variability.dev: Towards an Online Toolbox for Feature Modeling

Tobias Heß
University of Ulm
Germany
tobias.hess@uni-ulm.de

Lukas Ostheimer
University of Ulm
Germany

Tobias Betz
University of Ulm
Germany

Simon Karrer
University of Ulm
Germany

Tim Jannik Schmidt
University of Ulm
Germany

Pierre Coquet
University of Ulm
Germany

Sean Semmler
University of Ulm
Germany

Thomas Thüm
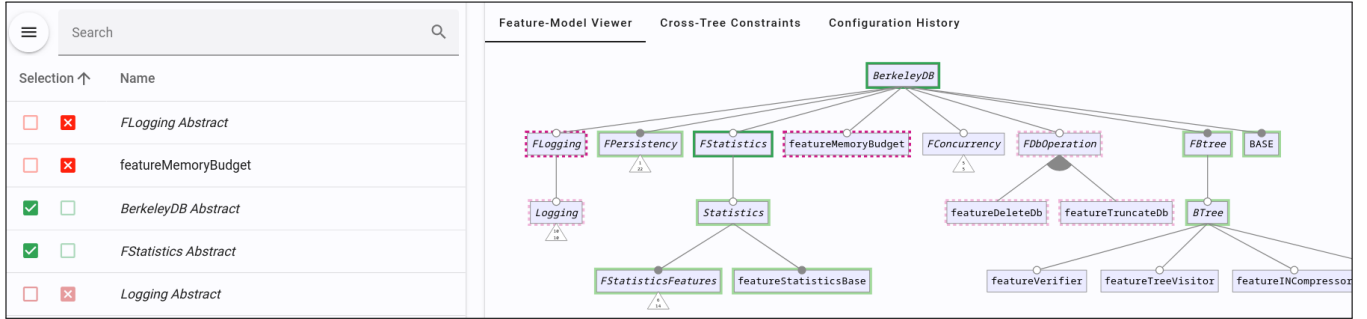University of Ulm
Germany

Figure 1: Interactive Configurator of variability.dev

ABSTRACT

The emergence of feature models as the default to model the variability in configurable systems fosters a rich diversity in applications, application domains, and perspectives. Independent of their domain, modelers require to open, view, edit, transform, save, and configure models as well as to collaborate with others. However, at the time of writing, the top five results when googling "Online Editor Feature Model" point to editors that either have minimal functionality, are unmaintained or defunct, or require an offline installation, such as FeatureIDE. In this work we present a preview of our in-development online toolbox for feature modeling, variability.dev. In particular, we showcase our collaborative feature-model editor and our online configurator both of which are built on top of the FeatureIDE library.

KEYWORDS

Feature Modeling, Feature-Model Editor, Online Configurator

1 INTRODUCTION

Alongside the emergence of feature models as the default for modeling the variability in configurable systems, many tools and frameworks have been developed in the past two decades [2, 5, 8, 9, 13, 15–18]. Their scopes and purposes range from fully fledged IDEs such as FeatureIDEs [15] that support the entire development process of software product lines, over frameworks for feature-model analysis [2, 5, 8, 9, 17], to prototypes [13] and online repositories [16, 18].

Of these, variED [13] and S.P.L.O.T [16] provide online feature-modeling tools. variED is a prototype of a feature-model editor with a focus on collaboration and conflict resolution [13]. S.P.L.O.T is a repository for feature models that also provides an online feature-model editor and a configurator [16], which are however cumbersome to use [18]. In addition, S.P.L.O.T has not been maintained for almost a decade and there exists a community effort within the MODEVAR initiative to replace it in the future [18].

With this work, we want to contribute to this effort [8, 14, 18] by showcasing our in-development collaborative feature-model editor and configurator. Both are built on top of the FeatureIDE library for means of transforming feature-model formats, anomaly detection,
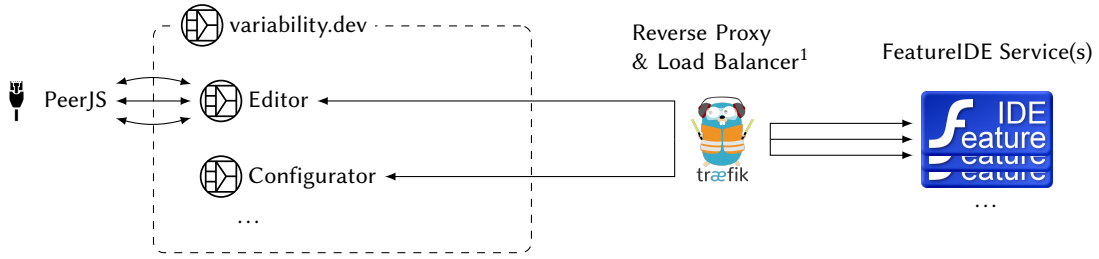
Figure 2: Architecture

well-formed edit operations, and more. We envision our toolbox as a lightweight frontend to the feature-modeling functionality in FeatureIDE. However, we are currently working on supporting additional and alternative backends for analyses not supported by FeatureIDE, such as model counting.

