REVIEW OF EVALUATION OF SOFTWARE UNDERSTANDABILITY USING SOFTWARE MATRICS

Apeksha Nayak[*],Anurag Chandna[1],Ankur Goel[2]

[*]M.Tech Scholar, Department of Computer Science and Engineering, Roorkee College of Engineering Uttarakhand Technical University, Dehradun India

[1]Asst. Professor, Computer Science & Engineering, Roorkee College of Engineering, Roorkee, India,

[2]Asst. Professor, Computer Science & Engineering, Roorkee College of Engineering, Roorkee, India

ABSTRACT

In this we discussed mainly the previous work, which was done on the Evaluation of understandability. For that we have gone through different techniques like Fuzzy set approach. After that, we have concentrated on particular metric like Spatial complexity.Here they consider the factors like Understan-dability of documentation, Understandability of structure, Understandability of components, Understandability of source code, and Unders-tandability of data. In order to explain these factors they considered few metrics to measure it. When a worker needed to reconstruct one software system as many times until he/she correctly reconstructs it, it can be considered the software system is difficult for him/her to understand.

Keywords: *Understandability, Fuzzy Matrix, Spatial Complexity*.

1-INTRODUCTION

Jin-Cherng Lin et al.[16]explained a model for measuring software understandability. For this they have considered the few factors which affect the understandability. Here they consider the factors like Understandability of documentation, Understandability of structure, Understandability of components, Understandability of source code, and Understandability of data. In order to explain these factors they considered few metrics to measure it. Comments Ratio(CR) is used to judge the Readability of Source Code(RSC), Quality of Documentation(QOD) is judged using Fog Index. They came with the statement like, Understandability of structure and the number of components are correlated. When a worker needed to reconstruct one software system as many times until he/she correctly reconstructs it, it can be considered the software system is difficult for him/her to understand. In this method they are evaluating software understandability sources from the fuzzy equivalent matrix. Here first they calculated the degree or distance of the similarities between different samples x1,x2,x3,...xn, they selected two clusters with the most similar degree or shortest distance to classify into one class, and then calculate the similar degree or distance between new class and other class, select two classes with the most similar degree or shortest distance to classify into one new class, each classification shall reduce one class until all samples to be classified into one class.

2-EVALUATION

Software Understandability Based on Fuzzy Matrix

Let samples x1,x2,x3,...xn in the software engineering, each sample xi(i=1,2,3...n) has m characteristic indexes, namely xi=(xi1,xi2,x3,...xim).

The data of n samples can be represented in matrix form.

$$X=\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nm} \end{bmatrix}$$

The dimension of m characteristics index are different, and consequently resulted in a shortage of one canonical in the classifying of the characteristic indexes, so, we must undertake data standardization to indexes. After that they establish fuzzy similar matrix between samples.

$$\tilde{R}=\begin{bmatrix} r_{11} & r_{12} & \dots & r_{1m} \\ r_{21} & r_{22} & \dots & r_{2m} \\ \vdots & \vdots & & \vdots \\ r_{n1} & r_{n2} & \dots & r_{nm} \end{bmatrix}$$

where rij($0 \leq rij \leq 1$,i,j=1,2,3...n) indicates the similar degree between sample xi and sample xj

Adopt max-min composition to establish fuzzy similar matrix.

$$r_{ij} = \frac{\sum_{i=1}^{m} \min(x_{ik}, x_{jk})}{\sum_{i=1}^{m} \max(x_{ik}, x_{jk})} (\, i, j \leq n \,)$$

If rij=0, indicating that sample xi is dissimilar to sample xj; If rij=1, indicating that sample xi is similar or equivalent to sample xj; if i=j,rij is similar degree between sample xi themselves and its value is permanently 1. Because of rij= rji, fuzzy similar matrix R˜ is symmetric and reflexive matrix.

Using Zadeh operator "◦", we do square self-synthesis operation to fuzzy similar matrix R˜. After that applied λ-cut operation at different values, to fuzzy classification of software understandability.

