TELKOMNIKA Telecommunication, Computing, Electronics and Control
Vol. 19, No. 5, October 2021, pp. 1507~1516
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v19i5.19278

Measure extendibility/extensibility quality attribute using object oriented design metric

Taghreed Riyadh Alreffaee, Marwah M. A. Dabdawb, Dujan B. Taha
Department of Software, College of Computer science and Mathematics, University of Mosul, Mosul, Iraq

Article Info

Article history:

Received Dec 31, 2020
Revised Mar 16, 2021
Accepted Mar 30, 2021

Keywords:

EA
Extensibility
Metrics
OO
QMOOD
Quality assurance
UML
XMI
XML

ABSTRACT

Software design is one of the very important phases of the software engineering. The costs of software can be minimized if improvements or corrections made during this stage. Several of the current computer aided software engineering (CASE) tools like enterprise architect (EA) v12 do not have the capability to improve the design. This work aims to develop an algorithm that helps the software engineers evaluating the design quality utilizing one of the object-oriented (OO) design models namely quality metrics for object-oriented design (QMOOD) which represents as hierarchical model that describes the relationship between quality attributes such as reusability, extendibility and properties of the design of OO design. This algorithm describesed how the assessment of the extendibility/ extensibility using the software metrics has been done and the impact of the involved metrics in the extendibility value. Results obtained demonstrate the effect of OO design metrics such as inheritance, polymorphism, abstraction and coupling in quality characteristics like extensibility. The results show that lower values of abstraction and coupling, obtain higher value of extendibility which means the class diagram is ready to accept additional improvements. The proposed algorithm has been tested on two different systems (test cases) that vary in their class diagrams, functionalities, and complexities.

Corresponding Author:

Taghreed Riyadh Alreffaee
Department of Software
University of Mosul
Mosul, Iraq
Email: taghreed_reyad@uomosul.edu.iq

1. INTRODUCTION

Quality management system deals with organizational structure, procedures, responsibility, compatibilities, activities and resources that together seek to ensure that software product will meet its intendant purpose [1]. Often, quality engineering and management is misunderstood. This is due to the fact that the term 'quality' is ambiguous. This confusion can be a result of several reasons like:

- Quality is a wide concept rather than a single idea.
- There can be several levels of interpretations for quality because any concept carries levels of abstraction.
- Quality is used in daily language so there is a difference between popular and professional uses.

After all, a general definition for quality can be set as "the degree to which a system, component, or process meets a customer or user's needs or expectations" [2]. For saving effort and avoid extra work, software quality should be predicted in the early stage of software development process and taking into

consideration this predicted quality must be as good as desired. So, whenever the prediction of software quality is done earlier, we got rid of working on the same software again and again [1].

The quality of software extremely depends on its design. In software building, the design stage is considered to be one of the most important stages and should be focused because when design is prepared, modifications on it will be difficult and expensive [1]. Recently, object-oriented design became common concept in software development environment. It is used to divide the problem into multi small objects which is more capable for modifications and much easier to understand in addition of benefits in decomposition, reliability, adaptability and reusability of a problem [3].

To assess software quality through the early phases of software development process, software metrics have been used. These metrics are measurements that can be used to explore the design quality. It is time to examine object-oriented metrics with regard to software quality because as mentioned before, object-oriented design methodologies acquired wide publicity. Many metrics have been proposed in the past for the object-oriented (OO) design, code and constructs which can help software companies in development their software within intendant quality [1].

Many research papers have been introduced in the object-oriented design metrics field. The difference in the gained outcomes is the result of utilizing a various methodology in each of these articles. To cover most of the previous work presented in this field, a glance is made back to 2013, when Hoffmann *et al.* [4] showed that it is possible to improve a robotics application in a continuous manner at run-time by integrating the object-oriented software framework specifically "the Java based robotics API", into "the dynamic module system OSGi".

Vir *et al.* [1] proposed in 2014 a hybrid approach for investigating the extendibility of classes in object-oriented design in the early stage of software development. This hybridization composed of subset of Chidamber and Kemerer (CK) metric suite and mood metric suite, then the proposed model has been analyzed using fuzzy logic approach. In the same year, Winn [5] pursued to measure the extensibility and changeability of aspect-oriented software (AO) where implemented in AspectJ. The analysis covered in this topic is based on mobile media and AspectJ projects by using self-organizing map (SOM). Also, in Yadav [6] described an improved analysis view of software attribute dependency relations for the assessment of high-level design quality attributes in object-oriented designs. It characterizes relations and dependencies of quality attributes such as reusability, flexibility, understandability, functionality, extendibility, effectiveness. The relations, or links, from properties of the design to attributes of the quality are weighted in accordance with their effect and significance.

