Automated Estimation of Functional Size from Code

Özgesu Özen
Department of Computer Engineering
Izmir Institute of Technology
Izmir, Turkey
ozenozgesu@gmail.com

Bora Özsoy
Department of Computer Engineering
Izmir Institute of Technology
Izmir, Turkey
boraozsoy@gmail.com

Busenur Aktılav
Department of Computer Engineering
Izmir Institute of Technology
Izmir, Turkey
busenuraktilav@gmail.com

Eren Can Güleç
Department of Computer Engineering
Izmir Institute of Technology
Izmir, Turkey
erencangulec@gmail.com

Prof. Dr. Onur Demirörs
Department of Computer Engineering
Izmir Institute of Technology
Izmir, Turkey
onurdemirors@iyte.edu.tr

***Abstract*—Determination of the size of a software project is challenging as well as crucial for both self-employed software developers and corporate businesses. That's why it is subjected to a lot of academic studies where it is discussed how to determine the size more accurately. Functional Size Measurement (FSM) is one the most popular measurement techniques for a software from the point of the delivered functionality. However, the aspects of know-how, the cost, time, and manual operation creates difficulties to apply FSM techniques. This study aims to solve these issues by automating the measurement process to approximate the functional size of a project using the COSMIC Functional Size Measurement. The end product of this study is called 'Cosmic APP' that utilizes the sequence diagram of a software after reverse engineering it from the given code using a third-party tool called 'SequenceDiagram'. The working principles, the estimation process, and the obtained results of 'Cosmic APP' are described thoroughly in this paper.**

Keywords—Functional Size Measurement, COSMIC, sequence diagram

I. INTRODUCTION

Functional Size Measurement (FSM) is a measurement technique for a software from the point of the delivered functionality. Despite the fact that Functional Size has many purposes to be used [1], it is mainly used at the planning stage as an input into project resource estimation calculations for cost, effort, and schedule. The function point count of delivered functionality provides input to productivity and quality performance indicators as well [2]. Functional Size Measurement provides numerical data to compare performance regarding the cost effectiveness and efficiency of the development and support teams throughout the project.

The unit of functional size measurement is called function point but the unit depends on the measurement method used [3]. Whichever method is used the count is a fundamentally useful value, that is obtained independently from the programming language or the program developer.[4] There are several FSM methods that are suitable for ISO standards: COSMIC(Common Software Measurement International Consortium)[5], IFPUG(International Function Point Users Group)[6], Mk II[7], NESMA(The Netherlands Software Metrics Association)[8], and FiSMA(The Finnish Software Measurement Association)[9].

The primary objective of this study is to support estimation of software project's effort by utilizing function point counts of existing projects. Organizations use sets of functional sizes of the existing projects and related parameters of the projects to produce an estimation model. However, the measuring the size of the existing projects takes time and requires significant effort. Our study focuses on automating this process by comparing the projects' predetermined functional size and estimated functional size that is computed using the sequence diagrams' interactions. Thus, significantly reducing the time that is needed for measuring the existing projects. Our goal is to minimize the effort needed to establish measurement sets of software projects.

The purpose of our study is to decrease the effort needed for measuring the size of the software from code. In order to achieve this purpose, we propose to automate the process of manual functional size measurement of software from code. Research methodology of this study is as follows: First, manual measurements on different projects are performed. Then we performed a literature research on the topic to understand the state of the art. Reverse engineering tools are tested, and the most suitable tool is selected for the generation of the sequence diagram. Later, the Cosmic APP is developed. The solution workflow of Cosmic APP is as follows: A Java application is acquired. The sequence diagram of the application is produced with the help of a reverse engineering tool. Sequence diagram is converted to the text format. Text format of sequence diagram is given to the Cosmic APP and the functional size estimation of the given Java application is calculated.

In the case study setup, the Cosmic APP estimation results of the acquired Java applications are compared with the manual functional size measurements of these Java applications in order to validate the Cosmic APP accuracy. It cannot detect data movements so we prepared a correlation table to compare the functional processes and the methods that are found by the Cosmic APP so that we could observe if the calculated CFP by Cosmic APP is a coincidence or not. Correlation table and the results are further discussed in the limitations of the study section.

