OPTI-SELECT: An Interactive Tool for User-in-the-Loop Feature Selection in Software Product Lines

Ahmed Eid El Yamany
College of Computing and Information Technology
Arab Academy for Science, Technology, and Maritime Transport
Cairo, Egypt
Ahmedeid100@gmail.com

Mohamed Shaheen
College of Computing and Information Technology
Arab Academy for Science, Technology, and Maritime Transport
Alexandria, Egypt
cshaheen@hotmail.com

Abdel Salam Sayyad
Lane Department of Computer Science and Electrical Engineering
West Virginia University
Morgantown, WV, USA
asayyad@mix.wvu.edu

ABSTRACT
Opti-Select is an Interactive Multi-objective feature analysis and optimization tool for software product lines configuration and feature models optimization based on an innovative UIL (User-In-the-loop) idea. In this tool, the experience of system analysts and stakeholders are merged with optimization techniques and algorithms.
Opti-Select interactive tool is an integrated set of techniques providing step by step feature model and attribute configuration, selecting and excluding features, solution set optimization, and user interaction utilities that can all together reach satisfactory set of solutions that fits stakeholder preferences.

Categories and Subject Descriptors
D.2.1 [Requirements]: Tools; D.2.13 [Reusable Software]: Domain engineering; G.1.6 [Optimization]: Constrained optimization.

General Terms
Algorithms, Management, Performance, Design, Economics, Reliability, Experimentation, Human Factors, modeling, features, optimal variant, feature modeling, product line engineering, Pareto front visualization, exploration

Keywords
User-in-the-loop (UIL), Software Product Lines, Feature Models, Optimal Feature Selection, Multi-objective Optimization, Search-Based Software Engineering.

1. INTRODUCTION
Opti-Select is a result of research experiments that investigated the best ways to empower the user in the process of feature model configuration.

The tool adopts a new exploration technique which merges the expert vision with evolutionary algorithms optimization to produce shortlisted set of solutions which fits best for user needs.

Opti-Select takes input from the user to specify the preferences that the user is most interested in and will enable the user to interact with the optimization process; which is called the "user-in-the-loop" approach.

The optimization process takes place in an incremental form. After each round of optimization, the user is provided with a concise presentation of the multiple solutions that make up the Pareto Front allowing the user to mark their preferred solutions to focus on producing related solutions in the following iterations.

The following sections discuss the background of our experiments then it provides a detailed demonstration for Opti-Select and how it is optimized to achieve user goals.[1]

SPLC '14 , Sep 15-19 2014, Florence, Italy
ACM 978-1-4503-2739-8/14/09.
http://dx.doi.org/10.1145/2647908.2655977

2. Feature Models Optimization
The application aims to optimize product lines represented as feature models to optimize dynamic set of objectives best fit user's environment.

2.1 Product Lines and Feature Models
Increasingly, software engineers spend their time creating software families consisting of similar systems with many variations [1]. Many researchers in industry and academia started using a feature-oriented approach to commonality and variability analysis after the Software Engineering Institute introduced Feature-Oriented Domain Analysis (FODA) [2].

A feature is an end-user-visible behavior of a software product that is of interest to some stakeholder. A feature model represents the information of all possible products of a software product line in terms of features and relationships among them. Feature models are a special type of information model widely used in software product line engineering.

2.2 Feature Models with Attributes
Optimizing feature models by attaching each feature a set of attributes is based on the idea of extending (or augmenting) feature models with quality attributes that was proposed by many, among them Czarnecki et al. [3], Zhang et al. [4] and Cordy et al. [5]. Many researchers used feature models with attached attributes to experiment with optimizing feature selection in Software

[1] This publication was made possible by NPRP grant # [09-1205-2-470] from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

Product Lines, including Benavides et al. [6], White et al. [7], [8], Bagheri et al. [9], Soltani et al. [10] and Guo et al. [11].

2.3 Multi-Objective Evolutionary Algorithms

A comparison among various multi-objective search-based software engineering methods was proposed by A. Sayyad [12]. It has shown that IBEA [13] performs much better than methods in widespread use especially with increased number of optimization objectives. IBEA works best since it makes most use of user preference knowledge. It also generates far more products without violations of domain constraints.

3. Application Features

Opti-Select interactive optimization tool is enriched with a set of user empowering features that helps analysts reach a desired and highly optimized Pareto front solution set.

3.1 Model and Attribute Configuration

The tool can load dynamically SXFM formatted feature model.

The Simple XML Feature Model (SXFM) format was defined by the SPLOT website [14].

Configuration attribute can be loaded or saved to a separate formatted file linking each feature with its attributes.

Pareto Front Results can be loaded from a saved Pareto front saved file where results from previous runs can be used for specifying next iterations indicators.

