24 Reasons Why Variability Models Are Not Yet Universal (24RWVMANYU)

Mathieu Acher, University of Rennes, Institut Universitaire de France

https://www.mathieuacher.com @acherm

Context of the Keynote

- Slides presented at MODEVAR 2024 co-located with VaMoS 2024: https://modevar.github.io/program/
- Thanks to Jessie Galasso and Chico Sundermann for the invitation
- Abstract: "Variability models (e.g., feature models) are widely considered in software engineering research and practice, in order to develop software product lines, configurable systems, generators, or adaptive systems. Numerous success stories of variability models have been reported in different domains (e.g., automotive, aerospace, avionics) and the applicability broadens. However, the use of variability models is not yet universally adopted... Why? Some examples: variability models are not continuously extracted from projects and artefacts; each time a variability model is used, its expressiveness and language should be specialized; learning-based models are decoupled from variability models; modeling software variability should cover different layers and their interactions, etc. The list is arguably opinionated, incomplete, and based on my own practical experience, but also on observations of existing works. I will give 24 reasons to occupy your day as a researcher or practitioner in the field of variability modeling."

Context

- Variability models (e.g., feature models) are widely considered in software engineering research and practice, in order to develop software product lines, configurable systems, generators, or adaptive systems.
- Numerous success stories of variability models have been reported in different domains (e.g., automotive, aerospace, avionics) and the applicability broadens.
- However, the use of variability models is not yet universally adopted... Why?
- I will give *24* ~~42~~ reasons to occupy your day as a researcher or practitioner in the field of variability modeling.

Disclaimer

- The list is arguably opinionated, incomplete, and based on my own practical experience, but also on observations of existing works!
- I might be wrong and provocative!
- Some VaMoS 2024 papers may invalidate some of the points I will raise!
- Thanks to Gilles Perrouin, Philippe Collet, and Olivier Barais for some inputs on the list.
- The order of slides is random...
- We will vote for the most important reasons at the end of the talk! (please note the titles of the slide, top 3!)
- We will vote for the worst reasons at the end of the talk! (please note the titles of the slide)

24

- Tooling Support Is Not Mainstream (1)
- Continuous Extraction Is Lacking (2)
- Reverse Engineering Is Not Mainstream (3)
- The Cost-Benefit Dilemma (4)
- Ad-hoc Variability Languages (5)
- No Tooling for Removing Variability (6)
- Lack of Clear Impact Analysis of Variability (7)
- Constraints Are Overrated (3D printing) (8)
- Constraints Are Overrated (feature toggles) (9)
- Constraints Are Overrated (CLI) (10)
- Constraints Are Overlooked (11)
- Semantics Mismatch (12)
- Specialization of Expressiveness (13)
- Variability Models Decoupled From Learning-based Models (14)
- Scaling Issues in Reasoning Procedures (15)
- Modeling Across Different Layers (16)
- Mapping Variability Models to Other Artefacts (17)
- Searching for the Killer App (18)
- Partial Support from LLMs (19)
- Educational Gaps (20)
- Industry Reluctance (21)
- Configuration Language vs Variability Modeling Language (22)
- The Illusion of Universal Solutions (23)
- Github: Where are variability models? (24)

1. Tooling Support Is Not Mainstream

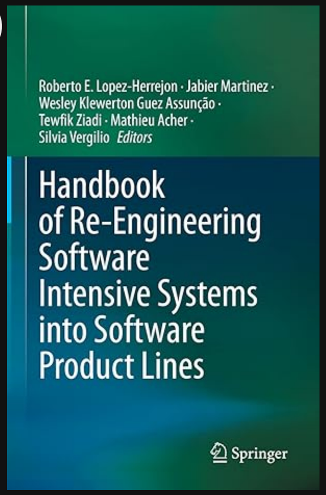
- "Hey I want to draw a feature model, what tool should I use?"
- "I'm developing in Python, how can I use a feature model and sample configurations?"
- "I'd like to add a Web configurator... Can I do it in UVL?"
- Tobias Heß et al. to the rescue with: variability.dev: Towards an Online Toolbox for Feature Modeling

2. Continuous Extraction Is Lacking

- Variability models are not automatically derived from projects and artefacts.
- JHipster is a good example of a project that could benefit from reverse engineering variability models.
  - promising results about sampling strategies Axel Halin et al. Test them all, is it worth it? Assessing configuration sampling on the JHipster Web development stack (2019).
  - good communication with core contributors that were interested in the results
  - the plan was to continue the work with the community, but it did not happen
- x264: interesting results and insights to share, but the first reaction is usually: what about the latest version?