The topics and contents of this study are as follows. *Section II* clarifies the background on COSMIC FSM and the sequence diagram, and it provides the comparison of reverse engineering tools. *Section III* overviews related works by explaining the similarities and differences between our study

978-1-7281-8541-5/20/$31.00 ©2020 IEEE

and their studies. *Section IV* provides a detailed explanation for the produced software tool Cosmic APP. *Section V* explains the studies of cases and overviews the comparisons of the results from manual and automated estimation of the function points, examines the results. Lastly, *Section VI* provides general conclusions and future work of opportunities and plans.

II. Background

A. FSM and COSMIC

In this study we have decided to claculate functional size using COSMIC method as COSMIC is the second generation method, is an ISO standart and conforms to measurment theory.

COSMIC, the Common Software Measurement International Consortium, defines the principles, rules and a process for measuring a standard functional size of a piece of software. The COSMIC method is used to measure the functional size of a software. Functional processes, data groups and data movements are identified in response to functional user requirements (FURs) in the COSMIC generic software model.[10] Before moving on, the vocabulary of COSMIC method is briefly explained. Functional process is a set of data movements that is unique to the software. Functional user is the intended recipient of data processed. Functional User Requirement (FUR) is the statement of functional requirements such as user stories. Object of Interest is anything in the world of the functional user, which the software being measured must process the data. Data group consists of one or more data attributes which all describe a single object of interest. Triggering entry is the data movement which is the start of each functional process. The data group moved by the triggering Entry is generated by a functional user in response to a triggering event.[12][13] There are four types of data movement:

- Entry (E)
- Exit (X)
- Write (W)
- Read (R)

An entry moves a data group from a functional user into the functional process. An exit moves a data group from a functional process to the functional user. Write moves a data group from inside a functional process to persistent storage area. Read moves a data group from persistent storage into the functional process. Each data movement is counted as 1 CFP (COSMIC function point). When the data movements are accumulated over all functional processes, the size of software is then defined.[12]

B. Sequence Diagram and Reverse Engineering Tools

The OMG's Unified Modeling Language™ (UML®) helps you specify, visualize, and document models of software systems, including their structure and design, in a way that meets all of these requirements.[14] A UML sequence diagram is also known as interaction diagram which shows the interaction between objects in a sequential order. It captures the high-level interactions between user and the system or the system and other systems. These interactions realize a use case or an operation.[15] The idea behind utilizing the sequence diagrams in this project is to encapsulate all the use cases and operations from a single point if it is visualized in one diagram. Representing those interactions between objects is critical to detect the data movements between objects. Therefore, reverse engineering tools play an important role to convert the given code into a sequence diagram.

Reverse engineering is the process of analyzing a subject system to identify the system's components and their interrelationships and create representations of the system in other form or at higher levels of abstraction. There are quite a few different reverse engineering tools in the market. As previously pointed out that the quality and correctness of the reverse engineering tools is a key factor for the estimations to be made accurately.[16] Depending on the reverse engineering tools, the produced sequence diagrams also vary. A few different reverse engineering tools are found: SequenceDiagram [17] for IntelliJ IDEA, Visual-Paradigm [23], ZenUML [22], ObjectAid UML Explorer [24]. These tools are tested to compare the produced sequence diagram. Comparison of the reverse engineering tools is represented in Table I. Reverse engineering tools that have been found are evaluated based on convenience of the tool and its accuracy. Convenience of using a tool is our first priority to select the most appropriate reverse engineering tool, because usage of a convenient tool maintains the repeatability of proposed methodology steps. We eliminate Visual Paradigm and ObjectAid UML according to this concern. Next, reverse engineering tools which generate accurate and workable sequence diagrams are searched. SequenceDiagram plugin of IntelliJ is used thanks to its feature which is exporting sequence diagram elements as text.

TABLE I. REVERSE ENGINEERING TOOL COMPARISON

	SequenceDiagram	**ZenUML**	**Visual Paradigm**	**ObjectAid UML**
Free	✓	✓	X	X
Open Source	✓	?	X	X
Output Configuration	✓	X	✓	?
Export as Image	✓	✓	?	?
Export as PDF	X	✓	?	?
Export as TXT	✓	X	?	?
Export as HTML	X	✓	?	?