As soon as a model loads to the application, the accompanied attributes are displayed allowing the user to alter the values of each attribute attached to every feature. As shown in figure 1, the user can navigate SXFM loaded features and display dynamically viewed attributes columns (based on custom objectives setting), change attributes values, or change the feature to be selected or excluded.

The user can select from a wide range of objectives to optimize. That dynamic objective selection allows the user to modify each related attribute value and dynamically apply optimization process based on selected objectives.

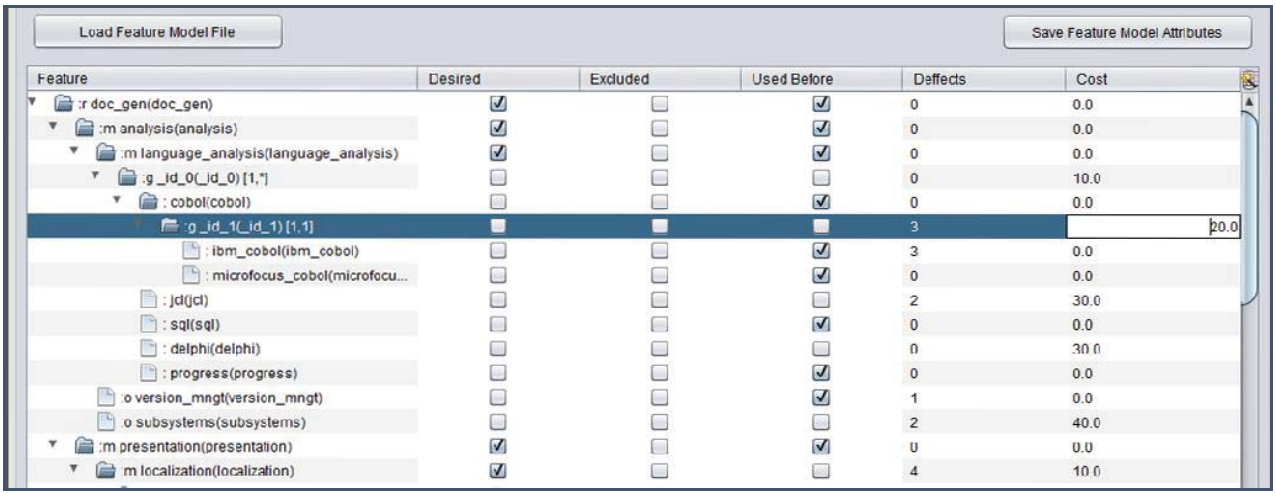

Figure 1. Attributes Configuration

3.2 Selecting and Excluding Features

Whether the current status is before the first iteration run or between optimization incremental iterations, the user can specify features to be selected (forced) or excluded which will make the next generated solution obey the user desires and prevent the mutation process from changing the values of those features.

The application detects whether forcing and excluding features will violate feature trees structure, cross tree constraints, group constraints, or feature model mandatory features. It will try to correct forced and excluded features situation automatically if it can. If not, it will notify the user with a list of corrective actions (see Figure 2) to choose between or it gives him the option to roll back his change. A more advanced option will be shown for experienced users to ignore corrective actions list and let the optimization process decide the new Pareto front solution set.

3.3 Solution Set User Interaction

The user can run the optimization process on many incremental steps. Each step can be granted a time limit in millisecond allowing the user to visualize or modify solution set between steps.

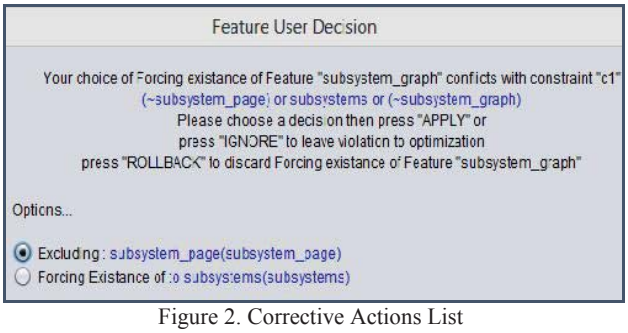

Figure 2. Corrective Actions List

After each decided limited time incremental run, the application will display the current summary of solution set to the user. The user can visualize details of each solution and then they can decide which of the current solutions satisfy their vision of the final solution. Based on that, the user will omit undesired solutions so that they do not enter the next evaluation run. Only

the marked solutions will be used as offspring for the rest of the optimization process.

3.3.1 Solution Details

The user can display the details of each solution in the solution set. Details display will show a tree structure respected view of this solution (Figure 3) and all objectives values related to this solution (Table 1). It gives the user a wider look of this solution to figure out which to pick out for next iteration if it serves their final desires or not.

