International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 3, pp 29-35, March-2023

https://doi.org/10.47001/IRJIET/2023.703005

Estimating Software Size Using Metrics from Analysis and Design Phases (ADM) of the Software Development Life Cycle (SDLC)

[1]AESHAA L. AL-SALEEM, [2]ASMA'A Y. HAMMO

[1,2]Department of Software Engineering, College of Computer Science and Mathematics, University of Mosul, Iraq

Abstract - **Due to the lack of available information in the early stages of project creation, early estimation of the project size is a difficult task. The information appears more detailed and clear as we progress in the stages of building the software. The aim of the research is to design and implement a model to estimate the size of the program in the two phases of analysis and design separately, and in the phases of analysis and design together using four measures of the class diagram, which are "number of classes", "number of features", "number of operations" and "number of relationships".**

Keywords: Software size, Source Line of Code (SLOC), Software Development Life Cycle (SDLC), analysis, design, Unified Modeling Language (UML), Class diagram, diagram.

I. INTRODUCTION

One of the basic aspects of software management successful is the effort estimation [1-3]. Calculating software effort is certainly a complex process and time consuming because it depends on several factors including software size [4]. Many methods have been proposed over the past years to estimate the software size. Source line of code (SLOC) represents the real size of a program excluding comments and white space lines. Estimating the volume of software in the advanced stages of the project life cycle is important for project managers for several reasons: including knowing the time needed to complete the system, because the difference in the scheduled date of delivery of the system to the customer will cause the customer to lose confidence in the company producing the system. Also, it helps in determining the appropriate budget for the project, as well as allocating resources efficiently [5]. Therefore, estimating code size early became an important research topic for many years [4]. There are several methods for estimating software size. One of the most used methods from the beginning to solve the problem of estimation is expert judgment, which depends on the expert's experience, but it is not subject to criteria [27]. As an alternative method for SLOC, the scientists proposed a function point that depends entirely on the number of basic function as that the system performs and thus represents the functional requirements of the user [37, 38]. Many researchers proposed different UML models for estimating software size (e.g. Sequence Diagram, Entity Relationship Diagram, Use Case Diagram, Activity Diagram and Class Diagram) [6-25]. Model-based estimation is divided into two main types: parametric such as COCOMO and its extensions [28], non-parametric model, where the estimation model is applied using machine learning such as a neural network, and thus we will get a more accurate model [29, 30]. In the agile software development process, software scaling method is called story point [31, 32]. The aim of the research is to design and implement a tool to estimate the size of the program in the analysis and design phases separately, and in the analysis and design phases together using metrics from Class diagram. The research structure is as follows. Section 2 discussed research related work. Section 3 provides enough information about class diagram in UML and software sizing. Section 4 presents the research methodology. Section 5 gives conclusion and future works.

II. RELATED WORK

Some researchers turned to study software estimation in many as follows:

Harizi 2012 [19] built a new method for estimating the size of the program using class diagram scales and giving them their own weights. Although it is an innovative method, the criteria used to allocate weights have not been experimentally validated.

Lazic et al.2012 [20] they studied four methods for estimating the size of the program, and as a result of this study, they derived a model for estimating the size through the multiple linear regression model. They proposed a method to calculate (LOC) for the system and validated the method by applying it to a number of samples taken from open source systems and industry.

Nur Atiqah Sia Abdullah et al 2013 [41] used Unified Modeling Language)UML(models such as Use Case diagram, sequence diagrams component diagram, object diagram and behavioral model elements to represent the workflow requirements and functional requirement for estimation based on a COSMIC function point .

Zhou et al 2014 [11] presented an investigation of the accuracy of Source Line Of Code(SLOC) based estimation models in early Software Development Life Cycle(SDLC) using six measures of class diagram. They used 100 Java systems and applied a number of techniques to them, including tree based model, instance based model, nonlinear model, and linear model. The results were that using an estimation model that create using UML diagrams size metric achieved the highest accuracy.

Ayyildiz and Koçyigit 2015 [23] Analyzed the congruence between metrics such as the number of nouns specified and verbs specified in the requirements document and solution metrics such as the total number of classes in the program and the number of methods. They used 14 software projects. They also in 2018 [22] estimated the voltage and compared it to the real effort using COSMIC function point. They noticed that there is a robust correlation between effort and size. and effort estimation model is more accurate than CFP.

