23rd International Systems and Software Product Line Conference

September 9 - 13, 2019
Paris, France

Taken from Pixabay under CC0 1.0

Proceedings - Volume A

EDITED BY:

Thorsten Berger, Philippe Collet, Laurence Duchien, Thomas Fogdal, Patrick Heymans, Timo Kehrer, Jabier Martinez, Raúl Mazo, Leticia Montalvillo, Camille Salinesi, Xhevahire Tërnava, Thomas Thüm, Tewfik Ziadi

SORBONNE UNIVERSITÉ

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

ICPS
Published by ACM

Table of Contents

Domain Implementation

Solution-Space Analysis

Challenges and Solutions

Emerging Application Areas

Community Efforts

Requirements Engineering

Metrics and Refactoring

Feature-Model Evolution

Configuration and Sampling

Problem-Space Analysis

Workshops

Tutorials

Achieving Change Requirements of Feature Models by an Evolutionary Approach

Extended Abstract

Paolo Arcaini
National Institute of Informatics
Tokyo, Japan
arcaini@nii.ac.jp

Angelo Gargantini
University of Bergamo
Bergamo, Italy
angelo.gargantini@unibg.it

Marco Radavelli
University of Bergamo
Bergamo, Italy
marco.radavelli@unibg.it

CCS CONCEPTS

• **Software and its engineering** → **Software product lines**; *Search-based software engineering*; • **Theory of computation** → *Evolutionary algorithms*;

KEYWORDS

feature models, software product lines, mutation, search-based software engineering

DESCRIPTION OF THE JOURNAL-FIRST PAPER

Software Product Lines (SPLs) are families of products that share some common features, and differ on some others. The variability of SPLs is usually described at design time by using *variability models*; one of the main used variability models are *feature models* (FMs).

Overtime, feature models need to be updated in order to avoid the risk of having a model with wrong features and/or wrong constraints. Two main causes for *change requirements* exist: either the model is incorrect (it excludes/includes some products that should be included/excluded), or the SPL has changed. The change requirements come from different sources: *failing tests* identifying configurations evaluated not correctly, or *business requirements* to add new products, to allow new features, to not support some products any more, and so on. All these change requirements identify configurations/features to add or remove, but do not identify a way to modify the feature model to achieve them. Manually updating a feature model to achieve all the change requirements can be particularly difficult and, in any case, error-prone and time consuming.

In the corresponding *journal-first paper* [1], we propose an approach (shown in Fig. 1) to automatically update a feature model upon change requirements. The user must only specify an *update request*, based on the change requirements coming from testing or from business requirements. The update request is composed of
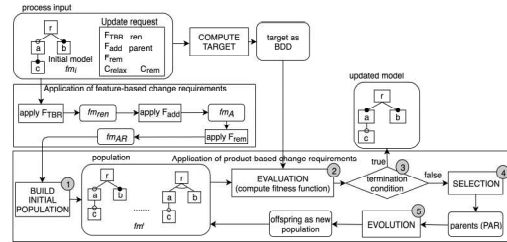

Figure 1: Proposed evolutionary approach

three kinds of *feature-based* change requirements and two kinds of *product-based* ones; the feature-based ones consist in features that must be renamed, and features that must be added to and removed from the products of the original feature model; the product-based ones, instead, consist in configurations that should be no more accepted as products by the final model, and configurations that should instead be accepted as new products. Starting from the update request, the approach tries to apply the feature-based change requirements directly on the starting model; however, some of these requirements could be not completely fulfilled. Then, by means of an evolutionary algorithm, the approach tries to obtain a feature model that captures all the change requirements: the feature-based ones not fulfilled in the previous phase, and the product-based ones. The process iteratively generates, from the current population of candidate solutions, a new population of feature models by mutation. Population members are evaluated considering primarily the percentage of correctly evaluated configurations, and secondly a measure of the *structural complexity* of the model, defined in terms of number of cross-tree constraints. When a correct model is found or some other termination condition holds, the process terminates returning as final model the one having the highest fitness value.

The approach has been evaluated on real-world feature models; experiments show that, on average, around 89% of requested changes are applied.

ACKNOWLEDGMENTS

REFERENCES

[1] Paolo Arcaini, Angelo Gargantini, and Marco Radavelli. 2019. Achieving change requirements of feature models by an evolutionary approach. *Journal of Systems and Software* 150 (2019), 64–76. https://doi.org/10.1016/j.jss.2019.01.045