Objective	Value
Correctness	100
Number Of Features	
Used Before	12
Deffects	6
Cost	500
Time	340

Table 1. Solution Details Attached Parameter Sample

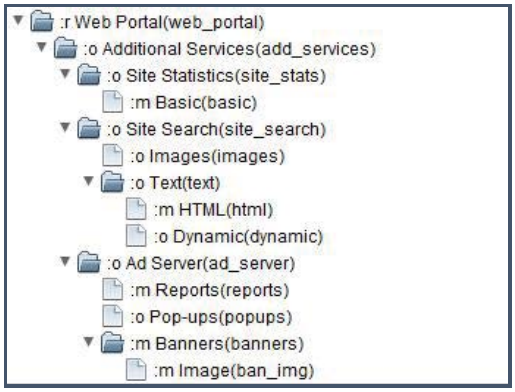

Figure 3. Sample Solution Details Tree View

3.4 Objectives Dynamic Settings

The user is allowed to specify needed objectives to be optimized prior to any optimization runs. The user is also allowed to change his desired objectives between runs and change each feature attached attribute value. That gives the user the power to use a desired solution set that resulted from some objectives optimization for specific time as an offspring for a specific objective optimization.

Figure 4 displays a wide list of quality attributes that can be expanded in next versions. Correctness and number of features are dimmed as application assumes that optimizing correctness and richness of product line is essential for good software design.

List of attributes attached to each feature is dynamically configurable based on objectives' settings. While in optimization process, each solutions is evaluated based on current objectives' settings by calculating their related attributes values.

4. Mechanism

The application reads and parses feature models using depth first algorithm into memory, Attributes lists are generated and linked to features based on objectives' settings.

First optimization iteration generates initial population with respect to user selected and excluded features. Next iterations preserve solutions selected by the user.

The application gives the user the power to dynamically control each step of the flow like loading feature models, objectives' settings, and interaction with optimization middle solution sets. That dynamic policy requires a careful technical and performance handling of application inputs [15].

Figure 4. Objectives Dynamic Setting

To achieve dynamic attributes and objectives' settings, an objective manager is introduced to map attributes lists attached to features nodes to evaluation generic method. It also links current settings to user interface enabling the user to reconfigure attributes values.

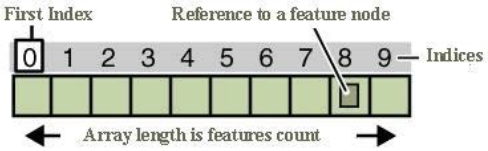

Figure 5. Feature Model Indices Array

Performance and memory management are essential especially when searching large feature model trees attached with dynamic multi-objective attributes. A sorted index array is introduced to hold references for tree features nodes (Figure 5). Features tree is traversed using depth first algorithm once prior to optimization iteration to generate a sorted index array. This Index provides O (1) direct access to features properties and attributes. Hence, evaluation processes and search performance are optimized by avoiding tree repetitive search and tree diving recursion overhead.

5. Related Work Discussions and Comparison

Botterweck G. [16] feature configuration tool S2T2 Configurator integrates a visual interactive representation of the feature model and a formal reasoning engine that calculates consequences of the user's actions and provides formal explanations. Still it didn't provide multi-objective support nor incremental configuration.

FAMA [17] is a framework for the automated analysis of feature models integrating some of the most commonly used logic representations and solvers proposed for automated analyses of feature models.

The Feature Model Plugin (FMP) [18] is implemented as an Eclipse plug–in. It supports configuration based on feature diagrams. But it does not have the analysis of FMs among its main goals. It does not support attributed feature models.
CaptainFeature is a feature modelling tool using the FODA notation to render and configure feature diagrams. It does not support the automated analysis of FMs.

Clafer [19] is a lightweight yet expressive language for structural modeling: feature modeling and configuration, class and object modeling.

	Feature Model Representation	Iterative Configuration	Multi-Objective Optimization	Features Automated Analysis	Attributed Feature Model
S2T2	☑	☐	☐	☐	☐
FAMA	☑	☐	☐	☑	☐
FMP	☑	☐	☐	☐	☐
CaptainFeature	☑	☐	☐	☐	☑
Clafer	☑	☐	☑	☑	☑
Opti-Select	☑	☑	☑	☑	☑

Table 2. Summary of Feature Configuration Proposals

6. Conclusion

We presented how Opti-Select interactive multi-objective feature analysis and optimization tool will add a value to product-line analysis and feature optimization tasks. Our application is a hybrid between utilizing system analysts and experts' knowledge and evolutionary algorithms optimization techniques that brings highly-optimized alternative solutions to hands of user in an expressive method.

