Indian Journal of Science and Technology, *Vol 9(17), DOI: 10.17485/ijst/2016/v9i17/92728, May 2016*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

An Approach for Optimized Feature Selection in Software Product Lines using Union-Find and Genetic Algorithms

Asad Abbas, Zhiqiang Wu, Isma Farah Siddiqui and Scott Uk-Jin Lee*

Department of Computer Science and Engineering, Hanyang University, South Korea; asadabbas@hanyang.ac.kr, hhhwwwuuu@hanyang.ac.kr, isma2012@hanyang.ac.kr, scottlee@hanyang.ac.kr

Abstract

In Software Product Line (SPL), feature model is highly recommended to manage the commonalities and variability of features under resource constraints of mandatory, optional and alternative. Features with mandatory constraints and high in dependency with other features are identified as crosscutting concerns; reduce the reusability of resources. It is important to find and modularize these concerns at modeling level. With this practice, these concerns do not effect if deletion or addition is required from entire system. In this paper we have applied Union-find algorithm to find crosscutting concerns in feature model. We evaluated our approach by applying on an automobile feature model with various dependencies between features, and found required crosscutting concerns. By this approach, identification of crosscutting concerns and their modularization made easier. Further, we have also applied genetic algorithm to get optimized feature selection under cost constraint with high performance. In SPL, as crosscutting concerns are mandatory features with fix cost and performance, optimization on feature model is necessary under consideration of crosscutting concerns. Our approach found all possible products according to crosscutting concerns, cost and performance at modeling level of an automobile feature model. At last, we found all products from minimum to maximum cost with respect to least maximum performance by using GA optimization technique.

Keywords: Feature Model, Genetic Algorithm, Optimization, Software Product Line, Union-find Algorithm

1. Introduction

SPL is set of product families that share the common resources and develop products according to need of particular market requirements. SPL is very common and effective paradigm to develop families of software by the reusability of resources from repository. Two major procedures are utilized to develop SPL namely, domain engineering and application engineering. The application engineering process comprises of product development (product derivation) according to market segments and requirements. Further, the domain engineering consists of common and variable features among all products. The common features are reusable in each product while, variable features differentiate products. High reusability of features can be achieved by well manage commonalities and variability, systematically[1].

Feature model is highly used and effective to manage the variability and commonalities in SPL. This is a tree structure of common and variable features which consists of terminal and parent features. Terminal features (leaf nodes) are used for development of products in which common features are easy to reuse however, variable features are even difficult to manage. Rules and constraints about use of common and variable features are precisely described in feature model of SPL[2]. It is compact picture of all possible products, feature combination, constraints and relationship among features. Relationship and constraints among features are described as optional, alternative and mandatory features. Mandatory features are significant for all products in SPL. These features can be crosscutting concerns which are scattered in overall system and cannot be removed because of tangled in multiple features[3].

** Author for correspondence*

Crosscutting concerns are hard to decouple and problematic for the reusability of features, where complexity of relationship among features is high. To increase the reusability, it is essential to find the crosscutting concerns from feature model and modularize them[4]. In large feature model, it is difficult and time consuming task to identify crosscutting concerns and their dependent feature.

Furthermore, it is also hard to choose best features within some constraints and user requirements for product derivation. Identification of crosscutting concerns at modelling level before development of product is very important to modularize and get high reusability which is known as aspect mining process[5]. The idea behind this study is to find crosscutting concerns from feature model, which are mandatory and having constant value, by using union-find algorithm resulting easy selection of features. To get best optimized feature selection we used Genetic Algorithm (GA) under constraints of cost and performance.

The specific objective of this study is to use union-find algorithm for the identification of crosscutting concerns in feature model. Subsequently, we used GA to optimize feature selection by considering resource constraints and crosscutting concerns.

The paper is organized as follows. Section 2 is about related work for searching crosscutting concerns and optimization techniques, section 3 gives background on feature model, section 4 identifies crosscutting concerns, section 5 is about union-find algorithm procedure, section 6 is about feature combinations with constraint problems and GA approach for optimization and, finally section 7 gives conclusion.