Other researchers proposed some new metrics instead of traditional software metrics. In this context, Cossentino *et al.* [7] estimated modularity and extensibility of "holonic multi agent systems (HMAS)" which is used for the development of software systems that are frequently designed in order to realize complex dynamical behavior for solving complicated problems. Pölzlbauer *et al.* [8] addressed message ID assignment, in a manner that the system can be extensible. In the beginning, they gave an assessment metric that provides a deep insight view of the extensibility of a given ID assignment. After that, they developed an effective ID-assignment policy which in turn will maximize the extensibility.

Couto *et al.* [9] suggested semiautomatic software restructuring method based on the attributes of the quality. This paper depends on the measurements of the quality metrics for object-oriented design (QMOOD) for recommending move method refactorings that improve the quality of the software. Another study was performed by Kim *et al.* [10] in 2018, proposed the extensibility metric for software architecture (EMSA), which adopts the architecture of the system in order to determine its extensibility.

Abdullah *et al.* [11] in 2019 recommended an investigation structure for extensibility estimation process and did a comprehensive survey on object-oriented extensibility. The researchers recommended a model by creating the relationship in the middle of design properties. Researchers justified the model attachment with the help of statistical measures, which showed that coupling, cohesion, inheritance, polymorphism are significantly affected with extensibility.

This work aims to design and implement an algorithm to measure the extendibility of a software system during its architectural design stage using QMOOD model metrics. The extendibility attribute depends on 4 metrics, each of these metrics has its mathematical equation based on the class diagram of the software to measure and calculate them. These four metrics constructs the mathematical equation which is considered the basic form to determine if the software extendible or not. This model has been applied for two software as will be explained later. Tested software must be follows the object-oriented principles, written in jave or any other object-oriented language.

The remainder of this paper is organized as follows: section 2 identifies object-oriented design metrics. Section 3 demonstrates extensibility quality attributes model. Section 4 exhibits the proposed methodology. Experimental tests and results are found is section 5. The paper is concluded in section 6.

2. OBJECT ORIENTED DESIGN METRICS

Although the term "metric" is used mutually with term "measurement" and indicates wide range of it, there is a delicate difference between them. In software engineering situation, a measure supply a quantitative indication of the amount, dimensions, capacity or size of some attribute of a process or product [12]. So, we can say that measurement is the action of locating a measure, while metric had been defined by The Institute for Electrical and Electronics Engineers Academic & Science (IEEE) as" a quantitative measure of the degree to which a system, component, or process possesses a given attribute" [12].

Metrics provide the software developer with a deep insight that is necessary to access the quality and consider critical source of information to make the right decision about designing the wanted software before building it and make changes. This will reduce complexity and improve the quality of the product in early phases. Some metrics may be converted to serve their purpose in a new environment [13].

In summary, it can be said that prediction of software processes depends on software metrics. Object oriented (OO) metrics are generally used for quality estimation. Relying on OO concepts, those metrics are regarding to measurement of some design features like encapsulation, message passing, information hiding and inheritance [14]. The researchers have proposed many metrics for OO software. These metrics are used in multi potential contexts like quality indicators, complexity measure and reliability measure [15]. Object oriented metrics are based on the objects and their characteristics. A lot of object-oriented metrics existed for the object-oriented software development process. These metrics are [3]:

- Chen.
- Lorenz and Kidd.
- Morris.
- Metrics for object-oriented software engineering (MOOSE).
- Extended metrics for object-oriented software engineering (EMOOSE).
- MOOD.
- QMOOD.
- Goal Question.
- Software assurance technology center (SATC) for object oriented.
- LI.

In this proposed work, only QMOOD metrics set are considered and for other metrics could be referred from other paper [16].

2.1. QMOOD model metric

The QMOOD is a global quality model that founds an obviously defined and empirically validated model to appreciate object oriented design quality attributes and relates it with structural object-oriented design properties through mathematical formulas [17]. This model extends from Dromey's quality framework which is based on 3 principles: product properties that effect quality, a group of quality attributes and a means of linking them [18]. QMOOD quality model defines the way to measure the quality attributes in terms of properties of the design through a group of mathematical formulas [19]. Design properties can be measured or observed using functionality, relationship and structure of design components [20]. QMOOD model consists of six OOD quality attributes and they are: (reusability, flexibility, understandability, functionality, extendibility, and effectiveness). These attributes had a relationship with eleven design properties; each calculated quality attribute serves as a parameter to provide a notion of the current quality of the software [17]. Table 1 show the design properties and the metrics utilized for these properties of the QMOOD model [17].