Opti-Select excels in providing a set of preferences and solutions to the users based on their interaction with the optimization algorithm instead of providing the user with a single solution or a huge set of solutions which may not be useful and confusing.

7. ACKNOWLEDGMENT

Our thanks to Dr. Tim Menzis and to Dr. Hany Ammar from West Virginia University for their valuable advices and providing access to benchmarks.

8. REFERENCES

[1] Coplien J., Hoffman, D., and Weiss D. 1998. Commonality and variability in software engineering. *Software.* IEEE, 15, 1, 37-45.

[2] Kang K. C., Lee J., and Donohoe P. 2002. Feature-oriented product line engineering. *IEEE software*, 19, 4, 58-65 pages.

[3] Czarnecki K., Helsen S., and Eisenecker U. 2005. Formalizing cardinality based feature models and their specialization. *Software process: Improvement and practice*, 10, 1, 7-29.

[4] Zhang G., Ye H., and Lin Y. 2011. Using knowledge-based systems to manage quality attributes in software product lines. In *Proceedings of the 15th International Software Product Line Conference*. ACM, 2, 32 (August. 2011).

[5] Cordy M., Schobbens P. Y., Heymans P., and Legay A. 2013. Beyond boolean product-line model checking: dealing with feature attributes and multi-features. In *Proceedings of the 2013 International Conference on Software Engineering* (May. 2013). IEEE Press, 472-481.

[6] Benavides D., Trinidad P., and Ruiz-Cortés, A. 2005. Automated reasoning on feature models. In *Advanced Information Systems Engineering* (Jan. 2005). Springer Berlin Heidelberg, 491-503.

[7] White J., Dougherty B., Schmidt D. C., and Benavides D. 2009. Automated reasoning for multi-step feature model configuration problems. In *Proceedings of the 13th International Software Product Line Conference* (Aug. 11-20, 2009), Carnegie Mellon University.

[8] White J., Dougherty B., and Schmidt, D. C. 2009. Selecting highly optimal architectural feature sets with filtered cartesian flattening. *Journal of Systems and Software*, *82*, 8, 1268-1284.

[9] Bagheri E., Asadi M., Gasevic D., and Soltani S. 2010. Stratified analytic hierarchy process: Prioritization and selection of software features. In *Software Product Lines: Going Beyond*. Springer Berlin Heidelberg, 300-315.

[10] Soltani S., Asadi M., Gašević D., Hatala M., and Bagheri E. 2012. Automated planning for feature model configuration based on functional and non-functional requirements. In *Proceedings of the 16th International Software Product Line Conference.* ACM, Vol. 1 (Sep. 2012), 56-65.

[11] J. Guo, J. White, G. Wang, J. Li, and Y. Wang. 2011. A Genetic Algorithm for Optimized Feature Selection with Resource Constraints in Software Product Lines. *Journal of Systems and Software*. 84, 12 (Dec. 2011), 2208–2221.

[12] Sayyad A. S., Menzies T., and Ammar H. 2013. On the value of user preferences in search-based software engineering: A case study in software product lines. In *Software Engineering (ICSE), International Conference* (May. 2013). IEEE, 492-501.

[13] Zitzler E. and Künzli S. 2004. Indicator-based selection in multiobjective search. In *Parallel Problem Solving from Nature-PPSN,* 3 (Jan. 2004). Springer Berlin Heidelberg, 832-842.

[14] Mendonca M., Branco M., and Cowan D. 2009. SPLOT: software product lines online tools. In *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications* (Oct. 2009). ACM, 761-762.

[15] Sayyad A. S., Ingram J., Menzies T., and Ammar, H. 2013. Scalable product line configuration: A straw to break the camel's back. In *Automated Software Engineering (ASE). IEEE/ACM 28th International Conference* (2013, November). IEEE, 465-474.

[16] Botterweck G., Janota M., and Schneeweiss D. 2009. A Design of a Configurable Feature Model Configurator. *VaMoS*, *29*, 165-168.

[17] Benavides D., Segura S., Trinidad P., and Cortés, A. R. 2007. FAMA: Tooling a Framework for the Automated Analysis of Feature Models. *VaMoS*, *2007*, 01.

[18] Czarnecki, K. and Kim, C. H. P. 2005. Cardinality-based feature modeling and constraints: A progress report. In *International Workshop on Software Factories* (2005, October), 16-20.

[19] Antkiewicz. M., Bąk K., Murashkin A., Olaechea R., Liang, J. H. J., and Czarnecki K. 2013. Clafer tools for product line engineering. In *Proceedings of the 17th International Software Product Line Conference co-located workshops (*2013, August). ACM*,* 130-135.