2. Related Work

One research[1] over feature selection discussed ant colony optimization in SPL feature model. According to authors, ant colony optimization technique took more time than GA. Ant colony optimization technique optimizes good paths in graph.

Another research[5] was on application of k-means algorithm to calculate attribute values of each method by data clustering, subsequently, measure the value of each method with respect to each group using vector space model.

A study[6], identified crosscutting concerns using static analysis approach by the conversion of attribute values to vector space model in genetic algorithm and find FAN-IN values in java program.

In another research[7], authors developed new approach namely GAAM (Graph Algorithm in Aspect Mining), by the combination of graph algorithm and aspect mining technique to find the crosscutting concerns. Authors also introduced GAAM for computation of attribute values, filtering, grouping and analysis.

Another research[8], discussed genetic algorithm of optimized feature selection under resource constraints. Authors discussed database case study with constraints of cost memory, and CPU. Their research described use of genetic algorithm to select optimized feature for a product in SPL. Genetic algorithm was applied on large scale feature model of database and selected optimized feature selection.

In this paper, for identification of crosscutting concerns in SPL feature model we have applied union-find algorithm. Identification of crosscutting concerns at modeling level reduces the optimization time and increase the accuracy of optimized feature selection under constraints and requirements of stakeholder. For optimized feature selection, we have applied GA under cost constraint and obtained high performance feature combination for product derivation.

3. Feature Model Background

Comprehensive analysis and information of feature model is essential for assets development of the product line, however, this information is not enough for product derivation. Information related to binding features such as timing and relationship to include in specific products and delivered to stakeholders also the derivation of components in SPL. Moreover, in product line assets, some features (bound features) must be provided at the time of development while, some of them (optional features) can be provided on the time of installation by the choice of costumer. Therefore, it is important to optimize features thoroughly for bounding to products, and such information is compulsory to provide for the purpose of component design[9].

Three ways can be adopted for the testing of feature binding, given as: 1) how (approaches), 2) when (timing) and, 3) what (units) features are bound. When a bounded feature is specified to a product, for the proper operation of features the exiting set of bounded features should be together. These exiting sets of features called as binding units (variability and commonalty units) which should further utilized to identify the points of variation to achieve composition components easily for feature bindings. While, the timing units of feature binding are also influence component design. In case of when feature (timing), the load table approach can be adopted for the component design, at the time of installation. It is therefore, the components need to be design for the provision of binding features when they are required[10].

The terminal features are required to bind for derivation of new product from feature model by finding their combination. It can be assumed that the development of new product is successful if it fulfills all constraints which are defined by the feature model. The constraints which are governed by a feature model on terminal features are "And" (select all), "Or" (select few or all) and "Alternative" (select only one) groups[7]. The motivating example used in this paper is given below in Figure 1.

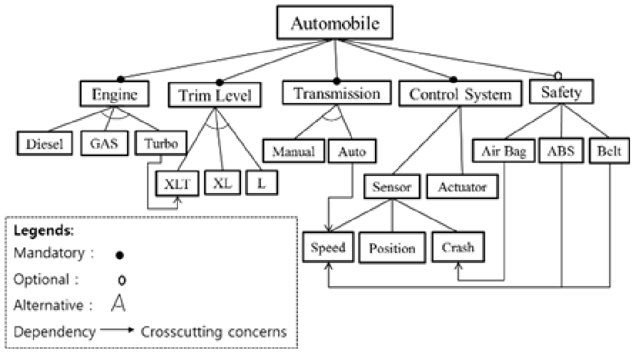


Figure 1. Automobile feature model.

Figure 1 depicts feature model of automobile where total number of features are 22 with different constraints of selection. We defined 3 features as crosscutting 'Turbo', 'Speed' and 'Crash'. 'Air Bag' depends on 'Crash', 'Auto Break System (ABS)' and 'Belt' depend on 'Speed' and 'Turbo' depends on 'Extra Large Turbo (XLT)'.