Object-Oriented Spatial Complexity Measures

Chhabra et al.[7][6] explained about the effect of spatial complexity in code understandability. Software comprehension accounts for over one third of the lifetime cost of a software system and the process of software comprehension is directly related to complexity of software. The computation of software complexity has been carried out by the researchers using various affecting attributes like control flow paths, the volume of operands and operators, identifier density, cognitive complexity and spatial complexity.

Spatial complexity metrics indicate the difficulty of understanding the logic of the program in terms of lines of code that the reader is required to traverse to follow control or data dependencies as they build a mental model. The spatial complexity metrics have been proposed for procedure-oriented software as well as object-oriented software. Spatial complexity of object-oriented software is the combination of class spatial complexity as well as object spatial complexity. The class spatial complexity measures the spatial complexity of methods and attributes. In this thesis work, we have concentrated on the object-oriented spatial complexity measures.

Object-oriented spatial complexity measures Two types of object-oriented spatial complexity measures have been proposed class spatial complexity measures, and object spatial complexity measures. The class spatial complexity measures are defined in terms of class attribute spatial complexity and class method spatial complexity measures.

(i) Class Spatial complexity measures

The class spatial complexity measures are defined in terms of the effort needed to understand the behavior of a class. Generally class consists of two entities: attributes and methods. Thus, class spatial complexity measures are defined in terms of the class attribute spatial complexity measures and class method spatial complexity measures.

Class attribute spatial complexity The class attribute spatial complexity of an attribute is measured using the distance between its definition and first use within the method and subsequently taking into account the distance between successive uses within the same method. Spatial complexity of an attribute is the average of distances of various uses of that attribute from its definition/previous use.

$$\mathrm{CSC} = \sum_{i=1}^{p} \mathrm{Distance}_i / p$$

where 'p' is the number of times of the usage of attribute.

Distance is measured in Lines Of Code (LOC) in between the successive use of that variable or definition to the use of that variable. In case of multiple files coming into picture for measurement of this distance, the distance is defined as

Distance = (distance of first use of the attribute from the top of the current file) + (distance of definition of the attribute from the top of the file containing definition)

Total Class Attribute Spatial Complexity of a class(TCASC) is defined as average of class attribute spatial complexity of all attributes of that class.

$$\mathrm{TCASC} = \sum_{i=1}^{q} \mathrm{CASC}_i / q$$

where 'q' is the count of attributes in the class.

Class method spatial complexity The Class Method Spatial Complexity (CMSC) of a method is defined as distance (in LOC) between the declaration and the definition of the method. In case of multiple files, distance is defined as

Distance = (distance of definition from the top of the file containing definition) + (distance of declaration of the method from the top of the file containing declaration)

Total Class Method Spatial Complexity (TCMSC) of a class is defined as average of class method spatial complexity of all methods of the class.

$$\mathrm{TCMSC} = \sum_{i=1}^{m} \mathrm{CMSC}_i / m$$

where 'm' is the count of methods in the class.

The class Spatial Complexity (CSC) of a class is defined as

$$\mathrm{CSC} = \mathrm{TCASC} + \mathrm{TCMSC}$$

(ii) Object Spatial complexity measures

The object spatial complexity is of two types - object definition spatial complexity and object member usage spatial complexity.

Object definition spatial complexity The Object Definition Spatial Complexity (ODSC) of an object is defined as the distance of the definition of the object from the corresponding class declaration and if

the object is defined in a different file than the file containing class declaration, then distance is calculated as:

Distance= (distance of object definition from top of current file) + (distance of declaration of the corresponding class from the top of the file containing class)

Object member usage spatial complexity The Object member usage spatial complexity (OMUSC) of a member through a particular object is defined as the average of distances (in LOC) between definition of the member in the corresponding class and call of that member through that object i.e.

$$\text{OMUSC} = \sum_{i=1}^{n} \text{Distance}_i / n$$

Where 'n' represents count of calls/use of that member through that object, and Distancei is equal to the absolute difference in number of lines between the method definition and the corresponding call/use through that object. In case of multiple files,

