https://dx.doi.org/10.22161/ijaers/3.11.21 *ISSN: 2349-6495(P) | 2456-1908(O)*

Software Complexity Prediction by Using Basic Attributes

Rasha Gaffer. M. Helali

Sudan University of Science and Technology, Sudan

Abstract— *Software complexity is one of the important quality attribute that affect the success of software. Predicting such attribute is a difficult task for software engineers. Current used measures for computing complexity are not sufficient. Data mining can be applied to software data to explore useful interesting patterns. In this paper we present a simple data mining based prediction model to predict software complexity based on some basic attributes. The article starts by considering the correlation between different features that describes software code structure then selecting some of these features to be used for complexity prediction. Results reveal the ability to use branching count feature as strong predictor of complexity.*

Keywords— Software complexity, LOC, McCabe, halstead, branch count.

I. INTRODUCTION

Software complexity is "a natural byproduct of the functional complexity that the code is attempting to enable" [1]. In literature, software complexity has been defined differently by many researchers [2]. Z use defines software complexity as the difficulty to maintain, change and understand software. Others view it as difficulty to develop, test, debug and maintain [2].Therefore, no standard definition exits for the same in literature. However, knowledge about software complexity represents an indicator of development, testing, and maintenance efforts, defect rate, fault prone modules and reliability [2].With multiple system interfaces and complex requirements, the complexity of software systems sometimes grows beyond control, rendering applications overly costly to maintain and risky to enhance [1]. The complexity is affected by many factors former to software development [3]. Understanding, predicting and resolving complexity of software are critical tasks that affect the success of software.

Software complexity can be measured by Direct Measures which is also known as internal attributes and Indirect Measures which is also known as external attributes. Direct Measures are measured directly such as Cost, effort, LOC, speed, memory. Indirect Measures cannot be measured directly. Example - Functionality, quality, complexity, efficiency, reliability, maintainability [4]. The common used complexity measures are:

- The cyclomatic complexity v(G) has been introduced by Thomas McCabe in 1976.The McCabe complexity is one of the more widely - accepted software metrics, it is intended to be independent of language and language format.
- The McCabe's [5] software complexity introduces the concept of Cyclomatic Complexity. It combines the number of flow graph edges, nodes and predicate nodes to represent the complexity. The Cyclomatic Complexity of a source code is the linearly independent paths count through the source code.
- The Halstead [6] software complexity measures the complexity by counting number of operators and operands in software. It measures the software's ability to understand and estimates the effort required to develop a software algorithm. Halstead metrics are difficult to calculate and it is very hard to count the distinct and total operators and operands in a software program.
- Metrics Suite for Object Oriented Design [7] proposed by Chidambaram Kamerer to measures complexity of object oriented software based on coupling and coherence between class.

Recently, many researches focused on predicting complexity because complexity prediction can help in estimating many other quality attributes like testability and maintainability. The main goal of this paper is to build predictive model by using data mining techniques to find out which attribute/s can help predicting complexity more than others. The subsequence sections are organized as follows: section II contains what had been done in the area of complexity prediction. Section III describes the proposed predictive model and used data set. Then the following sections highlight analysis, results and validation. Finally, conclusion is presented.

II. RELATED WORK

A number of studies investigate software complexity either as attribute to be predicted or as predictor to other attributes. Software complexity commonly used as indicator to fault prone class/modules. Moreover, several studies focused on the relationship between software complexity and software reliability and maintainability. More complex software is,

https://dx.doi.org/10.22161/ijaers/3.11.21 *ISSN: 2349-6495(P) | 2456-1908(O)*

less maintainability and reliability is Usha Chhillar and Sucheta Bhasin, pointed out that there is a relationship between complexity and possibility of faults [8]. Graylin et.al.Constructed a model to find correlation between McCabe's Cyclomatic Complexity (CC) and lines of code (LOC). Their model successfully predicts roughly 90% of CC's variance by LOC alone. D. Francis Xavier Christopher and E. Chandra[9], addressed software Requirements Stability Index Metric (RSI) that helps to evaluate the overall stability of requirements and also keep track of the project status. Their study proposes Multi-criteria Fuzzy Based approach for findingOut the complexity weight based on Requirement Complexity Attributes such as Functional Requirement Complexity, Non – Functional Requirement Complexity, Input Output Complexity, Interface and File Complexity. The advantage of their model is that it is able to estimate the software complexity early which in turn predicts the Software Requirement Stability during the software development life cycle.