Kiewkanya and Surak 2016 [24] Used multiple linear regression analysis, which is a statistical technique and class diagram, a method was proposed to estimate the size of C++ software. Through the implementation of the proposed model, an automated software tool was built to measure the size of the program based on the structural complexity of the class diagram.

Badri et al 2016 [25] conducted a comparative study between two methods for predicting the size of the software. They chose the use case metrics and the objective class point. They used simple linear regression methods to build the estimation models. The data were for four Java projects. The result of this study was that the Use Case metrics are more reliable estimating SLOC.

Daud, M. and Malik, A.A. 2021 [9] presented a special method for estimating the size of the program by comparing the category diagram at the design stage and the category diagram at the analysis stage. They derived four famous measures from the class plot, which are the number of classes, the number of methods, the number of attributes, and the number of relationships between classes. Moreover, compared two previous models of volume estimation before and after applying model on them.

III. CLASS DIAGRAM IN UML AND SOFTWARE SIZING

The class diagram is one of the most popular and widely used (UML) diagrams and gives a comprehensive view of the overall software structure by showing the system's classes, their features, operations (or methods), and the relationships amongst objects. The classes in a class diagram represent both the basic components interactions in the system, and the classes for which code is to be written. In the diagram, the class is represented in the form of a box divided horizontally into three parts. The first part contains the name of this class, written in bold, and takes a central location, and the first letter of the name is capitalized. The second part contains the features of class, and the end part contains the operations that the class performs, and it is similar in Adjectives with the second section, as the script is normal and aligned to the left, and the first letter is made of lowercase letters. There are several relationships between the classes, including Association, Aggregation, and Composition [44]. Commonly the systems analyst starts by doing the Analysis Class Diagram (ACD) in the analysis stage to comprehension the problem domain. In the next stage (design), the systems analyst converts the ACD into a Design Class Diagram (DCD) which include more accurate and specific information about the system. As it is known that estimating the size of the software using inputs from an early stage of the project life cycle is considered better, but waiting until more accurate and detailed information is obtained, we get a more accurate estimate [45]. In order for the estimate to be useful for project managers, its results must be closer to reality and more accurate. There are some differences between ACD and DCD firstly, when going from ACD to DCD, the level of abstraction decreases. Secondly, DCD contains more detailed and accurate information about the system than ACD.

IV. THE METHODOLOGY

The task of deriving the measures and determining their values is easy to do. In this research, the additional information from the design stage to the analysis stage and apply a new type of Metrics it's called the metrics from the design stage to the analysis stage (ADM). ADM extracts four metrics from the class diagram, which are number of classes (NOC), number of attributes (NOA), number of methods (NOM) and number of relationships (NOR). Figures (1) describe action steps.

First step: the class diagram of the analysis and design stages for the projects should be obtaining

For each project Class diagram should be drawn using enterprise architect (EA), one of the known tools, which

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

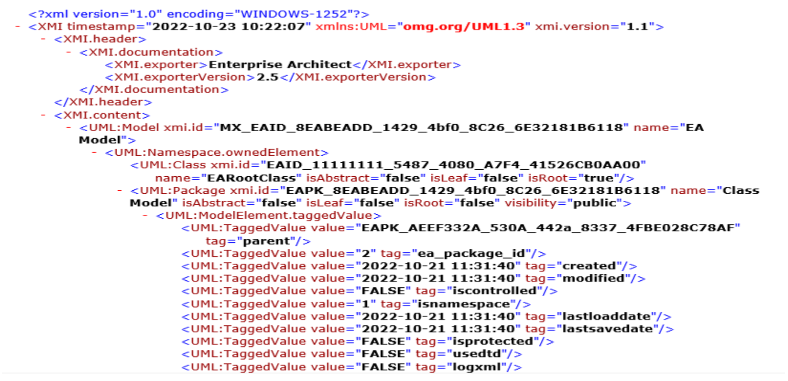
Volume 7, Issue 3, pp 29-35, March-2023

https://doi.org/10.47001/IRJIET/2023.703005

supports an overall modeling of UML and it is uses for creating and designing software systems.

