# Anomaly Detection for Early Warning in Object-oriented Programming Course

Shaoxiao Lu†, Xu Wang‡, Haici Zhou†, Qing Sun†, Wenge Rong†, Ji Wu†*
†*School of Computer Science and Engineering*, ‡*School of Software*
*Beihang University*
Beijing, China
{lusx, wangxu}@act.buaa.edu.cn, 1294552164@qq.com, {sunqing, w.rong, wuji}@buaa.edu.cn

2021 IEEE International Conference on Engineering, Technology & Education (TALE) | 978-1-6654-3687-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/TALE52509.2021.9678677

***Abstract*—As one mainstream of current software development, Object-oriented programming has become one key course for undergraduate students in Computer Science. Since Object-oriented concepts are difficult to understand for students, small programming exercises are used to train and help the students, and the study performance is evaluated based on the quality of the submitted source code. The common practice of code assessment in programming courses is checking whether the submitted projects pass carefully-designed test cases. However, even some projects pass all test cases, they may have bad software design and do not use the knowledge of Object-oriented programming well, especially in the early stage of courses. Therefore, we propose an anomaly detection approach for early warning in Object-oriented programming courses, which can automatically find the abnormal application of Object-oriented knowledge. In our approach, we conduct static analysis on the code submitted by students. Typical Objected-oriented metrics are extracted, and students are divided into two groups by K-means clustering: being good at Object-objected knowledge or not, and finally detect anomalous students based on the distance from cluster centers. We evaluate our approach on the realistic data sets collected from our Object-oriented programming course, and experimental results show the effectiveness of our method.**

***Keywords*—Object-oriented Programming, Anomaly Detection, Early Warning, Code Assessment**

## I. Introduction

With the development of computer science and software engineering, computer programs are increasingly used to solve practical problems. Meanwhile, programs are also becoming more and more complex. To deal with the cases reasonably, programming with thinking in Objected-oriented (OO) is created to reduce the difficulty and complexity. Therefore, Object-oriented programming is accounting for a larger proportion in software development. Mastering thinking in Object-oriented programming (OOP) is undoubtedly fundamental for students to cultivate programming ability and create their own perfect programming works. However, most colleges are facing the intractable teaching problems that teaching students how to learn OO concepts and program with the theories [1].

Although OOP was first proposed in 1970s, it was officially introduced as a formal method in computer education in 2001 at the beginning [2]. Subsequently, a number of programming educators find that some Object-oriented concepts (i.e., abstraction, class, inheritance, coupling) are complicated for undergraduate students, which makes it difficult for those students to master the knowledge and integrate theories into OO programming practice [1]. Therefore, a lot of approaches are proposed to solve this problem and further facilitate the development of OOP courses [3] [4] [5].

Recently, a number of educators have suggested that teaching OO concepts directly is not a sensible way for most students to learn related concepts. On the contrary, they consider that using intensive programming exercises consisting of a series of limited-size but high-quality codes is an effective way to improve students' OOP ability [6]. Therefore, we build an online course platform which could give relatively reasonable grades by using a pre-defined formula, where a small improvement is made on the basis of the number of passed carefully-designed test cases. The approximate algorithms of calculating grades can be found in Section IV. For more details about the algorithms, please refer to [7].

Giving advice to students who are struggling in understanding OO concepts and programming with the theories is a fundamental task in teaching activities. According to the given feedback of students, some useful guidance for these target students is made by instructors, and the teaching plan will be adjusted to solve corresponding problems as soon as possible. Meanwhile, students will benefit from it to improve their programming abilities in OO courses. Nevertheless, due to the difficulty of Object-oriented courses, students are required to learn some preceding courses (i.e., "Data Structure (C language)", "Java Programming Foundation"). At the same time, according to teaching plans, students should submit numerous codes in the process of completing their homework on account of continuous related coursework arranged by teachers. Apparently, to manually detect the code quality issues is a time-consuming task for instructors, which may not be able to identify personalized studying problems of each student. Therefore, there is no doubt that early warnings are critical for each student and teacher in teaching activities, but it is hard to achieve the goal of giving early warnings in the situation discussed above.

Code assessment is a technique to measure program quality through a quantitative method [8] which contains two main

*Corresponding author: Ji Wu, wuji@buaa.edu.cn.

978-1-6654-3687-8/21/$31.00 ©2021 IEEE

methods: the first is based on the Process-oriented source code measurement represented by Halstead and McCabe metrics [9]; the second is based on the Object-oriented source code measurement represented by Chidamber and Kemerer metrics [10]. With the metrics, the quality of programs submitted by students is able to be quantified, which makes it possible to perform some analyses on computers. In addition to code metrics, our school also builds an online course platform according to the purpose of scoring each student's coursework automatically, which contributes to subsequent analyses performed on data sets built by code metrics and homework scores. Generally, students who achieve high grades in coursework possess relative great OOP ability, which could be represented by code quality metrics. However, there are exceptional cases for some students with high scores, and some basic examples are discussed in Section IV. In conclusion, when code data sets could be converted into code metrics, we could evaluate the data automatically and achieve the goal of giving early warnings to some students without wasting much time.