III. Related Work

There are related works that have been done about automating the functional size measurement process. Each method has its own vocabulary and its own way of modeling software [18]. That's why they are distinct from one another. These related works are summarized. The comparison between our proposal and the related works is explained.

Kusumoto et al. [19] propose functional size calculations and moot the idea of automated Functional Size Measurement. They perform the FSM by IFPUG (International Function Point Users Group) and follow the

design specifications described by the UML (Unified Modelling Language) as the sequence and class diagrams which are acquired by Rational Rose®. They produced a FP measurement tool to automate the FP analysis process. The values which are calculated by their tool are nearly the same as manual values obtained by the specialist in IFPUG FP analysis.

A self-developed measurement library that monitors the data movements used in the studies of Tarhan and Akca [20]. Aforesaid method is proposed to apply for functional size measurement that is triggered from the GUI. From a single point all data movements could be traced. Thus, the result of the measurement is calculated by 92% accuracy.

Tarhan and Sağ [11] focus on automation of FSM from software code and develop a tool called 'COSMIC Solver' for COSMIC FSM of Java Business Applications. Cosmic Solver is the most related approach to our solution Cosmic APP. These two studies focus on the same problem and follow similar steps. However, they diverge greatly in the means of analyzing the UML sequence diagram. The methodology of this article is to extract the textual representation of the sequence diagram and with the usage of AspectJ technology tagging these textual representations. Then, calculating functional size of user scenarios from the tag information. However, AspectJ technology is not used in Cosmic APP due to increased complexity for clients to use the tool and slowed down functional estimation process of software. The study demonstrates that the effectiveness of COSMIC Solver is proven by 77% accuracy of functional sizing of the JBA. Compared to the manual measurement of CFP, the efficiency of the tool is 12 times faster in functional sizing of JBA.

Papers that use similar techniques (IFPUG, COSMIC) to measure the size of the software [11][19] resulted in significantly different results as the reverse engineering tool and algorithm behind the verification phase would be distinct with a goal of reaching higher accuracy in less computing time.

IV. COSMIC APP

We design the Cosmic APP with easy-of-use and speed in mind from the beginning. The main priorities are estimating the CFP of a Java Business Application accurately without needing an in-depth knowledge about the COSMIC method. The upcoming paragraphs include detailed explanation on how the Cosmic APP works and what someone needs to operate it successfully.

The Cosmic APP does not oblige a measurement expert or an information technologies specialist. That being said, it is recommended to have a general idea about the COSMIC method if more accurate estimations are desired. The reason is clarified in the following paragraphs.

The main methodology behind the Cosmic APP is utilizing the sequence diagram of the given project. To achieve this, we use the aforementioned reverse engineering tool called SequenceDiagram. This tool runs from the method where it is called and displays the flow of the program in a colored sequence diagram. The choice of the method has a significant importance in the end result of the estimation. That's why it is recommended to conduct the measurement using the main method where it is applicable. If there is no main method or the program depends on running via a build tool, it is recommended to find a place where the program starts to operate like a login page, main menu or their related method or constructor.

The Cosmic APP relies on SequenceDiagram's output to be able to perform its estimation. The output can be exported in a text file after choosing its options about the output's configuration. These include skipping mutator (setter) and accessor (getter) methods, constructor or private methods and a call depth for the methods etc. These options can be chosen by the user depending on the project. Then, the output file's location can be given to the Cosmic APP to start the estimation process.

The estimation takes place after the user specifies whether he or she prefers to exclude any method that is part of the sequence diagram but not related with the COSMIC method. This feature is offered for the users who have knowledge of COSMIC FSM or have noticed unrelated methods in their previous executions. The entry does not require the exact method name or its parameters. If the given name is contained within the SequenceDiagram output, the Cosmic APP automatically excludes it from the evaluation. This step is not mandatory for the evaluation process, but it may increase the accuracy of the estimation. It can be skipped if desired.

In the end, the assessment begins, and the results are displayed. Program displays the counted method list and the total CFP score. If any method or a group of methods seen that the user thinks is irrelevant to COSMIC measurement, the program can be re-run with these methods excluded to fine-tune the estimation result until no misinterpreted methods exist.