Second step: class diagram has been exported to (XML document)

Each class diagram for the analysis and design stages exported to (XML document) by the directive "export package to XML" and we choose the version (UML 1.3). Figure (2) gives an example of the (XML document).



```
  <?xml version="1.0" encoding="WINDOWS-1252"?>
- <XMI timestamp="2022-10-23 10:22:07" xmlns:UML="omg.org/UML1.3" xmi.version="1.1">
  - <XMI.header>
    - <XMI.documentation>
        <XMI.exporter>Enterprise Architect</XMI.exporter>
        <XMI.exporterVersion>2.5</XMI.exporterVersion>
      </XMI.documentation>
    </XMI.header>
  - <XMI.content>
    - <UML:Model xmi.id="MX_EAID_8EABEADD_1429_4bf0_8C26_6E32181B6118" name="EA
      Model">
      - <UML:Namespace.ownedElement>
          <UML:Class xmi.id="EAID_11111111_5487_4080_A7F4_41526CB0AA00"
            name="EARootClass" isAbstract="false" isLeaf="false" isRoot="true"/>
        - <UML:Package xmi.id="EAPK_8EABEADD_1429_4bf0_8C26_6E32181B6118" name="Class
          Model" isAbstract="false" isLeaf="false" isRoot="false" visibility="public">
          - <UML:ModelElement.taggedValue>
              <UML:TaggedValue value="EAPK_AEEF332A_530A_442a_8337_4FBE028C78AF"
                tag="parent"/>
              <UML:TaggedValue value="2" tag="ea_package_id"/>
              <UML:TaggedValue value="2022-10-21 11:31:40" tag="created"/>
              <UML:TaggedValue value="2022-10-21 11:31:40" tag="modified"/>
              <UML:TaggedValue value="FALSE" tag="iscontrolled"/>
              <UML:TaggedValue value="1" tag="isnamespace"/>
              <UML:TaggedValue value="2022-10-21 11:31:40" tag="lastloaddate"/>
              <UML:TaggedValue value="2022-10-21 11:31:40" tag="lastsavedate"/>
              <UML:TaggedValue value="FALSE" tag="isprotected"/>
              <UML:TaggedValue value="FALSE" tag="usedtd"/>
              <UML:TaggedValue value="FALSE" tag="logxml"/>
```

Third step: parsing the XML document

An XML parser was used to extract required information from an XML document; The XML parser will keeps all the tag values of the XML documents in lists containing most of class information. The JAVA Net Beans implementation programming language and the Document Object Model (DOM) are used to process the XML document. The (DOM) is an interface for programming application that deals with XML document and provides for the document's tag tree structural representation. XML documents contain a hierarchy of informational units called nodes. These nodes represent the 'tags'. The DOM describes those nodes and the relationships between them. Actually, we need only some metrics as shown below.

Fourth step: Calculate input metrics values from analysis class diagram (ACD)

Calculate input metrics from parsing the XML document of ACD. The needed ACD metrics are (AM1=NOC, AM2=NOA, AM3=NOM, AM4=NOR).

Fifth step: Calculate input metrics values from design class diagram (DCD)

In the same way, calculate input metrics from DCD. The used DCD metrics are (DM1=NOC, DM2=NOA, DM3=NOM, DM4=NOR).

Sixth step: Calculate (ADM) analysis and design metrics

Calculate (ADM) analysis and design metrics for each input metrics as in equation (1) [9].

$$\boldsymbol{ADMx = \frac{DMxi}{AMxi}} \qquad (1)$$

Seventh step: adjusting ADM value

The adjusting is done by multiplying ADM by the DCD metrics as shown in equation (2).

$$\boldsymbol{ADCDx = ADMx * DCDxi} \qquad (2)$$

Eighth step: Existing size estimation model

Finding the size of the program using the (LOC) method, based on the two model first that was found by (Lazic et al 2012), in which 8 programs written in C++ language taken from industry and 17 graduation projects for students written in Java were used as in equation (3). The second model is obtained from (bianco & Lavazza 2006) in which 12 student programs written in java and 5 open source projects were used as in equation (4).