4. Identification of Crosscutting Concerns

Large number of dependencies exist among features in feature model, resulting reduce reusability of common and variable feature. In order to manage the variability of SPL, it is important to eliminate the dependencies among features. As previously discussed in section I, the crosscutting concerns are scattered in overall system and cannot be separated from domain. These concerns impact and tangle with other non-crosscutting and crosscutting concerns. Crosscutting variables are commonly found in feature model span in other features. This crosscutting problem makes the reusability of features difficult. Therefore, to cater this problem, we modularize these concerns separately at modeling level[11]. In the motivating example of automobile feature model (Figure 1), the identification of crosscutting concerns at modelling level will provide flexibility to modularize separately for reusability. It is slightly easy to find and locate these crosscutting concerns accurately in small feature model but in case of complex feature model it is difficult and the accuracy is questioned due to the existence of large number of features.

5. Union-find Algorithm

The utilization of well-organized and effective data structure sometime depends on efficiency of an algorithm. An appropriate selection of the data structure can minimize the execution time of an algorithm. Union-find algorithm was introduced by Tarjan in 1970s which makes disjoint sets by union operation[12]. It tracks elements in tree and compare with other elements. In our approach union-find algorithm compare X feature with Y, if X is called in Y then it declares X is crosscutting.

As an example, set of N elements which are divided into all possible subsets to retain the connectivity of every component in a particular subset or based on relationship among each other. This operation can be done by using union-find data structure. Let's consider there are five persons A, B, C, D and E. A and B are friends, B and C are friends, D and E are friends. Relation between friends become:

- A has direct relation with B and indirect relation with C via B.
- D has direct relation with E.

Union-find data structure can be used to determine direct or indirect relation between friends. We can also find disconnected subsets, here two disconnected subsets are {A, B, C} and {D, E}.

Two operations are performed here:

- Union (A, B) - join two friends A with B.
- Find (A, B) – determine the path joining elements A and B.

The parent node in tree feature model is mostly used to sustain a number of disjoint sets. If two or more than two disjoint sets have no common members (no direct or indirect dependency), their intersection set is empty i.e. not dependent to each other. Numbers of disjoint sets keep grouping of some elements, so that each element is the part of one disjoint set. Two main operations that union-find algorithm performs[13]:

- Keep track if two elements belong from same set i.e. they have relationship with each other (FIND operation).
- Combine two sets.

Union-find algorithm works with disjoint set data structure named as union-find data structure which maintain partition of finite sets without loss generality. Disjoint set data structure aims to perform union with arbitrary sequence and perform operation on two disjoint sets to get one set. Steps for union-find algorithm:

- Set of all possible elements
- {S1, S2, . . . ,Sk}
- Union command: connect two elements of same category.
- Union(x, y): Function union both sets which containing x and y.
- Find Query: Find the target elements and create new set for these elements.
- Find(x): This function finds the ID of the set x belongs.

5.1 Identification of Crosscutting Concerns using Union-find

Several aspect mining techniques exist to identify crosscutting concerns from feature model. In complex and large feature model it is hard to find crosscutting concerns for aspect modularization. Since, the crosscutting concerns are spread in whole system and develop relationship with other features directly or indirectly[14]. Union-find is an easy and simple algorithm to find crosscutting concerns. Union-find algorithm process matches with crosscutting problems and usable to find crosscutting concerns. In this study, we adopted terminal features for aspect mining because non-terminal features will select automatically by the selection of terminal features.

Algorithm 1. Union-find algorithm for searching Feature Model

Part I: Merge(a,b)

```
• Input: the number of features and feature model
  with F={fi} 1 ≤ i ≤ n RF (Relationship of features)
• S' = x  y
• x.total<-- get the children number both x and y
• y.parent<-- setting the parent of y is x
• x.children<-- setting the children of x is y
• Output: Union of all disjoint sets
```

In order to find terminal features, we give input of initial population of all features in feature model and determine the relationship between them. We get the union of all disjoint sets with comparison of terminal with terminal feature (child node) or terminal with non-terminal feature (Parent of any terminal node).