V. CASE STUDY SETUP

COSMIC Functional Points of several software products are calculated manually and conscientiously. The size obtained by the Cosmic App is compared with the sizes recorded manually in order to predict the margin of error. In case the hit ratio is less than projected value (~85%), the software and the logic-behind the produced-tool would be revised and the assessments would be performed again. Throughout the study, this cycle is followed. The closest study to ours is Cosmic Solver and they achieved 77% accuracy. Therefore we determine the acceptance criterion as 85% so that our study could contribute to the scientific community by a different approach and better accuracy rate to the stated problem.

A. Application Selection and Information Gathering

With respect to knowledge gained from the research and analysis, a solution is developed. Thanks to our fellow student friends, some of their homework projects from the CENG431 Building Software Systems course in Izmir Institute of Technology are obtained to be assessed as part of our study. One of this course's main subjects is to teach different design patterns that are used while developing a software. Homework projects ask to implement solutions with the given software design pattern. Measuring different projects that utilize various design patterns gives great opportunity to see developed algorithm's consistency with respect to manual measurements. In addition, various GitHub repositories are used to test the Cosmic APP. They establish a great code source to test Cosmic APP's correctness.

We gathered sample software projects for our case studies. These projects are heavily dependent on the reverse engineering tool — SequenceDiagram. Firstly, this tool requires the IntelliJ IDEA. Therefore, the projects to be assessed should be written in a programming language which the IntelliJ IDEA supports, in this case Java. Secondly, in our estimation process, we have encountered several circumstances where the SequenceDiagram failed to produce an output. The projects that satisfy these conditions are used in the solution developing process.

B. Manual Measurement and Automated Estimation Results of Case Studies

Functional point sizes are calculated manually for the projects. The estimated size obtained by the Cosmic APP is compared with the sizes recorded in order to predict the margin of error. As part of the case studies, only eight projects are presented in Table III.

At first, the acceptance criterion was determined as 85% COSMIC point accuracy. However, there are some negative aspects that reduce the accuracy of the Cosmic APP which are discussed in the upcoming paragraphs and in the conclusion in detail. In the end, our solution, Cosmic APP achieved 87.8% accuracy. Counting each CFP as a weight for each project, accuracy rate is calculated by Weighted Mean Relative Error using the formula:

$$WMRE = \frac{\sum \text{(Absolute Error)}}{\sum \text{(True Value)}} x\ 100$$

The accuracy rate is 79.325% according to the Mean Relative Error which is:

$$MRE = \frac{Absolute\ Error}{True\ Value} x\ 100$$

The error rates in Table III is also calculated using the MRE formula. However, some of the cases resulted in much higher error rates than the achieved accuracy. This problem is caused by several things. One of the most common issues that is encountered is programming in a design pattern which does not cover COSMIC data movements in its methods. For instance, gathering a lot of work under one method results in an estimation that is significantly lower than the actual number since they are counted once. Besides, the evaluation of the sequence diagrams generally performed from the main methods, but in some cases, these main methods were not invoking all the functionalities of the program, thus reducing the estimated value. On the other hand, in our case, these issues are seen in the cases with low CFPs. That's why they do not affect the overall accuracy greatly even though their particular error rate is high.

One of the case studies is a hotel management system. It is obtained from GitHub and the project is called "Java Simple Hotel Management" [21]. Hotel administration can manage who has access to the system. Room reservation is possible. Hotel administration can control the customers' and the rooms' management. Additional rentals are also recorded in the booking diary. Also, payment is calculated according to the room, additional renting, food price and payment and checkout can be made.

The manual COSMIC function point result of the hotel management project is 104. The Cosmic APP estimation is 101. There is a %2 error between the actual and Cosmic APP results. The correlation between the functional processes and related methods is 100% as represented in Table IV.

The other sample case is the Shopping Centre App. In this application, customers can make product selection. Selected products can be added to or deleted from the shopping bag. Payment can be made for the items in the shopping bag. The manual COSMIC function result of the Shopping Centre project is 17 and the Cosmic APP estimation is 24. There is a 41% error between the actual and Cosmic APP results and the correlation between the functional processes and related methods is 60% for its 5 functional processes as stated in Table II. It means that Cosmic APP could only located 3 out of 5 FPs.

