Visualization and Exploration of Optimal Variants in Product Line Engineering

Alexandr Murashkin, Michał Antkiewicz, Derek Rayside, Krzysztof Czarnecki
University of Waterloo, Waterloo, Canada

ABSTRACT

The decision-making process in Product Line Engineering (PLE) is often concerned with variant qualities such as cost, battery life, or security. *Pareto-optimal* variants, with respect to a set of objectives such as minimizing a variant's cost while maximizing battery life and security, are variants in which no single quality can be improved without sacrificing other qualities. We propose a novel method and a tool for visualization and exploration of a multi-dimensional space of optimal variants (*i.e.*, a *Pareto front*). The visualization method is an integrated, interactive, and synchronized set of complementary views onto a Pareto front specifically designed to support PLE scenarios, including: understanding differences among variants and their positioning with respect to quality dimensions; solving trade-offs; selecting the most desirable variants; and understanding the impact of changes during product line evolution on a variant's qualities. We present an initial experimental evaluation showing that the visualization method is a good basis for supporting these PLE scenarios.

Categories and Subject Descriptors

D.2.11 [**Software Architectures**]: Languages; D.2.13 [**Reusable Software**]: Domain engineering; F.4.1 [**Mathematical Logic**]: Logic and constraint programming; F.4.3 [**Formal Languages**]: Decision problems; G.1.6 [**Optimization**]: Constrained optimization; H.5.2 [**User Interfaces**]: Interaction styles

Keywords

Pareto front, optimal variant, feature modeling, product line engineering, visualization, exploration, Clafer, ClaferMoo, ClaferMoo Visualizer

1. INTRODUCTION

Product Line Engineering (PLE) is an approach to engineering a family of different products with a shared architecture and common components [5]. Individual product variants are characterized by a selection of features from the product family's feature model [6]. For example, a product line of mobile phones defines connectivity features such as `GSM` and `LTE` (communication standards). A particular variant of the product line may include the `GSM` feature but exclude the `LTE` feature. Some features can contribute positively or negatively to the measurable qualities of a variant [1]. For example, `LTE` increases a phone user's productivity but increases the phone's cost. The challenge faced by product-line engineers (herein, "engineers") is discovering which variants are optimal with respect to a set of objectives (such as to *maximize productivity* and to *minimize cost*) and what trade-offs (*e.g.*, productivity level versus acceptable cost) to resolve when choosing among optimal variants.

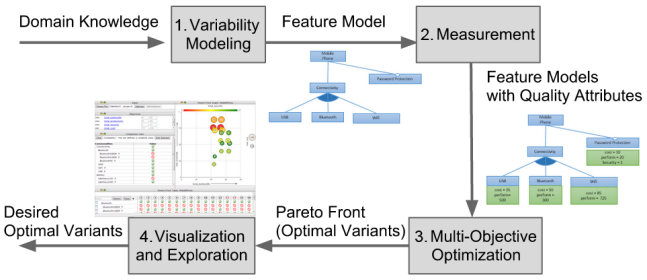

Figure 1: Decision Making Workflow in PLE.

In this paper, we consider the following workflow (Fig. 1). In the first phase, *Variability Modeling*, the engineers create a product line's feature model. In our example, the feature model (expressed in Clafer [3]) describes the product line `MobilePhone` with a set of features (Fig. 2a). Each line defines a feature, and indentation indicates feature nesting (*e.g.*, `GSM` is a subfeature of `Connectivity`). The question mark `?` indicates optional features. The keyword `xor` indicates exclusive-or feature groups (*e.g.*, `xor Bluetooth`). In Clafer models, features are mandatory by default unless specified otherwise.

In the next phase, *Measurement*, the engineers define quality attributes and individual contributions of each feature. They can define a single `Feature` concept with four quality attributes (Fig. 2b) and then specify the exact contributions for each `Feature`, like `LTE` (Fig. 2b). Alternatively, they could define more specific concepts like `SecurityFeature` or `BatteryFeature` for convenience. Performing the actual measurement and discovering the exact values of individual feature contributions is outside the scope of this paper [15].