In this paper, we conduct learning analytic on a novel education practice in an OO programming course in our university. Predicting the performance of students according to their daily learning situation by neural networks is commonly used in learning analytic research [11] [12]. However, the poor interpretability of neural networks impedes the deeper explanation of educational phenomena. Moreover, even in the case of high predicting accuracy, high passing rates on the designed test suites can not guarantee that the students have mastered the OOP thinking.

To solve the limitations mentioned above, we build data sets via combining code metrics with assignment scores obtained from our online course platform to predict each student's performance on OO courses. Specifically, OOP skills of students are distinguished through K-means clustering algorithm, and we integrate clustering results into a customized sorting method and all students are divided into four categories, namely, a) students with great Object-oriented skills and high scores in homework; b) students with great Object-oriented skills but low scores in homework; c) students with poor Object-oriented skills but high scores in homework; d) students with poor Object-oriented skills and low scores in homework. For the first type of students, we do not need to intervene. The other types of students should change their learning methods to solve the corresponding problems. It can be discovered that the third group of students is the most difficult to find in practical teaching activities for high course grades they possess. Naturally, we believe that those students in the category grasp the thinking in OOP. Meanwhile, students also consider that there is no problem with their programming methods and Object-oriented programming skills, which is detrimental for those students to gain further development in OOP. To deal with these situations reasonably, in the early semester period, we leverage the anomaly detection method [13] to find related students who may fail to pass the final examination or have bad OOP custom and give early warnings to these students to avoid negative circumstances happening.

The rest of the paper is structured as follows: Section II introduces our proposed algorithm. Section III introduces the implementing details of tools using in this paper. Meanwhile, we give some pictures obtained from our course platform to display the results generated by our approach. Section IV reports the results of experiments. Finally, section V concludes this paper.

## II. Our Proposed Approach

The approach we propose includes three parts. Firstly, Object-oriented metrics and assignment grades are obtained to build complete data sets. Then, we leverage an anomaly detection algorithm to gain anomalous students through analyzing data sets generated in early stage of semester. Finally, according to clustering types computed by our algorithm, all students are divided into four categories and given early warnings to related students to solve their own problems respectively.

### A. Obtaining Object-oriented Metrics

Design quality assessment is an effective and powerful method for analyzing Object-oriented programming languages to obtain some metrics, which can be used to reveal properties of source code quality. With these features, programs can be converted into data forms without obliterating information through the method, which is a suitable sort for further analyses. To figure out the quality of source codes, we choose four metrics introduced in paper [14] and add two metrics mentioned in book [15] to evaluate Java programs submitted by students.

To access a more precise analyzing result, we evaluate all Java source codes at the class granularity and method granularity. The meanings of all metrics we use are detailed in this section, and the implementation of all metrics will be introduced in Section III. All metrics are listed as follows.

- **WMC** (Weighted Methods per Class): The number of weighted methods in a class. The calculation method about this metric is to sum up the cyclomatic complexity [15] of all methods in a class, so the sum of weighted result is mainly affected by the number of methods and the complexity of each method. Apparently, with the increasing numbers of weighted methods per class, many sub classes will be influenced by WMC metric, which contributes to result in poorer reusability in relevant classes and consume more time to develop and maintain these classes.
- **NC** (Number of Children): The number of sub classes in a class. Contrary to WMC metric, this metric is proportional to the reusability in a class within a certain extent.
- **DIT** (Depth of Inheritance Tree): The depth of inheritance tree in a class. Inheritance tree, as the name implies, is a tree structure which transforms super and child classes of Java source codes into super and child nodes through consecutively recursive searches. From the above

description, we consider that the larger this metric, the more times of a class are inherited, and the more reusable code snippets will be.

- **LCOM** (Lack of Cohesion in Methods): This metric reveals the extent to which methods of a class works towards realizing a single responsibility. The metric we use is a customized implementation referring this paper [16].
- **NOPF** (Number of Public Fields): The number of public variables in a class. This metric can better represent the public variables in the current class and is used to reflect the encapsulation characteristics of Object-oriented codes.
- **PC** (Parameter Count): The number of parameters in a method.

These code metrics are used to estimate students' homework and converted into data results for further analyses.

### B. Anomaly Detection

Different from neural network methods, clustering methods are unsupervised learning algorithms, which could revel the inherent properties and principles of data through models which are trained from data sets without marking information. In addition, clustering algorithms are also particularly suitable for anomaly detection [13] and easy to interpret generating results, which is consistent with the task we are currently dealing with.