Table 1. QMOOD model

Design Property	Metrics used for Design Property	Design Property	Metrics used for Design Property
Design Size	design size in classes (DSC)	Cohesion	Cohesion Among Methods of Classes (CAM)
Hierarchies	Number of Hierarchies (NOH)	Composition	Measure of Aggregation (MOA)
Abstraction	Average Number of Ancestors (ANA)	Inheritance	Measure of Functional Abstraction (MFA)
Encapsulation	Data Access Metric (DAM)	Polymorphism	Number of Polymorphic Methods (NOP)
Coupling	Direct Class Coupling (DCC)	Messaging	Class Interface Size (CIS)
		Complexity	Number of Methods (NOM)

3. EXTENSIBILITY QUALITY ATTRIBUTE

We initially investigate the meaning of "software extensibility". In ISO/IEC-25010 standard, characterizes the extensibility of the software as "the relative exertion to expand the ability or performance of the software by enhancing current functions or by including new functions or data". IEEE-1219 standard does not characterize extensibility, however, characterizes a comparable quality attribute \"maintainability"\

as the exertion for adjusting the product of the software in accordance with the modification in the requirements or environment [10]. Ciraci and Broek [21] characterized another comparable quality attribute\ evolvability" which is the ability to adjust to the new environment to maintain a service. Usually, these attributes are referring to modifies in the current system of the software that are made to satisfy a new environment or a new requirement [10].

Saeed *et al.* [22], the authors characterized extensibility which is the existence and utilize of features in the existing design that permits integration with the new requirements. When designing the system of the software and has not taken into account the extensibility and evolution elements, usually, may well become prey to the phenomenon known as "code rot" or "design rot" and may as a result of this phenomenon be abandoned as the system becomes too difficult to expand and maintain. Changing a system for new features, repairing old features, cleaning up obsolete features or even a customer supply with a particular product is all expensive work, and it may become even more expensive when each change may lead new problems in the form of errors and violations in the design of the system [23].

In ISO 9126, attributes: "functionality", "reliability", "efficiency", "usability", "maintainability", and "portability"- were chosen in QMOOD model as the first set of quality attributes. The "portability" term in the context of software quality implementation is more appropriate and was exchanged with "extendibility" which reflects this characteristic better in design [24]. To measure extendibility attribute, four QMOOD metrics must be computed first, as the following [25]:

– ANA: is computed by specifying the ratio between the classes number on all paths from the root class (number of ancestors) to all classes in the structure (number of total classes), which means ANA metric measured as the following equation:

$$ANA = \frac{Sum(Number\ of\ Ancestors)}{Total\ Number\ of\ Classes} \quad (1)$$

The property of abstraction (ANA) is for classes that have one or more successors [24].

– DCC: It is a measure of the other objects number which an object must access normally. This means determine a different number of message passing in a method or attribute of each class that a class is directly related to, as in (2).

$$DCC = \frac{Sum(Numberof\ Attribute\ and\ Parameter)}{Total\ Number\ of\ Classes} \quad (2)$$

– MFA: It is the ratio of the numbers of methods inherited by a class to the all methods accessible to member methods of the class [24]. In (3) is depicted as follows:

$$MFA = \frac{Sum\left(\frac{NumMethodInh}{AllMethods}\right) of\ all\ Classes}{Total\ Number\ of\ Classes} \quad (3)$$

where; NumMethodInh: refers to the number of methods inherited by a class,
AllMethods: refers to all methods accessible (inherited + class's methods) to the class member.

– NOP: This metric is measured by determining the number of the methods which have polymorphic behavior. This is done by (4) [3].

$$NOP = \frac{Sum(Number\ of\ PolyMorphicMethod)}{Total\ Number\ of\ Classes} \quad (4)$$

According to Bansiya and Davis [24], an extendibility attribute is a combination of four property of the design as mentioned above, this attribute measured according to the formula given by [24] and shown in the following equation:

$$Extendibility = 0.5 \times ANA - 0.5 \times DCC + 0.5 \times MFA + 0.5 \times NOP \quad (5)$$

where; ANA : Abstraction metric.
DCC : Coupling metric.
MFA : Inheritance metric.
NOP : Polymorphism metric

4. THE PROPOSED ALGORITHM FOR EXTENSIBILITY QUALITY ATTRIBUTE

The overall work of the proposed algorithm can be depicted as shown in Figure 1. It consists of the following steps:

Step1: Drawing test cases as unified modeling language (UML) class diagram which is a design of software system using enterprise architect (EA) computer aided software engineering (CASE) tool.
Step2: Exporting UML class diagrams from the enterprise architect. This will generate XMI (XML (extensible markup language) metadata interchange) document where the information is stored as XMI files.
Step3: Passing XMI document to the XMI parser where the required information will be extracted from XMI document such as operations, classes and attributes of a class diagrams.
Step4: Computing the design metrics as it was clarified in section 3 previously and according to (1)-(4).
Step5: Calculating extendibility quality attributes according to the formula (5) in section 3.
In the following sections each of the above steps will be described in more details.