3. Reverse Engineering Is Not Mainstream

- Practical tools for reverse engineering variability models are scarce.
- Hard to find an agnostic tool that can be applied to any kind of software system, language, or domain.
- Handbook of Re-Engineering Software Intensive Systems into Software Product Lines 1st ed. Springer 2023 Edition by Roberto E. Lopez-Herrejon, Jabier Martinez, Wesley Klewerton Guez Assunção, Tewfik Ziadi, Mathieu Acher, Silvia Regina Vergilio


Roberto E. Lopez-Herrejon · Jabier Martinez · Wesley Klewerton Guez Assunção · Tewfik Ziadi · Mathieu Acher · Silvia Vergilio *Editors*

Handbook of Re-Engineering Software Intensive Systems into Software Product Lines

Springer

4. The Cost-Benefit Dilemma

- We highly value the benefits of variability models for configuring, reasoning, driving the testing strategy, etc.
- While variability models offer benefits, they also introduce costs.
- For instance, specifying variability models from scratch is hard and time-consuming.
  - you might need to maintain the effort through the evolution of the system
- Same for extractors (reverse engineering): you need to maintain them and it has a cost!

5. Ad-hoc Variability Languages

- Prevalence of informal variability representations (e.g., JSON, XML).
- Lots of ad-hoc variability language or directly in the code (in-disguise feature models)
- Spreadsheets
- And people are happy with it!

6. No Tooling for Removing Variability

- Challenges in declaring features dead and removing code and dependencies.
- The tooling support for removing variability points is not mainstream.
- Mathieu Acher, Luc Lesoil, Georges Aaron Randrianaina, Xhevahire Tërnava, and Olivier Zendra. A Call for Removing Variability (2023). In VaMoS 2023

7. Lack of Clear Impact Analysis of Variability

- Difficulty in identifying code and dependencies affected by a specific option.
  - eg Spring properties. I would like to quickly show only the code, test and dependencies that use this option at some points.
- Visualizing a feature model is not the same as visualizing a variability model together with other artefacts. The feature model itself is a partial representation and the visualization should be adapted to what the variability is related.
- This depends on the model (pure domain, metrics for analysing it, relations/mapping to other formalisms).

8. Constraints Are Overrated ~~Overlooked~~

- I would argue logical constraints are a key aspect of variability models. Yet, many tools and platforms ignore the importance of constraints in variability models
- *3D printing domain*
  - there is still no support for cross-tree constraints in Thingiverse: https://customizer.makerbot.com/docs since the exploratory study of Acher et al. Customization and 3D Printing: A Challenging Playground for Software Product Lines (SPLC'14).
  - the consequence is that learned constraints cannot be made explicit for further reuse and enforcing configuration knowledge (as done in Amand et al. Towards Learning-Aided Configuration in 3D Printing: Feasibility Study and Application to Defect Prediction VaMoS 2019)

9. Constraints Are Overrated ~~Overlooked~~

- *Feature toggling*
  - huge interest in industry… and some academic works
  - Kastner et al. Feature Flags vs Configuration Options – Same Difference? ICSE-SEIP 2020.
  - Tërnava et al. On the Interaction of Feature Toggles VaMoS 2020
  - Jézéquel et al. From feature models to feature toggles in practice SPLC 2022
- No support of logical constraints in most feature toggling tools
  - need systematic study to support my claim, but TTBOMK, feature toggling libraries/platforms (e.g., LaunchDarkly, Split.io, Unleash, Togglz, etc.) do not fully support logical constraints
  - Kastner et al. "feature flag practitioners most lack behind the product line community and use often at most ad-hoc mechanisms and sparse comments spread across various configuration or source files"

