International Journal of Enhanced Research in Science, Technology & Engineering
ISSN: 2319-7463, Vol. 6 Issue 10, October-2017, Impact Factor: 4.059

An Understandability Estimation Tool for UML Class Diagram (UETCD)

Atica M. Altaie

Software Engineering Department, Computer and Mathematics Science College, University of Mosul, Mosul, Iraq

ABSTRACT

A Class diagram is a kind of static structures for UML diagrams that describes the structure of a target system. However, no effective measures of a class diagram's understandability yet exist. The paper presents a linear regression for structural properties of class diagrams based on set of metrics defined for UML class diagrams, may be used as indicators to evaluate class diagrams and develop the understandability Estimation Tool for UML Class Diagrams (UETCD). JAVA programming language is used to implement the tool and document object model (DOM) for manipulating XMI document and providing the structural representation of the document. The obtained results reveal that the better understanding of class diagrams when they have low abstraction, coupling, polymorphism, complexity, and design size values with a high cohesion and encapsulation values to make the system more understandable. Furthermore, it may influence the effort needed to review, maintain or reuse UML class diagrams.

Keywords- Software metrics; software understandability; UML class diagram; object-Oriented design;

1. INTRODUCTION

Software quality is an important characteristic in information system (IS) success. The management of software quality is still immature branch in the research of software engineering. The quality model describes what is meant by quality and represents it in a structured way [6].

Boehm[7] defines software quality to include portability, reliability, efficiency, human engineering, understandability, modifiability, and testability [5]. Understandability is defined as the ability of the product to enable the user to understand how it should be used [8]. Software understanding aims to provide enough information about the system that should be written by software engineers and programmers during software development life-cycle processes [9]. But there is no distinction between different types of understandability, nor is there an indication of the specific characteristics and processes that contribute to understandability [8].

Understanding is important and useful for maintenance and reuse activities.

- Software understandability is an important quality factor that allows the software engineers to easily understand the structure of the system, which simplifies the task of software maintenance [8]. The software engineer can't maintain a system that doesn't understand - at least not easily and completely and make changes to the system if not understand how the system as a whole will work once the changes are made [25].
- The component to be reused later by different software engineers should be well documented and understandable. Imagine that software engineer have to understand completely the components made by other, change and modify them to make the system better. So, software engineer should have little effort spend for reading and understanding of components [10].

An understandable system enhances its own maintainability and reusability; this is so because most maintenance work and reuse activities require that software engineers first try to understand the system's components completely before making any subsequent system modifications or extensions [9].

The Unified Modeling Language (UML) is a family of graphical modeling language, backed by a single meta-model that help in describing, visualizing, constructing, and documenting the artifacts of software systems, particularly object oriented (OO) systems [11]. A class diagram is a type of static structure UML diagram that describes the structure of a target system by showing the objects and classes inside the system and the relationships between them [2].

This paper is organized in 6 Sections. The related work is described in Section 2. The understandability design properties, metrics and tool architecture are described in Section 3. The case studies and results are described in Section 4. Finally, some conclusion of the understandability estimation model and tool are described in Section 5 and Section 6.

2. RELATED WORK

M. Genero et al (2008) [3], present the definition and validation of ER diagram structural complexity metrics that can be used as early, easily and objectively calculated indicators of ER diagram understandability. They show that the understandability of an ER diagram is affected by its structural complexity where the contributors to this structural complexity are the diagram's attributes and relationships.

Nugroho (2009) [23] focused on the understandability of models in the development phase and evaluate Level of Detail (LoD) in UML models and experimentally investigate the effect of correctness and efficiency in comprehending UML models. The results show a better understanding of models when they have a high level of detail [24].

Nazir el at. (2010) produced a model for Understandability Quantification based on metrics. They proved that the readability and understandability of the software has a lot of influence on the factors that directly or indirectly affect software quality [25].

S.W.A et al. (2010)[22], propose a Maintainability Estimation Model for Object-Oriented Software in Design Phase (MEMOOD), which provide an opportunity to improve understandability of class diagram. They build a model for understandability based on Number of Classes and Number of Generalization Hierarchies to estimate maintainability in terms of understandability and modifiability.

