# How patterns in source codes of students can help in detection of their programming skills?

Štefan Pero
Institute of Computer Science, Faculty of Science, Pavol Jozef Šafárik University
Košice, Slovakia
stefan.pero@student.upjs.sk

Tomáš Horváth
Institute of Computer Science, Faculty of Science, Pavol Jozef Šafárik University
Košice, Slovakia
tomas.horvath@upjs.sk

## ABSTRACT
A technique to detect patterns in student's program source codes. First, we represent a source code in the form of an Abstract Syntax Tree (AST). The detection of patterns is done with the SLEUTH algorithm for frequent subgraph mining on trees. We provide experiments using real data from a programming course at our university. In the paper, we discuss the relation between patterns and skills as well as some use cases and further directions of our research.

## Keywords
pattern, source code, student, skills

## 1. INTRODUCTION
One of the best-known problems in educational data mining (EDM) is predicting student's performance [6]. A great deal of algorithms have been applied to predict academic success of students. However, we are interested mainly in the following issues/questions: What lies in the background and prerequisities of students' success? What skills do students have and what is their level? We are inspired by a real situation from one programming course at our university. Students solve programming tasks where their produce a program source code. Evaluation of the solutions for the tasks solved by students is a complex process driven mainly by subjective evaluation criteria of a given teacher. Each teacher is somehow biased meaning how strict she is in assessing grades to solutions. Besides the teacher's bias there are also some other factors contributing to grading, for example, teachers can make mistakes, the grading scale is too rough-grained or too fine grained, etc. Latent programming skills of students are somehow "encoded" in their source codes provided. Automatic detection of these latent skills (with or without the assistance of the teacher) remains still an open issue.

Pardos and Heffernan presented a model called "Knowledge tracing" [5] and they used it to model students' knowledge and learning over the time assuming that all students share the same initial prior knowledge. However, by considering the needed (listed) skills as attributes of the task, it is straightforward to use them also as features in prediction models [7]. Desmarais [3] introduces different linear models of student skills for small, static student test data that does not contain missing values. They compare the predictive performance of their model to the traditional psychometric Item Response Theory approach, and the k-nearest-neighbors approach. In [1] they present wrapper-based method for finding the number of latent skills.

This work focuses on designing the technique to detect programming patterns from students' source program codes. Using real data, we illustrate how do patterns related to skills of students predefined by the teacher (author) are discovered. The contributions of this work are the following: i) we introduce a model for representing source codes in a tree-structure, ii) we propose an approach to detect patterns in source codes utilizing pattern mining algorithm.

## 2. THE PROPOSED APPROACH
Our approach is based on pattern detection from source codes and on the analysis of the relationships between the found patterns and the skills for programming tasks predefined by the teacher.

**Source Code Representation** is a critical issue in designing the process of pattern recognition. We utilize a representation scheme of source codes in the form of Abstract Syntax Trees which provide detailed information about the source code which can be used for various types of analysis [4]. Since AST contains lots of abundant information from a pattern detection point of view, we have implemented our own filter to generate a representation of a source code in XML format from AST. which provides us with a better abstraction of a source code in different levels which allows us to better specify its important parts needed for the next step of our approach.

**Pattern Mining Method** which we use in our approach is SLEUTH, an efficient algorithm for mining frequent, unordered, embedded subtrees in a database of labeled trees [8]. Given the particular source codes, first, we represent them in relevant trees in XML format at the given level of abstraction. Second, we apply SLEUTH on the prepared dataset of trees.The aim is to find all patterns (frequent, unordered, embedded subtrees) in the input dataset. We are

especially interested in so-called *maximal frequent patterns*, i.e. maximal frequent subtrees which are defined as those frequent subtrees none of which proper supertrees are frequent [2]. Finally, we cluster the resulting maximal frequent patterns (since many of them may be similar) and extract a set of representative patterns from each cluster of maximal frequent patterns.

**Relation between Patterns and Skills.** Consider the following instance of the *for loop* construction in the Java programming language.

```
for(int i=0;i<5;i++) {...}
```
(1)

To understand this construction of a for loop, and thus, to be able to use it during programming, we must first understand the following four programming concepts we call *prerequisites* for the for loop : i) variable declaration (`int i`), ii) variable assigment (`i=0`); iii) relational operators (`i<5`), iv) increment/decrement operators (`i++`). An important issue to mention is "The whole is greater than the sum of its parts" principle. For example, if one knows all of these prerequisites for the for loop individually it does not necessarily mean that she is also able to construct a for loop itself.