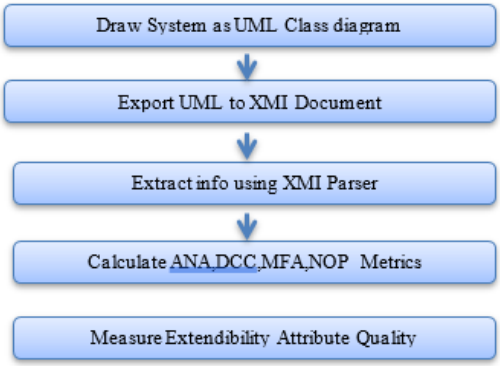

Figure 1. Extendibility attribute flowchart

4.1. UML class diagram and EA

In this work, UML class diagrams (for the test cases, automated teller machine (ATM) machine and online exam systems) have been drawn using one of the famous CASE tools, enterprise architect (EA) which supports a comprehensive modeling of UML and uses for building and designing software systems. It covers all sides of the software development life cycle with full traceability. Usually, this tool does not support design metrics of the software systems.

Each system has been drawn (as will be mentioned later in section 5) depending on its function; the first test case is ATM system. The main function of this device is to enables customers to achieve financial transactions, like deposits, cash withdrawals, and funds transfers, at any time without the need for direct interaction with the staff of the bank. The second test case is the online exam system where examinations are given online to evaluate the student effectively. The fundamental goal of this system is to reduces the required time and obtain accurate results in a fast way. QMOOD model metrics and extendibility quality attribute have been computed for each of these systems, each one gives different result according to its design. This will be mentioned later in Table 2.

4.2. XMI

After the class diagrams were drawn for each system at the design stage and clarified the basic functions for each of them using classes and the relationship between them, each diagram has been exported to (XMI document). It is an open standard format of the file which enables interchange the model information between tools and models. XMI is a way of saving the diagrams of UML (class diagram in this work) in details such as the name, attributes, operations of each class, and relationships.

XMI document has a large set of tags as depicted in Figure 2, some are important but others are not such as the class style, and date of creation. The aim of exporting class diagrams to XMI file is to use these files as input to the parser which will extract all necessary information used to calculate metrics such as no. of classes and no. of operation of each class.

4.3. XMI parser

XMI parser is utilized to extract data from XMI document; XMI parser will store all values of XMI document tags in lists which contain all information about the class diagram. JAVA Eclipse programming language is used for implementation and document object model (DOM) for manipulating XMI document. Document object model (DOM) is an application programming interface which manipulating XMI document and providing a tree structural representation of the document tags. XML documents have an informational

unit hierarchy called nodes. DOM describing those nodes and the relationships between them. Figure 3, depicts the use of DOM parser to manipulate XML document of the class diagrams as will be mentioned later using java eclipse.

```
<?xml version="1.0" encoding="windows-1252"?>
<!DOCTYPE XMI SYSTEM "UML_EA.dtd">
<XMI xmi.version="1.1" xmlns:UML="omg.org/UML1.3" timestamp="2020-04-05 11:02:08">
    <XMI.header>
        <XMI.documentation>
            <XMI.exporter>Enterprise Architect</XMI.exporter>
            <XMI.exporterVersion>2.5</XMI.exporterVersion>
        </XMI.documentation>
    </XMI.header>
    <XMI.content>
        <UML:Model name="EA Model" xmi.id="MX_EAID_C92C7C77_8AB1_41e3_857D_A7A536A14BBA">
            <UML:Namespace.ownedElement>
                <UML:Class name="EARootClass" xmi.id="EAID_11111111_5487_4080_A7F4_41526CB0AA00" isRoot="true" isLeaf=
                <UML:Package name="Class Model" xmi.id="EAPK_C92C7C77_8AB1_41e3_857D_A7A536A14BBA" isRoot="false" isl
                    <UML:ModelElement.taggedValue>
                        <UML:TaggedValue tag="parent" value="EAPK_21B70DA0_FBAD_47be_83AA_C162E3952FD1"/>
                        <UML:TaggedValue tag="ea_package_id" value="2"/>
                        <UML:TaggedValue tag="created" value="2019-11-14 11:11:12"/>
                        <UML:TaggedValue tag="modified" value="2019-11-14 11:11:12"/>
                        <UML:TaggedValue tag="iscontrolled" value="FALSE"/>
                        <UML:TaggedValue tag="isnamespace" value="1"/>
                        <UML:TaggedValue tag="lastloaddate" value="2019-11-14 11:11:12"/>
                        <UML:TaggedValue tag="lastsavedate" value="2019-11-14 11:11:12"/>
                        <UML:TaggedValue tag="isprotected" value="FALSE"/>
                        <UML:TaggedValue tag="usedtd" value="FALSE"/>
                        <UML:TaggedValue tag="logxml" value="FALSE"/>
                        <UML:TaggedValue tag="tpos" value="6"/>
                        <UML:TaggedValue tag="packageFlags" value="isModel=1;VICON=3;CRC=0;"/>
                        <UML:TaggedValue tag="batchsave" value="0"/>
                        <UML:TaggedValue tag="batchload" value="0"/>
                        <UML:TaggedValue tag="phase" value="1.0"/>
                        <UML:TaggedValue tag="status" value="Proposed"/>
                        <UML:TaggedValue tag="complexity" value="1"/>
                        <UML:TaggedValue tag="ea_stype" value="Public"/>
                        <UML:TaggedValue tag="tpos" value="6"/>
                    </UML:ModelElement.taggedValue>
                    <UML:Namespace.ownedElement>
```

