An automated approach for classifying reverse-engineered and forward-engineered UML class diagrams

2018 44th Euromicro Conference on Software Engineering and Advanced Applications

An Automated Approach for Classifying Reverse-engineered and Forward-engineered UML Class Diagrams

Mohd Hafeez Osman*¶, Truong Ho-Quang‡ and Michel R.V. Chaudron‡
*Department of Computer Science,
Technical University of Munich, Germany
Email: hafeez.osman@tum.de
‡Department of Computer Science and Engineering
Chalmers & Gothenburg University of Technology, Sweden
Email: {truongh,chaudron@chalmers.se}
¶Faculty of Computer Science and Information Systems
University Putra Malaysia, Malaysia

***Abstract*—UML Class diagrams are commonly used to describe the designs of systems. Such designs can be used to guide the construction of software. In practice, we have identified two main types of using UML: i) FwCD refers to diagrams are hand-made as part of the forward-looking development process; ii) RECD refers to those diagrams that are reverse engineered from the source code; Recently, empirical studies in Software Engineering have started looking at open source projects. This enables the automated extraction and analysis of large sets of project-data. For researching the effects of UML modeling in open source projects, we need a way to automatically determine the way in which UML used in such projects. For this, we propose an automated classifier for deciding whether a diagram is an FwCD or an RECD. We present the construction of such a classifier by means of (supervised) machine learning algorithms. As part of its construction, we analyse which features are useful in classifying FwCD and RECD. By comparing different machine learning algorithms, we find that the Random Forest algorithm is the most suitable algorithm for our purpose. We evaluate the performance of the classifier on a test set of 999 class diagrams obtained from open source projects.**

1. Introduction

In the early stages of the SDLC, class diagrams may be used to represent the architectural software design. As development progresses, class diagrams can be used to represent information that is closer to the construction of the system. During or after the implementation of source code, a class diagram may be recovered using reverse engineering techniques. Such a reverse engineered class diagram is closely based on the source code and reflects the fine-grain implementation structure of software systems [1].

Hebig et. al [2] present Lindholmen dataset which is a repository of UML diagrams built to serve as an informative collection of UML models. This repository holds more than 26,000 UML class diagrams and includes links to the projects on GitHub where the diagrams were found. As such it forms a valuable resource for empirical studies on projects that use some forms of UML modeling. We argue that different practices of UML use (such as use of forward-design (FW) and reverse engineering (RE) diagrams) could lead to different effects on various aspects of software systems. However, these effects are often overlooked within a small set of softwares. The automated classification of FwCD and RECD will hopefully enable researcher to study the effects on a bigger population of open source software systems.

Goal. This study aims at providing an automated classification model for classifying Forward-designed Class Diagram (FwCD) [3] and Reverse Engineered Class Diagram (RECD) [3]. We apply supervised machine learning techniques as the method for the classification of diagrams. We use a dataset of 999 class diagrams that are collected from Lindholmen dataset. In order to obtain a ground truth, this dataset is labelled by experts that have experience in working with UML diagrams. The classification features are extracted based on (i) experts judgment, (ii) the work by Osman and Chaudron [4], and (iii) the work by Nugroho and Chaudron [5]. We evaluate 11 classification algorithms that cover the diverse type of algorithms in supervised machine learning, in order to select the best classification algorithm for this problem.

Contribution. The contributions of this study are the following: (i) identification of features that can be used to classify FwCD and RECD diagrams, (ii) a suitable machine learning algorithm for classifying FwCD and RECD, (iii) a dataset with ground truth for classifying FwCD and RECD and, (iv) comparative analysis of the performance of various machine learning algorithms for our problem.

This paper is structured as follows. Section 2 discusses the related work. Section 3 describes the research questions. Section 4 explains the approach. The analysis of results is in Section 5. This followed by conclusions in Section 6.

978-1-5386-7383-6/18/$31.00 ©2018 IEEE
DOI 10.1109/SEAA.2018.00070

2. Related Work

To our best knowledge, there is no work that is directly focused on automatically classify of FwCD and RECD. Therefore, we broaden our discussion to works that have used class diagram information (e.g. metrics) and machine learning classification algorithms for classification purposes.

Maneerat and Muenchaisri [6] proposed a method for predicting bad-smell from software design model. This study used 27 standard software metrics proposed by Abreu in [7]. Halim [8] proposed a method to predict fault-prone classes using the complexity metrics of UML class diagram. The prediction models were built using two classification algorithm i.e. Naive Bayes and k-Nearest Neighbors. Bagheri and Gasevic [9] investigated through controlled experimentation whether a set of structural metrics can be good predictors of the three main sub-characteristics of maintainability: analyzability, changeability, and understandability.