SPLC 2013 August 26 - 30 2013, Tokyo, Japan

```
abstract MobilePhone
  Connectivity
    xor Bluetooth
      Bluetooth20EDR : Feature ?
      Bluetooth21EDR : Feature ?
      Bluetooth40 : Feature ?
    GSM : Feature
    LTE : Feature ?
    WiFi : Feature ?
    USB : Feature ?
  xor Battery
    LiBattery1150 : Feature ?
    LiBattery1400 : Feature ?
```

(a) Feature Model.

```
abstract Feature
  productivity : integer
  batterylife : integer
  security : integer
  cost : integer

abstract MobilePhone
...
  LTE : Feature ?
     [ productivity = 12 ]
     [ batterylife = -15 ]
     [ security -10 ]
     [ cost = 44 ]
...
```

(b) Feature Qualities.

```
abstract MobilePhone
...
  total_productivity = sum Feature.productivity
  total_batterylife = sum Feature.batterylife
  total_security = sum Feature.security
  total_cost = = sum Feature.cost
```

(c) Variant Qualities.

```
aPhone : MobilePhone

<< max aPhone.total_productivity >>
<< max aPhone.total_batterylife >>
<< max aPhone.total_security >>
<< min aPhone.total_cost >>
```

(d) Optimization Objectives.

Figure 2: Feature Modeling with Quality - Example.

Next, the engineers define variant's quality attributes by an **arbitrary constraint**: just a sum of all individual feature contributions (Fig. 2c) or, if necessary, a more complex formula accounting for feature interactions. Finally, the engineers define optimization objectives (Fig. 2d).

In the third phase (Fig. 1), *Multi-objective Optimization*, the engineers compute a set of non-dominated optimal variants—a *Pareto front*—with respect to the stated optimization objectives. Throughout the paper, we refer to such variants as *Pareto-optimal*, or just *optimal* variants. In this phase, one can use a multi-objective optimizer such as ClaferMoo [11]. ClaferMoo takes a model with quality attributes and objectives as an input, performs a multi-objective optimization, and generates the set of optimal variants: in our example, 16 variants. Now the engineer must decide which of the optimal variants to actually produce.

Optimizers like ClaferMoo only focus on Pareto front computation without supporting its exploration: the final step in the workflow (Fig. 1) still requires tool support due to the following challenges. First, the engineers need a global view of the Pareto front to observe the variants in relation to each other, quality ranges, and feature occurrence frequency. Next, they need to perform a trade-off analysis and filter the variants according to the desired features and quality values. Finally, they need to explore and understand the impact of changes during product line evolution that can cause Pareto fronts to change, *e.g.*, some previously optimal variants may become sub-optimal when the feature model evolves.

To address these challenges, we propose a method and a tool for interactive Pareto front visualization and exploration. The tool, called *ClaferMoo Visualizer*[1], accepts an attributed feature model with quality attributes and optimization objectives, invokes ClaferMoo to solve the optimization problem, and presents the engineers with a graphical user interface (GUI) in which they can explore a Pareto front. The GUI is a web-based, integrated, interactive, and synchronized set of complimentary views onto a Pareto front. The visualizer could be used with Pareto-front computation tools other than ClaferMoo.

As the initial evaluation of our Pareto-front visualization and exploration tool, we performed a small controlled experiment. The evaluation confirmed that both the visualization method and the tool are effective in helping participants perform a set of tasks. Additionally, the evaluation revealed new functional requirements for the tool which we have subsequently implemented.