Figure 2. XMI format

```
package XML_DOM_Parser;
import java.io.IOException;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.soap.Node;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;
public class MyDomclass {
    public static void main(String[] args)
    {
    DocumentBuilderFactory factory= DocumentBuilderFactory.newInstance();
    try {
        DocumentBuilder builder= factory.newDocumentBuilder();
        Document doc=builder.parse("XMLFile4.xml");
        NodeList operationlist= doc.getElementsByTagName("Operation");
    for(int i=0;i<operationlist.getLength();i++){
        org.w3c.dom.Node p =operationlist.item(i);
        if (p.getNodeType()==Node.ELEMENT_NODE){
```

Figure 3. Manipulate XML using DOM parser

4.4. Compute (ANA, DCC, MFA, NOP) metrics

In this step, these four metrics must be computed. To do so, metrics explained in section 3 (1)-(4) have been implemented. These metrics considered as a prerequisite to compute the extendibility quality attributes.

4.5. Compute extendibility quality attributes

The final step is to compute the extendibility quality attribute. After computing the four metrics (ANA, DCC, MFA and NOP). In (5) mentioned in section 3 has been implemented to find the final value.

5. EXPERIMENTAL TESTS AND RESULTS

For the purpose of executing the algorithm, test cases must be used and analyzed. Two test cases were used on two different class diagrams. Then, clarified the QMOOD extendibility assessment results.

5.1. First example

The first example represents the implementation of ATM system. Each metrics value needed will calculated from the class diagram of the system. Figure 4 shows the class diagram of ATM system.