3. RESEARCH METHODOLOGY

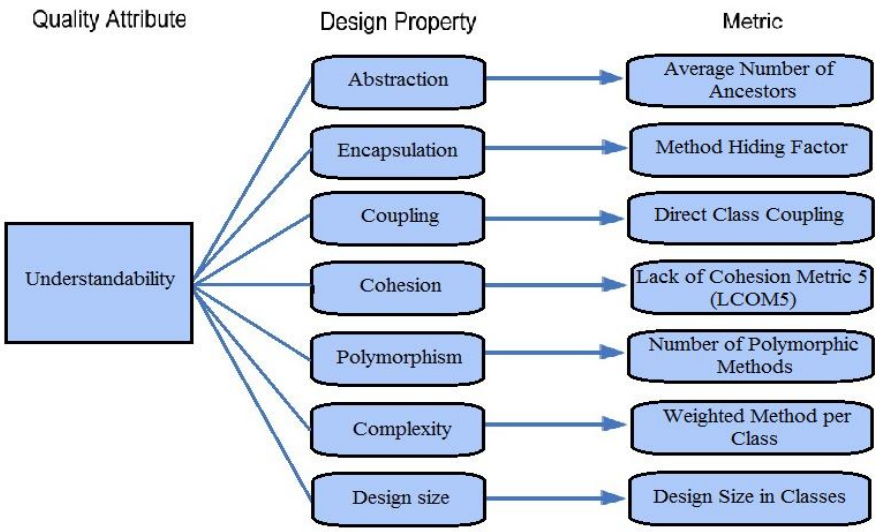
The Understandability Estimation Model

At present, software engineering is an important technology and software developers have tried constantly to develop new technologies as computer programs have grown. The new developed technologies focused on object oriented presentation technologies.

The main factor for any branch of engineering is measurement. Without measurement or metrics it is impossible to measure quality characteristic before it is released. Therefore measurement characteristic is very important in the software project management [12].

The paper selects a quality factor understandability based on models like QMOOD. QMOOD (Quality Model for Object Oriented Design) is the complete model that assesses quality attributes like reusability, functionality, understandability, flexibility [16].

Understandability formula= (-0.33*abstraction) + (0.33*encapsulation)-(0.33*coupling) + (0.33*cohesion) - (0.33*polymorphism) - (0.33*complexity) - (0.33*design size) [1][16].



Figuren1: Understandability design properties and metrics

Quality Attribute and Metric used in the model are customized as follow:

1) Abstraction: It is a measure of the generalization-specialization aspect of the design. Classes in a design which have one or more offspring carry this characteristic of abstraction [1].

- The Average Number of Ancestors (ANA) metric is calculated by determining the number of classes along all paths from root to all classes in the structure of inheritance [18].

2) Encapsulation: It is defined as attach data and behavior within a single structure. In the design of object-oriented property refers specifically to design classes that prevent access to the attribute declarations by defining them to be private, thus preserving the internal representation of objects [2].

- Method Hiding Factor (MHF) is defined as the ratio of the sum of all private methods defined in all classes to the total number of methods defined in the system [20] [17].

3) Coupling: It is defined as the interaction degree with the object has other objects, measured by the number of links it has with these objects [15].

- Direct Class Coupling (DCC) measures the number of other classes that interact with a specific class [15].

4) Polymorphism: It is defined as the ability to replace an object with its sub objects. The ability of an object-variable to contain, not only object, but also all of its sub objects [2].

- Number of Polymorphic methods (NOP): This metric is a count of the methods that can carry polymorphic behavior characteristic.

5) Complexity: It is defined as the degree of difficulty to predict the characteristics of the system, due to the characteristics of the systems parts [14].

- Weighted Method per Class (WMC) is defined as the sum of the complexities of all methods of a class [19]. The high value of WMC refers to the class is more complicated than that of the low values [4].

6) Cohesion: It is defined as a measure of the elements of a module belong together *[17]*.

- Lack of Cohesion Metric 5 (LCOM5) that measures a lack of cohesion across multiple methods by looking at the number of methods that reference of each attribute. They define LCOM5= (a-kl)/(l-kl), where l is the number of attributes, k is the number of methods, and a is the summation of the number of attributes accessed by each method in a class[13].