After these results, the validation of the assessment comes as a question. Since the Cosmic APP cannot detect the data movements between functional user requirements, the answer to this question has significant importance in the approval process. We would like to try to evaluate this via comparing the two result sets that are obtained after the end of the case studies. First, the overall error rate of the cases in the Table III equate to 87.8% which is in the same class as the studies mentioned in the related work section. Secondly and most importantly, the overall correlation between functional processes and related detected methods resulted at 93.4% for all the cases. This number confirms the validity of the Cosmic APP and proves that the estimations are not coincidence.

TABLE II. SHOPPING CENTRE APP METHOD CORRELATIONS

FP Name	**CFP Size**	**Related Method**
Add product to shopping centre	3	-
Add product to bag	3	addProduct
Get a bag	3	-
List product in bag	3	passThroughCounter
Pay	5	checkOrderOf
Total	**17**	
Unrelated Methods	checkExtensionFactor, isProductInBag, contains, peek, checkInitialization, pop, push, ensureCapacity, add, getFrequencyOf, shoppingBagSize, shoppingBagCapacity, shiftEntries, enqueue, dequeue, shiftQueueEntries, isInQueue, putToDesk, grab, toString, numberOfProductNotInOrder	

TABLE III. CASE STUDIES' CFP RESULTS, ERROR RATES AND ASSESMENT DURATIONS

No.	Project Name	Functional size obtained by the MANUAL FSM procedure (CFP)	Functional size estimated by the AUTOMATED FSM procedure (CFP)	Error Rate (%)	Approximate duration of measurement MANUAL (min)	Approximate duration of estimation AUTOMATED (min)
1	FlightTableApp	30	20	33.3	30	2
2	AutoCorrectionConsoleApp [25]	12	17	41.6	30	2
3	Addressbook [26]	13	11	15	30	2
4	Simple Employee Payroll Management System [27]	59	57	8	120	2
5	Java Simple Hotel Management [21]	104	101	2	240	2
6	Shopping Centre App	17	24	41	30	2
7	Video Management App	50	56	12	90	2
8	Shopping List Application	8	9	12.5	20	2

TABLE IV. JAVA SIMPLE HOTEL MANAGEMENT METHOD CORRELATIONS

FP Name	CFP Size	Related method	FP Name	CFP Size	Related method
Add Room	**5**	btn_addActionPerformed	Display Food	**2**	table_foodMouseClicked
Edit Room	**5**	btn_editActionPerformed	Add Item	**5**	btn_addActionPerformed
Delete Room	**5**	btn_deleteActionPerformed	Edit Item	**5**	btn_editActionPerformed
Display Room	**2**	table_roomsMouseClicked	Delete Item	**5**	btn_deleteActionPerformed
Add Room Type	**4**	btn_addRoomTypeActionPerformed	Display Item	**2**	table_itemMouseClicked
Edit Room Type	**4**	btn_editRoomTypeActionPerformed	Add Order	**2**	jButton1ActionPerformed
Delete Room Type	**2**	btn_deleteRoomTypeActionPerformed	Search Booking	**4**	searchHelper
Display Room Type	**2**	table_roomTypeMouseClicked	Generate Payment	**4**	jButton1ActionPerformed
Add Customer	**5**	btn_addCustomerActionPerformed	Check-out	**4**	btn_checkOutActionPerformed
Edit Customer	**5**	btn_editCustomerActionPerformed	Display Payment Property Change	**3**	table_paymentPropertyChange
Delete Customer	**5**	btn_deleteCustomerActionPerformed	Print Memo	**3**	jButton2ActionPerformed
Display Customer	**2**	table_customerMouseClicked	Search Customer	**4**	searchCustomerHelper
Add Food	**2**	btn_addActionPerformed	Room Up	**2**	btn_room_upActionPerformed
Edit Food	**2**	btn_editActionPerformed	Check-in Property Change	**3**	date_checkInPropertyChange
Delete Food	**2**	btn_deleteActionPerformed	Save Booking	4	btn_saveBookingActionPerformed
			Total	**104**	
Unrelated Methods	initComponents,actionPerformed, getSize, getElementAt, roomsToRoomObjectList, flushAll, ObjectCreation, boolToString, flushStatmentOnly, createNewCustomer, roomObjectCreation, foodObjectCreation, itemObjectCreation				

C. Limitations of the Study

There are a few threats which are capable of peril our solution. The first peril is the heavy dependency to the reverse engineering tool for estimating the size of the software because of two reasons. The domain of our proposal, java applications, degrade to Java applications which are run on IntelliJ and the sequence diagram can be produced from by using the plugin [17].