$$\mathbf{LOC^{L}=241.41+10.2*NOA+9.547*NOM-24.84*NOC} \qquad \mathbf{(3)}$$

$$\mathbf{LOC^{B}=5.7*NOM+3.3*NOA} \qquad \mathbf{(4)}$$

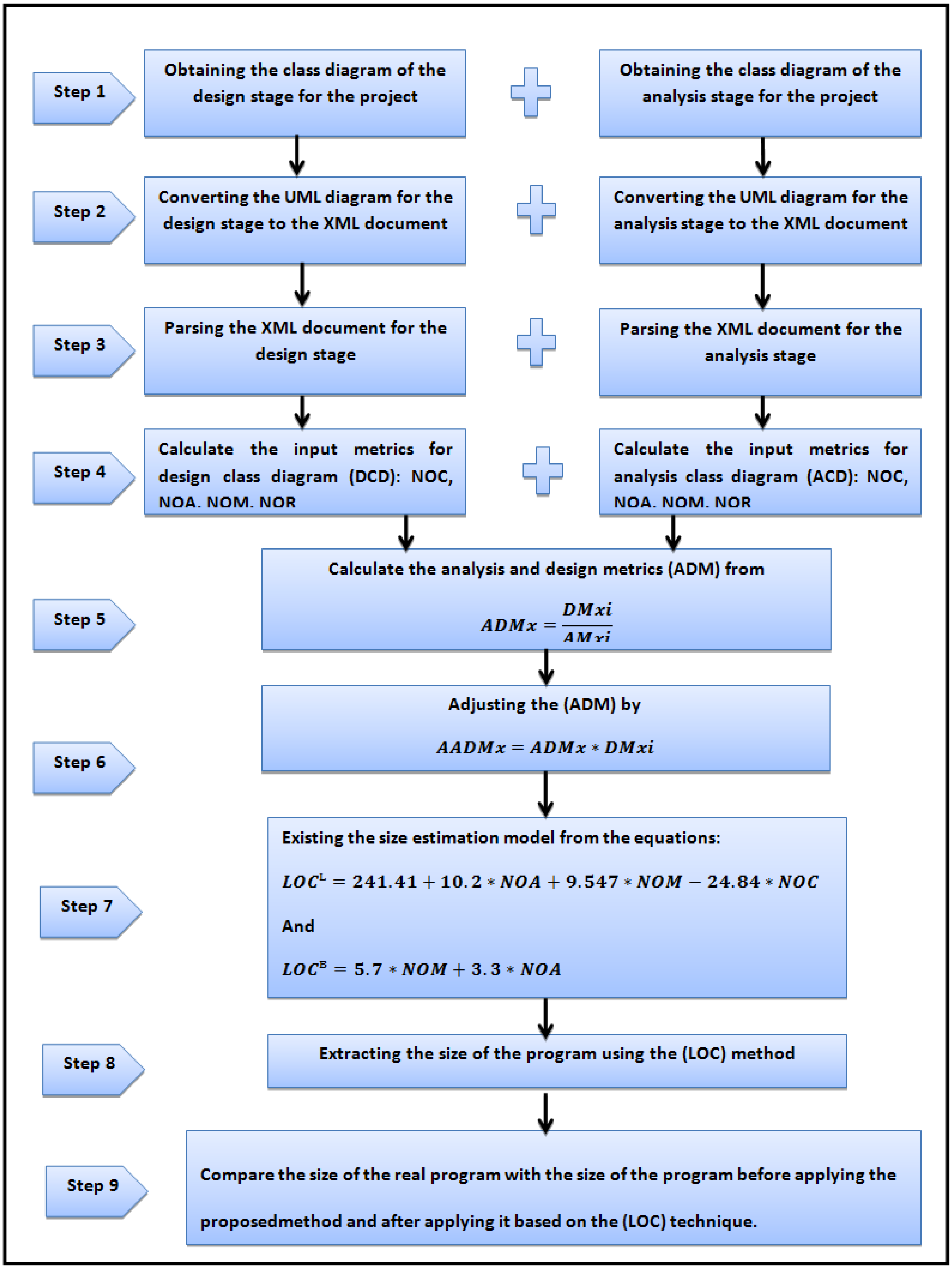
Ninth step: extracting the size of the program using the (LOC) method

Programs similar to the selected schemes were searched on GitHub. The programs were matched with the charts. Then calculate (LOC) for it. LOC here is the source code with eliminating comments and blank lines.