N. J. Pizzi et.al. [10] Investigated a computational intelligence based strategy, random feature selection, as a classification system to determine the subset of software measures that yields the greatest predictive power for module complexity. Sabharwal.et.al.In[11]discussed how to use fuzzy logic based approach to predict complexity.On the same direction M S. Dattathreya, and H Singh used Fuzzy logic techniques for developing, modeling and analyzing the software complexity prediction metric. The authors propose five non-technical factor metrics based on the current software development process to predict future Army Vehicle software complexity [3].

Some studies moved towards computing software complexity differently, Henry and Kafura [3] provide the measure of couplings between modules in terms of number of parameters, global variables and function calls. In[13] authors introduce The Entropy software complexity measure based on the average information content of each operator in a software program's source code. An attempt was made by Jingqiu Shao and Yingxu Wang [3] to models the software complexity based on the cognitive functional size of the software. Although, many studies considered software complexity, still much research is required. Above literature leads to a conclusion that we need to find a way to use current available measures of software attributes to give an indicator to how software complexity is. We describe the proposed software complexity prediction model in a step wise manner as follows:

1- Data set and feature selection
2- Find correlation between software attributes.
3- Applying data mining techniques to predict complexity.

III. COMPLEXITY PREDICTIVE MODEL

Our analysis is divided into two main steps. The first step is to determine the software attributes (metrics) that can yield acceptable predictability then find the correlation between these attributes to select the most related attributes to complexity. The second step is using the selected attributes to build prediction models. These two steps/ phaseswere presented in the following two subsections.

- **Feature selection**

The study investigates the ability to use some basic attributes that describe software code to predict its expected complexity. The features/ attributes that are suggested simply including LOC, number of operatorsand number of operands, branch count, an estimation of complexity and used programming Language. We tried to find dataset including these features to be used for prediction purpose. The data used in this study is retrieved from online public repository PROMISE [14]. The original data is made available by Software Research Laboratory of Bogazici University [15]. The utilized data sets areembedded software products implemented in C. It contains the measurements of 21 static code attributes (complexity metrics) and 1 defect information (false/true) of tens tohundreds of modules. Module attributes were collected using "Prest Metrics Extraction and Analysis Tool" [15]. The collected attributes contains:

%
% 1.loc : numeric % McCabe's line count of code
% 2.v(g) : numeric % McCabe "cyclomatic complexity"
% 3.ev(g) : numeric % McCabe "essential complexity"
% 4.iv(g) : numeric % McCabe "design complexity"
% 5.n: numeric % Halstead total operators + operands
% 6.v: numeric % Halstead "volume"
% 7.l: numeric % Halstead "program length"
% 8.d: numeric % Halstead "difficulty"
% 9.i: numeric % Halstead "intelligence"
% 10.e: numeric % Halstead "effort"
% 11.b: numeric % Halstead
% 12.t: numeric % Halstead's time estimator
% 13.lOCode: numeric % Halstead's line count
% 14.lOComment : numeric % Halstead's count of lines of comments
% 15.lOBlank : numeric % Halstead's count of blank lines
% 16.lOCodeAndComment: numeric
% 17.uniq_Op: numeric % unique operators
% 18.uniq_Opnd: numeric % unique operands
% 19.total_Op: numeric % total operators
% 20.total_Opnd: numeric % total operands
% 21: branch Count : numeric % of the flow graph

https://dx.doi.org/10.22161/ijaers/3.11.21 *ISSN: 2349-6495(P) | 2456-1908(O)*

% 22.defects:{false,true} % module has/has not one or more
% % reported defects

According to suggested features only 18 attributes were used from the above list. We omit 4 attributes (design complexity, essential complexity, and b and time estimator t). It is important to note that all attributes were measured by using traditional known metrics (LOC, McCabe and Halstead). Our goal here is to figure out which of the above attributes can be used to predict complexity. Spearman's Correlation is done between these attributes table "1" below shows correlation results.