[1]https://github.com/gsdlab/claferMooVisualizer

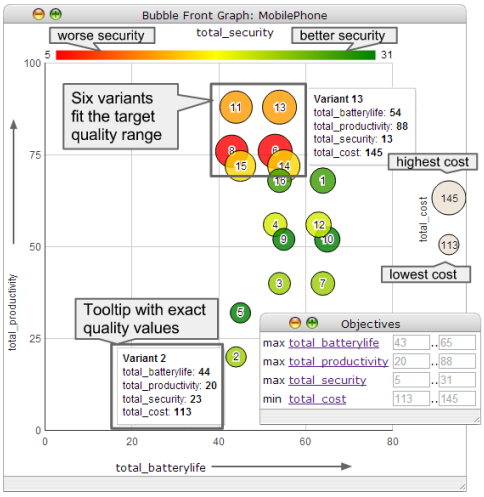

Figure 3: The *Bubble Front Graph* and *Objectives* views for the mobile phone example. The graph shows all 16 generated variants. The six variants fit the quality target range with security less than 20 and cost within [125..145].

Our paper describes examples of Pareto front exploration with supporting visualizations, the experimental evaluation, the related work in this area, and our work's conclusions.

2. VISUALIZATION AND EXPLORATION

This section considers the *Visualization and Exploration* phase of the workflow in the context of our mobile phone example and our users—product line engineers. We propose the *Bubble Front Graph* as a visualization to give the engineer a global view for exploring the Pareto front.

2.1 Bubble Front Graph

The Bubble Front Graph (Fig. 3) represents each variant as a bubble.[2] The graph uses four representations: horizontal axis X (bottom), the vertical axis Y (left), bubble color Z (top) and bubble size T (right) to visualize up to four quality dimensions simultaneously.

In our example (Fig. 3), `productivity` is represented as a vertical coordinate (Y); `batterylife` is represented as a horizontal coordinate (X); `security` is represented as a color (Z); and `cost` is represented as a size (T).

The Bubble Front Graph (Fig. 3) shows all 16 variants that ClaferMoo has generated for our example. The interpretation of bubble representations depends on the optimization objectives: for maximization, the larger the value, the better; for minimization, the smaller the value, the better. In our example, the greener the bubble, the better security, and the smaller the bubble size, the better (smaller) cost. The graph also supports tooltips to allow inspection of exact quality values. For instance, for variant 2, battery life is 44, productivity is 20, security is 23, and cost is 113.

[2]Our graph is implemented as an extension to Bubble chart of Google Chart Tools: https://developers.google.com/chart

The graph offers many interesting opportunities of exploration. First, an engineer can identify ranges of optimal variants' quality values by looking either at bubble placement (`batterylife` [43..65]) or directly at the label (`cost` [113..145]). Next, all variants are sorted by X and Y values: the variant 2 is the least productive, while the top productivity belongs to the variants 11 and 13. An engineer can sort variants in different dimensions by assigning third and fourth quality attributes to the X and Y representations.

An engineer can see how variants are distributed, identify locations with a high density. The engineer can consider exploring the largest bubbles (6, 8, 11, 13, 14 and 15) separately from all other bubbles because these ones are quite close to each other in all four dimensions: their X and Y positions differ by small amounts; the variants' cost is almost the same; and the variants' security is below the average.

One way of focusing on a graph area is filtering the Pareto front by specifying *target quality ranges* in the view `Objectives`. For example, to show the six desired ones only (Fig. 3), an engineer can set the ranges as follows: cost [125..145], (upper and lower bounds), security [..20] (upper bound only to exclude top security values), productivity [..] (all values), battery life [..] (all values). This operation does not recalculate the Pareto front, just narrows it down. We consider such filtering as a form of the top-down exploration approach: *quality range-driven* exploration.

An engineer can also discover dependencies and correlations among the qualities. In our example, the engineer can notice that the larger the cost, the more productive and less secure the variant is.

The Bubble Front Graph supports a top-down exploration of the Pareto front focused on quality metrics. Our tool offers another visualization for top-down exploration—a Feature and Quality Matrix.