Nugroho performed a study on predicting defects based on diagram metrics [5]. For this he developed metrics that aim to capture the level of detail/abstraction of UML diagrams. Examples of the metrics that they propose are: the percentage of association-relations that has labels, the percentage of methods that have signatures. His study showed that higher level of detail in UML models correlates with fewer defects in source code. This suggests that when our classifier uses level of detail metrics, it might also be used in quality assurance of models. Osman et. al [10] proposed an approach to condense reverse engineered class diagram by using machine learning techniques. They used standard object-oriented design metrics i.e. size measures and coupling measures. Based on this work, Thung et. al [11] improved the classification result by adding Network measure (e.g. Baycenter, PageRank).

3. Research Questions

This section describes the research objective and associated questions.

The overall goal of this study is to automatically distinguish forward-designed class diagrams from reverse engineered diagrams. To this end, we need to find out the following: **RQ1**: Which features of UML diagrams are influential predictors for classifying diagrams into FwCD and RECD? And also, **RQ2**: Which machine learning algorithms are suitable for this task?

4. Approach

This section describes our approach that consists of Data collection, Features extraction & Creating ground truth, Model learning, and Evaluation of results.

4.1. Data Collection

For this study, we use UML class diagrams images that were collected from the Lindholmen dataset [2]. By scanning 4443 projects, we collected 2000 class diagrams that are stored in various image file formats. We refined the dataset by selecting reasonable-quality images and by removing duplicate images for the dataset. We finalized a list of 999 class diagram images for this study. The dataset can be found at [12].

Because the collected class diagrams were stored in image formats, it was necessary to extract the (model) content of the diagrams in a file format: for this we used the XMI (XML Metadata Interchange) standard. The conversion of UML class diagrams in image formats into XMI format was done using the Image2UML tool [13]. In doing so, we extended the Image2UML tool's capabilities to capture additional information such as operator parameters to which the original version did not support.

4.2. Feature Extraction & Ground Truth

Supervised machine learning needs a labelled dataset for learning purposes. As there are no explicit rules on how to distinguish FW- from RE-diagrams, this labelling was done manually by three selected UML experts who have at least five years of experience on using UML class diagrams. The labelling process consists of two phases: (i) All the experts gathered together in a brainstorming session to outline the characteristics of FwCD and RECD. After the experts defined the FwCD and RECD characteristics, we randomly selected 30 class diagrams images from the dataset and let the experts discuss and refine the original characteristics for classifying the diagrams. (ii) Each expert is randomly assigned a set of 323 class diagrams. For each diagram, every expert needs to classify it into different categories: "FW Design", "RE Design" and "For Discussion". Diagrams in "For Discussion' were discussed in follow-up group meetings. The loop of individual classification and group discussion is continued until all class diagram images are classified into FW and RE.

In earlier work, Osman et. al [4] reported several characteristics that are typical of UML diagrams that are obtained by reverse engineering technique (by using selected commercial UML-CASE tools). These characteristics are the basis for our selection of candidate classification features. To this set, we added features that the aforementioned experts elicited when labelling diagrams as FwCD and RECD. More detailed explanations on the candidate features are described in Table 1. Finally, we merge the data file with the label information (isFwd - "Yes" or "No").

4.3. Model Learning

Mining data is experimental. There is no algorithm that fits all situations and all purposes. We start the selection of machine learning algorithm by conducting an exploratory experiment on a range of machine learning algorithms.

The algorithms in this experiment are selected from the different set of algorithms representative for different approaches. For example, Decision Trees, Stumps, Tables and Random Trees or Forests all divide the input space up

TABLE 1. LIST OF FEATURES & INFOGAIN RESULTS

No	Features	Description	Data Type	InfoGain Value
1	avgParaOper	Average parameter per operation	Numeric	0.3191
2	numPara	The total number of parameter in the class diagram	Numeric	0.2897
3	extOperPara	"true" if the operation parameter exist and "false" if it is not exist	Nominal (Binary)	0.2371
4	avgOperCls	Average operation per class	Numeric	0.2249
5	maxOperCls	Select the highest number of operation (for a class) in the class diagram	Numeric	0.1602
6	avgAssocCls	Average association per class	Numeric	0.1597
7	numCls	The total number of classes in the class diagram	Numeric	0.1319
8	numAssoc	The total number of association in the class diagram	Numeric	0.1304
9	numOper	The total number of operation in the class diagram	Numeric	0.1265
10	numOrpCls	The total number of orphan classes in the class diagram	Numeric	0.858
11	avgOrpCls	Average orphan classes per class diagram	Numeric	0.0668
12	avgAttrCls	Average attribute per class	Numeric	0.0605
13	numAttr	The total number of attribute in the class	Numeric	0.0551
14	maxAttrCls	Select the highest number of attribute (in a class) in the class diagram	Numeric	0.0377
15	extOrpCls	"true" if the orphan (unconnected) classes exist and "false" if it is not exist	Nominal (Binary)	0.0238
16	numAssocType	Count the number of assocation type that exist in the class diagram	Numeric	0.0118