In Section 3 and Section 4 we showcase and discuss our editor and configurator, respectively. We discuss tools and frameworks with similar scopes in Section 5 and future plans for our toolbox in Section 6.

2 OVERVIEW AND USAGE SCENARIO

variability.dev is organized as a collection of dockerized microservices that may communicate with each other via restful APIs. Figure 2 depicts the stack deployed for this showcase. All components will also function without the others, albeit with limited functionality in some cases. While we will discuss the feature-model editor in more detail in Section 3, consider the following scenario for an illustration of the interaction of the various components:

User Alice opens a feature model in the editor. In order to display anomalies in the model, the editor asynchronously calls the FeatureIDE service (a restful API wrapping parts of the FeatureIDE library) to analyze the model. Possible anomalies are displayed in the model, once the request's results are available.

Alice's colleague Bob wants to view the same model that Alice is currently viewing. Alice shares her editing session with Bob via the PeerJS[2] integration. Bob finds a flawed cross-tree constraint in the model and requests edit rights from Alice. Alice grants the edit rights to Bob, who fixes the constraint. Alice notices that a previously dead feature is no longer dead, as the editor requested a reanalysis of the model after Bob's edit. Finally, Alice export and saves the fixed model.

3 FEATURE-MODEL EDITOR

We intended our feature-model viewer and editor to be an lightweight online alternative to editors in offline tools such as FeatureIDE [15], with a focus on rendering speed and extensibility. Currently, our editor is capable of rendering the largest available models in fractions of a second, has functionality to efficiently navigate large models, allows for collaboration and feature-model
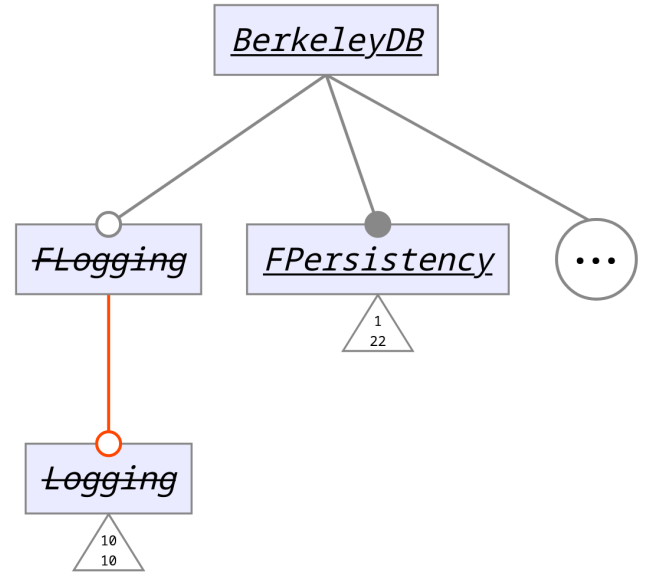
---

[1]Traefik Logo by https://github.com/traefik/traefik#credits (CC BY 3.0 Deed)
[2]https://peerjs.com/

Figure 3: Horizontal and Vertical Collapsing and Anomaly Annotation in variability.dev

analysis via restful calls to other services, and supports typical editing scenarios, including feature deletion by means of feature-model slicing.

3.1 Viewing Feature Models

The feature-model viewer in variability.dev follows the classical depiction of feature models as, for instance, in FeatureIDE [15]. Features are denoted by rectangles, with abstract features [23] typed in italic. Core features (features that are selected in every valid configuration [4]) are underlined and dead features (features that are not part of any valid configuration [4]) are stroked through. Mandatory and optional child features are depicted with the usual edges, alternative and or groups with the typical cone at the parent. Opposed to FeatureIDE, we opted to mark the edge of an false-optional feature (an optional feature that must always be selected together with its parent due to cross-tree constraints). In Figure 3, BerkeleyDB is an abstract and core feature, FLogging is abstract, optional, and dead, and Logging is abstract, false-optional, and also dead.

While even the largest models are rendered well below 200 ms by our viewer, large models are likely to visually overwhelm the user. Therefore, our viewer supports two types of collapsing, depicted in Figure 3. Sibling features and nodes on the same level can be hidden and are replaced by (⋯) to denote that something is hidden. Sub trees below can be hidden and are replaced by triangles as in Figure 3. The top number of the triangle refers to the hidden nodes directly below and the second number refers to the total number of hidden features below.