Tenth step: Software size estimation models are made to compare the actual software projects' size with before and after applying model (ADM)

Finding the real (LOC) before applying the (ADM) method in the design stage. And to find the expected (LOC) resulting from the application of the (ADM) method, and to find the difference or ratio between the two outputs.

ISSN (online): 2581-3048
Volume 7, Issue 3, pp 29-35, March-2023
https://doi.org/10.47001/IRJIET/2023.703005


Step 1
Obtaining the class diagram of the design stage for the project
Obtaining the class diagram of the analysis stage for the project
Step 2
Converting the UML diagram for the design stage to the XML document
Converting the UML diagram for the analysis stage to the XML document
Step 3
Parsing the XML document for the design stage
Parsing the XML document for the analysis stage
Step 4
Calculate the input metrics for design class diagram (DCD): NOC, NOA, NOM, NOR
Calculate the input metrics for analysis class diagram (ACD): NOC, NOA, NOM, NOR
Step 5
Calculate the analysis and design metrics (ADM) from
$ADMx = \frac{DMxi}{AMxi}$
Step 6
Adjusting the (ADM) by
$AADMx = ADMx * DMxi$
Step 7
Existing the size estimation model from the equations:
$LOC^L = 241.41 + 10.2 * NOA + 9.547 * NOM - 24.84 * NOC$
And
$LOC^B = 5.7 * NOM + 3.3 * NOA$
Step 8
Extracting the size of the program using the (LOC) method
Step 9
Compare the size of the real program with the size of the program before applying the proposedmethod and after applying it based on the (LOC) technique.

V. TEST AND RESULT FOR METHOD

For the purpose of executing the algorithm, test cases must be used and analyzed. Three test cases were used on three different class diagrams. Then, clarified the results of method. The first example represents a graduate research for a master's student examining the possibility of discovering copying codes using the Dart language, the second represents the ATM system, and the third represents the student information system table (1).

Table 1: Result before and after applying size estimation method

The model used	Project name	Real (LOC)	(LOC) before applying method	(LOC) after applying method
Model (1) "**LOC**^{L}"	P1	1500	566	1482
	P2	434	451	427
	P3	1334	903	1220
Model (2) **LOC**^{B}	P1	1500	237	622
	P2	434	234	313
	P3	1334	437	633

The method was applied to our data set, and the size of the programs was estimated based on two models, M1 and M2, and the results showed that M1 gave better results and closer to reality than M2. This is because M1 calculates more metrics than M2.

VI. CONCLUSION AND FUTURE WORK

This research examines the measures taken from the Class diagram, which can be used to estimate the size of software at an early stage of the software development life cycle. A new method was used in estimating the size based on the analysis and design stages, both separately and together. The results were also tested on a data set using two estimation models. The results were in favor of one model over the other.

In the future we plan to apply this method to a wider number of projects, preferably from industry.

REFERENCES

[1] Badri, M., Badri, L., Flageol, W., &Toure, F. (2017). Source code size prediction using use case metrics: an empirical comparison with use case points. Innovations in Systems and Software Engineering, 13(2), 143-159.

[2] Nassif AB, Ho D, Capretz LF (2013) Towards an early software estimation using log-linear regression and a multilayer perceptron model. J Syst Softw 86(1):144–160.

[3] Ochodek M, Nawrocki J, Kwarciak K (2011) Simplifying effort estimation based on use case points. Inf Softw Techno 53: 200–213.

[4] Lagerstroemia R, von Wurttemberg LM, Holm H, Luczak O (2012) Identifying factors affecting software development cost and productivity. Softw Qual J 20(2): 395–417.

[5] Zhou Y, Yang Y, Xu B, Leung H, Zhou X (2014) Source code size estimation approaches for object oriented systems from UML class diagrams: a comparative study. Inf Softw Technol 56: 220–237.

[6] Silhavy, R., Silhavy, P. and Prokopova, Z. (2021) Using actors and use cases for software size estimation. Electron., 10, 1–20.