into disjoint smaller sub-spaces and make a prediction based on the occurrence of positive classes in those sub-spaces. K- Nearest Neighbour (k-NN) and Radial Basis Functions (RBF) Networks are similar local approaches, but the subspaces here are overlapping. In contrast, Logistic Regression and Naive Bayes model parameters are estimated based on potentially large numbers of instances and can thus be seen as more global models [10]. OneR [14] is selected to represent the simplest classification algorithm compared to the algorithms mentioned above. We use the result of ZeroR as the baseline (only on the accuracy measurement). ZeroR shows the probability of a guess of whether a class diagram is ReCD or FwCD. The detail explanations of the aforementioned algorithms can be found at [15]. The classification models are constructed by using all aforementioned (16) features. The stratified 10-fold cross-validation is used for evaluating the classification model performance. To ensure a more accurate validation result, the 10-fold cross-validation is repeated ten (10) times for every classification model. This activity is supported by WEKA tool [15].

4.4. Evaluation of Results

To measure predictive power of predictors, we used the information gain with respect to the class [15]. Univariate predictive power means measuring how influential a single predictor is in prediction performance. This study uses this analysis for the data exploration. We use WEKA's Information Gain Attribute Evaluator (InfogainAttrEval) in conducting this experiment. The WEKA's InfogainAttrEval produces a value from 0 to 1. The higher value of InfoGain (close to 1) denotes a stronger influence of the predictor.

In general, we use three evaluation measures to evaluate the classification algorithms performance i.e. (i) Percentage of correct (accuracy) [16], (ii) Precision [17] and, (iii) Recall [17]. If required, we extend this evaluation into more detail measures such as F-Measure [18] and Area Under Receiver Operating Characteristic (ROC) Curve (a.k.a AUC) [19].

5. Result and Findings

This section evaluates the performance of the selected features and the classification algorithms.

5.1. RQ1: Analysis of Selected Features

Table 1 shows InfoGain results for our 16 features. (predictors) produce InfoGain score >0. This result shows that every single feature used in this study has some predictive power. The average number of operation parameters (*avgParaOper*) is the most influential predictor. The three most influential features are related to the operation parameters. Meanwhile, the other most influential features are the average number of operations per class (*avgOperCls*) and the maximum number of operation per class (*maxOperCls*). Both features are related to the operations in class diagrams. Thus, this result indicates that the class diagram operations plays a major role in classifying the RECD and FwCD. In the future, we would like to evaluate the influence of a group of features in classifying FwCD and RECD. We also plan to analyze the correlation between features and come out with better predictor/feature set.

5.2. RQ2: Classification Model Performance

The accuracy value for ZeroR is 80.68, which means our dataset is imbalanced: the majority of the class diagram in the dataset is RECD. The results show (see Table 2) that all classification models produce a significant improvement compared to ZeroR. The relative improvement of accuracy values ranges from 7% to 10%.

The second baseline is OneR. The OneR classification algorithm uses only one (most influential) feature to construct the classification model. This benchmark experiment is conducted because according to Holte [14], there is a possibility that a simple algorithm works well in a dataset. OneR represents the simplest classification model. Thus, complex classification algorithms should perform better to

TABLE 2. CLASSIFICATION PERFORMANCE

Performance Measure	Accuracy	Precision	Recall	F-Measure	AUC
OneR	88.01	0.94	0.91	0.92	0.84
Decision Table	88.33	0.93	0.93	0.93	0.93
Naive Bayes	88.10	0.97	0.88	0.92	0.91
RBFNetwork	89.32	0.95	0.91	0.93	0.92
Logistic Reg.	88.90	0.94	0.93	0.93	0.94
SVM	87.28	0.88	0.97	0.93	0.71
K-NN (1)	89.16	0.93	0.93	0.93	0.88
K-NN (5)	87.89	0.93	0.92	0.92	0.93
Decision Stump	87.69	0.98	0.87	0.92	0.89
J48	88.59	0.94	0.91	0.93	0.85
Random Tree	87.89	0.92	0.93	0.93	0.81
Random Forest	90.74	0.95	0.93	0.94	0.96

produce a significant improvement because they require greater computational effort (e.g. for extracting all features). The results show that the performance of SVM, k-NN (5), Decision Stump and Random Tree are slightly lower than OneR (based on accuracy value). Thus, we exclude these algorithms from further evaluation.

After benchmarking the classification algorithms against the baselines, we compare the classification algorithms' performance based on precision and recall. Decision Table, Logistic Regression, k-NN (1) and Random Forest show a balance score between precision and recall. Hence, based on this result, Decision Table, Logistic Regression, k-NN (1) and Random Forest are suitable algorithms for our problem and dataset. Based on the F-Measure and AUC score, the Random Forest is the best performing classification algorithm for our purpose.