Distance = (distance of call from the top of the file containing call) + (distance of definition of the member from the top of the file containing definition)

Total Object-Member Usage Spatial Complexity (TOMUSC) of an object is defined as average of object-member usage spatial complexity of all members being used of that method.

$$\text{TOMUSC} = \sum_{i=1}^{k} \text{OMUSC}_i / k$$

where 'k' is the count of object-members being called through that object. The Object Spatial Complexity of an object is defined as

OSC = ODSC + TOMUSC

3-RESULT

In this they calculated class spatial complexity, object spatial complexity. Class spatial complexity includes the distance between definition &use of attributes and methods. They explained this spatial complexity with the help of Reverse engineering time. We had a notion, LOC may affect the code understandability most, but they try to prove that class spatial complexity may affect more in object-oriented programming languages.

4-CONCLUSION

At the end they said that due to the high class spatial complexity, the reverse engineering time is more to generate class diagram from source code. In addition to this they explained the effect of object spatial complexity. In this they consider perfective maintenance time for a few projects and explained the effect of that spatial complexity.

5-REFERENCES

[1] Understandability metrics. Website. http: www. Aivosto .com/project/help.
[2] Using uml part two- behavioral modelling diagrams. Website. http://www.sparxsystems.com.
[3] Krishan K. Aggarwal, Yogesh Singh, and Jitender Kumar Chhabra. An integrated measure of software maintainability. pages 235-240, GGS Indraprastha University, Delhi and Regional Engineering College, Kurukshetra, 2002. 2002 PROCEEDINGS
[4] Annual RELIABILITY and MAINTAINABILITY Symposium.
[5] Richard H. Carver, Steve Counsell, and Reuben V. Nithi. An evaluation of the mood set of object-oriented software metrics. IEEE Trans. Software Eng., 24(6):491-496, 1998.
[6] Jitender Kumar Chhabra, K. K. Aggarwal, and Yogesh Singh. Code and data spatial complexity: two important software understandability measures. Information & Software Technology, 45(8):539-546, 2003.
[7] Jitender Kumar Chhabra, K. K. Aggarwal, and Yogesh Singh. Measurement of object-oriented software spatial complexity. Information & Software Technology, 46(10):689-699, 2004.
[8] Jitender Kumar Chhabra and Varun Gupta. Evaluation of object-oriented spatial complexity measures. ACM SIGSOFT Software Engineering Notes, 34(3):1-5, 2009.
[9] Nicolas E. Gold, Andrew Mohan, and Paul J. Layzell. Spatial complexity metrics: An investigation of utility. IEEE Trans. Software Eng., 31(3):203-212, 2005.
[10] Maurice H. Halstead. Elements of Software Science. ISBN:0-444-00205-7. Amsterdam, 1977.

[11] Seyyed Mohsen Jamali. Object oriented metrics. Department of Computer Engineering Sharif University of Technology, 2006.
[12] Feng Jiang, Yuefei Sui, and Cungen Cao. A rough set approach to outlier detection. volume 37, pages 519-536. International Journal of General Systems, october 2008.
[13] K.Shima, Y.Takemura, and K.Matsumoto. An approach to experimental evaluation of software understandability. Proceedings of the 2002 International Symposium on Empirical Software Engineering(ISESE'02), 2002.
[14] Xiangjun LI and Fen RAO. An rough entropy based approach to outlier detection. Journal of Computational Information Systems, pages 10501-10508, 2012. Department of Computer science and Technology, Nanchang University,Nanchang 330031, China and College of Economy and Management, Nanchang University, Nanchang 330031, China.
[15] Jin-Cherng Lin and Kuo-Chiang Wu. A model for measuring software understandability. In CIT, page 192, 2006.
[16] Jin-Cherng Lin and Kuo-Chiang Wu. Evaluation of software understandability based on fuzzy matrix. In FUZZ-IEEE, pages 887-892, 2008.