2.2 Feature and Quality Matrix

Feature and Quality Matrix (further just "matrix") (Fig. 4) is an intuitive way to represent variants' features and quality values. The first column presents the product line's feature model (similarly to Fig. 2a). Each feature of the product line is shown in a new row without the feature's quality attributes. The last rows represent variants' quality values, *e.g.*, `total_batterylife`. The remaining columns, labeled by numbers, represent variants (from 1 to 16). A content cell is: a green tick or a crossed circle to indicate the presence or absence of a feature; an empty cell to indicate the presence of an *effectively mandatory* feature (present in every variant); or a variant's numeric quality value. For example, the feature `Connectivity` is effectively mandatory; the variant 1 has `Bluetooth40`, does not have `WiFi`, and costs 125.

Since the matrix deals with features, more scenarios are possible. First, the engineer can immediately see the effectively mandatory features, they are marked with an inactive but checked checkbox. Next, the engineer can also see features that commonly or rarely occur. For example, `USB` is included in 14 out of 16 variants. All variants exclude `Bluetooth20EDR`—it can be considered for deprecation.

Some features change their occurrence when filtering by a target quality range, because the matrix is filtered as soon as the graph is filtered. For example, filtering by the target quality range, as described above, causes the matrix to show only the six variants (6, 8, 11, 13, 14 and 15), and `WiFi` and `USB` are included in all the six variants. This may indicate

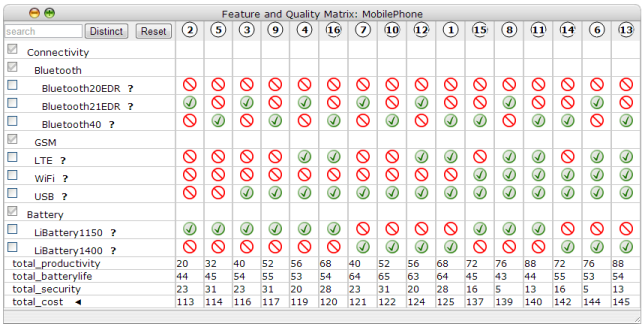

Figure 4: The Feature and Quality Matrix. Variants sorted in ascending order by `cost`.

that these two features contribute to cost.

The matrix also allows engineers to instantly filter variants by selecting features that should be present in all variants and eliminating features that should not be present in any of the variants without recalculation of the Pareto front. For example, in order to only see variants with `WiFi`, the engineer ticks the corresponding checkbox in front of `WiFi` (Figure 5). The matrix now shows only the six variants (6, 8, 11, 13, 14 and 15). Now, in order to see variants without `Bluetooth21EDR`, the engineer crosses out the corresponding checkbox and only four variants remain. Effectively mandatory features cannot be eliminated. We refer to this kind of *top-down exploration* as *feature-driven* exploration.

Feature-driven exploration can be combined with *quality range-driven* exploration. For example, instead of excluding `Bluetooth21EDR` explicitly, the engineer could specify the target quality range for `security` as [10..] to exclude the red bubbles and arrive at the same four variants.

The engineer can also sort the variants in the matrix by a variant number (default) and by each quality in ascending or descending order. In Fig. 4, the engineer sorted by `cost` in ascending order (as indicated by the small black triangle) and observed that features `USB` and `WiFi` begin to appear with variants 3 and 15, respectively, as the cost increases.

The matrix offers a function to make differences among variants more visible for feature impact and trade-off analysis. Pressing the button `Distinct` causes dimming of uniform rows and effective highlighting of the rows with differences. For example, Fig. 5 shows that the difference between these variants in terms of features is `LTE` and battery type: `LiBattery1150` or `LiBattery1400`. By looking at quality values, the engineer sees that `LTE` combined with `LiBat-`

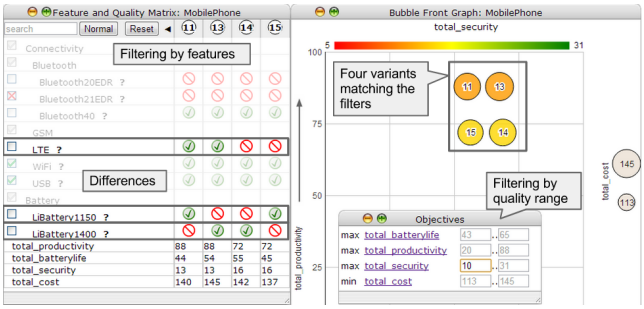

Figure 5: Filtering by features or by quality attribute values.