In this paper, K-means clustering algorithm [17] is selected to fit code quality data sets. Moreover, the data sets should be preprocessed before using the origin data. Specifically, due to inconsistent ranges of each code metric mentioned in Section II-A, the metrics with large ranges may seriously reduce the influence of small range metrics when the clustering method calculates internal distance. Thus, large range metrics occupy a major impact on the algorithm, and we use Min-Max scaler to normalize the unbalanced ranges of metrics to [0,1].

Due to two clusters should be generated, the k value in K-means methods should be set to 2, which could divide all students into two types: being good at OOP skills or not. A classification result will be generated through finite iterations of the algorithm finally.

When the results of the clustering algorithm are gained, we calculate the sum of Euclidean distance between each metric vector and two clustering center points. After this operation, all distances are sorted in descending order.

We consider that the samples far away from the center points of the two clusters are anomalous points [13]. The distance calculation formula is shown as follows (1):

$$dis = \sqrt{(\boldsymbol{x_i} - \boldsymbol{c_1})^2 + (\boldsymbol{x_i} - \boldsymbol{c_2})^2} \tag{1}$$

Among them, $\boldsymbol{x}_i$ represents a code metric vector of student homework numbered i, and $\boldsymbol{c_1}, \boldsymbol{c_2}$ represent the center point of the first and second cluster, respectively.

We draw an example of a clustering result's projection diagram in Fig. 1, in which it is assumed that the triangle represents the center point of the student cluster with great Object-oriented programming ability named GOOP temporarily, and the pentagonal star represents the center point of the student cluster with poor Object-oriented programming capabilities called POOP temporarily. Meanwhile, the clustering result shows that P1 and P3 belong to GOOP, and P2 belongs to POOP. Apparently, P1 represents students with great OOP skills, and P2 represents students with poor OOP qualifications. As for P3, although both P3 and P1 belong to students with good OOP ability, it is obvious that P3 is closer to POOP, while P1 is farther from POOP. Therefore, we believe that P1 belongs to the student with relatively better OOP ability in GOOP and P3 belongs to the student with relatively weaker OOP skills in GOOP. Then, we consider that (1) is a reasonable equation which could distinguish students' OOP abilities.

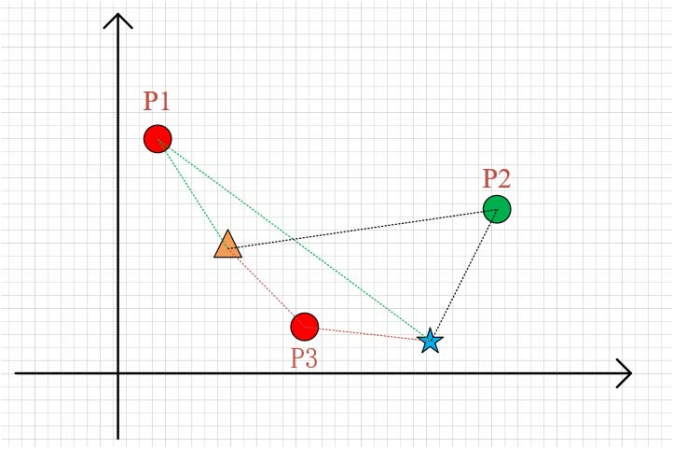


Fig. 1. An example of a clustering result's projection diagram

In addition to the method of calculating distance, we also tried two other methods to compare. Consequently, we discover that this method could generally obtain better results than others. The results will be introduced in Section IV.

### C. Early Warning for Anomalous Students

In an Object-oriented programming course, we can provide warnings for the detected anomalous students at the early stage of semester to help them improve their study performance. For example, we collect 12 programming exercises in the spring semester 2019. Three assignments are chosen to be analyzed to achieve anomaly detection in the early semester period. Subsequently, we will generate a report about the anomalous students and give some advice to them respectively. How to judge the clustering types is detailed as follows.

**Judgement of clustering types**. According to the papers [14] [18] and our teaching experience, we believe that if a class has satisfactory Object-oriented properties, there will be more public variables in the class. Meanwhile, if a class possesses the property of high reusability, the number of other classes which are inherited from the current class will be more, corresponding to greater depth of inheritance tree.