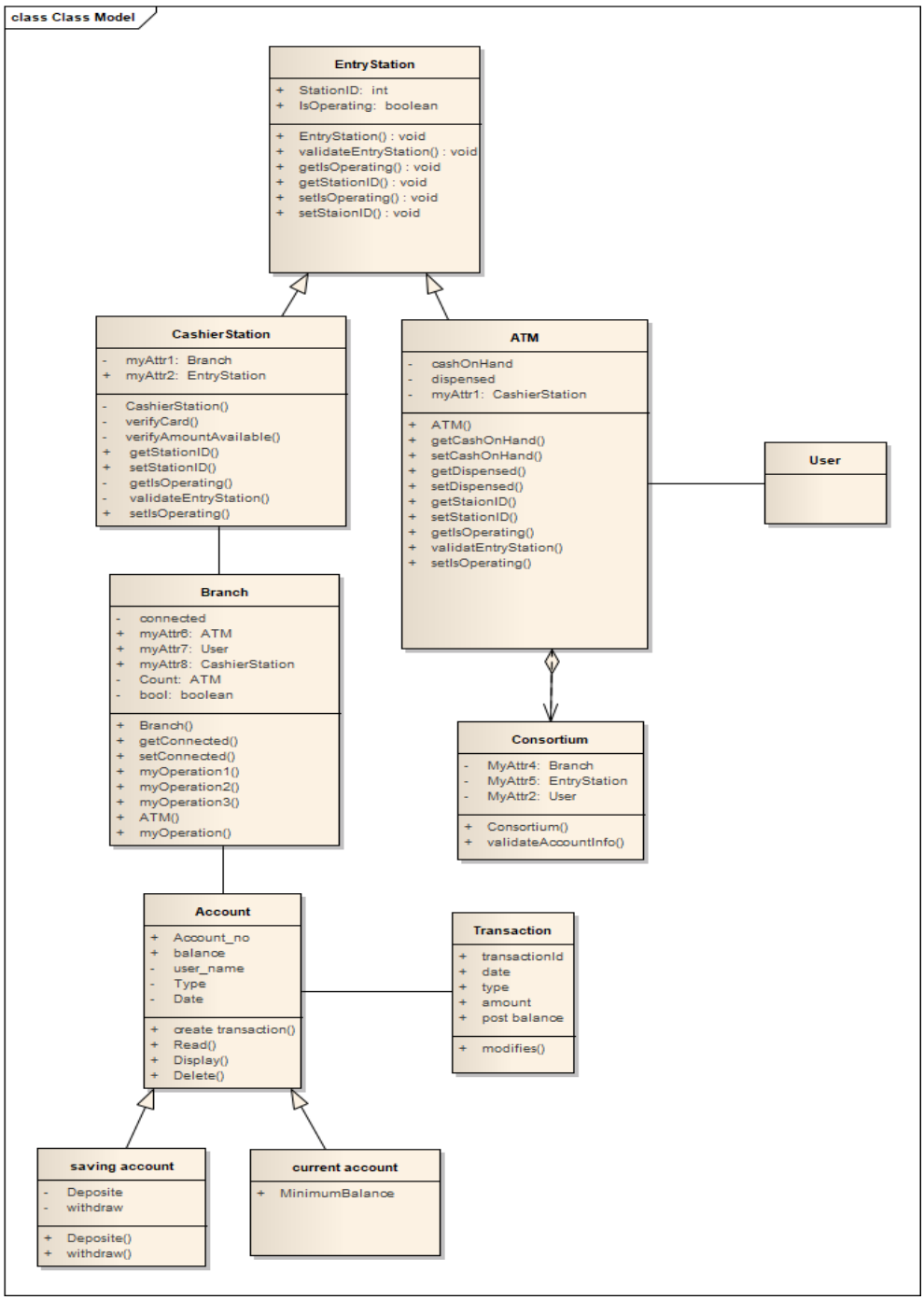

Figure 4. Simple class diagram for ATM machine

5.2. Second example

The second case study represents the implementation of online exam system. The class diagram is shown in Figure 5. Figure 4 and Figure 5 show object-oriented design for ATM Machine and online exam system. Table 2 shows the values of metrics.

