabagh and Alenoush Saroyan. 2014. Professors' Perceived Barriers and Incentives for Teaching Improvement. *International Education Research* 2, 3 (2014), 18–40.

[61] Christoph Seidl and Irena Domachowska. 2014. Teaching Variability Engineering to Cognitive Psychologists. In *Proc. Int'l Workshop on Software Product Line Teaching (SPLTea)*. ACM, 16—-23.

[62] Chico Sundermann, Kevin Feichtinger, Dominik Engelhardt, Rick Rabiser, and Thomas Thüm. 2021. Yet Another Textual Variability Language? A Community Effort Towards a Unified Language. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)*. ACM, 136–147.

[63] Chico Sundermann, Tobias Heß, Michael Nieke, Paul Maximilian Bittner, Jeffrey M. Young, Thomas Thüm, and Ina Schaefer. 2023. Evaluating State-of-the-Art #SAT Solvers on Industrial Configuration Spaces. *Empirical Software Engineering (EMSE)* 28, 29 (2023), 38.

[64] Mikael Svahnberg, Jilles van Gurp, and Jan Bosch. 2005. A Taxonomy of Variability Realization Techniques: Research Articles. *Software: Practice and Experience* 35, 8 (2005), 705–754.

[65] Reinhard Tartler, Daniel Lohmann, Julio Sincero, and Wolfgang Schröder-Preikschat. 2011. Feature Consistency in Compile-Time-Configurable System Software: Facing the Linux 10,000 Feature Problem. In *Proc. Europ. Conf. on Computer Systems (EuroSys)*. ACM, 47–60.

[66] Scientific United Nations Educational and Cultural Organization (UNESCO). 2019. Recommendation on Open Educational Resources (OER). https://www.unesco.org/en/legal-affairs/recommendation-open-educational-resources-oer. Accessed: 2024-03-18.

[67] Frank J. van der Linden, Klaus Schmid, and Eelco Rommes. 2007. *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. Springer.

[68] Mahsa Varshosaz, Mustafa Al-Hajjaji, Thomas Thüm, Tobias Runge, Mohammad Reza Mousavi, and Ina Schaefer. 2018. A Classification of Product Sampling for Software Product Lines. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)*. ACM, 1–13.

[69] Hillel Wayne. 2024. The Hunt for the Missing Data Type. https://www.hillelwayne.com/post/graph-types. Accessed: 2024-03-22.