On the contrary, if a class includes more conditional branches, the class will be more complicated than other

| students with poor Object-oriented skills but high scores in coursework | students with poor Object-oriented skills and low scores in coursework | students with great Object-oriented skills but low scores in coursework |
|---|---|---|
| 17*****5 | 17*****1 | 17*****3 |
| 17*****2 | 17*****5 | 17*****6 |
| 17*****9 | 75*****2 | 17*****0 |
| 17*****0 | 17*****8 | |
| 17*****5 | 17*****3 | |

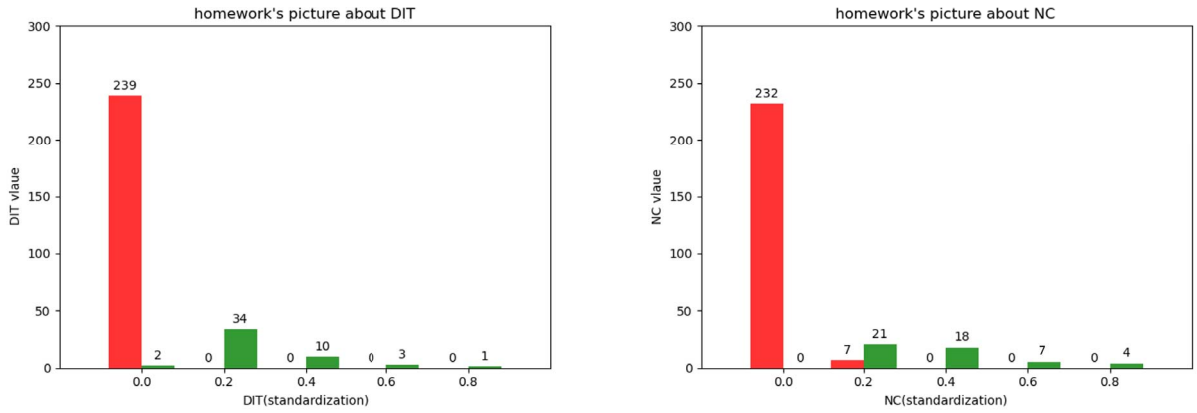


Fig. 2. Result of coursework3 in our platform

classes. And if a method requires more parameters, it will be more restricted for its scalability. At the same time, the less cohesion of a method, the more single responsibility can be achieved by this method, which will simplify its design.

$$quality = NOPF + NC + DIT – WMC – LCOM – PC \quad (2)$$

Based on the above discussion, we consider that among all metrics, the larger NOPF, NC, DIT metrics and the less WMC, LCOM and PC metrics, the higher quality of codes will be acquired. Therefore, we use (2) to distinguish the code quality of two clustering center points, where the large value represents the cluster with high code quality, and the small value represents the cluster with low code quality.

With clustering types distinguished, we understand the meanings of each cluster and attain four categories of students via integrating the results generated by the process of anomaly detection. Then, we should set a threshold to limit the number of students and obtain feedback on these students' programming problems. For all anomalous students detected in the cluster, how to advise them to solve their programming problems is discussed as follows: For students with great Object-oriented skills but low scores in homework, we need remind them to strengthen their basic training of programming. For students with poor Object-oriented skills but high scores in homework, we need prompt them to bolster their programming abilities with Object-oriented methods. For students with poor Object-oriented skills and low scores in homework, we need remind them to study diligently and reinforce their programming skills.

## III. Tools Implementation

### A. Approach Implementation

In this subsection, we use Java programming language to implement those metrics introduced in Section II-A. The calculation processes of all metrics are detailed below.

- **WMC** (Weighted Methods per Class): The calculation method about this metric is to calculate the cyclomatic complexity of all methods in a class. Cyclomatic complexity reflects the complexity of decision logic in codes, and its calculation method is to sum up the number of all conditional and loop statements in a method.
- **NC** (Number of Children): The calculation of this metric is to scan all Java classes in the project and obtain the number of all sub classes which inherit from current class.
- **DIT** (Depth of Inheritance Tree): The calculation of this metric is to find all child and super classes in current class through a recursive method and convert all classes into nodes which could be used to construct a n-ary tree. Then, a n-ary tree is built with all nodes we attain, and the height of the n-ary tree is recursively calculated as the depth of current class's inheritance tree.

