Acapulco: An extensible tool for identifying optimal and consistent feature model configurations

Jabier Martinez
Tecnalia, Basque Research and Technology Alliance (BRTA)
Derio, Spain

Daniel Strüber
Chalmers | University Gothenburg, Sweden, and Radboud University Nijmegen
The Netherlands

Jose Miguel Horcas
CAOSD Group, ITIS, Universidad de Málaga
Málaga, Spain

Alexandru Burdusel
Department of Informatics, King's College London
United Kingdom

Steffen Zschaler
Department of Informatics, King's College London
United Kingdom

ABSTRACT

Configuring feature-oriented variability-rich systems is complex because of the large number of features and, potentially, the lack of visibility of the implications on quality attributes when selecting certain features. We present *Acapulco* as an alternative to the existing tools for automating the configuration process with a focus on mono- and multi-criteria optimization. The soundness of the tool has been proven in a previous publication comparing it to SATIBEA and MODAGAME. The main advantage was obtained through consistency-preserving configuration operators (CPCOs) that guarantee the validity of the configurations during the IBEA genetic algorithm evolution process. We present a new version of *Acapulco* built on top of FeatureIDE, extensible through the easy integration of objective functions, providing pre-defined reusable objectives, and being able to handle complex feature model constraints.

CCS CONCEPTS

• **Software and its engineering → Software product lines**.

KEYWORDS

Variability management, software product lines, genetic algorithms

ACM Reference Format:
Jabier Martinez, Daniel Strüber, Jose Miguel Horcas, Alexandru Burdusel, and Steffen Zschaler. 2022. Acapulco: An extensible tool for identifying optimal and consistent feature model configurations . In *26th ACM International Systems and Software Product Line Conference- Volume B (SPLC '22), September 12–16, 2022, Graz, Austria.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3503229.3547067

1 INTRODUCTION

Feature modelling is a widely spread formalism to represent configuration spaces within variability-rich systems, including Software Product Lines (SPLs) [5]. Given a feature model, it is desired to find the optimal configuration for an objective or for multiple objectives simultaneously[12]. This task is challenging when the number of features is large (usually the case for industrial configurable systems), when only partial knowledge of the features is available (it is difficult that stakeholders are experts in all features), when there exist complex cross-tree constraints [6] (i.e., arbitrary relationships in propositional logic), or when it is accepted that feature interactions can make error-prone a manual estimation of the fitness of the configurations. Although several tools exist to address the optimization challenge to some extent [3] such as SATIBEA [2], MODAGAME [10], SPL Conqueror [13], ClaferMoo [9], and SPL Config [11], some of them tend to generate invalid configurations that need to be repaired during the optimization process, others have limitations regarding extensibility of the objective functions, or cannot be seamlessly integrated with other SPL-focused tools.

We present a new tool named *Acapulco* which is a search-based optimization tool for feature models, and SPLs in general, that instantiates the IBEA algorithm (Indicator-based Evolutionary Algorithm) [2] and improves it through the use of *consistency-preserving configuration operators* (CPCOs) [4]. The tool name was inspired by the acronym CPCO (ACaPulCO). CPCOs are special mutation operators that guarantee the validity of the configurations during the search by bundling coherent sets of changes (i.e., the activation or deactivation of a particular feature with other changes that are needed to preserve validity) without relying on SAT solvers. The benefits of *Acapulco* were shown against two state-of-the-art tools [2, 10] outperforming both in speed and solution quality.

We report on a new version of the tool, making the previous research pipeline [4] more usable, resulting in a user friendly tool built on top of FeatureIDE [15]. A key challenge in solving the optimal-configuration problem is the significant diversity of objectives to guide the search [12] and technological stacks needed for their calculation (e.g., tools to measure derived variants). *Acapulco*, thus, required extensibility capabilities to easily adapt to domain-specific objectives. A set of predefined generic objectives are included, and an extension point with dedicated interfaces is provided. This addresses a limitation for the uptake of other tools which are only based on feature attributes (e.g., [2, 9–11]), or even predefine the number and type of attributes (e.g., [2, 10, 11]).