The cases which the plugin could not produce a sequence diagram could not be estimated in the means of software size and there is not an alternative way to do the estimation. In addition, our solution trusts the aforementioned reverse engineering tool [17] to produce high quality and correct sequence diagrams. However, this may not be the case for every software and if the produced sequence diagram has flaws, then a way to address this issue is not being suggested. Due to failure of the reverse engineering tool, the results are unreliable.

Second peril of our proposal is the garbage-in garbage-out problem meaning that if the given input is flawed then the output will be flawed. It is assumed that a meaningful input is given to the program. In the case of an incomplete or erroneous sequence diagram produced by the reverse engineering tool, the calculated output of the program will be inaccurate. Therefore, the correctness of given inputs is the user's responsibility.

Third peril of our proposal is the failure of detecting the data movements described in the COSMIC method. Therefore, our solution only estimates the COSMIC functional size measurement so the results of the estimation may vary in different cases and may not be accurate. For this reason, Cosmic APP may not be suitable or scalable for corporate businesses or any area where precision is essential.

Another peril to our solution is that manual measurements are made by the authors of this study so it might be considered as bias. However, the example case study is a public repository on GitHub so that everyone could check the manual measurement result. For this reason, our case studies can be verifiable.

The argument of Cosmic APP being more accurate, efficient, and effective than other studies would not be yet asserted and more case studies should be evaluated. In the future, we plan to overcome these threats and the solution to these threats are being discussed in Section VI.

D. Potential Threats to Validity

There is number of validity threads to discuss for this study. First, this study has taken place in ten months. That's why it is open to any discussion about the effect of time on the authors that their expertise in the COSMIC FSM method may have improved over time. This aspect needs to be taken into consideration about the manual assessment of the case studies under maturation topic. Additionally, these manual assessments and Cosmic APP estimations may have impacted by the statistical regression due to repeated trials of the development phase. Also, this study is effective on projects with sample features that are written in Java and whose sequence diagram can be created by the SequenceDiagram plugin. Therefore, the methods have been used for estimation cannot be generalized for all projects. Furthermore, the knowledge and experience accumulation about Functional Size Measurement of the people doing this study are at different levels. In this case, internal validity may be discussed considering this selection bias factor.

VI. Conclusion And Future Work

Rapid development in the software industry reveals the greatest importance of FSM automation. In this article we conduct a scientific study on automated estimation of Cosmic Functional Size of a software project. We clarified the background on the COSMIC Functional Size Measurement and sequence diagram. We referred to why we need a reverse engineering tool and the comparison of reverse engineering tools and we give details of working principles of the SequenceDiagram IntelliJ IDEA plugin that we chose for our study in the development process. We overviewed related works mainly focused on Cosmic Solver [11] by explaining the similarities and differences between our study and their studies. However, detailed analysis could not be achieved because these two studies examine different types of projects. Only overall accuracy results are compared between them. We introduced our produced software tool called Cosmic APP and we presented the results of several case studies on comparing the Cosmic Functional Size obtained manually and the Cosmic Functional Size estimated by Cosmic APP. We specified one main research goal that we should get more than 85% accuracy when estimating the functional size. The obtained results and error rate calculations indicate that we have 87.8% accuracy between measured Functional Size manually and estimated Functional Size by Cosmic APP.

We specified some of the positive and negative features of our produced tool Cosmic APP such that Cosmic APP has higher accuracy, provides an easy-use, calculates Functional Size much faster than Cosmic Solver, nonetheless Cosmic APP does a poor job while detecting the data movements than Cosmic Solver and that causes the uncertain results for Cosmic APP. In addition, Cosmic APP strongly depends on the performance of reverse engineering tool. Despite these facts, this study has been accepted as a success by those carrying out this research, as it has been shaped according to scientific research steps and will contribute to scientific studies and the desired goal has been achieved.