Part II: Find_Crosscutting(n)

```
• foreachfeature∈ N do
• numParent<-- amount of parents
• numChildren<-- amount of children
• if numParent>1 and numChildren>1 then NotTer
  minalCrosscutting(feature);
• else
• if (numParent>1 and numChild =0)
•     or property is mandatory
•     thenflag<--true
• else  JugeParentIsCC(feature);
• end
• end
• end
• Output: crosscutting features
```

The description of above pseudo code is given as; first, terminal feature, if called by any other feature i.e. dependent on that feature, is considered as crosscutting concern. Comparison has applied between all features and found relations among them. Terminal features (Crosscutting concerns) can be parent of other features but not with direct relationship, and only need functionality by any other feature in feature model (dependent). Second, if non-terminal feature is crosscutting concern then all of its related features are also crosscutting concerns.

Part III: NotTerminalCrosscutting(n)

```
• Foreachi∈length of children do
• Foreachi∈length of parents do
• If ithparent is crosscutting then
• This feature is crosscutting
```

If feature is crosscutting but not terminal feature, set its children to crosscutting feature and set its children flag to True. If one of the feature's parent is crosscutting then execute this function.

Part IV: Search crosscutting concerns

```
• Input: the number of feature model with N, relationship for features with Rf={x,y} 0<=x,y<n
• Initial feature model
• repeat
•         input relationship
•         Merge(x,y)
• until stopping condition
• Find Crosscutting (N)
• Output: crosscutting features &Mandatory
```

Crosscutting concerns identified with respect of their relationships with other feature. It repeats the operation of find again and again until all existing crosscutting concerns and their respective dependent features identified.

6. Feature Combination with Constraints Problem

Language of feature model first proposed and formulized by Kang et al. Feature model comprise the Information of common and variable features of software product line. Feature model is hierarchical structure of parent and node with different relationships and capture the scope and different level of abstraction of SPL[16].

Feature selection in SPL feature model with resource constraints from stakeholder is a big challenge. Since, the cost is most common constraint from the customer; therefore, in this study we discuss the appropriate selection of features combination under cost constraint with respect to performance evaluation of the product. We present a case study of automobile feature model with constraints and crosscutting concerns.

Example of feature selection: we randomly selected cost of terminal features from 100$ to 500$ and found best feature combinations with least high performance. As cost increased the performance of product also increased. Crosscutting concerns are already identified by using union-find algorithm.

6.1 Genetic Algorithm (GA)

GA is stochastic and heuristic approach for global search optimization which quickly scan large population for best feature selection. It can quickly optimize big papulation of resources and get best solutions. With highly constraints problems GA work very well and effectively. For the best optimization the developers need to satisfy resource limitations and specific product feature selection[8].

GA procedure cannot directly applied for synthesis mechanism. Modifications related to objective function are obligatory in the original algorithm according to the behavior of feature constraints to avoid impulsive combination of feature selection. This study describes a GA formulation to consider the constraints and follow the algorithm procedure concerning the global best solution[15].

In GA, initial population is generated randomly such that satisfies all given constraints. In GA, features of SPL in feature model are connected with each of the product which can be denoted by either real numbers or binary strings. The binary strings have advantage over real number because of simple and easy in operation of crossover and mutation. All random generated feature selections are assessed with objective function and separately checked according to constraints[8].

In optimization where constraints exist, the solution space is limited to restrictions, therefore, GA cannot be applied directly on data structure. Generally, there are three methods to formulate genetic algorithm to consider constraints. First, allow unreasonable elements after disciplining them by assigning low fitness values, therefore there is low selection probability of such numbers for the next features generation. While high fitness values are ensured high probability for next generation of feature selection. Second, continuous crossover and mutation with random feature selection is occurred until the selection of valid feature combination. Third, by defining constraints, the crossover and mutation process changes to keep the feature combination remain inside in the valid region[6].