10. Constraints Are Overrated ~~Overlooked~~

- I would argue logical constraints are a key aspect of variability models. Yet, many tools and platforms ignore the importance of constraints in variability models
- *Command-line parameters (CLI)* and environment variables
    - no support for cross-tree constraints in the most popular libraries and frameworks (e.g., yargs, commander, click, argparse, etc.)
        - again, need systematic study to support my claim
    - the consequence is that learned constraints cannot be made explicit for further reuse and enforcing configuration knowledge

11. Constraints Are ~~Overrated~~ Overlooked

- Without constraints, lots of invalid configurations can be generated
  - pressure is on the developer to ensure that configuration is valid
  - we need to integrate our tools in such contexts (eg 3D printing, feature toggling, CLI, etc.), but requires to engage with the communities and provide tools that are easy to use and deploy
  - Think about SAT solvers in the Linux community: Franz et al. Configfix: interactive configuration conflict resolution for the linux kernel ICSE-SEIP 2021
- Necessity for domain-driven constraints in specific reverse engineering contexts. Specialization of configuration spaces:
  - Hugo Martin, et al. A comparison of performance specialization learning for configurable systems In SPLC 2021
  - Paul Temple et al. Using Machine Learning to Infer Constraints for Product Lines (SPLC 2016)

12. Semantics Mismatch

- Differences in semantics between variability modeling languages and existing languages
- eg KConfig: widely used (eg in the Linux kernel) but has its own semantics…
    - the language evolves over time
    - hard to cover all language constructs
    - tools on top of KConfig are the only way to understand the semantics of KConfig
    - Kaan Yaman, Jan Wittler and Christopher Gerking. Kfeature: Rendering the Kconfig System into Feature Models (VaMoS 2024)
- Prankur Agarwal, Kevin Feichtinger, et al. On the Challenges of Transforming UVL to IVML and Back

13. Specialization of Expressiveness

- Each use of a variability model requires customization of its expressiveness and language.
  - Experience in the video domain: basic variability mechanisms are useful but not enough; attributes and multi-features are of prior importance; meta-information and specific constructs are relevant for scalable and purposeful reasoning over variability models. See Mauricio Alférez et al. Modeling Variability in the Video Domain: Language and Experience Report (2019).
  - VaryLaTeX: ideally, we would like to express preferences... but it would boil down to devise a new language with new constructs, semantics, and reasoning procedures (see Acher et al. VaryLaTeX: Learning Paper Variants That Meet Constraints (VaMoS 2018) and https://github.com/diverse-project/varylatex)
- "Modeling variability is not flexible enough wrt expressiveness (real problems need attributes, relation to artefacts... but not always, I’m happy with a propositional formula in easier settings)"

14. Variability Models Decoupled From Learning-based Models

- Huge interest in learning-based models for performance prediction, debugging, documentation, optimization, etc.
  - Pereira et al. Learning Software Configuration Spaces: A Systematic Literature Review JSS 2021
- Learning-based models: performance prediction and interpretability?; how to encode such prediction/knowledge into variability models?
- Performance-based feature interactions can hardly be modeled with variability models (note: feature interaction != logical constraints)
- Variability model + classifier *vs* Variability models + constraints extracted out of classifiers (e.g., SVM, RF, etc.) as in Temple et al. Using Machine Learning to Infer Constraints for Product Lines (SPLC 2016)

15. Scaling Issues in Reasoning Procedures

- Current reasoning techniques do not scale well, especially for complex systems or richer formalisms.
  - progress: Chico Sundermann et al. On the benefits of knowledge compilation for feature-model analyses Annals of Mathematics and Artificial Intelligence, 1-38
- We are not there yet? Quentin Plazar et al. Uniform Sampling of SAT Solutions for Configurable Systems: Are We There Yet? ICST'19
  - Sampling does not scale to eg Linux
- Sampling with numerical attributes or strings or ... eg
  - Munoz et al. Uniform Random Sampling Product Configurations of Feature Models That Have Numerical Features SPLC 2019
  - Lukas Güthing et al. Sampling Cardinality-Based Feature Models VaMoS 2024
  - you may have to alter encoding of VMs to scale (e.g., to SAT)