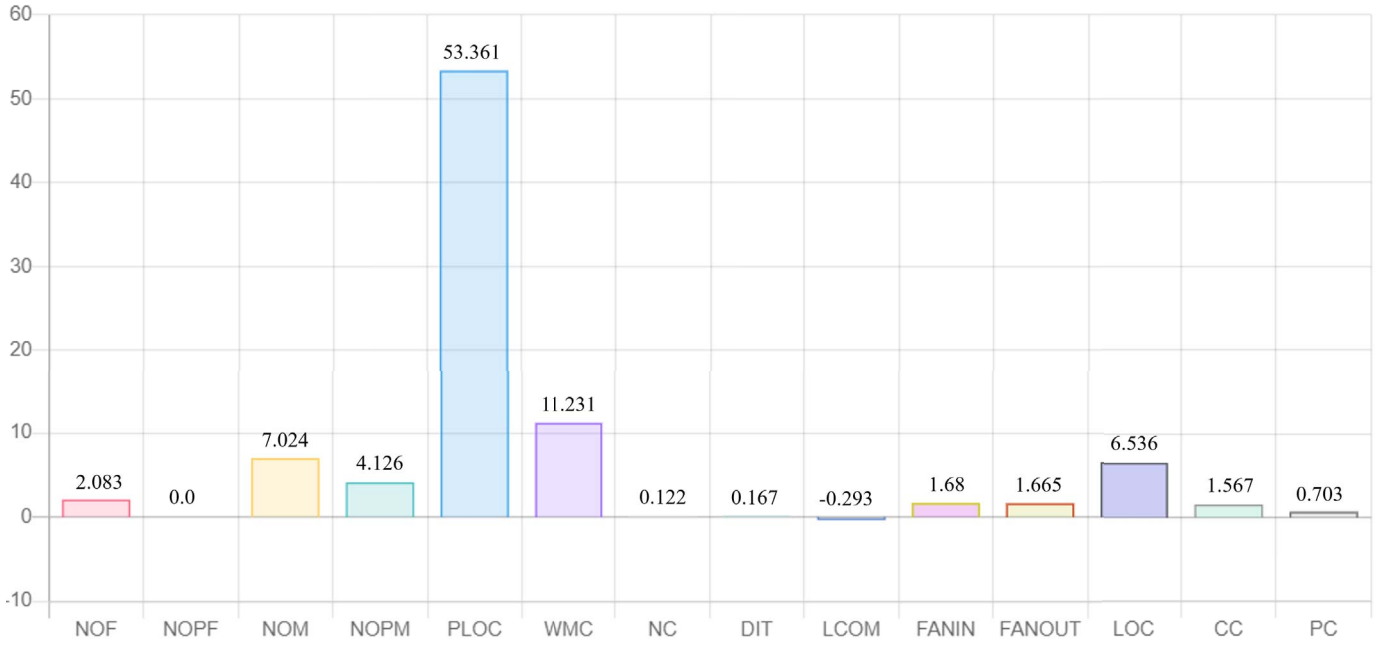


Fig. 3. An example of a student's total code metrics

- **LCOM** (Lack of Cohesion in Methods): We implement the metric referring this paper [10]. Nevertheless, we use -1 as LCOM value for a class when the class contains insufficient information or LCOM is not applicable (for instance, for an interface).
- **NOPF** (Number of Public Fields): Counting the number of all public variables in current class.
- **PC** (Parameter Count): The calculation of this metric is to count the number of parameters in each method and add all results together as the PC metric of current class.

After completing calculating processes, we will obtain all metrics of each class. Then, we add the metrics of all classes in the current project together and average the summing results, which could gain a metric vector representing an assignment submitted by a student.

In the part of using code metrics, all student IDs as the key corresponding to code metrics will be kept, and all metrics will be preprocessed with Min-max normalization. Then, we use K-means class in sklearn lib [19] to fit our data sets where the parameter of clustering center number is set to 2. After clustering algorithm converges, the two 6-dimensional vectors of the clustering center points will be obtained.

The results obtained by (1) are coupled with student's assignment grades judged by our online course platform, which could gain four categories of students described in the introduction section after recognizing types of two clusters via (2). Finally, setting a threshold for every type of students and computing and sorting the sum of distance between two clustering center points and each code metric vector, students with related problems within the threshold can be analyzed and displayed.

### B. Our Online Platform for Early Warning

We obtain the third coursework from realistic data sets and the result generated by our approach is displayed in Fig. 2, where the threshold is set to 5 and student IDs have been processed to protect their privacy. In particular, there are only three students with great OOP knowledge but low scores in this assignment.

Three types of students who should be urged to solve their related problems in OO courses are shown in Fig. 2, in which we should not meddle in students with great Object-oriented skills and high scores in homework. Besides, we also display distribution of DIT and NC metrics in coursework 3. There are 289 students submitting the assignment where 239 students belong to the cluster of being good at OOP and the other 50 students belong to the cluster of being not good at OOP. Meanwhile, it can be clearly observed from Fig. 2 that there are significant differences between students in the two clusters represented by metrics.

In addition to implementation of six metrics we discussed in Section III-A, we also implement other design quality assessments to depict student's code quality. In Fig. 3, we draw a realistic picture about a student's code metrics in coursework 3 to show all metrics we implement. The meanings and implementation of other metrics can be found in [15] [20], and all metrics depicted in Fig. 3 are not standardized to range [0,1].

## IV. Experiment

### A. Data Sets

A complete coursework data set consists of grade files and codes submitted by each student who takes the OO course.

Since students commit codes on GitLab platform, some non-source code files(i.g., ".gitignore file") are inevitable in the project. Therefore, we should ignore these unrelated files in the process of analyzing code quality metrics. Meanwhile, if some students do not submit assignments, our online course platform will automatically create empty project directories to standardize subsequent analyzing operation.

Regarding grades of student coursework, this part is designed by the OO course staff. Generally, the score calculation is composed of multiple test cases where some test cases are the correctness of the program, and the other test cases are the program performance based on measurement of running time [14]. Evaluating codes quality and gaining grades of each student's homework from our online course platform, all information about grades is written into a file.