In order to navigate large models, we implemented a search that suggests matching features ordered by their Levenshtein distance. For visual guidance, not only the found feature is highlighted, but also the path from the root feature to the found feature.

Cross-tree constraints are listed similarly to the constraint view in FeatureIDE. Features appearing in constraints may be selected and are highlighted with a colored border. The interaction of multiple cross-tree constraints can be illustrated by selecting multiple cross-tree constrains, as the colored borders around a feature stack.

3.2   Editing Feature Models

We support the typical feature model operations, namely adding, moving, and deleting features, as well as adding, editing, and deleting cross-tree constraints. Features may be added via the context menu or, with quick edits enabled, by clicking highlighted areas around features. Features may be moved laterally or arbitrarily, if semantic drag and drop is enabled. As the order of sibling features may encode semantics [7], moving of features via drag and drop can also be disabled entirely.

For feature deletion, we employ the slicing functionality in FeatureIDE [1, 11]. This ensures that cross-tree constraints remain meaningful after feature appearing in the are removed from the model and handles non-leaf features.

3.3   Collaborative Editing

Similar to variED [13], our editor allows for multiple users to collaboratively edit, analyze, and view the same feature model. To circumvent the challenges incurred by simultaneous edit operations, we, opposed to variED [13], opted to only allow one user to edit the model at a time. This allows us to use the full set of edit operations known from FeatureIDE, including the use of feature-model slicing when deleting features.

An editing user may share his editing session via link or QR code. After joining, users can control their view of the model independently of the host's and ask to claim edit rights at any time. Both host and the currently editing user can grant edit rights, with the host being additionally able to reclaim edit rights at any time. All users share the edit history and are therefore capable of undoing or redoing edit operations.

3.4   Import and Export

Our feature-model editor natively supports the FeatureIDE XML format, other feature-model formats are translated using the FeatureIDE service. Currently, edited models can be exported to the formats available in FeatureIDE, including DIMACS and UVL. As our editor is based on SVG, the model can also be exported as SVG.

3.5   FeatureIDE Integration

To harness the functionality of FeatureIDE, we built the FeatureIDE service[3], a command-line interface and restful API around the FeatureIDE library. This service currently supports the transformation of feature-model formats, anomaly detection, decision propagation, and t-wise sampling with YASA [12]. Requests to the restful API are handled asynchronously and are lazily loaded into the editor, once available.

4   FEATURE-MODEL CONFIGURATOR

Besides the feature-model editor, we are also developing an online feature-model configurator, whose user interface is depicted in Figure 1. The configurator reuses many parts of the editor, for instance the feature-model viewer and the constraint view. Features may be configured from the list of features, or by clicking on features in the feature model. Every configuration decision may be rolled back using the configuration history. Selection, deselection, or freeing of a feature invokes the built-in decision propagation which selects and deselects features that are implied by the current partial configuration.

When the FeatureIDE service is available, the configurator uses the decision propagation of FeatureIDE which also marks open features, for instance when the parent of an or group is selected but none of the child features have been selected so far. Core and dead features are selected and deselected automatically when loading the model, respectively. Users may disable decision propagation and configure freely but potentially invalid. For the future, we are planning to use the explanation and configuration reparation functionalities from FeatureIDE as well. Furthermore, we are currently testing the incorporation of model counting information into the configuration process.

Like the default feature-model view, we designed the configuration view to be as inclusive as possible with regard to colorblind people. Therefore, selected features are always depicted with a solid border and deselected features with a dashed border. Only the distinction between explicitly (dark) and implicitly (light) selected or deselected features is made by color. In the future, we plan to collapse the feature model based on configuration, for example, that sub trees below a deselected feature will always be collapsed and sub trees below a selected feature will be revealed upon selection.

5   RELATED WORK

In this section, we present work that is related to ours, in particular, previous and ongoing efforts to provide the product-line community with a platform facilitating collaboration.

S.P.L.O.T.[4] is a web-based platform providing a feature-model repository, functionality for editing and analyzing feature models, as well as an interactive configurator [16]. However, S.P.L.O.T. has not been maintained or updated in almost a decade and the models in its repository are not representative of current real-world feature models, with the majority being insignificantly tiny [18]. Its automated analysis functionality is rather cumbersome to use (e.g., blocking the page on computation) and does not scale to current models. Most notably, analysis results are not stored or cached.