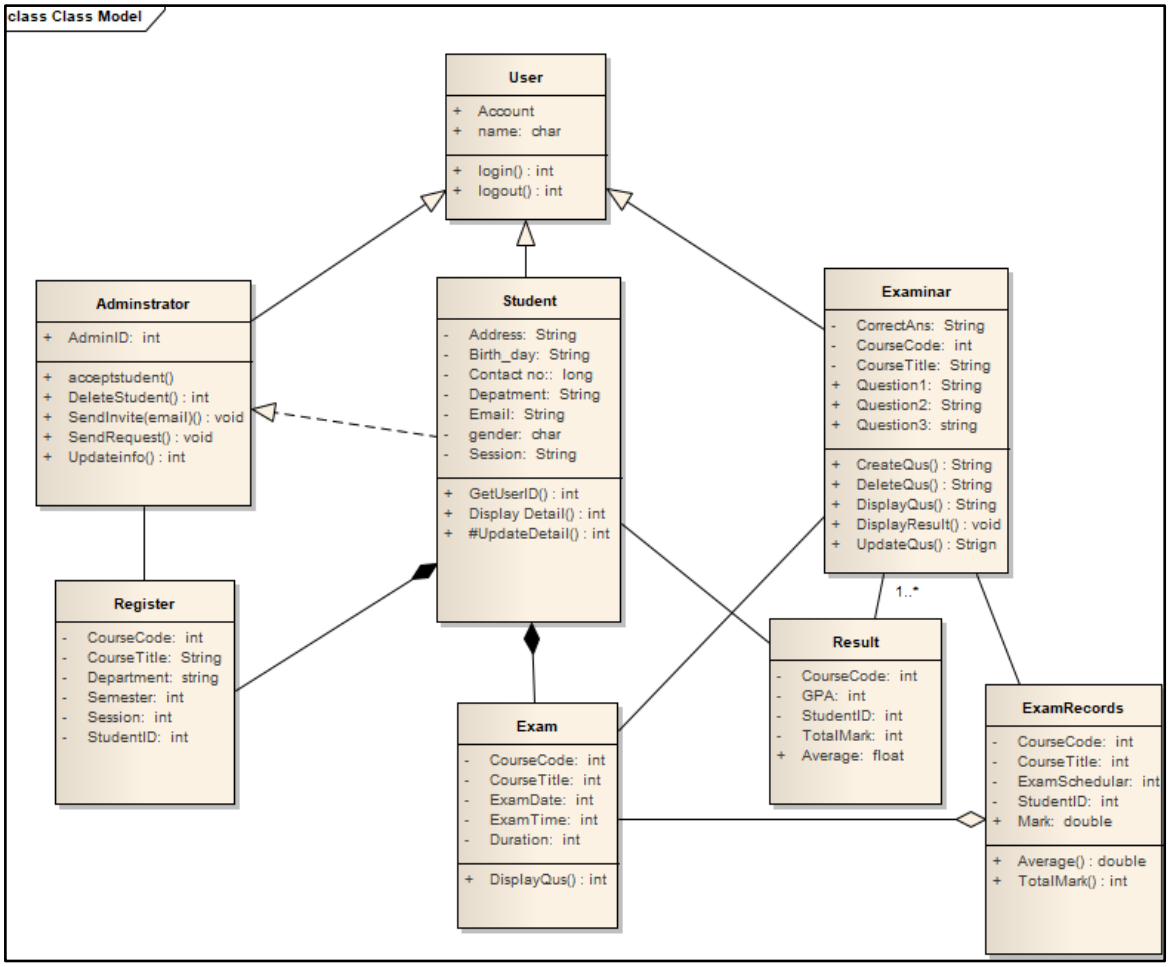

Figure 5. Simple class diagram for online exam system

Table 2. Extendibility metrics and its value for ATM and online exam systems

Property	Metric	ATM Machine	Online Exam
Abstraction	(ANA)	0.4	0.375
Coupling	(DCC)	1	0
Inheritance	(MFA)	0.24	0.12
Polymorphism	(NOP)	0	0
Extendibility= 0.5×ANA-0.5×DCC+0.5×MFA+0.5×NOP		-0.18	0.24

Table 2 shows the calculated values of the extendibility/extensibility quality attribute for the two systems based on the values of the four metrics in property column. After export each diagram (online exam and ATM machine) to XMI document, the parser extracts the necessary information from these documents for measuring the metrics later as explained in Figure 1. After these steps, mathematical (1)-(4) were applied to measure the four metrics of the QMOOD model (ANA, DCC, MFA, NOP) depending on the information extracted from the class diagram using DOM parser. Finally, to know if these systems are extendible/extensible or not, (5) was applied. It can be noted from Table 2, the value of the metrics (abstraction, and coupling) is inversely proportional to the extendibility quality attribute value. In the other words, If the values of abstraction and coupling are minimized, then class diagram is ready to accept additional improvements. So, when abstraction and coupling in online exam system are less than in ATM

machine system, then the value of extendibility for the online system was better than ATM and it is ready to accept additional refinement.

6. CONCLUSION

The important contribution of this work is to propose an algorithm for measuring the extendibility of the class diagrams in the design phase and the impact of the involved metrics in the extendibility value. We have conducted experimental study on two test cases to see the influence of the (ANA, DCC, NOP, and MFA) values on the extendibility attribute. The proposed algorithm consists of five steps, each one has explained in details. The experimental results demonstrated that if the value of abstraction and coupling metrics decreased, the extendibility value is increases and vice versa. Hence when the equation of extendibility was applied on the two test cases (ATM machine and online exam systems), the value of extendibility for the online exam system was greater than the extendibility value for the ATM machine system, which means that the system of the online exam is more extensible than the other system and it is ready to accept any additional improvements. This gives a judgment to software engineers about the ability to expand the software in early phases of software development. The future work may include applying the algorithm to a variety of object-oriented design quality attributes like reusability, flexibility, understandability, functionality and effectiveness, and providing a comprehensive experimental study of the effect of each metrics values on these attributes.

ACNOWLEDGMENT

We are grateful to the University of Mosul for all the support and for making this work achieved. We also would like to thank the department of Software for providing us with all the possible support in performing this research.

REFERENCES

[1] R. Vir, P. Dhillon, and J. Dhillon " Fuzzy Logic Approach to Forecast the Extendibility/Extensibility in Object Oriented Design using an Integrated Model," *International Journal of Computer Applications*, vol. 94, no. 9, May 2014, doi: 10.5120/16371-5811.

[2] L. Alzubaidy and K. A. Ibrahim, "Constructing an Add-in Tool for Enterprise Architect v7.5 To Measure the Quality of Object Oriented Design (Class Diagram)," *International Journal of Computer Science and Information Security*, vol. 13, no. 7, pp. 72-85, July 2018.

[3] P. K. Goyal and G. Joshi, "QMOOD metric sets to assess quality of Java program," *International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), IEEE*, 2014, doi: 10.1109/icicict.2014.6781337.

[4] A. Hoffmann, A. Angerer, A. Schierl, M. Vistein, and W. Reif, " Managing Extensibility and Maintainability of Industrial Robotics Software," *16th International Conference on Advanced Robotics (ICAR) © 2013 IEEE*, doi: 10.1109/icar.2013.6766561.

[5] K. Z. N. Winn, "Quantifying and Validation of Changeability and Extensibility for Aspect-Oriented Software," *InProc. Int. Conf. Advances in Engineering and Technology*, Mar. 2014, doi: 10.15242/iie.e0314075.