7) Design size: It is a measure of the number classes used in a design [1].

- Design Size in classes (DSC): This metric is a count of the total number of classes in the design.

Finally, we are proposed to divide understandability value by design size to obtain understandability ratio. The little value means more understandability of class diagram.

Tool Architecture

Figure 2 presents the overall architecture of the UETCD tool for evaluate understandability of class diagram.

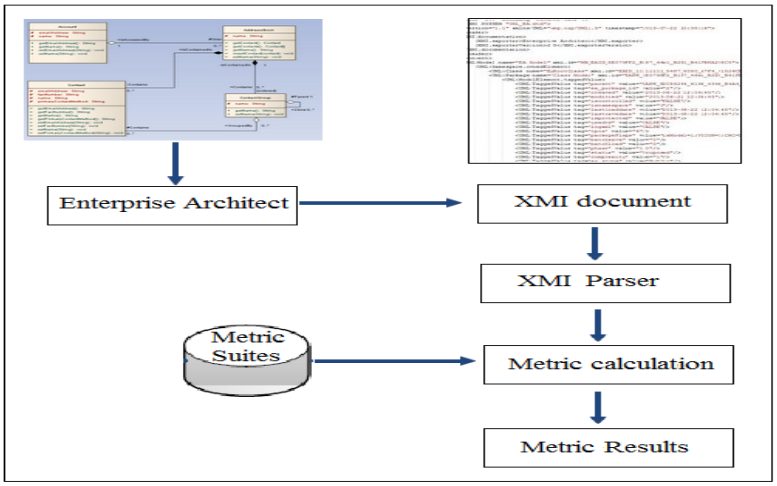


Figure 2: UETCD Tool Architecture

Scenarios are first modeled as UML class diagram is a design of software system using a CASE tool Enterprise Architect (EA) [21]. The UML class diagram is then exported from the Enterprise Architect in the vendor-independent XMI document (XML Metadata Interchange is an open standard file format that enables the interchange of model information between models and tools [11]) which the information in this system is stored in XMI files.

XMI parser that developed extracts all the required information for our analysis such as classes, attributes, operations, associations, dependencies, generalizations, and stereotypes of class diagrams for UETCD tool.

Metric Calculation for calculate a understandability estimation of class diagram by using the object- oriented design metric suites and computation formula for quality attribute of understandability.

Display Metrics Results for display data as a set of seven design properties of the class diagrams.

JAVA programming language is used to implement the tool and Document Object Model (DOM) for manipulating XMI document and providing a tree structural representation of the document tags.

Calculate understandability Attribute of the class diagrams in design phase by using a computation formula for Quality Attributes is useful for review, reuse, and maintain UML class diagrams.

4. CASE STUDY

To test the implemented tool, object oriented class diagram is taken and calculated the value of understandability based on quality attribute and metrics used.

Figure 3 and Figure 4 show OO design and Table 1 shows the values of metrics obtained from OO design depicted in figure 3 and 4.

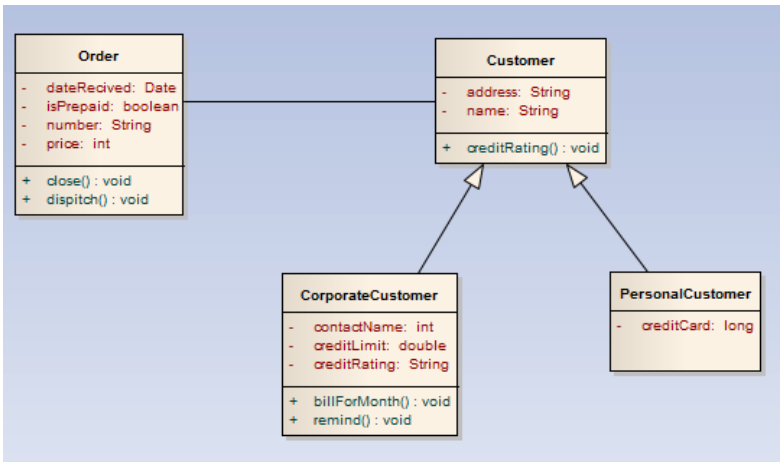


Figure 3: class diagram for customer ordering system (COS)

