Improving the structural quality of UML class diagrams with the genetic algorithm

Olga Deryugina\textsuperscript{1,}\textsuperscript{a}

\textsuperscript{1}Moscow Technological University (MIREA), Institute of Integrated Security and Special Instrument Engineering, 119454 Moscow, Russia

Abstract. The problem of improving the structural quality of UML class diagrams can be formulated as an optimization problem. The Genetic algorithm is concerned to be able to solve such problems. This paper focuses on the ways in which the Genetic algorithm can be applied to the problem of improving structural quality of UML class diagrams. It develops the theme of semantically equivalent transformations of UML class diagrams during the evolutionary search. This paper suggests the structural semantics of the UML class diagrams. It also formulates the problem of improving the structural quality of a UML class diagram during the evolutionary search and proposes a solution of the problem based on the Genetic algorithm. The paper presents the results of the computational experiment aimed at improving of the structural quality of the UML class diagram with the help of the Genetic algorithm and identifies issues for future work.

1 Introduction

Recent advances in SBSE (Search Based Software Engineering) have provided the possibility of applying evolutionary algorithms to the problems of software engineering, many of which can be formulated as the optimization problems.

The evolutionary algorithm is a bio-inspired algorithm, which iteratively searches for a solution of the problem. The evolutionary algorithms include the genetic algorithm [1], genetic programming [2], simulated annealing [3], swarm intellect [4], bio-geographic algorithm [5], etc.

The Genetic algorithm (GA) is an evolutionary algorithm based on a natural selection mechanism.

A number of studies have investigated different ways of object-oriented software architecture design based on evolutionary algorithms and UML diagram transformations. For example, in [6] hierarchical decomposition of the system has been performed. In [7] pattern based evolutionary transformations have been performed. In [8] authors solve the class responsibility assignment problem with the help of the Genetic algorithm. However, to date it remains unclear which transformations (design patterns, for example) can be automatically performed on the architecture design stage and whether these transformations are semantically equivalent.

The aim of this work is to formalize structural semantics of UML class diagrams and to suggest an algorithm of UML class diagram evolutionary transformation aimed at improving the diagram structural quality.

The evolutionary transformation of UML diagrams is significant because it allows implementing automatic refactoring of software architecture.

The primary contributions of this paper are:

1. To propose the structural semantics of the UML class diagrams to be able to check equivalence of two UML diagrams.
2. To formalize the problem of improving the structural quality of a UML class diagram during the evolutionary search.
3. To propose a solution of the problem based on the GA.
4. To present the results of the computational experiment aimed at improving of the structural quality of the UML class diagrams.

2 Structural semantics of the UML class diagrams

In paper [9], the structural semantics of UML class diagrams was suggested. This semantics provides a formal way of describing UML diagram transformations (interface insertion, Façade pattern applying).

Let $d$ be a UML class diagram $d = \{C, I, R\}$, where $C$ is a set of classes $C = \{c_0, c_1, \ldots, c_n\}$; $I$ is a set of interfaces $I = \{i_0, i_1, \ldots, i_m\}$, $R$ is a set of relations $R = \{r_1, r_2, \ldots, r_n\}$.

Then $c_i$ is a class $c_i = \{A_i, M_i, F_i\}$, where $A_i$ is a set of attributes $A_i = \{a_{i0}, a_{i1}, \ldots, a_{in}\}$, $M_i$ is a set of methods $M_i = \{m_{i0}, m_{i1}, \ldots, m_{in}\}$, $F_i$ is a set of features $F_i = \{s_{i0}, v_{i0}, abs_i\}$, $s_{it}$ is an isStatic feature $s_{it}$ $\{0, I\}$, $v_i$ is a visibility

\textsuperscript{a} Corresponding author : o.a.deryugina@yandex.ru
parameter $v_i \in \{\text{public, private, protected}\}$, $abs_i$ is an isAbstract feature $abs_i \in \{0, 1\}$.

In addition, $i_j$ is an interface $i_j = \{m_0, m_1, \ldots, m_m\}$ characterized by a set of methods $\{m_0, m_1, \ldots, m_m\}$.

The following kinds of relations $r_i$ are possible between classes: association ($c_1 \rightarrow c_2$), generalization ($c_1 \rightarrow c_2$), aggregation ($c_1 \rightarrow c_2$), composition ($c_1 \rightarrow c_2$), and dependency ($c_1 \rightarrow i_j$), where $\{p_1, p_2\}$ is a power of a relation.