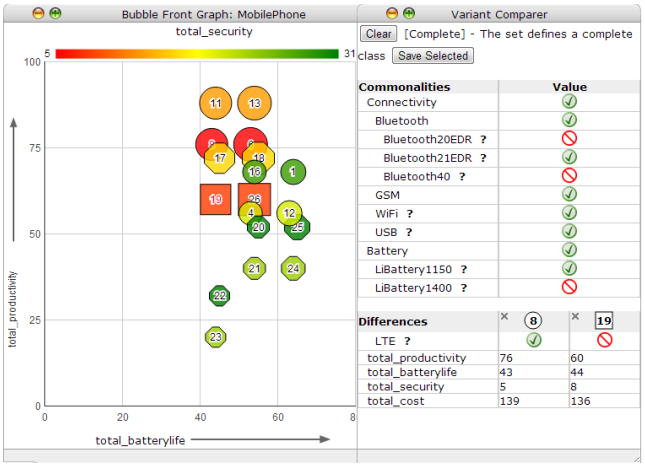

Figure 6: Comparison of two Pareto fronts.

tery1400 gives the highest cost of 145 and the top productivity of 88, while exclusion of LTE combined with the inclusion of the same battery gives the cost of 142, which can be considered as small in comparison to the productivity drop from 88 to 72. Therefore, the engineer chooses the variant 13 as it represents the most desired trade-off. It is important to emphasize that previously the engineer could only understand the individual contributions of a single feature to the overall variant quality. Now, the visualization and exploration allows engineers to achieve deeper insight into a cumulative effect of the presence of a feature set for analyzing more complex trade-offs.

The Feature and Quality Matrix supports several tasks related to Pareto front exploration by feature and comparison of variants. More complex comparisons of arbitrary selections of variants require another view—Variant Comparer.

2.3 Variant Comparer

The engineer may need to understand evolution of the whole Pareto front as the product line evolves. To support such scenarios, the tool allows loading pre-configured variants (*e.g.*, previously computed optimal variants or existing manually configured variants), and visualizing and exploring them together with the newly calculated variants (Fig. 6).

For example, consider a set of optimal variants of our product line at the time when the feature LTE was not yet available. After adding an optional feature LTE, the engineer generates a new Pareto front and loads the variants saved before, and they merge with new optimal variants. Figure 6 shows the resulting Bubble Front Graph. Pre-configured (previously generated) variants that exactly match the newly generated optimal variants are shown as octagons; otherwise, they are shown as squares. The engineer observes that eight old variants remain unchanged and that two old variants, 19 and 26, are not optimal. The engineer can see the achieved quality values: in the old Pareto front, the highest level of productivity achieved was 72, but now it is equal to 88. The engineer can compare non-optimal variants to optimal ones: non-optimal variants 19 and 26 are located below the similar-size optimal variants 8 and 6. The engineer first wants to compare the variants 19 to 8 to identify whether the former has evolved to the latter. To help with comparing arbitrarily selected variants, we designed the third view: *Variant Comparer*.