Table.1: Correlation results

	Complexity v(g)	
	r-value	p-value
LOC (McCabe's line count)	.889	0.00
Design complexity	.826	0.00
total operators + operands	.869	0.00
Volume	.872	0.00
Length	**-0.832**	0.00
Difficulty	.861	0.00
Intelligence	.733	0.00
Effort	..883	0.00
Loc code (Halstead's line count)	.569	0.00
Locomments	.635	0.00
Loblank	.676	0.00
LOC and comments	**-0.062**	.170
uniq_Op	.887	0.00
Uniqu_oernd	.843	0.00
total_Op	.869	0.00
total_Opnd	.860	0.00
Branch_ count	.999	0.00
Defects	.169	0.00

By considering r- value it is obvious that the most related attributes to complexity estimation which measured by McCabe are branch count, LOC and unique operators. In contrast, 2 attributes are not related to complexity such as Length whichmeasured by Halstead and (LOCand commands)measure that appears from negative r value resulting from correlation process. So, both of negative attributes are omitted from selected features. Finally, the rest of attributes (16 attributes) were fed to predictive model.It is important to note that for prediction purpose we assign two classes for complexity the first class is "high"if complexity value is greater than 20, second class is "Low" if complexity is less than 20.

- **Proposed predictive model**

The proposed predictive model as mentioned above consists of two main phases: feature selection phase and analysis/ prediction phase. Data mining generally used to extract previously unknown patterns help to improve or even predict new knowledge[16]. Data mining are integrated in analysis / prediction phasethat classifyingsoftware data as high or low complexity based on labeled training data. If complexity estimation is less than 20 it labeled as low, if greater than 20 labeledhigh. Decision tree classification algorithm C5.0 is used to perform classification process.The decision trees algorithms are classification algorithms for use in predictive modeling. They build a data mining model by creating a series of splits in the tree [16].The C5.0 algorithm was chosen for the following reasons. Firstly, it's simplicity. Secondly, it has boosting feature that mean using multiple classifiers instead of one to provide better classification accuracy. The output of this phase is three sets of software data in addition to set of classification rules. Figure 1 bellow shows the proposed prediction model.

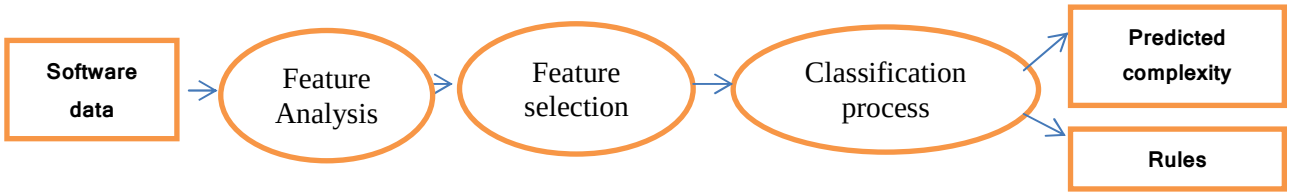

Fig.1: Prediction model

Total number of records fed to classifier is498records. Records are randomly split into two sets, a training set and a testing set. The training set used to create the mining model. The testing set used to check model accuracy.Trainingdata represents40% of total data/records. Results are listed in the following section.

IV. ANALYSIS AND RESULTS

Results confirm the existence of strong relation between branch count and complexity. C5.0 algorithm generates set of classification rules learned from training data as follows:

- If branch count between 1and 20 then complexity class is low.

https://dx.doi.org/10.22161/ijaers/3.11.21 *ISSN: 2349-6495(P) | 2456-1908(O)*

- If branch count is greater than 20 then complexity will be high.
- Special cases if branch count in [27 32 33 34 41 43 49 59 63 71] also complexity class Low

So, branch count can work as a good predictor to software complexity. Predication accuracy, prediction precision, and recall rate are commonly used metrics to evaluate the binary prediction models [14]. Classification accuracy is estimated by used mining algorithmequal to 100%. Table “2”presents the results after applying C5.0 to data.