Moreover, the following relations $r_i$ are possible between a class and an interface: realization ($c_1 \rightarrow i_j$) and dependency ($c_1 \rightarrow i_j$).

Now we assume that the structural semantics $S_i$ of the UML class diagram $d_i$ can be described as follows:

$$S_i = \{i_1, i_2, \ldots, i_n\},$$

where $S_i$ is a semantic value of the class diagram $d_i$; $i_1, i_2, \ldots, i_n$ are the classes of the class diagram $d_i$; $i_1, i_2, \ldots, i_n$ are interfaces of the class diagram $d_i$.

Assume that we are given a UML class diagram $d_1 = \{C_0, I_0, R_0\}$, where $C_i$ is a set of classes $C_i = \{C_{i0}, \ldots, C_{in}\}$, $I_i$ is a set of interfaces $I_i = \{C_{i0}, \ldots, C_{i0}\}$, $R_i$ is a set of relations $R_i = \{r_{i0}, \ldots, r_{in}\}$; $i_0, \ldots, i_n$ are the semantic value of $d_1$ is denoted as $S_i$.

Moreover, there is a function $f(d_1)$ evaluating a structural quality of the UML class diagram $d_1$.

In addition, we are given a set of semantically equivalent transformations $T = \{t_0, t_1, \ldots, t_k\}$. An example of such transformation (Interface insertion) is given at Figure 1:

$$f(d_1) \equiv \sum_{p \neq 0} CBO_p,$$

where $CBO_p$ is an average $CBO$ value of a class diagram $d_i$ with $n$ classes $c_j$, $j = 0 \ldots n$, which reflects a degree of dependence between the components of the system; $CBO_p$ is a $CBO$ value of the class $c_j$; $d_i$ is a number of classes connected with $c_j$ except from the dependency relation; $n$ is a number of classes located at the diagram $d_i$.

In other words, it is required to find such a set of transformations $T^* \in T$, which transfers the $d_1$ into the $d_2$:

$$\{T^* \in T \mid f(d_1) \leq f(d_2)\} \equiv \{T \mid f(d_1) \leq f(d_2)\} \equiv \{T \mid f(d_1) \leq f(d_2)\}$$

$$\{T^* \in T \mid f(d_1) \leq f(d_2)\} \equiv \{T \mid f(d_1) \leq f(d_2)\} \equiv \{T \mid f(d_1) \leq f(d_2)\}$$

### 3 Solution of the problem based on the GA

In order to use GA, the UML class diagram $d_i$ can be considered as an individual, which corresponds to a chromosome consisting of genes, number of which is equal to the number of classes $C' \in d_i$. Each of genes stores an information about whether or not the transformation $t_0 \in T$ has been applied to a class $c_m$.

This approach is similar to that proposed in [7], where the authors applied design patterns to classes, but our transformation approach includes the essential check of semantic equivalence of the transformation.

Individual mutation occurs by adding/deleting a random transformation to/from a gene.

In order to simplify the representation of UML class diagrams in RAM to process them with the GA, the abstract data structure (ADS) UML Map was proposed [10]. UML Map is based on hash maps; therefore, the evaluation of search complexity is $O(1)$.

A scheme of applying the GA in order to improve the structural quality of the UML class diagram is shown at Figure 2.

Let us formulate $f(d_i)$ in accordance with the Low Coupling principle. For example, we can use a CBO (Coupling Between Objects) metric:

$$CBO_i = \frac{1}{n} \sum_{j=0}^{n} CBO_j,$$

where $CBO_i$ is an average $CBO$ value of a class diagram $d_i$ with $n$ classes $c_j$, $j = 0 \ldots n$, which reflects a degree of dependence between the components of the system; $CBO_j$ is a $CBO$ value of the class $c_j$; $d_i$ is a number of classes connected with $c_j$ except from the dependency relation; $n$ is a number of classes located at the diagram $d_i$.
4 Conducting a computational experiment