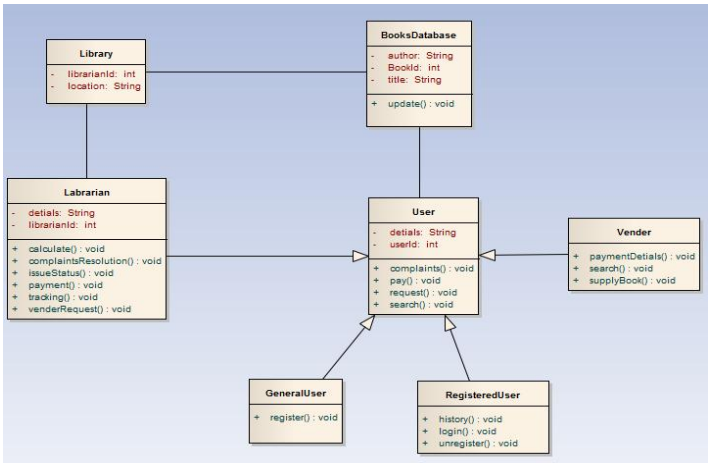


Figure 4: Class diagram for library management system (LMS)

Table 1: Depicts the Understandability Formula

Property name	Metric	Value of COS	Value of LMS
Abstraction	**Average Number of Ancestors**	**1**	**1**
Encapsulation	**Method Hiding Factor**	**1**	**1**
Coupling	**Direct Class Coupling**	**1**	**1**
Cohesion	**Lack of Cohesion Metric 5**	**1.25**	**1.06**
Polymorphism	**Number of Polymorphic methods**	**0**	**0**
Complexity	**Weighted Method per Class**	**5**	**18**
Design size	**Design Size in classes**	**4**	**7**
Understandability formula		**2.88**	**8.32**
Understandability ratio		**0.72**	**1.18**

Table 1, contains the understandability values calculated using developed model. Result shows that understandability value for LMS is highly than COS computed through model. This mean COS is more understandable than LMS. The understandability value indicates that the design model needed a low abstraction, coupling, polymorphism, complexity, and design size values with a high cohesion and encapsulation values to make the design model more understandable. Furthermore, it is an indicator to software engineers to may change the way that information is presented.

CONCLUSION

The paper has developed model to quantify understandability of the class diagrams in design phase. The model has been developed and implemented using the technique of multiple linear regressions and java programming language. This tool for help a software designer based on metrics to:

- When abstraction, coupling, polymorphism, complexity, and design size values decrease, understandability value increases and when cohesion and encapsulation values increase, understandability value increases.
- Review the design and take appropriate corrective measures, early in the development life cycle and exhibit good design characteristics.
- Decrease the effort needed to maintain or reuse UML class diagrams by decrease ambiguity.

FUTURE WORK

The work may be enhanced in the future by the following aspects:

- More sophisticated model may be developed in future, by conducting a larger scale study with variety domains.
- The tool may be used to assess other quality factors such as reliability, testability and maintainability by integrates formulas to calculate them.