Crossover operation is formulated to use binary string rather than of real numbers. It takes two features/parents from initial population and makes comparison according to fitness values of the objective function. Subsequently, mutation operation applies on single bit of the binary string; if objective function is satisfied then it inverts the binary string and makes comparison with next generation. Finally, new valid feature combination is generated from old feature model[15].

7. Initial Population

The initial population presents all combinations of features in SPL ($S \subseteq 2^F$), while each 'n' binary string is generated to present chromosomes.

Let initial population F={fi}; 1 ≤i≤ n F contains n features denote feature model

Further, each random binary string and random generated feature presents features of specific product in feature model and evaluates according to given objective function, respectively. The initial population is generated as input of GA, in which further binary code is used as chromosome (e.g. 111001011) and all chromosomes are compared under fitness value.

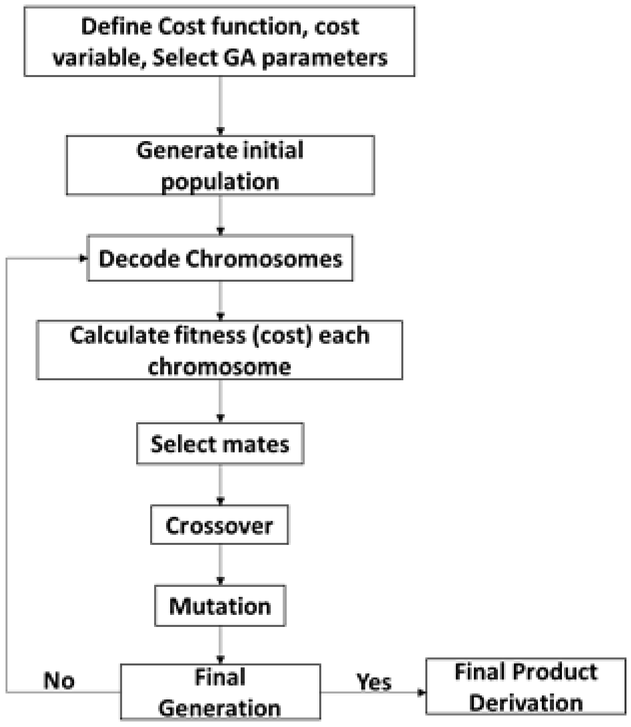

Define Cost function, cost variable, Select GA parameters
Generate initial population
Decode Chromosomes
Calculate fitness (cost) each chromosome
Select mates
Crossover
Mutation
Final Generation
No
Yes
Final Product Derivation

Figure 2. Genetic algorithm process.

Figure 2 describes the complete process of GA. Third step (decode chromosomes) is repeated until object function fulfill.

8. Genetic Algorithm for Optimized Feature Selection

GA performs various process to select the features combination for final product derivation as describe in Figure 2. Complete procedure is described in Algorithm 2 for optimized feature selection to satisfy object function[8].

Algorithm 2. Genetic Algorithm

```
GA for optimized feature selection
•  Generate feature model with n resources F
•  F = {fi}, 1 ≤ i ≤ n, C (Constraints), R (Feature), V(F)
   value of features, F(R) feature relationship and Rc
   (resource constraints).
•  Evaluate the fitness f(i) of each feature i in the
   feature model.
•  Set of new features according to fitness of each.
•  Select two parent p1 and p2
•  Crossover p1 and p2 to get offspring
•  Mutation to get new offspring (update set of
   features).
•  Population of new offspring
•  Use new generated population for a further run of
   the algorithm
•  If the required optimized feature selection satisfied
   then stop, otherwise repeat from step 3
```

Figure 2 shows the complete procedure of GA for feature optimization in feature model of SPL. We put two constraints, cost for optimized feature selection.

