International Journal of Enhanced Research in Science, Technology & Engineering
ISSN: 2319-7463, Vol. 5 Issue 5, May-2016

Optimizing Bit Vector Algorithm for Anti-Pattern Detection

Parul[1], Brahmaleen Kaur Sidhu[2]

[1]M. Tech Research Scholar at Punjabi University Patiala
[2]Assistant Professor at Punjabi University Patiala

ABSTRACT

A lot of good and bad practices are known to software engineering, which proves to be a threat to software quality. With reference to software quality in software, in the field of software design and source code quality, Design patterns are referred to as good practices and Anti-Patterns are referred to as bad practices. Anti-Pattern are for example known strategies, which are poor solutions of recurring design problems which decrease software quality. There are several methods of detecting occurrence of anti-patterns in software models, some of them are: an OCL Query, Similarity Scoring Algorithm, Bit Vector Algorithm and a Rule based approach for design patterns detection. A proposed contribution to the research field related to anti-patterns: combining bit vector algorithm and similarity scoring algorithm for anti-pattern detection.

Keywords: Software Quality, Anti-Pattern, Smell Detection, Bit Vector Algorithm, extension, Similarity Scoring Algorithm

.

INTRODUCTION

Software quality is one of the most important aspects of software engineering. In context to it software quality can be defined through two distinct but related notions : software functional quality and software structural quality. Software functional quality is enforced typically and measured through software testing, whereas,software structural quality is evaluated through the analysis of the software inner structure and its source code at the unit level. In this paper we lay our focus on structural quality of the software systems.

Software systems are continually evolving to deal with ever increasing industrial and massive software development. But with constant evolution and development also comes a number of problems in designing and implementation of software systems, smells being the most recent and discussed problem. A lot of good and bad practices are known to software engineering, which prove to be a threat to software quality. With reference to software quality in software, in the field of software design and source code quality, Design patterns are referred to as good practices and Anti-Patterns are referred to as bad practices. Anti-Pattern are for example known strategies,which are poor solutions of recurring design problems which decrease software quality. They are outlined as violations of various quality rules and are applied in an inappropriate context. One example of an anti-pattern is Attribute Name Overriden, which is characteristic of procedural thinking in object oriented programming. The class defines a property with the same name as an inherited attribute.

PREVIOUS WORK

There are several methods of detecting occurrence of anti-patterns in software models, some of them are: an OCL Query, Similarity Scoring Algorithm, Bit Vector Algorithm and a Rule based approach for design patterns detection.
Various Algorithms and works have been done for the detection of anti-patterns in software models, few of the work is listed below:

1. Kaczor et al. in [1] described a technique to identify design patterns in models by using Bit-vector algorithm on string representation. Firstly, models and design patterns are transformed into directed graphs which are further

transformed into string representation by Eulerian path, which is created by traversing each edge of the graph exactly once.

2. Tsantalis et al. in [1] introduced an approach to design pattern identification based on algorithm for calculating similarity between vertices in two graphs, namely Similarity Scoring algorithm. System model and patterns are represented as the matrices reflecting model attributes like generalizations, associations, abstract classes, abstract method invocations, object creations etc.

3. Moha et al. [1] proposed a method called DECOR that defines steps for the specification and detection of code and design smells.

4. Ramaswamy et al. in [1] introduced method for pattern matching in the string by using approximate fingerprinting, which involves sliding fingerprinting window by more than one byte for faster processing and for reducing memory access while keeping a low number of false poitives.

5. Gueheneuc et al. proposed an experimental study [1] of classes playing roles in design patterns. Machine Learning Algorithm is used to find commonalities among classes playing a role in design pattern.

6. Kramer and Prechelt[2] define the way to represent good reusable design patterns using the rule-based metainterpreter in Prolog. They use this paradigm for detection patterns inside the software.

Introduction to bit vector algorithm

Bit-vector algorithm is a string matching algorithm based on dynamic programming. The underlying operational principle which makes it an important algorithm for the detection of anti-patterns is discussed further.Suppose we are given two sequences of characters-pattern $P = p_1 p_2 . . . p_m$ and text $T = t_1 t_2 . . . t_n$. The problem of approximate string matching is to find all the locations in the text *T*, that contain the pattern *P* “approximately”. That is, we wish to find all the substrings in the text, that are similar to P under some measure of similarity. The most common measure used is the edit distance - the minimum number of character insertions, deletions or substitutions required to obtain one string from another.

The identification of occurrences of a design pattern consists in identifying, in a program, classes whose structure and organization may strictly or approximately match with the structure and organization of classes as suggested by the design pattern.

Introduction to Similarity Scoring Algorithm

Tsantalis et al. in[3] introduced an approach to pattern identification based on algorithm for calculating similarity between vertices in two graphs.