The target users of *Acapulco* are domain or application engineers dealing with variability-rich systems with complex configuration optimization needs, as well as researchers that want to build on, improve, or benchmark the tool to advance the SPL optimization field.

Tool: **https://github.com/acapulco-spl/acapulco**
Video: **https://youtu.be/WPAJT9kVHe4**

SPLC '22, September 12–16, 2022, Graz, Austria

ACM ISBN 978-1-4503-9206-8/22/09…$15.00
https://doi.org/10.1145/3503229.3547067

This paper is structured as follows. Section 2 presents background information. Section 3 details *Acapulco* functionality and architecture. Section 4 summarizes our previous experience with real-world SPLs. Section 5 concludes and outlines future work.

2 BACKGROUND AND MOTIVATION

Principle-guided CPCO generation. In our recent paper [4], we present an algorithm for automatically generating a suite of CPCOs. The generated suite provides for each real-optional feature two operators, one for activation and one for deactivation. The generation of these operators is guided by a notion of *principles*, rules that avoid one particular constraint violation from arising, in response to the (de)activation of a feature involved in a constraint. The starting point for the generation of an operator is one feature, together with the choice to either activate or deactivate it. From there, the principles are considered recursively, as long as one of them is applicable. We now list the set of activation principles, that is, principles that are applied in response to the activation of a feature (in analogy, a set of deactivation principles exists [4], omitted for space reasons):

Activation principles. Given a feature f to be activated:

(1) ActMand: Activate all mandatory children of f.
(2) ActPar: If g is f's parent feature and g is not a core feature, activate g.
(3) ActReq: If f requires another feature g (via a *requires* relation) and g is not a core feature, activate g.
(4) ActGroup: If f is an "or" or "xor" group, activate one of f's children.
(5) ActXor: If f is a feature in an "xor" group, deactivate all of f's siblings.
(6) ActExc: If f excludes or is excluded by a feature g, deactivate g.

Fig. 1 shows an example of a feature model and the CPCO generated to activate the feature F3. We can observe how feature F1 is activated because it is an ancestor of F3. Features F4 and F5 are deactivated as they belong to the same "xor" group. Feature F6 is activated because of the "requires" CTC, and F2 is activated because it is an ancestor of F6. In this way, the CPCO is self-contained for bringing any configuration to a valid state after activating F3. In general, there are multiple ways of applying a principle (e.g., choosing the child to activate in ActGroup), all of which can be used to ensure validity. Our core technical contribution in paper [4] is an efficient algorithm that avoids combinatorial effects during the naïve application of principles, and encodes the resulting CPCO variants in an efficient way. To this end, we introduce a new data structure of *feature activation diagrams*, which can be computed efficiently and incrementally.

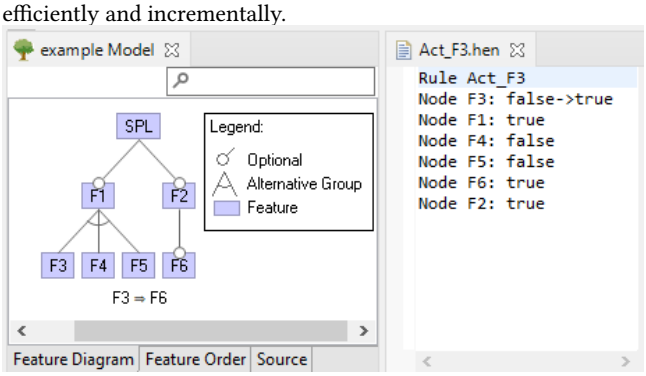


Figure 1: Illustrative example of a feature model (left) and a concrete CPCO to activate the feature F3 (right).

Complex cross-tree constraints in feature models. The hierarchy of the feature model and the definition of optional and mandatory features, "or" and "xor" groups, already define certain constraints for configurations. In addition, cross-tree constraints (CTCs) such as a feature requiring, or mutually excluding another feature can be defined. CPCOs [4] assumes this basic feature model dialect (i.e., FODA [5]), and thus, the generation of the CPCOs will only be possible in this case. In other approaches (e.g., SATIBEA), complex CTCs are less problematic because they rely on SAT solvers. To address this limitation of CPCOs, in this tool we add support for the transformation method proposed by Knüppel et al. [6]. The method automatically transforms complex and pseudo-complex constraints into basic "requires" and "excludes" constraints. Pseudo-complex constraints are those that can be transformed directly, whereas complex constraints require the addition of additional features and feature groups to the feature model to represent the same configuration space. *Acapulco* reuses the implementation of this method already available in FeatureIDE.