## 3. FIRST EXPERIMENTS

Experiments were performed with a real-world dataset, labeled "PAC"[1]. The dataset contains the following information about students' solutions: *studentID, taskID, teacherID, grade, review, solution (source code)*. Main characteristics of the dataset are described in the table 1. Each task belong to one set of tasks, i.e. we consider a set of tasks as one complex task containing several subtasks. We realized experiments

**Table 1: Characteristics of the dataset used**

| Dataset PAC | #Students | #Sets | #Tasks | #Codes |
|---|---|---|---|---|
| 2011/2012 A | 82 | 7 | 33 | 578 |
| 2011/2012 B | 36 | 9 | 21 | 381 |
| 2012/2013 A | 85 | 6 | 28 | 769 |
| 2012/2013 B | 33 | 10 | 20 | 397 |
| 2013/2014 A | 78 | 7 | 31 | 510 |

on the sets of tasks according to the described steps of our approach above, such as representation in AST, conversion to XML, pattern mining with SLEUTH and clustering the maximal patterns.

The result shown in Figure 1 refer to the number of patterns and the number of maximal patterns detected in the data for different sets of tasks. Using maximal patterns we are able to filter out repetitive and meaningless patterns. In pattern mining method we used support $0.8, 0.9, 1$ corresponding to 80%, 90% and 100% coverage, respectively.

## 4. CONCLUSIONS

We presented a model for mining patterns in source codes in order to map these patterns to corresponding programming skills. The proposed model consists of several phases such as source code representation in the form of AST and its transformation to XML at different levels, mining frequent maximal patterns and choose their representatives by utilizing clustering techniques. Since our work is in its beginning we provided only some early-bird experiments. We have also discussed three use cases of the mined programming patterns we would like to focus on in our future work.

[1]Collected from the "Programming Algorithms Complexity" course at the Institute of Computer Science at Pavol Jozef Šafárik University during the years 2011–2014.

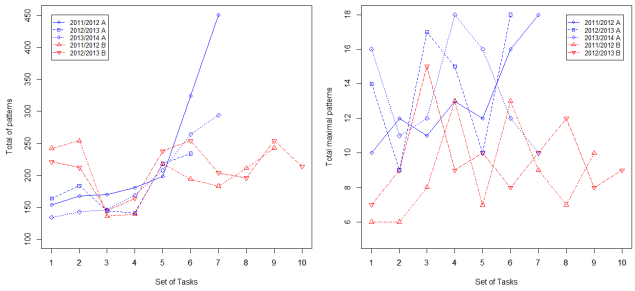


**Figure 1: The number of detected patterns and maximal patterns with support=1 (i.e. 100% coverage).**

## 5. ACKNOWLEDGMENTS

This publication is the result of the Project implementation: **University Science Park TECHNICOM for Innovation Applications Supported by Knowledge Technology**, ITMS: 26220220182, supported by the Research & Development Operational Programme funded by the ERDF and partially supported by the research grants *VEGA 1/0475/14* and *VVGS-PF-2013-102*.

## 6. REFERENCES

[1] B. Beheshti, M. C. Desmarais, and R. Naceur. Methods to find the number of latent skills. *International Educational Data Mining Society*, 2012.

[2] Y. Chi, R. Muntz, S. Nijssen, and J. Kok. Frequent subtree mining - an overview. 2005.

[3] M. C. Desmarais, R. Naceur, and B. Beheshti. Linear models of student skills for static data. In *UMAP Workshops*, 2012.

[4] M. K. and S. Yamamoto. A case tool platform using an xml representation of java source code. *Proccedings of Fourth IEEE International Workshop on Source Code Analysis and Manipulation*, 2004.

[5] Z. A. Pardos and N. T. Heffernan. Modeling individualization in a bayesian networks implementation of knowledge tracing. *Proceedings of International Conference on User Modeling, Adaptation and Personalization, (UMAP 2010)*, 2010.

[6] C. Romero and S. Ventura. Data mining in education. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2012.

[7] N. Thai-Nghe, L. Drumond, T. Horváth, and L. Schmidt-Thieme. Multi-relational factorization models for predicting student performance. *ACM SIGKDD 2011 Workshop on Knowledge Discovery in Educational Data*, 2009.

[8] M. J. Zaki. Efficiently mining frequent embedded unordered trees. *Fundamenta Informaticae*, 2005.