After introducing the structure of data sets, we can begin to produce data sets we actually use. In particular, we collect students' project data throughout the spring semester 2019 where 336 students engaged in the OO course. Since our target is giving early warnings to related students, we decide to collect three assignments in the early period of semester to analyse code quality. We gain a total of 1008 valid code projects which could be analyzed, and achieve statistical analysis results of students through combining projects with grades generated by our online course platform. Consequently, there are two kinds of data actually collected, namely, a) students' committing codes in the GitLab repository for each assignment, b) a score file generated by our OO course system.

### *B. Algorithm Results and Analysis*

Initially, we hope to utilize a set of code quality metrics of each student's three Object-oriented homework in the early stage of semester to predict final exam passing rate (score $\geq$ 60). With this idea, we want to early urge students who may fail to pass the examination to learn diligently and not fail at the end of the semester.

The specific operation is to combine code quality metrics generated by students' assignments with grades of the final examination in a given year. Then, we use LSTM [21] and GRU [22] neural networks to fit the metric data sets. It is found that the final consequence will always be over-fitting: all students are predicted to be qualified. Nevertheless, this phenomenon is not a reasonable conclusion we want.

When students finish all assignments in a whole semester, we find that the earlier the time, the more difficult it is for students to distinguish their levels of Object-oriented skills, which causes great difficulties of obtaining a direct classification result of passing examination or not. In addition, the learning abilities of computer science students has increased in recent years and the difficulty of coursework has also changed. Therefore, it is difficult to accurately measure the current students' abilities via a model trained by students' homework code data in other years. Naturally, we consider that current data sets may not be suitable for the usage of neural networks, and clustering method is devised to detect students' performance.

After students' OOP abilities are classified by the clustering method, we can utilize the distance calculation method introduced in Section II-B to allocate a category for each student. Then, we use a graphic primitive called dimensional anchor (DA) [23] to display the classification results generated by our algorithm, which could project a N-dimensional data set into a 2D space.

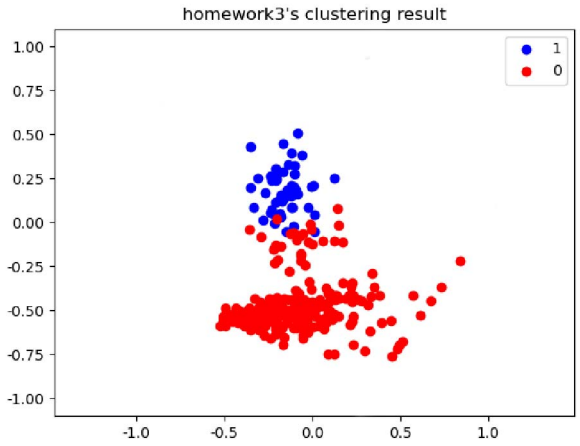


Fig. 4. Homework3's clustering result

We draw a diagram about the third coursework as an example to display the classification result shown in Fig. 4, where the blue points represent students with good OOP abilities and the red points represent students with poor OOP abilities. Analyzing the classification result depicted in Fig. 4, it can be apparently observed that the students' OOP skills are clearly separated. Therefore, the classification result is acceptable, which does work that combining K-means clustering algorithm with design quality assessments to measure students' OOP abilities.

In addition to the distance calculation method introduced in Section II-B, we also try two other distance calculation methods to classify students' OOP skills.

- **DSCC** (Distance from Single Clustering Center): We calculate distances between a single clustering center point and other points which belong to the cluster, and sort all distances in descending order.
- **EMCC** (Experience Method without Clustering Center): According to (1), without clustering centres, quality values of each clustering point are sorted in descending order.
- **DBCC** (Distance from Both Clustering Centers): We calculate the sum of Euclidean distance between each metric vector and two clustering center points. The detail of this method has been discussed in Section II-B.

Combining three distance calculation methods with the three assignments, obtaining coursework grades given by our OO course platform and setting the threshold to 10, we gain the following results through checking the number of students who

fit the clustering algorithm results manually. In particular, we consider that students whose coursework grades are greater than or equal to 90 are adept at passing test cases and students whose coursework scores are less than 60 are not good at passing test cases.