Flexible optimisation objectives in feature configuration. A traditional evolution objective is driven by *extended feature models* [1] where each feature has an associated value for a non-functional attribute, and the minimization or maximization objective is based on adding the values of the selected features in a given configuration. For instance, MODAGAME considers the floating-point attributes "usability", "battery consumption" and "memory footprint". SATIBEA uses the attributes "used before" (boolean), "known defects" (float) and "cost" (float). These attributes are fixed in these tools, making it difficult to use the tools in other contexts that require different attributes. A systematic literature review on learning configuration spaces [12] evidenced the large diversity of evolution objectives that can be considered, ranging across footprint, code complexity, different time measures, energy consumption, or even user likeability. Also, simply adding attributes associated to individual features is not always appropriate due to feature-interaction effects. Thus, a tool aiming to be generic must support extensibility on this aspect considering extended feature models and any other objective including those requiring to analyze the variant derived from the configuration (e.g. by running performance tests).

3 FUNCTIONALITY AND ARCHITECTURE

Main characteristics and functionalities:

- **Built on top of FeatureIDE,** enabling the use of a single environment when dealing with the management of the SPL or configurable system. In particular, *Acapulco* brings multi-objective optimization capabilities to FeatureIDE users.
- **Visual UI for user-defined parameters.** Fig. 2 shows the launch dialog of *Acapulco* with its configurable functionality. The stopping condition for the search can be configured with a number of evolutions or a timeout in milliseconds (both values to be defined in the "Stopping value" field). This is followed by standard genetic algorithms' parameters (i.e., population size, mutation, and crossover probability). Finally, the objectives are specified. In the example in Fig. 2, three objective were added and configured (i.e., maximize usability, minimize battery consumption, and minimize memory footprint).
- **Standard formats for the output.**

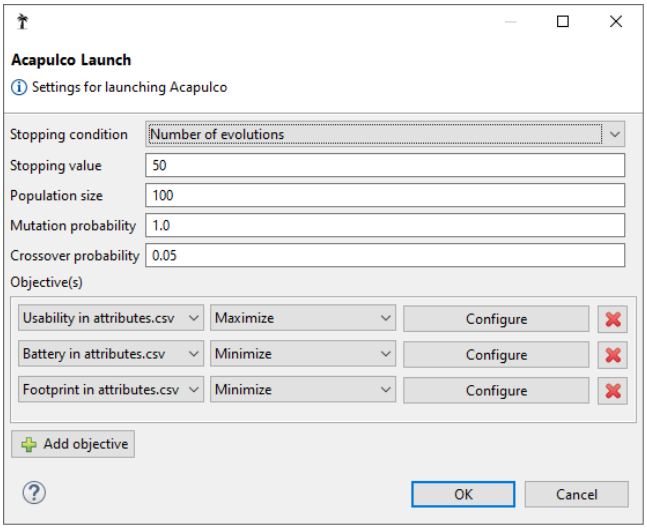


Figure 2: Launch Dialog of *Acapulco*.

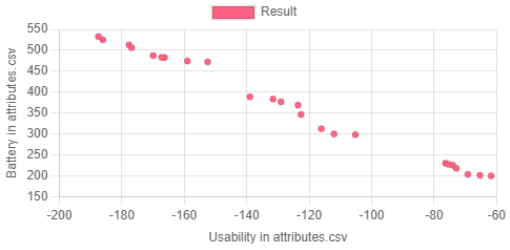


Figure 3: Visualization of the Pareto-front generated by *Acapulco* in an optimization problem with two objectives.