[7] Densumite, S. and Muenchaisri, P. (2017) Software size estimation using activity point. IOP Conf. Ser.: Mater. Sci. Eng., 185, 1-8.

[8] Ungan, E. (2013) A functional software measurement approach bridging the gap between problem and solution domains. Ph.D. dissertation. In The Department of Information Systems. Middle East Technical University, Ankara, Turkey.

[9] Daud, M. and Malik, A.A. (2021) Improving the accuracy of early software size estimation using analysis-to-design adjustment factors (ADAFs). IEEE Access, 9, 81986–81999.

[10] Kim, S., Lively, W. and Simmons, D. (2006) An effort estimation by UML points in the early stage of software development. In Proc. SERP 06, Las Vegas, Nevada, June 26–29, pp. 415–421. CSREA Press, Las Vegas, Nevada.

[11] Zhou, Y., Yang, Y., Xu, B., Leung, H. and Zhou, X. (2014) Source code size estimation approaches for object oriented systems from UML class diagrams: a comparative study. Inf. Softw. Technol., 56, 220–237.

[12] Misic, V.B. and Te ˇ sic, D.N. (1998) Estimation of effort and ˇ complexity: an object-oriented case study. J. Syst. Softw., 41, 133–143.

[13] Antoniol, G., Lokan, C., Caldiera, G. and Fiutem, R. (1999) A function point-like measure for object oriented software. Empirical Softw. Eng., 4, 263–287.

[14] Antoniol, G., Fiutem, R. and Lokan, C. (2003) Object-oriented function points: an empirical validation. Empirical Softw. Eng., 8, 225–254.

[15] Chen, Y., Boehm, B.W., Madachy, R. and Valerdi, R. (2004) An empirical study of eServices product UML sizing metrics. In Proc. ISESE 04, Redondo Beach, CA, August 19–20, pp. 199– 206. IEEE, Washington, DC.

[16] Bianco, V.D. and Lavazza, L. (2005) an empirical assessment of function point-like object oriented metrics. In Proc. METRICS 05, Como, Italy, September 19–22, pp. 1–10. IEEE, Washington, DC.

[17] Tan, H.B.K. and Zhao, Y. (2006) Sizing data-intensive systems from ER model. IEICE - Trans. Inf. Syst., E89-D, 1321–1326.

[18] Tan, H.B.K., Zhao, Y. and Zhang, H. (2009) Conceptual data model-based software size estimation for information systems. ACM Trans. Softw. Eng. Methodol., 19, 1–37.

[19] Harizi, M. (2012) the role of class diagram in estimating software size. Int. J. Comput. Appl., 44, 31–33.

[20] Lazic, L., Petrovic, M. and Spalevic, P. (2012) Comparative study on applicability of four software size estimation models based on lines of code. In Proc. ECC 12, Prague, Szeh Republic, September 24–26, pp. 71–80. WSEAS Press, Zografou, Athens, Greece.

[21] Alashhb, M.I. and Lazic, L. (2016) A critical review of source code size estimation approaches for object-oriented programming languages: a comparative study. In INFOTEH-JAHORINA 16, Jahorina, Bosnia, March 16–18, pp. 535–540. IEEE, Washington, DC.

[22] Ayyildiz, T.E. and Koyiit, A. (2018) Size and effort estimation based on problem domain measures for object oriented software. Int. J. Softw. Eng. Knowl. Eng., 28, 219–238.

[23] Ayyildiz, T.E. (2015) Size and effort estimation based on correlations between problem and solution domain measures for object oriented software. Ph.D. dissertation. In The Department of Information Systems. Middle East Technical University, Ankara, Turkey.

[24] Kiewkanya, M. and Surak, S. (2016) Constructing C++ software size estimation model from class diagram. In Proc. JCSSE 16, Khon Kaen, Thailand, July 13–15, pp. 1–6. IEEE, Washington, DC.

[25] Badri, M., Badri, L. and Flageol, W. (2016) Source and test code size prediction-A comparison between use case metrics and objective class points. In Proc. ENASE 16, Rome, Italy, April 27– 28, pp. 172–180. Springer, Berlin.

[26] Object Management Group. About the UML specification version 2.5.1. https://www.omg.org/spec/UML/About-UML/ (accessed October 11, 2021).