We see there is a possibility to enhance this work by the following: (i) reconfigure the classification parameter, (ii) combination of classification algorithms, (iii) extend our approach to larger datasets, and (iv) classifying class diagrams into application domains by using text mining techniques.

6. Conclusions

This work presented the construction and evaluation of an automated classifier for differentiating forward-designed (FwCD) from reverse-engineered (RECD) class diagrams. This classifier was constructed using machine learning algorithms. We investigated various properties of class diagrams as features of our classifier. We have shown that the features that relate to parameters of operations are the most influential features for this classification purposes. We found that Random Forest is the most suitable classification algorithms for our classification model.

As for the conclusion, this study has formulated an automated classification model to classify FwCD and RECD. The classification model performed reasonably well based on the scores benchmark. As part of our future work, we would like to apply this model to our recently collected corpus of class diagrams (in total 24000+) from Lindholmen dataset. Through this research, we expect to get an understanding of the different ways in which UML diagrams are used in open source projects and ultimately an understanding of the effectiveness of various modeling and documentation practices.

References

[1] M. H. B. Osman, "Interactive scalable condensation of reverse engineered UML class diagrams for software comprehension," Ph.D. dissertation, Leiden University, 2015.

[2] R. Hebig, T. H. Quang, M. R. V. Chaudron, G. Robles, and M. A. Fernandez, "The quest for open source projects that use uml: mining github," in *19th MODELS Conference*. ACM, 2016, pp. 173–183.

[3] E. J. Chikofsky and J. H. Cross, "Reverse engineering and design recovery: A taxonomy," *IEEE software*, vol. 7, no. 1, 1990.

[4] H. Osman and M. R. V. Chaudron, "An assessment of reverse engineering capabilities of UML case tools," in *2nd Int. Conf. on Software Engineering Application*, 2011, pp. 7–12.

[5] A. Nugroho, B. Flaton, and M. R. V. Chaudron, "Empirical analysis of the relation between level of detail in UML models and defect density," in *Conference, MoDELS 2008, Toulouse, France, 2008. Proceedings*, ser. LNCS, vol. 5301. Springer, 2008, pp. 600–614.

[6] N. Maneerat and P. Muenchaisri, "Bad-smell prediction from software design model using machine learning techniques," in *2011 Eighth International Joint Conference on Computer Science and Software Engineering (JCSSE)*, May 2011, pp. 331–336.

[7] F. B. e Abreu, "The mood metrics set," in *proc. ECOOP*, vol. 95, 1995, p. 267.

[8] A. Halim, "Predict fault-prone classes using the complexity of UML class diagram," in *2013 Int. Conf. on Computer, Control, Informatics and Its Applications (IC3INA)*, Nov 2013, pp. 289–294.

[9] E. Bagheri and D. Gasevic, "Assessing the maintainability of software product line feature models using structural metrics," *Software Quality Journal*, vol. 19, no. 3, pp. 579–612, 2011.

[10] M. H. Osman, M. R. V. Chaudron, and P. v. d. Putten, "An analysis of machine learning algorithms for condensing reverse engineered class diagrams," in *2013 IEEE International Conference on Software Maintenance*, Sept 2013, pp. 140–149.

[11] F. Thung, D. Lo, M. H. Osman, and M. R. V. Chaudron, "Condensing class diagrams by analyzing design and network metrics using optimistic classification," in *Proc. of the 22nd Int. Conf. on Program Comprehension (ICPC)*. ACM, 2014, pp. 110–121.

[12] Replication package. [Online]. Available: http://oss.models-db.com/Downloads/SEAA2018/ReplicationPackage/

[13] B. Karasneh and M. R. V. Chaudron, "Img2uml: A system for extracting uml models from images," in *2013 39th Euromicro Conference on Software Engineering and Adv. Applications*, Sept 2013, pp. 134–137.

[14] R. C. Holte, "Very simple classification rules perform well on most commonly used datasets," *Machine learning*, vol. 11, no. 1, 1993.

[15] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009.

[16] M. Junker, R. Hoch, and A. Dengel, "On the evaluation of document analysis components by recall, precision, and accuracy," in *Document Analysis and Recognition, 1999. ICDAR'99. Proceedings of the Fifth International Conference on*. IEEE, 1999, pp. 713–716.

[17] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation," in *Australasian joint conference on artificial intelligence*. Springer, 2006, pp. 1015–1021.

[18] D. Billsus and M. J. Pazzani, "Learning collaborative information filters." in *Icml*, vol. 98, 1998, pp. 46–54.

[19] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (roc) curve." *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.