  - Pareto-front results are FeatureIDE configuration files.
  - CSV file with data of all configurations in the final pareto-front. Notably, the measure of each objective per configuration.
  - CSV file with data for each evolution including elapsed time and current population and approximation set results.
  - HTML file with a JavaScript-based visualization is generated in case of multi-objective optimization (see Fig. 3).
- **Automatic generation of CPCOs and their usage during the search.** CPCOs guaranteeing the validity of configurations are automatically generated by *Acapulco*. CPCOs (two for each feature—for activation and deactivation) can be manually inspected. They use a simplified textual variant of the Eclipse Henshin notation [14], a rule-based model transformation language that is highly human-readable (see CPCO example in Fig. 1).
- **Support for complex cross-tree constraints.** Currently, CPCOs can only be created with basic FODA feature models. To solve this limitation, *Acapulco* uses the Knüppel et al. method [6] to convert complex constraints into simple constraints. This step is transparent to the user in the case of basic CTCs or pseudo-complex CTCs. The user receives a warning in case of complex CTCs allowing the user to decide between applying the Knüppel et al. method or removing the complex constraints.
- **Objectives can be freely combined to specify the multi-objective problem at hand.** Users can choose from:
  - A set of predefined, parametrisable, generic objectives:
    - *Extended Feature Models data based on feature attributes in CSV files.* The user selects a CSV file and the target attribute.
    - *Minimize or maximize the number of features.*
    - *Number of derived files or lines of code.* Files can be filtered by extension (e.g., only counting *java* files for Java SPLs, or *cpp* and *h* for C++ SPLs).
    - *Configuration validity.* When not using the Knüppel et al. method [6], the search may produce invalid configurations. The objective helps to minimise invalid configurations.
  - Domain-specific objectives. An extension point is provided to add other types of (potentially further configurable) objectives. This way, *Acapulco* is not limited to extended feature models or other predefined objective calculations. As an example, *Acapulco* includes a set of objective definitions for the FeatureIDE Images composer. These objectives are used for product lines of images, composing pictures from partial images depending on features selected. The first two objectives are related to finding the images that are closest to a user-defined colour, or to a given image provided by the user. The third objective illustrates the use of *Acapulco* for Interactive Genetic Algorithms (IGA) where, contrary to fully automatic calculations, humans provide the score of each individual (e.g., Likert scales for paintings [7] or usability test results in UI design [8]).
  - Analysis of implementation artifacts. As a special type of domain-specific objective, *Acapulco* offers an abstract implementation for objectives that require a specific variant product to be derived. The objectives to minimize or maximize the number of derived files or lines make use of this abstract class.

Architecture. Figure 4 illustrates, at the bottom, how *Acapulco* is a layer on top of the FeatureIDE layer, which is on top of the Eclipse IDE. *Acapulco* currently consists of only one plugin implementing all the functionality. It exposes the *acapulco.objective* extension point. Extensions must implement the *IObjective* interface where the most important method is *evaluate*, returning a number for a given list of features[1]. For instance, the "Number of features" objective returns the size of the feature list. Figure 4 also shows an abstract class *AbstractPostDerivationObjective* that implements *IObjective*. This class will ask FeatureIDE to derive the variant independently of the composer used in the SPL (e.g., annotative or compositional), so the developer will only need to implement *evaluateDerivation()* where the path to the derived variant is provided as parameter.

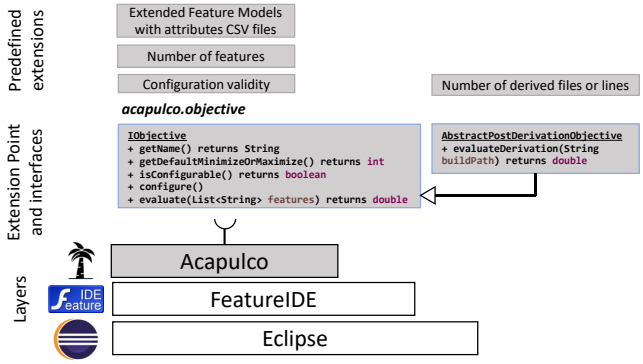


Figure 4: Layers and extensibility for objectives.