---

[3]https://github.com/OBDDimal/FeatureIDE-Service
[4]http://www.splot-research.org/

Similarly, we find that their feature-model editor and configurator are not intuitive to use for users that are familiar with, for instance, FeatureIDE.

FeatureIDE[5] is an Eclipse-based framework for feature-oriented software development [15]. In addition to feature modeling, FeatureIDE supports the entire scope of software product-line development [15]. As part of variability.dev, we built an restful API around the FeatureIDE library and use many of its functionality with regards to feature modeling.

variED[6] is a prototype of a collaborative feature-model editor, were multiple users may edit the same model concurrently [13]. Resulting conflicts are automatically detected and users are provided with conflict-resolution strategies. As discussed in Section 3.3, the collaboration feature in variability.dev circumvents the issues incurred by concurrent edits, by only allowing one user in the collaborative session to edit the model. This enables edit operations, such as feature deletion by means of feature-model slicing that are not covered by variED.

6 FUTURE WORK

Besides quality-of-life improvements, both already planned and as an reaction to feedback from the community, we plan to extend the functionalities of variability.dev as follows.

Analysis Integration. Currently, feature models are displayed in the common format in a view that highlights anomalies in a colorblind-friendly visualization (cf. Section 3). In the future, we envision additional views on a feature model that, for instance, color the features with respect to their commonality [19], other model-counting metrics [22], or syntactic information, such as constraint interaction. Besides such views, we are currently integrating feature-model fact labels [10] into the editor.

Knowledge Compilation. In addition to downloading the edited model in the various file formats or as SVG, we want to provide functionality to also download analysis results, samples, and knowledge-compilation artifacts such as BDDs or d-DNNFs in the future. Likewise, we plan to enhance the configurator with BDD-guided configuration [6] and to visualize configurations in general, not only with the model but also on graphical representations of BDDs and d-DNNFs.

Textual Feature-Model Editor. Our editor renders even the largest models in under 200 ms on entry-level machines. In addition, it possesses functionality to quickly navigate to features of interest and to hide parts of a model. Nevertheless, there are benefits to editing a model in a textual representation, such as UVL [20]. Stemming from the MODEVAR initiative, UVL is designed to be read and written by humans [20, 21] aided by a language server protocol [14]. We aim to support textual editing of feature models in our editor and especially an integration of UVL and its language server.

---

[5] https://featureide.github.io/
[6] https://github.com/ekuiter/variED

7 CONCLUSION

We presented two in-development tools for feature-model editing and configuration and the underlying architecture. Our editor already supports typical editing operations and is capable of handling the largest currently available models.

We presented variability.dev, our proposal for a web-based, collaborative platform to exchange feature models, analysis results, and related artifacts. However, our platform is not merely a static repository but every shared feature model is converted into other formats, analyzed, and sampled, when possible. Furthermore and to the best of our knowledge, variability.dev is the first platform to treat the evolution history of feature models as first-class citizens, an aspect on which we are planning to expand heavily in the future. Last not least, with features like the mobile-friendly feature-model viewer and editor, with its live collaboration, we also make an effort to lower the barrier for elementary product-line education.

REFERENCES

[1] Mathieu Acher, Philippe Collet, Philippe Lahire, and Robert B. France. 2011. Slicing Feature Models. In *ASE*. IEEE, Washington, DC, USA, 424–427. https://doi.org/10.1109/ASE.2011.6100089

[2] Mathieu Acher, Philippe Collet, Philippe Lahire, and Robert B. France. 2013. Familiar: A Domain-Specific Language for Large Scale Management of Feature Models. *SCP* 78, 6 (2013), 657–681.

[3] David Benavides, Rick Rabiser, Don Batory, and Mathieu Acher. 2019. First International Workshop on Languages for Modelling Variability (MODEVAR 2019). In *SPLC*. 323–323.

[4] David Benavides, Sergio Segura, and Antonio Ruiz-Cortés. 2010. Automated Analysis of Feature Models 20 Years Later: A Literature Review. *Information Systems* 35, 6 (2010), 615–708.

[5] David Benavides, Sergio Segura, Pablo Trinidad, and Antonio Ruiz-Cortés. 2007. FAMA: Tooling a Framework for the Automated Analysis of Feature Models. In *VaMoS* (Limerick, Ireland). Technical Report 2007-01, Lero, Limerick, Ireland, 129–134.

[6] Daniel Bischoff and Wolfgang Küchlin. 2017. Adapting Binary Decision Diagrams for Visualizing Product Configuration Data. *Jahrestagung der Gesellschaft für Informatik* (2017).