16. Modeling Across Different Layers

- The need to cover various layers and their interactions in software variability.
- Deep software variability (hardware, inputs, workloads, operating system, compiler, version, hypervisor, etc.)
  - Luc Lesoil et al. Deep Software Variability: Towards Handling Cross-Layer Configuration (2021). In VaMoS 2021
- Cross-system variability: it is not just deep in terms of layers from a model down to the internals of the CPU/storage, it’s also between systems or layers that interoperate between systems (variability inside a system that depends on the variability of another system, another API, etc.)
  - Paul Temple et al. Learning-Contextual Variability Models (2017). In IEEE Software

17 Mapping Variability Models to Other Artefacts

- Poor integration with other artefacts
  - requires significant engineering effort, see Damien Foures et al. Experience in Specializing a Generic Realization Language for SPL Engineering at Airbus (2023). In MODELS 2023
- Mappings features to other elements are often rigid / not flexible.
- Domain engineering may be a self-contained model but once you relate to lower implementation-related levels, you need to represent mappings, to relate the variability model to other elements and reason about them, without pulling all the heaviness of a full implementation
  - library dependencies, runtime dependencies are hard to manage
  - "often, when you update some dependencies, it could have an impact on your runtime constraints. I use Quarkus 3.6.4, it has an impact on the Dockerfile too"

18. Searching for the Killer App

- What is the "Hello World" of variability models?
- What are the arguments to convince a practitioner to put some effort and use variability models?
- The quest for a universally compelling application for variability models

19. Partial Support from LLMs

- Large language models (LLMs) offer some support for feature models, but it's not comprehensive.
  - Galindo et al. Large Language Models to generate meaningful feature model instances, SPLC 2023
- By the way, is it a bad thing that LLMs do not support variability models?
  - should we trust LLM for logical constraints?
- Btw LLMs+variability:
  - Greiner et al. Generative AI And Software Variability – A Research Vision, VaMoS 2024
  - Acher et al. A Demonstration of End-User Code Customization using Generative AI, VaMoS 2024
  - Achet et al. On Programming Variability with Large Language Model-based Assistant, SPLC 2023

20. Educational Gaps

- Variability models are not sufficiently taught in software engineering courses.
- https://teaching.variability.io/
- Resolution for 2024: I will teach feature models at undergraduate level, early in the curriculum

21. Industry Reluctance

- A preference in the industry for informal methods like Excel spreadsheets.
- See cost-benefit dilemna
- Feature models are not that well-known in industry (unless they are accompanied by companies like Pure or Gears), recent example from Alstom: https://arxiv.org/abs/2310.20395
    - "They seem to prefer encoding features and configurations of the products in Excel spreadsheets as front-end rather than using UVL or FeatureIDE directly. They model the UVL hierarchy in Excel! Interestingly, this work uses UPPAAL (a timed automata formalisms) which has a much more complex visual syntax and semantics than feature models though being visual. Developers are fine with automata modeling but not FM modeling. Is the simple visualization of FM unappealing to some? Semiotics issue? Education issue?"

22. Configuration Language vs Variability Modeling Language

- Apple has released a SOTA configuration language; not a variability modelling language https://github.com/apple/pkl
- Different communities (see SPLASH conference); the design space of configuration languages is huge!
- Dhall configuration language talk @ MODEVAR: remember https://modevar.github.io/splc-2020/program/?

23. The Illusion of Universal Solutions

- Recognizing that a one-size-fits-all language and approach is impractical.
- we already know it is an illusion
- Make things more configurable?
- What is the scope of variability modeling?
- out of the scope**s**?
- Universal is an ill-defined objective

24 Github: Where are variability models?

- Services for variability?
  - Matrix build for CI and testing configuration https://docs.travis-ci.com/user/build-matrix/
  - branching strategies
  - feature toggles (eg Unleash on Gitlab)
- On (open-source) projects? Numerous contributions of the community are applicable, but not directly in the form of variability models

Summary

- Variability models (e.g., feature models) are widely considered in software engineering research and practice
- The use of variability models is not yet universally adopted
  - not a problem per se: we cannot expect a single solution to fit all... and some domains or organizations are fine with their current practices
  - but we are certainly missing opportunities
- I gave *24* reasons to occupy your day as a researcher or practitioner in the field of variability modeling
- I am sure the applicability will broaden in the future
- What are the most important ~~reasons~~ research directions to consider? What are the worst and pointless directions? Let's vote, discuss, and have fun!