TABLE I
STUDENTS WITH POOR OBJECT-ORIENTED SKILLS BUT HIGH SCORES IN COURSEWORK

| accuracy | DSCC | EMCC | DBCC |
|---|---|---|---|
| coursework1 | 0.4 | 0.7 | **0.9** |
| coursework2 | 0.5 | 0.5 | **0.6** |
| coursework3 | 0.5 | **0.7** | **0.7** |

TABLE II
STUDENTS WITH POOR OBJECT-ORIENTED SKILLS AND LOW SCORES IN COURSEWORK

| accuracy | DSCC | EMCC | DBCC |
|---|---|---|---|
| coursework1 | 0.5 | **0.6** | **0.6** |
| coursework2 | 0.7 | 0.6 | **0.8** |
| coursework3 | **0.8** | 0.7 | **0.8** |

TABLE III
STUDENTS WITH GREAT OBJECT-ORIENTED SKILLS BUT LOW SCORES IN COURSEWORK

| accuracy | DSCC | EMCC | DBCC |
|---|---|---|---|
| coursework1 | 0.5 | **0.7** | **0.7** |
| coursework2 | 0.6 | **0.7** | **0.7** |
| coursework3 | **1.0** | **1.0** | **1.0** |

Analyzing comparison results in these tables, a conclusion can be summarized that we can generally gain a better result via DBCC. Later, we choose one student from the three coursework respectively to display their codes and judge whether each student's code meets our expectations. The reason why three distance methods obtain 1.0 accuracy in table III is that there are only three students with great OOP knowledge but low grades in our coursework 3 and those students can be observed in Fig. 2.

As for coursework 1, we choose the type of students with poor OOP skills but high grades in the assignment. In particular, the content of homework 1 is obtaining simple polynomial derivatives. The structure of the chosen student's codes is displayed in Fig. 5 in which it is found that the student does not follow the coding rule of Camel-Case and override the default constructor, and all methods are static methods. Meanwhile, there are no public fields set in class, while only static private variables are provided. Thus, we consider that the student is not skilled at OOP.

In coursework 2, we choose the type of students with poor OOP skills and low grades in the assignment. In particular, the content of homework 2 is obtaining derivatives of simple power functions and simple sine and cosine functions. The structure of the chosen student's codes is shown in Fig. 6 in which there are the same problems with previous student we discussed. In addition, the number of test cases that the student fails to pass is relatively more than the previous student.

```
public class Homework1 {
    private static ArrayList<String> result = new ArrayList();
    private static Pattern pattern = Pattern.compile("");
    static boolean judge(String s) {...}
    static String Simplify(String s) {...}
    static String buildmult(BigInteger ratiobig, BigInteger indexbig) {...}
    static String expspl(String s) {...}
    static String combine(String ratio, String index) {...}
    static void insert(String s) {...}
    public static void main(String [] args) {...}
}
```

Fig. 5. A student with poor OOP skills but high grades

```
public class main {
    public static void main(String [] args) {...}
    private static String inputdeal(Scanner line) {...}
    private static String calindex(String str) {...}
    private static String caltri(String str) {...}
    private static String calall(String str,int flag) {...}
    public static String removeCharAt(String s, int pos) {
        return s.substring(0, pos) + s.substring(pos + 1);
    }
}
```

Fig. 6. A student with poor OOP skills and low grades

In coursework 3, we choose the type of students with great OOP skills but low grades in the assignment. In particular, the content of homework 3 is obtaining derivatives of complex power functions and sine and cosine functions. Meanwhile, this homework is paid more attention to the performance of programs. In this assignment, the chosen student writes nine classes and the inheritance dependency diagram of these classes is shown in Fig. 7 where the black lines with arrow represent the inheritance relationship. It can be found from Fig. 7 that the student makes use of the inherited thinking in Object-oriented programming, and all written classes are named normatively. Nevertheless, the student fails to pass many test cases which leads to only a score of 29.11 he obtains.

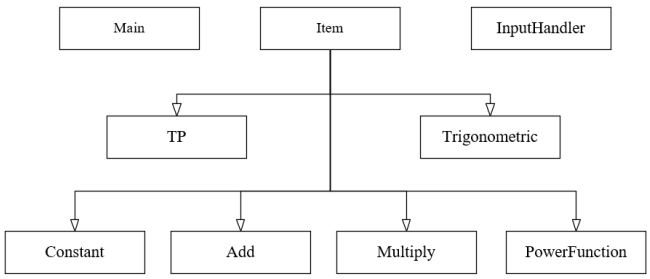


Fig. 7. Inheritance dependency diagram

We can clearly observe the distinguishable results where our method indeed recognizes students' problems in Object-oriented programming and gives feedback to instructors and related students. Therefore, there is no doubt that experimental results show the effectiveness of our approach.

## V. Conclusion and Future Work

In this paper, we provide a novel teaching method that is practiced in our Object-oriented programming course for undergraduate students in Computer Science. The contribution of our work can be summarized into the following three aspects.

First, we propose a systematic assessment model to detect students' practicing problems in our OO courses via integrating code quality metrics with coursework scores, rather than merely delivering experience reports or proposals with no evaluations. Meanwhile, our approach can distinguish four types of students and give some advice to related students in early semester period. Moreover, this study can provide feedback to teachers for optimizing their course design through problems detected by our approach, and thus better improves students' achievement. Finally, the analysis results help us to fully understand students' OOP skills reflected by the code quality metrics and coursework grades in early curriculum process based on our assessment model, which could give some suggestions to related students as soon as possible. With the received early warnings about their programming skills and coursework grades, those students will have opportunities to obtain more targeted guidance from instructors and teaching assistants in our OO course.