[6] V. Yadav, "Software Quality Analyser," *Matrix Academic International Online Journal of Engineering and Technology*, vol. 2, no. 2, pp. 21-28, Oct. 2014.

[7] M. Cossentino, C. Lodato, S. Lopes, P Ribino, and V. Palermo, "Metrics for Evaluating Modularity and Extensibility in HMAS Systems," *In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems,* May 2015.

[8] F. Pölzlbauer, R. I. Davis, and I. Bate, "A practical message ID assignment policy for Controller Area Network that maximizes extensibility," In *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, pp. 45-54. 2016, doi: 10.1145/2997465.2997484.

[9] C. M. S. Couto, H. Rocha, and R. Terra, "Quality-oriented Move Method Refactoring," *BENEVOL 2017-16th BElgian-NEtherlands software eVOLution symposium*, 2017.

[10] J. Kim, S. Kang, J. Ahn, and S. Lee, "EMSA: Extensibility Metric for Software Architecture", *International Journal of Software Engineering and Knowledge Engineering,* vol. 28, no. 3, pp. 371–405, 2018, doi: 10.1142/s0218194018500134.

[11] D. Abdullah, H. Mahfuzul, and Y. Hagos, "A Methodology to Evaluate Extensibility of Object Oriented Design: A Product Transition Perspective," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 8, no. 10, Oct. 2019, doi: 10.17148/ijarcce.2019.81001.

[12] N. S. A. A. Bakar, "The Analysis of Object-Oriented Metrics in C++ Programs," *Lecture Notes on Software Engineering*, vol. 4, no. 1, Feb. 2016, doi: 10.7763/lnse.2016.v4.222.

[13] S. K. Punia, "A Review of Software Quality Metrics for Object-Oriented Design," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 6, no. 8, Aug. 2016 .
[14] B. Kochar, S. S. Gaur, and D. K. Bhardwaj, "Identification, Analysis & Empirical Validation (IAV) of Object Oriented Design (OO) Metrics as Quality Indicators," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 5, no. 8, pp. 31-40, 2017.
[15] A. Singh, R. Bhatia, and A. Singhrova, "Taxonomy of machine learning algorithms in software fault prediction using object oriented metrics," *International Conference on Computational Intelligence and Data Science*, vol. 132, pp. 993-1001, 2018, doi: 10.1016/j.procs.2018.05.115.
[16] A. Sharma and S. K. Dubey, "Comparison of Software Quality Metrics for Object-Oriented System," *International Journal of Computer Science & Management Studies*, Special Issue of vol. 12, pp. 12-24, June 2012.
[17] Z. C. Ani, S. Basri, and A. Sarlan, "A Reusability Assessment of UCP-Based Effort Estimation Framework using Object-Oriented Approach," *Journal of Telecommunication, Electronic and Computer Engineering*, vol. 9, no. 3-5, 2017.
[18] C. Singh *et al*. "Toward Software Measurement and Quality Analysis of MARF and GIPSY Case Studies, a Team 8 SOEN6611-S14 Project Report," *arXiv preprint arXiv:1407.1328*, 2014.
[19] S. Jain, P. Shantanu, and S. Raghuraj, "Predictive Object Points (POP) Sizing Metric: A Good Predictor of Quality of OO Software," *Indian Journal of Science and Technology,* vol. 11, no. 20, pp. 1-8, 2018, doi: 10.17485/ijst/2018/v11i20/48215.
[20] A. Shaheen, U. Qamar, A. Nazir, R. Bibi, M. Ansar, and I. Zafar, "Oocqm: object-oriented code quality meter," *International Conference on Computational Science/Intelligence & Applied Informatics*, Jan. 2020, pp. 149-163, doi: 10.1007/978-3-030-25225-0_11.
[21] S. Ciraci and P. v. d. Broek, "Evolvability as a quality attribute of software architectures," *The International ERCIM Workshop on Software Evolution*, 2006.
[22] M. G. Saeed, M. T. Alasaady, F. L. Malallah, and K. H. Faraj, "Three Levels Quality Analysis Tool for Object Oriented Programming," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 11, 2018, doi: 10.14569/ijacsa.2018.091173.
[23] N. Johansson, A. Löfgren, and C. Olsson, "Designing for extensibility: An action research study of maximising extensibility by means of design principles," Bachelor of Applied Information Technology Thesis, University of Gothenburg, 2009.
[24] J. Bansiya and C. G. Davis, "A hierarchical model for object-oriented design quality assessment," *IEEE Transactions on software engineering*, vol 28, no.1, Jan. 2002. doi:10.1109/32.979986.
[25] D. Rizk, "Software Quality Attribute Measurement and Analysis Based on Class Diagram Metrics," Theses and Dissertations The American University in Cairo, 2009.