REFERENCES

[1]. M. Thirugnanam, J.N. Swathi, "Quality Metrics Tool for Object Oriented Programming", International Journal of Computer Theory and Engineering, Vol. 2, No. 5, October, 2010.
[2]. S. K. Dhiman, A. Sharma, A. Kaur ," Comprehensive Study of Object-Oriented Analysis and Design by using the Concept of OOSE ", International Journal of Research in Education Methodology, Vol.1, No.1 June 2012.
[3]. M. Genero, G. Poels, M. Piattini, "Defining and validating metrics for assessing the understandability of entity–relationship diagrams", Elsevier journal , Data & Knowledge Engineering 64, 2008, P.p 534–557.
[4]. Neelamegam , Dr. M. Punithavalli , " A Survey - Object Oriented Quality Metrics ", Global Journal of Computer Science and Technology, P.p 183-186.
[5]. N. Gorla, S. Lin, "Determinants of software quality: A survey of information systems project managers", Information and Software Technology, Vol. 52, Issue: 6, 2010, Pp: 602-610.
[6]. S. Wagner," A Bayesian Network Approach to Assess and Predict
[7]. Software Quality Using Activity-Based Quality Models ", Information and Software Technology, 2010, Vol. 52, Issue: 11, Pp: 1230-1241.
[8]. Boehm, "Practical Strategies for Developing Large Software Systems", Addison Wesley Longman, 1975.
[9]. Alonso-Ríos, A. Vázquez-García, E. Mosqueira-Rey, and V. Moret-Bonillo, " Usability: A Critical Analysis and a Taxonomy", Intl. Journal of human–computer interaction, 2010, Vol. 26, Issue 1, Pp. 53-74.
[10]. C. N. Sant'Anna, A. F. Garcia, C. F. G. Chavez, C. J. P Lucena, A. Staa, " On the Reuse and Maintenance of Aspect-Oriented Software: An Assessment Framework", Proceedings XVII Brazilian Symposium on Software Engineering, Vol.26, 2003.
[11]. Kumar, "Software Reuse Libraries Based Proposed Classification for Efficient Retrieval of Components", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 6, 2013, Pp. 884-890.
[12]. Seth, H. Agarwal, A. R. Singla, "Unified Modeling Language for Describing Business Value Chain Activities", International Conference on Advances in Computer Applications (ICACA) 2012, Proceedings published by International Journal of Computer Applications (IJCA), Pp.17-22.
[13]. V. Krishnapriya, Dr. K. Ramar ," Comparison of Class Inheritance and Interface Usage in Object Oriented Programming through Complexity Measures ", International Journal of Computer Science & Information Technology (IJCSIT), Vol 2, No 6, December 2010,Pp. 28-36.
[14]. J. Al Dallal, "Improving the Applicability of Object-Oriented Class Cohesion Metrics", Information and Software Technology, Volume 53, Issue 9, September 2011, Pp. 914–928.
[15]. T. Fässberg, Å. Fasth, F. Hellman, A. Davidsson, J. Stahre, "Interaction between complexity, quality and cognitive automation", Proc. 4th CIRP Conference on Assembly Technologies and Systems (CATS 2012), 2012.
[16]. Bandar Alshammari, Colin Fidge and Diane Corney, " Security Metrics for Object-Oriented Designs ", Proceedings of the 21st Australian Software Engineering Conference (ASWEC 2010), 6- 9 April, 2010, Hyatt Regency, A uckland.Pp. 55 – 64.
[17]. S. Chawla, " Review of MOOD and QMOOD metric sets ", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 3, March 2013, Pp.448-451.
[18]. N. A. Nemade, D. D. Patil, N. V. Ingale, " Study and Evaluation for Quality Improvement of Object Oriented System at Various Layers of Object Oriented Matrices", International Journal of Engineering Science and Innovative Technology (IJESIT) ,Vol. 2, Issue 3, May 2013,Pp.376-380.
[19]. M. Genero, M. Piattini, C. Calero, " A Survey of Metrics for UML Class Diagrams ", Journal of object technology, Vol. 4, No. 9, November-December 2005, pp. 59-92.
[20]. Chhikara, R.S.Chhillar," Analyzing the Complexity of Java Programs using Object - Oriented Software Metrics", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 3, January 2012,Pp. 364-372.
[21]. S. M. Jamali, "Object Oriented Metrics," Department of Computer Engineering , Sharif University of Technology , January 2006. http://www.sparxsystems.com .
[22]. S. W. A. Rizvi and R. A. Khan, "Maintainability Estimation Model for Object- Oriented Software in Design Phase (MEMOOD)", Journal of computing, Vol 2, Issue 4, April 2010, Pp.26-32.
[23]. Nugroho,, "Level of detail in UML models and its impact on model comprehension: A controlled experiment". Information and Software Technology, Vol . 51, Issue 12, 2009, Pp. 1670-1685.
[24]. M. Fernández-Sáez, M. Genero, and M. R.V. Chaudron, " Does the Level of Detail of UML Models Affect the Maintainability of Source Code?", Proceedings of the 2011th international conference on Models in Software Engineering, Pp. 134-148.
[25]. M. Nazir, R. A. Khan and K. Mustafa, " A Metrics Based Model for Understandability Quantification", JOURNAL OF COMPUTING, VOLUME 2, ISSUE 4, APRIL 2010, Pp.90-94.