System model and patterns are represented as the matrices reflecting model attributes like generalizations, associations, abstract classes, abstract method invocations, object creations etc. Since, Similarity Scoring algorithm is not matrix type dependent, thus other matrices could be added whenever needed. The advantages of matrix representation, due to which similarity scoring algorithm is considered to be a good approach for pattern identification are 1) easy manipulation with the data, and 2) higher readability by computer researchers.

A matrix type is created for system model and pattern of the system model and similarity of this pair of matrices is calculated. This process repeats for every matrix type and all similarity scores are summed and normalized.

Introduction to bit vector algorithm for design patterns

Kaczor et al. in [1] described a technique to identify design patterns in models by using bit-vector algorithm on string representation. In this technique programs and design models can be seen as directed graphs. Their string representations are built by going through every edge in their graph. The next step involves the transformation of the program graphs into Eulerian graphs.A directed graph is Eulerian if and only if every vertex has an equal in-degree and out-degree.Afterwards string representation is created with Eulerian path by traversing each edge of graph exactly once. String representation consists of connected triples *Vertex Edge Vertex* (or *Class Relationship Class* in model terms). These triplets from design pattern string representation are sequentially parsed and searched in model string representation.

Adaption of various extensions to the bit vector algorithms have been discussed by various authors. These extensions extend their searching capabilities for smell detection. Structural features included in extended methods are:

- associations (with cardinality)
- generalizations
- class abstraction (whether a class is concrete, abstract or interface).

Some of the bit-vector algorithm extensions[2] :

1. Names of methods/attributes.
2. Association types.
3. Many methods/attributes/associations.
4. The extended algorithm.

FUTURE WORK

Due to inherent parallelism, both the algorithms can be optimized with parallel computing methods simultaneously creating the model of the system and the strings/matrices for detecting the model flaws. The representation of programs as directed graphs and their transformation of eulerian graphs into strings in bit vector algorithm, And the representation of patterns into matrices in similarity scoring algorithm give us an opportunity to combine the two algorithms where the matrices can be converted into strings and further compared with the strings which represent patterns and help in the detection of anti-patterns.
A matrix can be converted into a string in matlab using the following syntaces:

```
str = mat2str(A)
str = mat2str(A,n)
str = mat2str(A, 'class')
str = mat2str(A, n, 'class')
```

DESCRIPTION

str = mat2str(A) converts matrix A into a string.
str =mat2str(A,n) converts matrix A using n digits of precision.
str = mat2str(A, class) creates a string with the name of the class A included. This option ensures that the result of evaluating strwill also contain the class information.
str = mat2str (A, n 'class') uses n digits of precision and includes the class information.

The only limitation in the conversion of matrix to a string is the mat2str function is intended to operate on scalar, vector or rectangular array inputs only. In case of a multidimensional array an error will occur. Future work holds the development of a technique which matches the strings that had been converted from the matrices using similarity scoring algorithm to the strings that had been transformed from eulerian graphs, and hence detect the presence of anti-patterns, in system models.

REFERENCES

[1]. O. Kaczor, G. G. Y., Hamel and S., "Efficient identification of design patterns with bit - vector algorithm," in Proceedings of the 10th European Conference on Software Maintenance and Reengineering, March 2006.
[2]. I. Polasek, "Anti-Pattern Detection as a Knowledge Utilisation," FIIT STU, Bratislava, Slovakia.
[3]. C. Bouhours, H. Leblanc and C. Percebois, "Bad smells in design and design patterns," Journal of object technology, vol. 8, no. 3, 2009.
[4]. B. Zamani and G. B. , "Smell Detection in UML Designs which utilise pattern languages," IRANIAN JOURNAL OF ELECTRICAL AND COMPUTER ENGINEERING,, vol. 8, no. 1, 2009.
[5]. T. Arendt, M. Burhenne and G. Taentzer, "Defining and Checking Model Smells: A Quality Assurance Task for Models based on the eclipse modeling framework," Philipps-Universit¨at Marburg, FB12 - Mathematics and Computer Science, Hans-Meerwein-Strasse, D-35032 Marburg, Germany, December 3,2010.
[6]. N. Tsantalis, Chatzigeorgiou, A., Stephanides, G., Halkidis and S.T, "Design pattern detection using similarity scoring," in IEEE Transactions on Software Engineering, vol. 32, 2006.
[7]. R. Wieman, "Anti-Pattern Scanner: An Approach to detect Anti-Patterns and Design Violations," 2011.
[8]. G. Meyers, "A Fast Bit-Vector Algorithm for Approximate String Matching," Dept. of Computer Science, University of Arizona Tucson, March 27, 1998.