Different from merely relying on the industry principles, pedagogical guidelines or teachers' teaching experience, instructors will better understand how students' programming skills are developed based on the results given by our method. Integrating developing trend of students' programming skills with coursework scores, teachers could gain relevant feedback on quality of instruction in the teaching practice.

Although we collect the realistic course data from our online teaching platform, the scale of the data sets is relatively small due to the student number and we use the traditional machine learning algorithm which is suitable for the small data sets. However, the larger data sets are preferred. We will evaluate the effectiveness of our proposed algorithm on large-scale data sets and compare with other popular approaches such as neural models in the future.

## Acknowledgment

We would like to thank the anonymous reviewers for their insightful feedback and comments, and all the teaching assistants for their significant contributions to the course and the online platform. This research is supported by the Teaching Reform Projects of Beihang University.

## References

[1] S. Xinogalos, "Object-oriented design and programming: an investigation of novices' conceptions on objects and classes," *ACM Transactions on Computing Education (TOCE)*, vol. 15, no. 3, pp. 1–21, 2015.

[2] L. F. Capretz, "A brief history of the object-oriented approach," *ACM SIGSOFT Software Engineering Notes*, vol. 28, no. 2, p. 6, 2003.

[3] M. Kölling and J. Rosenberg, "Guidelines for teaching object orientation with java," *ACM SIGCSE Bulletin*, vol. 33, no. 3, pp. 33–36, 2001.

[4] R. Or-Bach and I. Lavy, "Cognitive activities of abstraction in object orientation: an empirical study," *ACM SIGCSE Bulletin*, vol. 36, no. 2, pp. 82–86, 2004.

[5] A. Fjuk, J. Bennedsen, O. Berge, and M. E. Caspersen, "Learning object-orientation through ict-mediated apprenticeship," in *IEEE International Conference on Advanced Learning Technologies, 2004. Proceedings.*, pp. 380–384, IEEE, 2004.

[6] M. Torchiano and G. Bruno, "Integrating software engineering key practices into an oop massive in-classroom course: An experience report," in *Proceedings of the 2nd International Workshop on Software Engineering Education for Millennials*, pp. 64–71, 2018.

[7] Q. Sun, J. Wu, W. Rong, and W. Liu, "Formative assessment of programming language learning based on peer code review: Implementation and experience report," *Tsinghua Science and Technology*, vol. 24, no. 4, pp. 423–434, 2019.

[8] V. R. Basili, L. C. Briand, and W. L. Melo, "A validation of object-oriented design metrics as quality indicators," *IEEE Transactions on software engineering*, vol. 22, no. 10, pp. 751–761, 1996.

[9] B. Curtis, S. B. Sheppard, P. Milliman, M. Borst, and T. Love, "Measuring the psychological complexity of software maintenance tasks with the halstead and mccabe metrics," *IEEE Transactions on software engineering*, no. 2, pp. 96–104, 1979.

[10] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on software engineering*, vol. 20, no. 6, pp. 476–493, 1994.

[11] Ş. Aydoğdu, "Predicting student final performance using artificial neural networks in online learning environments," *Education and Information Technologies*, vol. 25, no. 3, pp. 1913–1927, 2020.

[12] Y. Ma, J. Zong, C. Cui, C. Zhang, Q. Yang, and Y. Yin, "Dual path convolutional neural network for student performance prediction," in *International Conference on Web Information Systems Engineering*, pp. 133–146, Springer, 2020.

[13] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[14] Q. Sun, J. Wu, and K. Liu, "Toward understanding students' learning performance in an object-oriented programming course: The perspective of program quality," *IEEE Access*, vol. 8, pp. 37505–37517, 2020.

[15] M. Lanza and R. Marinescu, *Object-oriented metrics in practice: using software metrics to characterize, evaluate, and improve the design of object-oriented systems*. Springer Science & Business Media, 2007.

[16] T. Sharma and D. Spinellis, "Do we need improved code quality metrics?," *arXiv preprint arXiv:2012.12324*, 2020.

[17] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[18] Q. Sun, J. Wu, and K. Liu, "How are students' programming skills developed: an empirical study in an object-oriented course," in *Proceedings of the ACM Turing Celebration Conference-China*, pp. 1–6, 2019.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[20] B. Henderson-Sellers, *Object-oriented metrics: measures of complexity*. Prentice-Hall, Inc., 1995.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[23] P. Hoffman, G. Grinstein, and D. Pinkney, "Dimensional anchors: a graphic primitive for multidimensional multivariate information visualizations," in *Proceedings of the 1999 workshop on new paradigms in information visualization and manipulation in conjunction with the eighth ACM internation conference on Information and knowledge management*, pp. 9–16, 1999.