Our objective function is used to get high performance feature selection for the specific product within the cost constraint provided by stakeholders.

$$S.t. \sum_{i=1}^{n} S_i C_i \leq Cost \qquad (1)$$

$$Maximise \sum_{i=1}^{n} S_i P_i \qquad (2)$$

$$S_i \in \{0,1\}$$

Equation (1) determines the sum of cost for all selected features under given cost constraints from stakeholder. Figure 3 depicts that the cost is directly proportional to the performance of the product which showed that if the performance of the product will increase with cost. Further, at one point where both constraints (cost and performance) will be constant and no further performance will increase as by increasing cost. In our experimental results we optimized features within limited cost with respect to performance. In GA, features are presented by binary string whereas, value of not selected and selected features in binary string are represented by 0 and 1, respectively.

$S \subseteq 2^F$; S denotes all possible combination of features in feature model

In equation (2) Pi and Si defines the performance of all n features and selected features, respectively. It calculates the sum of all features which are selected for specific product with least maximum performance according to cost, from F= {fi}. In this study, we used random value of feature performance to selected best features.

Part 1. chromsomesChange(int min, int max)

```
• foreachi∈[min,max] do
• if this bit is not crosscutting then
• This bit will be swap
• end
• end
```

We applied algorithm according to our own objective function and consider the crosscutting concerns in feature model for optimization. Part 1 check crosscutting concern for each bit in binary string, if any bit which is not from set of crosscutting then it swaps and check other bits by changing random chromosomes. Set of crosscutting concerns is:

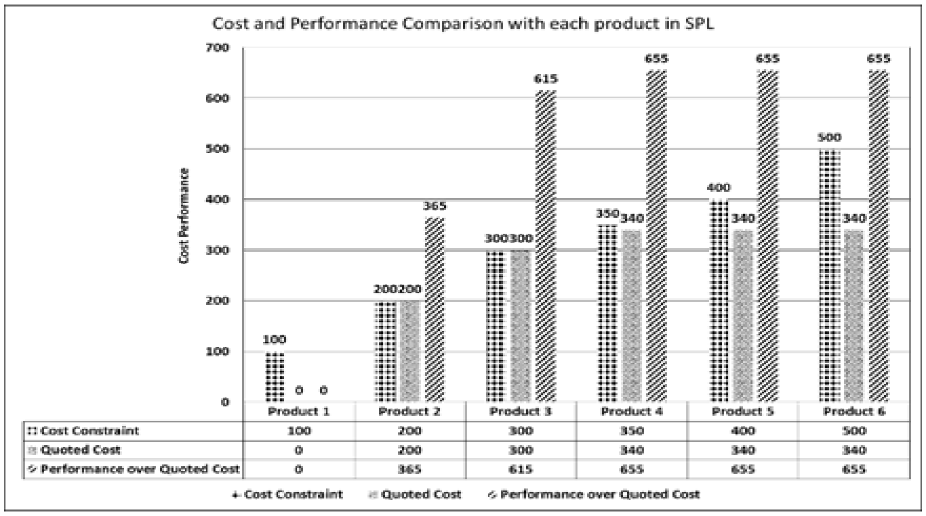
CCi={n}; CCi denotes all crosscutting concerns in feature model. Where 1≤ i ≤n.

Part 2. Mutation(bool parent[])

```
• Num = Generate a random number [0,99]
• ifnum is 7 then
• Changenum = generate a random number [0,n)
• ifchangenum-th bit is crosscutting then
• Skip;
• else
• Change this bit
• end
• end
```

The program will generate a random number from 0 to 99. Two scenarios may be occurred, first, the mutation probability should be less than ‘0.1’ and second the mutation should be occurred if random number is ‘1’ performed mutation operation and optimizes the best feature solution.

Figure 3 shows the experimental result for the values of cost constraints and performance of each product under domain of SPL. We have selected random values for cost and performance for each feature. Results determined product cost is started from 200$ with performance 365.



	Product 1	Product 2	Product 3	Product 4	Product 5	Product 6
▦ Cost Constraint	100	200	300	350	400	500
▒ Quoted Cost	0	200	300	340	340	340
▨ Performance over Quoted Cost	0	365	615	655	655	655

Figure 3. Derivation of Products in SPL.