[27] R. T. Hughes, ‘‘Expert judgment as an estimating method,’’ Inf. Softw. Technol., vol. 38, no. 2 pp. 67–75, Jan. 1996.

[28] R. Valerdi, B. W. Boehm, and D. J. Reifer, ‘‘COSYSMO: A constructive systems engineering cost model coming of age,’’ in Proc. INCOSE Int. Symp., vol. 13, no. 1. Hoboken, NJ, USA: Wiley, 2003, pp. 70–82.

[29] Y. Singh, P. K. Bhatia, and O. Sangwan, ‘‘ANN model for predicting software function point metric,’’ ACM SIGSOFT Softw. Eng. Notes, vol. 34, no. 1, pp. 1–4, Jan. 2009.

[30] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, "An effective approach for software project effort and duration estimation with machine learning algorithms,’’ J. Syst. Softw., vol. 137, pp. 184–196, Mar. 2018, doi: 10.1016/j.jss.2017.11.066

[31] M. Salmanoglu, T. Hacaloglu, and O. Demirors, ‘‘Effort estimation for agile software development: Comparative case studies using COSMIC functional size measurement and story points,’’ in Proc. ACM Int. Conf. Proc., 2017, pp. 41–49, doi: 10.1145/3143434.3143450.

[32] Pasuksmit, J., Thongtanunam, P., & Karunasekera, S. (2022). Story points changes in agile iterative development. Empirical Software Engineering, 27(6), 1-55.

[33] Roger S Pressman and Bruce R Maxim, 2020 “software engineering a practitioners approach”.

[34] Del Bianco, V., & Lavazza, L. (2006, October). Object-oriented model size measurement: experiences and a proposal for a process. In Workshop on Model Size Metrics, part of the ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MoDELS 2006), Genova.

[35] Amanullah, K., & Bell, T. (2019, October). Evaluating the use of remixing in scratch projects based on repertoire, lines of code (LOC), and elementary patterns. In 2019 IEEE Frontiers in Education Conference (FIE) (pp. 1-8). IEEE.

[36] Aswini, S., & Yazhini, M. (2017). An assessment framework of routing complexities using LOC metrics. 2017 Innovations in Power and Advanced Computing Technologies (i-PACT), 1-6.Software Size Estimation with Deep Learning Model, 2020.

[37] Morrow, P. (2018). Software sizing for cost/schedule estimation (Doctoral dissertation, Ulster University).

[38] Agresti, A. (2010). Analysis of ordinal categorical data (Vol. 656). John Wiley & Sons.

[39] Stern, S., & Gencel, C. (2010, November). Embedded software memory size estimation using COSMIC: a case study. In Int’l Workshop on Software Measurement (IWSM) (Vol. 39).

[40] Abdullah, Nur Atiqah Sia, Nur Ida Aniza Rusli, and Mohd Faisal Ibrahim. "A case study in COSMIC functional size measurement: angry bird mobile application." In 2013 IEEE Conference on Open Systems (ICOS), pp. 139-144. IEEE, 2013.

[41] Chander Diwaker, Astha Dhiman, “Estimating Size and Effort Estimation Techniques for Software

Development," International Journal of Software and Web Sciences (IJSWS), pp. 2279–0071, May. 2013.

[42] Ahmed, A.T. and Taha, D.B., Webapp Effort Estimation using Cosmic Method. International Journal of Computer Applications, (2018): 975, p.8887.

[43] https://en.wikipedia.org/wiki/Class_diagram

[44] Ungan, E., Trudel, S. and Abran, A. (2018) Analysis of the gap between initial estimated size and final (true) size of implemented software. In IWSM /Mensura 18, Beijing, China, September 19–20, pp. 123–137. CEUR Workshop Proceedings, Aachen, Germany.

Citation of this Article:

AESHAA L. AL-SALEEM, ASMA'A Y. HAMMO, "Estimating Software Size Using Metrics from Analysis and Design Phases (ADM) of the Software Development Life Cycle (SDLC)" Published in *International Research Journal of Innovations in Engineering and Technology - IRJIET*, Volume 7, Issue 3, pp 29-35, March 2023. Article DOI https://doi.org/10.47001/IRJIET/2023.703005

****