[1]Feature names are unique in FeatureIDE

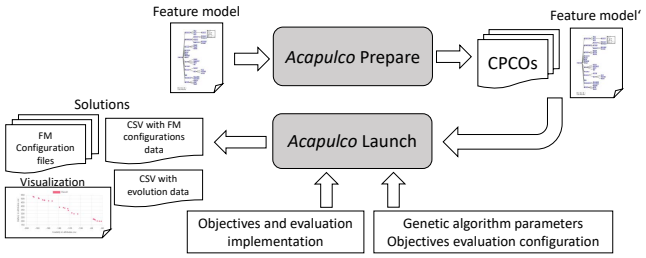
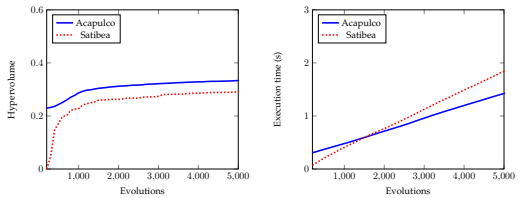


Figure 5: Two steps for the user: Preparation and Launching.

User perspective. Figure 5 illustrates the process consisting of two steps, namely, preparation and launching. For the preparation, the input is only the feature model in any format supported by FeatureIDE. The output are the CPCOs and eventually, in the case of complex or pseudo-complex constraints, a modified feature model is created as a separate file. If there are no modifications in the feature model, this step only needs to be conducted once, and the output is reused for as many *Acapulco* launches as desired (e.g., with different objective functions). The inputs for launching are the user-defined genetic algorithm parameters where the default values can be changed, and the selection of objectives and the user-defined configuration of each of them. Obviously, to use the objectives, they must be registered through the mentioned extension point. Otherwise, the objective and its evaluation implementation must be created. As mentioned before, the results of this step are the configuration files of the Pareto-front, a CSV file with data about each of those configurations, and an HTML file with a visualization of the Pareto-front for the case of multi-objective optimization.

4 EXPERIENCE WITH REAL-WORLD SPL

In [4], we report on experimental evaluation of *Acapulco* on a set of software product lines. These include small-scale product lines like MobileMedia, but also large industrial product lines, such as the Linux and Automotive product lines often used for experimental evaluations in the SPL literature. Our evaluation shows that *Acapulco* scales to these large-scale product lines and improves over state-of-the-art tools like MODAGAME [10] and SATIBEA [2]. Once the product line has been prepared for use with *Acapulco* (including the generation of the CPCOs), *Acapulco* finds better solutions faster than both tools, as can be seen in Fig. 6 for the Linux 2.6 feature model [6]. In this case with large number of features and constraints, MODAGAME did not return any valid solutions.



(a) Hypervolume. (b) Execution time.

Figure 6: Convergence of solutions (a) and performance (b) of *Acapulco* compared to SATIBEA for the Linux 2.6 feature model (extracted from [4]).

5 CONCLUSIONS

Acapulco is a tool that provides mono- and multi-criteria optimization functionalities in FeatureIDE. It guarantees the validity of the obtained configurations when there are no complex constraints. In addition, *Acapulco* supports transformation of complex constraints based on an existing method. *Acapulco* allows the flexible contribution of objective functions, opening the door to its usage in very diverse application domains.

From a user perspective, it might be difficult to adjust the parameters of the genetic algorithm. Empirical research can be performed to provide certain guidelines or automatically recommended values in this regard. Also, for recommendation, once the Pareto-front is obtained, functionality to make the final decision could be included.

ACKNOWLEDGMENTS

The work of Jose-Miguel Horcas was supported by the Spanish SRUK/CERU (On the Move) 2018/2019, and the projects *MEDEA* (RTI2018-099213-B-I00), *IRIS* (PID2021-122812OB-I00), *Rhea* (P18-FR-1081), *LEIA* (UMA18-FEDERIA-157), *DAEMON* (H2020-101017109), *OPHELIA* (RTI2018-101204-B-C22), *COPERNICA* (P20_01224), and *METAMORFOSIS* (FEDER_US-1381375). The work of Daniel Strüber was partially supported by the Deutsche Forschungsgemeinschaft (DFG), grant 413074939.