The highest cost 340$ and maximum performance of the product in SPL is 655. Figure 3 shows the comparison between cost and performance of each product and depicts Product 1 with null values because there is no feature selection at cost 100$.

9. Conclusion

Optimization is important for best features combination for a specific product according to the constraints and limitation of stakeholders. In this paper, we have discussed about union-find algorithm for feature model to identify crosscutting concerns. We have examined that the searching of crosscutting concerns at modelling level is beneficial at feature combination for product derivation. For best feature combination, we used GA optimization technique and derived all possible products according to cost and performance for each product. We randomly selected cost for each feature in feature model and derived products under cost constraint from stakeholder with maximum least performance. Our experimental result shows product's cost and performance according to cost constraints of stakeholder.

10. References

1. Wang YL, Pang JW. Ant colony optimization for feature selection in software product lines. Journal of Shanghai Jiaotong University (Science). 2014; 19(1):50–8.
2. Lee K, Kang KC, Kim M, Park S. Combining feature-oriented analysis and aspect-oriented programming for product line asset development. In 10th Software Product Line Conference, 2006 10th International. IEEE; 2006. p. 10-19.
3. Holdschick H. Challenges in the evolution of model-based software product lines in the automotive domain. In Proceedings of the 4th International Workshop on Feature-Oriented Software Development. ACM; 2012 Sep. p. 70–73.
4. Stoiber R, Meier S, Glinz M. Visualizing product line domain variability by aspect-oriented modeling. In 2nd International Workshop on Requirements Engineering Visualization IEEE; 2007 Oct. p. 8–13.
5. Şerban G, Moldovan GS. A new k-means based clustering algorithm in aspect mining. In Proceedings of the 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, IEEE; 2006 Sep. p. 69–74.
6. Mhawish M, Gupta M. A new genetic algorithm tool for clustering-based aspect mining using static analysis and vector-space model. International Journal of Scientific Research Engineering & Technology. 2015; 4(4):434–40.
7. Serban G, Moldovan GS. A graph algorithm for identification of crosscutting concerns. Studia Universitatis Babes-Bolyai, Informatica. 2006; LI(2):53–60.
8. Guo J, White J, Wang G, Li J, Wang Y. A genetic algorithm for optimized feature selection with resource constraints in software product lines. Journal of Systems and Software. 2011 Dec; 84(12):2208–21.
9. Thüm T, Kästner C, Benduhn F, Meinicke J, Saake G, Leich T. Feature IDE: An extensible framework for feature-oriented software development. Science of Computer Programming; 2014 Jan. p. 70–85.
10. Lee J, Kang KC. Feature binding analysis for product line component development. Software Product-Family Engineering Springer Berlin Heidelberg; 2003 Nov. p. 250–60.
11. Tizzei LP, Rubira CM, Lee J. An aspect-based feature model for architecting component product lines. Software Engineering and Advanced Applications 38th EUROMICRO Conference. IEEE; 2012 Sep. p. 85–92.
12. Wilkinson MH, Roerdink JB. Fast morphological attribute operations using Tarjan's union-find algorithm. Mathematical Morphology and its Applications to Image and Signal Processing; Springer: US; 2002. p. 311–20.
13. Carlinet E, Geraud T. A comparative review of component tree computation algorithms. IEEE Transactions on Image Processing. 2014; 23(9):3885–95.
14. Abufardeh S, Magel K. Software internationalization: crosscutting concerns across the development lifecycle. International Conference on New Trends in Information and Service Science, 2009. NISS'09, IEEE; 2009 Jun. p. 447–50.
15. Said Gaena, Mahmoud AM, El-Horbaty ESM. A comparative study of meta-heuristic algorithms for solving quadratic assignment problem. International Journal of Advanced Computer Science and Applications. 2014; 5(1):1–6.
16. Ajoudanian S, Hosseinabadi SHM. On Formalization of Extended Feature Model using Promotion Technique in Z. Indian Journal of Science and Technology, 2015; 8(17):1-14.