The *Variant Comparer* view (Fig. 6, right) is designed to present the commonalities and differences among the selected set of variants, identification of complete classes (explained further), and trade-off analysis. The view is dynamically updated whenever a variant is selected or unselected in the Bubble Front Graph or the Feature and Quality Matrix. The view is implemented as two tables: the top one for the common features, the bottom one is for the distinct features and all quality values. In our example, in order to compare the variants 19 and 8, the engineer first selects both variants. In the table *Differences*, the engineer realizes that 8 differs from 19 only by presence of the feature LTE, and that by adding this feature 19 would become optimal. The engineer can also see the impact on variant quality: a boost in productivity at the cost of small increase in price and decrease of battery life.

The view also shows that the selected set defines a complete class. A complete class is the largest set of variants that have exactly the same commonality—adding another variant would remove at least one feature from the commonality table. This notion helps to identify sets of similar variants—if the set is incomplete, the tool suggests which other variants can be added to make it complete. We refer to selecting variants and analyzing their commonalities and differences and complete classes as *bottom-up exploration* approach.

3. METHOD AND TOOL EVALUATION

We performed a small controlled experiment (details in [8]) with three participants knowledgeable about product-line engineering to evaluate the effectiveness of the visualization method and the tool. The participants were required to perform 11 predefined tasks and answer questions. They used an older version of the too—the material presented in Sec. 2 includes all feedback we have subsequently implemented. Here we only describe the most significant results.

All three participants found the bubble chart visualization both useful and intuitive. Participants were able to make decisions in 4-dimensional space, compare bubbles and select the ones they needed. All participants reported difficulty comparing variants that differ by a small amount in bubble opacity and size, whereas the differences were easily noticeable for bubble position on two dimensional graph plane. In the example, variant security was represented as bubble opacity with a single color and it had three discrete values: 0, 5 and 10. One participant was confused by the opacity representing the value zero, which was not completely transparent (intuitively no color is interpreted as corresponding to the value 0). To address that, we subsequently implemented a three-color red-yellow-green gradient. Fortunately, all three participants could rely on the tooltip feature of the graph to know the exact values of quality attributes.

The lack of a label for the bubble size representation made interpretation difficult for two subjects. We have since implemented a label which shows a large bubble with the maximum value and a small bubble with the minimal value, which makes the meaning of the bubble size clear (Fig. 3).

Two participants noticed the correlation between mass, security and performance, without being asked to do so. It was a good indicator that users can reason among multiple dimensions with this visualization.

One participant noticed a limitation: given limited screen space and big bubble size, bubbles will overlap and make the

exploration difficult. So far we do not have a good estimate of optimal number of bubbles and distribution density.

All three participants mentioned that filtering by features is important and needed in top-down decision making: finding variants that have given features included or excluded. Participants wanted to group variants together by some algorithm (features, common values of quality attributes, or proximity) and explore them separately. We since implemented filtering by features and introduced the notion of target quality range. Two participants found idea of clustering by feature useful, another one said that this would be probably useful. Clustering appears to be an important prospective feature as well.

4. RELATED WORK

We are not aware of any research or industrial tools that implement interactive exploration of Pareto fronts tailored for PLE scenarios. However, similar visualizations are used to some extent for different use cases. The Feature and Quality Matrix is simply an extension of commonly used matrix (*feature matrix* [9], *product-feature matrix* [7], and *variant matrix* [2]) with quality attributes.

Poles et al. [12] use bubble charts for representing their multi-objective optimization results, but in a different way: bubble position reflects the two quality values, bubble color and size represent standard deviations of the two quality values. Sasaki et al. [14] use a visualization similar to bubble charts to display solutions of the multi-objective optimization problem in three and four-dimensional spaces. However, the shapes representing variants are not labeled, and therefore it is not possible to enumerate and explore them individually, in contrast to our approach.

Other approaches to visualization *e.g.*, 3D scatter plots [17], Level Diagrams [4], heatmaps [13], self-organized maps [10] require user studies to evaluate them in the PLE context.

The tool SPL Conqueror [16] focuses on the measurement phase and supports some optimization of non-functional properties without exploration of optimal variants. Loesch and Ploedereder [7] described notions of shared, distinct, rarely used or never used features in terms of concept analysis. We can incorporate concept analysis into our tool in the future as it may offer more exploration opportunities.