Suppose we are given a UML class diagram 'E-commerce' $d_1$, which is showed at Figure 3. Its semantic value $S_1$ is as follows:

$$S_1 = \{ \{ c_1, \ldots, c_{17} \}, \emptyset, \{ c_17 \rightarrow c_8 \}, \{ c_1 \}, \{ c_{17} \rightarrow c_2 \}, \{ c_5 \}, \{ c_6 \}, \{ c_{16} \} \} \rightarrow \{ c_1 \};$$

$$c_1 \rightarrow c_3; c_{12} \rightarrow c_7; c_{17} \rightarrow c_{10}; c_{16} \rightarrow c_{14};$$

$$c_2 \rightarrow c_5; c_6; c_{13} \rightarrow c_9; c_2; c_7; c_8; c_{10}; c_{14} \rightarrow c_4;$$

$$c_11 \rightarrow c_{12}; c_{16}; c_{17} \rightarrow c_{15}; c_2; c_5 \rightarrow c_3; c_6; c_8; c_{13} \rightarrow c_7;$$

$$c_13 \rightarrow c_8; c_9 \rightarrow c_{10}; c_5; c_9 \rightarrow c_{14} \} \rightarrow \{ c_1 \};$$

A computational experiment has been conducted with the number of individuals in population = 200, the probability of mutation = 0.5, the percentage of crossover participants = 25%. The attained dynamics of $f(d_j)$ is showed at Figure 5.

$$\text{Figure 3. The null-diagram } d_1 \text{ 'E-commerce'}$$

$$\text{Figure 4. The dependence of } f(d_j) \text{ values from the iterations count}$$

On the 100 000th iteration has been achieved an individual $d_2$ with the lowest value of $CBO=0.941$, which is showed at Figure 5. The semantic value $S_2$ of the $d_2$ is as follows:

$$S_2 = \{ \{ c_1, \ldots, c_{17} \}, \emptyset, \{ c_17 \rightarrow c_6 \}, \{ c_6 \}, \{ c_{17} \rightarrow c_2 \}, \{ c_5 \}, \{ c_6 \}, \{ c_{16} \} \} \rightarrow \{ c_1 \};$$

$$c_1 \rightarrow c_3; c_{12} \rightarrow c_7; c_{17} \rightarrow c_{10}; c_{16} \rightarrow c_{14};$$

$$c_2 \rightarrow c_5; c_6; c_{13} \rightarrow c_9; c_2; c_7; c_8; c_{10}; c_{14} \rightarrow c_4;$$

$$c_11 \rightarrow c_{12}; c_{16}; c_{17} \rightarrow c_{15}; c_2; c_5 \rightarrow c_3; c_6; c_8; c_{13} \rightarrow c_7;$$

$$c_13 \rightarrow c_8; c_9 \rightarrow c_{10}; c_5; c_9 \rightarrow c_{14} \} \rightarrow \{ c_1 \};$$

Applying the rule (2) to the expression (7), we attain:

$$S_3 = \{ \{ c_1, \ldots, c_{17} \}, \emptyset, \{ c_17 \rightarrow c_6 \}, \{ c_6 \}, \{ c_{17} \rightarrow c_2 \}, \{ c_5 \}, \{ c_6 \}, \{ c_{16} \} \} \rightarrow \{ c_1 \};$$

$$c_1 \rightarrow c_3; c_{12} \rightarrow c_7; c_{17} \rightarrow c_{10}; c_{16} \rightarrow c_{14};$$

$$c_2 \rightarrow c_5; c_6; c_{13} \rightarrow c_9; c_2; c_7; c_8; c_{10}; c_{14} \rightarrow c_4;$$

$$c_11 \rightarrow c_{12}; c_{16}; c_{17} \rightarrow c_{15}; c_2; c_5 \rightarrow c_3; c_6; c_8; c_{13} \rightarrow c_7;$$

$$c_13 \rightarrow c_8; c_9 \rightarrow c_{10}; c_5; c_9 \rightarrow c_{14} \} \rightarrow \{ c_1 \};$$

From (8) it follows that $S_1 \equiv S_2$, so the obtained diagram $d_2$ is semantically equivalent to the diagram $d_1$.

5 Discussion

The results presented here demonstrate that GA has a potential to conduct automatic refactoring of UML class diagrams, but further studies are needed.

The limitation of the present research is that the only one type of transformations has been applied. The rate of the convergence might have decreased if the experiment had been carried out with a significantly larger set of transformations $T'$ and it would be beneficial to investigate this further [11].

In addition, the current study did not investigate the qualitative effect of different transformations on the values of various structure quality parameters.
6 Conclusions

In this work, we have formalized the structural semantics of UML class diagrams and the semantically equivalent transformations. In addition, we have formalized the problem of improving the UML class diagram structural quality.

We have proposed a solution of the problem based on the GA and on the UML Map data structure.

The study indicates that it is possible to perform automatic refactoring of UML class diagrams using semantically equivalent transformations. The results of this research can serve as a basis for further studies into evolutionary UML diagram transformations.

References