[7] Claudia Fritsch, Richard Abt, and Burkhardt Renz. 2020. The Benefits of a Feature Model in Banking. In *SPLC* (Montreal, QC, Canada). ACM, New York, NY, USA, Article 9, 11 pages. https://doi.org/10.1145/3382025.3414946

[8] José A. Galindo, José Miguel Horcas, Alexander Felfernig, David Fernández-Amorós, and David Benavides. 2023. FLAMA: A collaborative effort to build a new framework for the automated analysis of feature models. 16–19. https://doi.org/10.1145/3579028.3609008

[9] Tobias Heß, Tobias Müller, Chico Sundermann, and Thomas Thüm. 2022. ddueruem: A Wrapper for Feature-Model Analysis Tools. In *SPLC* (Graz, Austria). ACM, New York, NY, USA, 54–57. https://doi.org/10.1145/3503229.3547032

[10] Jose M. Horcas, Jose A. Galindo, Mónica Pinto, Lidia Fuentes, and David Benavides. 2022. FM Fact Label: A Configurable and Interactive Visualization of Feature Model Characterizations. 42—-45. https://doi.org/10.1145/3503229.3547025

[11] Sebastian Krieter, Reimar Schröter, Thomas Thüm, Wolfram Fenske, and Gunter Saake. 2016. Comparing Algorithms for Efficient Feature-Model Slicing. In *SPLC* (Beijing, China). ACM, New York, NY, USA, 60–64. https://doi.org/10.1145/2934466.2934477

[12] Sebastian Krieter, Thomas Thüm, Sandro Schulze, Gunter Saake, and Thomas Leich. 2020. YASA: Yet Another Sampling Algorithm. In *VaMoS* (Magdeburg, Germany). ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.1145/3377024.3377042

[13] Elias Kuiter, Sebastian Krieter, Jacob Krüger, Gunter Saake, and Thomas Leich. 2021. variED: An Editor for Collaborative, Real-Time Feature Modeling. *EMSE* 26, 2 (March 2021), 24. https://doi.org/10.1007/s10664-020-09892-x

[14] Jacob Loth, Chico Sundermann, Tobias Schrull, Thilo Brugger, Felix Rieg, and Thomas Thüm. 2023. UVLS: A Language Server Protocol for UVL. In *SPLC* (Tokyo, Japan). ACM, New York, NY, USA, 43–46. https://doi.org/10.1145/3579028.3609014

[15] Jens Meinicke, Thomas Thüm, Reimar Schröter, Fabian Benduhn, Thomas Leich, and Gunter Saake. 2017. *Mastering Software Variability with FeatureIDE*. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-319-61443-4

[16] Marcílio Mendonça, Moises Branco, and Donald Cowan. 2009. S.P.L.O.T.: Software Product Lines Online Tools. In *OOPSLA*. ACM, New York, NY, USA, 761–762.

[17] Michael Nieke, Gil Engel, and Christoph Seidl. 2017. DarwinSPL: An Integrated Tool Suite for Modeling Evolving Context-aware Software Product Lines. In *VaMoS* (Eindhoven, Netherlands). ACM, New York, NY, USA, 92–99. https://doi.org/10.1145/3023956.3023962

[18] David Romero, José A. Galindo, José Miguel Horcas, and David Benavides. 2021. A First Prototype of a New Repository for Feature Model Exchange and Knowledge Sharing. 80–85. https://doi.org/10.1145/3461002.3473949

[19] Chico Sundermann. 2020. *Applications of #SAT Solvers on Product Lines*. Master's Thesis. TUBraunschweig, Germany. https://doi.org/10.24355/dbbs.084-202009161329-0

[20] 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 *SPLC* (Leicester, United Kingdom). ACM, New York, NY, USA, 136–147. https://doi.org/10.1145/3461001.3471145

[21] Chico Sundermann, Kevin Feichtinger, José A. Galindo, David Benavides, Rick Rabiser, Sebastian Krieter, and Thomas Thüm. 2022. Tutorial on the Universal Variability Language. In *SPLC* (Graz, Austria). ACM, New York, NY, USA, 260:1. https://doi.org/10.1145/3546932.3547024

[22] Chico Sundermann, Michael Nieke, Paul Maximilian Bittner, Tobias Heß, Thomas Thüm, and Ina Schaefer. 2021. Applications of #SAT Solvers on Feature Models. In *VaMoS* (Krems, Austria). ACM, New York, NY, USA, Article 12, 10 pages. https://doi.org/10.1145/3442391.3442404

[23] Thomas Thüm, Christian Kästner, Sebastian Erdweg, and Norbert Siegmund. 2011. Abstract Features in Feature Modeling. In *SPLC* (Munich, Germany). IEEE, Washington, DC, USA, 191–200. https://doi.org/10.1109/SPLC.2011.53