Table.2: Prediction Results Details

	High	Low
Predicted high	24	0
Predicted Low	0	474

Accuracy =acc =100%
Probability of false alarm = pf = 0%
Probability of detection = pd = recall = 474/474=1
Precision = prec=474/474= 1

For validation purpose six programs are selected and tested according to the above rules. The selected programs are: stack,Gzip, print tokens, and arraysorting, binary search and replace programs. Branch count ismeasured for the five programs and also complexity estimated by using McCabe metric then rules are checked for validation.According to Validation results we can say branch count predict complexity correct 100%. Table 3 shows validation results.

Table.3: Validation Results

Program name	Branch count	Complexity v(G)	Class	Check
Stack	2	18	Low	correct
Gzip	100	1260	high	correct
Array sorting	2	6	Low	correct
replace	28	92	high	correct
Print tokens	61	79	high	correct
Binary search	2	4	low	correct

V. CONCLUSIONS

In this paper, a simple data mining based complexity prediction model were presented. Model depends on some attributes measured using traditional metrics from code structure.The most important aspect of the model was to figure out which attribute could be used as predictor to software complexity. Results find strong relation between complexity and branch count feature.

REFERENCES

[1] Application Analytics Software,What is Software Complexity:http://www.castsoftware.com/glossary/software-complexity.
[2] Chhillar, Usha, and SuchetaBhasin. "Establishing Relationship between Complexity and Faults for Object-Oriented Software Systems." IJCSI International Journal of Computer Science Issues 8.5 (2011).
[3] Dattathreya, Macam S., and Harpreet Singh."Army Vehicle Software Complexity Prediction Metric-Five Factors."
[4] Bhatnagar, Anurag, NikharTak, and Shweta Shukl. "A LITRERATURE SURVEY ON VARIOUS SOFTWARE COMPLEXITY MEASURES." International Journal of Advanced Studies in Computers, Science and Engineering 1.1 (2012): 1.
[5] McCabe, T.J. A Complexity Measure, IEEE Trans. On Software Engg., SE-2, 4, 1976, pp. 308-320
[6] Halstead, M.H. Elements of Software Science, New York: Elsevier North Holland, 1977.
[7] Jamali, Seyyed Mohsen. "Object oriented metrics." A survey approach Technical report, Department of Computer Engineering, Sharif University of Technology, Tehran, Iran (2006).
[8] Chhillar, Usha, and SuchetaBhasin. "Establishing Relationship between Complexity and Faults for Object-Oriented Software Systems." IJCSI International Journal of Computer Science Issues 8.5 (2011).
[9] International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.6, November 2012 doi : 10.5121/ijsea.2012.360 8 101 prediction of software requirements stability based on complexity point measurement u sing m ulti – c riteria fuzzy approach d. francisxavierchristopher 1 and e.chandra
[10] N. J. Pizzi, "A Computational Intelligence Strategy for Software Complexity Prediction," Neural Networks, 2006.IJCNN '06. International Joint Conference on, Vancouver, BC, 2006, pp. 4727-4733. doi: 10.1109/IJCNN.2006.247127
[11] S. Sabharwal, R. Sibal and P. Kaur, "Software complexity: A fuzzy logic approach," Communication, Information & Computing Technology (ICCICT), 2012 International Conference on, Mumbai, 2012, pp. 1-6.
[12] Harrison, Warren. "An entropy-based measure of software complexity." Software Engineering, IEEE Transactions on 18.11 (1992): 1025-1029.
[13] S.W. QingWANG, L.I. Mingshu, Software defect prediction, J SoftwMaintEvol: Res Practice, 19 (7) (2008), pp. 1565–1580
[14] S. J. Sayyad and T. J. Menzies, The PROMISE Repository of Software Engineering Databases, School

https://dx.doi.org/10.22161/ijaers/3.11.21 *ISSN: 2349-6495(P) | 2456-1908(O)*

of Information Technology and Engineering, University of Ottawa, Can-ada, http://promise.site.uottawa.ca/SERepository.

[15] Graylin, J. A. Y., et al. "Cyclomatic complexity and lines of code: empirical evidence of a stable linear relationship." Journal of Software Engineering and Applications 2.03 (2009): 137.

[16] Yousef, Ahmed H. "Extracting software static defect models using data mining." Ain Shams Engineering Journal 6.1 (2015): 133-144.