REFERENCES

[1] David Benavides, Pablo Trinidad, and Antonio Ruiz-Cortés. 2005. Automated reasoning on feature models. In *CAISE*. https://doi.org/10.1007/11431855_34
[2] Christopher Henard, Mike Papadakis, Mark Harman, and Yves Le Traon. 2015. Combining Multi-Objective Search and Constraint Solving for Configuring Large Software Product Lines. In *ICSE*. 517–528. https://doi.org/10.1109/ICSE.2015.69
[3] José Miguel Horcas, Mónica Pinto, and Lidia Fuentes. 2022. Empirical analysis of the tool support for software product lines. *Software and Systems Modeling* (2022). https://doi.org/10.1007/s10270-022-01011-2
[4] Jose Miguel Horcas, Daniel Strüber, Alexandru Burdusel, Jabier Martinez, and Steffen Zschaler. 2022. We're Not Gonna Break It! Consistency-Preserving Operators for Efficient Product Line Configuration. *IEEE Transactions on Software Engineering* (2022). https://doi.org/10.1109/TSE.2022.3171404
[5] 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. Carnegie-Mellon University Software Engineering Institute.
[6] Alexander Knüppel, Thomas Thüm, Stephan Mennicke, Jens Meinicke, and Ina Schaefer. 2018. Is There a Mismatch between Real-World Feature Models and Product-Line Research?. In *ESEC/FSE*. 53–54. https://dl.gi.de/20.500.12116/16312
[7] Jabier Martinez, Jean-Sébastien Sottet, Alfonso García Frey, Tegawendé F. Bissyandé, Tewfik Ziadi, Jacques Klein, Paul Temple, Mathieu Acher, and Yves Le Traon. 2018. Towards Estimating and Predicting User Perception on Software Product Variants. In *17th ICSR*. https://doi.org/10.1007/978-3-319-90421-4_2
[8] Jabier Martinez, Jean-Sébastien Sottet, Alfonso García Frey, Tewfik Ziadi, Tegawendé F. Bissyandé, Jean Vanderdonckt, Jacques Klein, and Yves Le Traon. 2017. Variability Management and Assessment for User Interface Design. In *Human Centered Software Product Lines*. Springer, 81–106.
[9] Rafael Olaechea, Steven Stewart, Krzysztof Czarnecki, and Derek Rayside. 2012. Modelling and Multi-Objective Optimization of Quality Attributes in Variability-Rich Software. In *NFPinDSML*. 2:1–6. https://doi.org/10.1145/2420942.2420944
[10] Gustavo G. Pascual, Roberto E. Lopez-Herrejon, Mónica Pinto, Lidia Fuentes, and Alexander Egyed. 2015. Applying multiobjective evolutionary algorithms to dynamic software product lines for reconfiguring mobile applications. *Journal of Systems and Software* 103 (2015), 392–411. https://doi.org/10.1016/j.jss.2014.12.041
[11] Juliana Alves Pereira. 2014. Search-Based Product Configuration in Software Product Lines. http://hdl.handle.net/1843/ESBF-9Q4FQ9
[12] Juliana Alves Pereira, Mathieu Acher, Hugo Martin, Jean-Marc Jézéquel, Goetz Botterweck, and Anthony Ventresque. 2021. Learning software configuration spaces: A systematic literature review. *Journal of Systems and Software* 182 (2021).
[13] Norbert Siegmund, Marko Rosenmüller, Martin Kuhlemann, Christian Kästner, Sven Apel, and Gunter Saake. 2012. SPL Conqueror: Toward optimization of non-functional properties in software product lines. *Software Quality Journal* 20, 3-4 (2012), 487–517. https://doi.org/10.1007/s11219-011-9152-9
[14] Daniel Strüber, Kristopher Born, Kanwal Daud Gill, Raffaela Groner, Timo Kehrer, Manuel Ohrndorf, and Matthias Tichy. 2017. Henshin: A usability-focused framework for EMF model transformation development. In *ICGT*. 196–208.
[15] Thomas Thüm, Christian Kästner, Fabian Benduhn, Jens Meinicke, Gunter Saake, and Thomas Leich. 2014. FeatureIDE: An extensible framework for feature-oriented software development. *Science of Computer Programming* 79 (2014), 70–85. https://doi.org/10.1016/j.scico.2012.06.002