5. CONCLUSION

We presented a novel visualization method: an integrated, interactive, and synchronized set of complementary views onto a Pareto front specifically designed to support a number of product line exploration scenarios. We implemented a web-based tool and evaluated it in a small controlled experiment. We demonstrated that Pareto front can be effectively visualized and explored using the proposed visualizations. The user experiment confirmed the feasibility of our approach and provided good feedback we successfully incorporated into our tool. In the future, we plan to explore industry use cases, conduct a new user experiment with a newer version of the tool and with professional engineers.

5.1 Acknowledgments

We thank Neil Vincent Redman for implementing many extensions to the tool.

6. REFERENCES

[1] D. Benavides, P. Trinidad, and A. Ruiz-Cortés. Automated reasoning on feature models. In *Advanced Information Systems Engineering: 17th International Conference*, LNCS, 2005.

[2] D. Beuche. Modeling and building software product lines with pure::variants. *SPLC*, 2008.

[3] K. Bąk, K. Czarnecki, and A. Wąsowski. Feature and meta-models in Clafer: Mixed, specialized and coupled. In *Intl. Conf. on Soft. Language Eng.*, 2010.

[4] X. Blasco, J. M. Herrero, J. Sanchis, and M. Martínez. A new graphical visualization of n-dimensional Pareto front for decision-making in multiobjective optimization. *Information Sciences*, 178(20), 2008.

[5] P. C. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley, 2001.

[6] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson. Feature-oriented domain analysis (FODA) feasibility study. Technical report, SEI, 1990.

[7] F. Loesch and E. Ploedereder. Optimization of variability in software product lines. In *Proceedings of the 11th International Software Product Line Conference*, SPLC, 2007.

[8] A. Murashkin. Web-based GUI for Pareto front design and analysis. Technical Report GSDLAB-TR 2013-02-04, University of Waterloo, 2013.

[9] D. Nestor, L. O'Malley, A. J. Quigley, E. Sikora, and S. Thiel. Visualisation of variability in software product line engineering. In *VaMoS*, 2007.

[10] S. Obayashi and D. Sasaki. Visualization and data mining of Pareto solutions using self-organizing map. In *Evolutionary Multi-Criterion Optimization*, 2632, LNCS. Springer Berlin / Heidelberg, 2003.

[11] R. Olaechea, S. Stewart, K. Czarnecki, and D. Rayside. Modeling and multi-objective optimization of quality attributes in variability-rich software. In *Intl. Workshop on Non-functional System Properties in Domain-Specific Modeling Lang.*, 2012.

[12] S. Poles, P. Geremia, F. Campos, S. Weston, and M. Islam. MOGA-II for an automotive cooling duct optimization on distributed resources. In *Evolutionary Multi-Criterion Optimization*, vol. 4403, LNCS. 2007.

[13] A. Pryke, S. Mostaghim, and A. Nazemi. Heatmap visualization of population based multi objective algorithms. In *Evolutionary Multi-Criterion Optimization*, vol. 4403, LNCS. 2007.

[14] D. Sasaki, S. Obayashi, and K. Nakahashi. Navier–Stokes optimization of supersonic wings with four design objectives using evolutionary algorithm. 11:14, 2001.

[15] N. Siegmund, M. Rosenmuller, C. Kastner, P. Giarrusso, S. Apel, and S. Kolesnikov. Scalable prediction of non-functional properties in software product lines. In *SPLC*, 2011.

[16] N. Siegmund, M. Rosenmuller, M. Kuhlemann, C. Kaestner, S. Apel, and G. Saake. SPL Conqueror: Toward optimization of non-functional properties in software product lines. *Software Quality J.*, 20, 2012.

[17] T. Tušar and B. Filipič. Visualizing 4D approximation sets of multiobjective optimizers with prosections. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO, 2011.