In future work, we plan to review the project in line with making it able to detect data movements and raising accuracy as a first task. Also, by integrating the manual part that is the acquiring diagram part of the project into the code, there is the opportunity to make the project fully automatic by reducing the manual part in the estimation process. There are several opportunities for future work. The project can become more user friendly by designing a graphical user interface in the future. Web integration of the Automated Estimation of Functional Size from Code project may lead to full performance in the scenarios that require collaboration as well as individual scenarios and it will increase user satisfaction. Making a version of the project in the form of a plugin can be given. The estimation of the size of the software may be performed with more customizable results by adding a machine learning algorithm on the Cosmic APP which examines and learns the pattern of the relationship between the structure of the software and the user options, in the future.

REFERENCES

[1] Ozkan B., Turetken O., Demirors O. (2008) Software Functional Size: For Cost Estimation and More. In: O'Connor R.V., Baddoo N., Smolander K., Messnarz R. (eds) Software Process Improvement. EuroSPI 2008. Communications in Computer and Information Science, vol 16. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-85936-9_6

[2] Tarhan A., Demirors O. (2008) Assessment of Software Process and Metrics to Support Quantitative Understanding. In: Cuadrado-Gallego J.J., Braungarten R., Dumke R.R., Abran A. (eds) Software Process and Product Measurement. Mensura 2007, IWSM 2007. Lecture Notes in Computer Science, vol 4895. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-85553-8_9

[3] C. Gencel and O. Demirors, "Conceptual Differences Among Functional Size Measurement Methods," First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), Madrid, 2007, pp. 305-313, doi: 10.1109/ESEM.2007.43

[4] Cigdem Gencel and Onur Demirors. 2008. Functional size measurement revisited. ACM Trans. Softw. Eng. Methodol. 17, 3, Article 15 (June 2008), 36 pages. DOI:https://doi.org/10.1145/1363102.1363106

[5] ISO/IEC (2011) ISO/IEC 19761: Software engineering - COSMIC: A functional size measurement method

[6] ISO/IEC (2009) ISO/IEC 20926: Software and systems engineering - Software measurement - IFPUG functional size measurement method.

[7] ISO/IEC (2002) ISO/IEC 20968: Software engineering - Mk II Function Point Analysis - Counting Practices Manual.

[8] ISO/IEC (2005) ISO/IEC 24570: Software engineering - NESMA functional size measurement method version 2.1 - Definitions and counting guidelines for the application of Function Point Analysis.

[9] ISO/IEC (2008) ISO/IEC 29881: Information technology – Software and systems engineering – FiSMA 1.1 functional size measurement method

[10] The COSMIC Software Sizing Methodology. (n.d.). Retrieved from https://cosmic-sizing.org/cosmic-fsm/

[11] Kolukısa Tarhan, Ayça & Sağ, Muhammet. (2018). COSMIC Solver: A Tool for Functional Sizing of Java Business Applications. Balkan Journal of Electrical & Computer Engineering (BAJECE). 6. 1-8. 10.17694/bajece.401986.

[12] Abran, A. (2019). Software Development Velocity with COSMIC Function Points. Retrieved from https://cosmic-sizing.org/wp-content/uploads/2019/04/COSMIC-FP-for-Developers.pdf

[13] Introduction to COSMIC Function Points. (2020, March 08). Retrieved from https://www.scopemaster.com/introduction-to-cosmic-function-points/

[14] What is UML. (n.d.). Retrieved from https://www.uml.org/what-is-uml.htm

[15] What is Sequence Diagram? (n.d.). Retrieved from https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/

[16] Osman, Mohd Hafeez & Chaudron, Michel. (2012). Correctness and Completeness of CASE Tools in Reverse Engineering Source Code into UML Model. The GSTF Journal on Computing (JoC). 1.

[17] "SequenceDiagram Plugin" Available at: https://plugins.jetbrains.com/plugin/8286-sequencediagram

[18] Ghislain Levesque, Valery Bevo, and De Tran Cao. 2008. Estimating software size with UML models. In Proceedings of the 2008 C3S2E conference (C3S2E '08). Association for Computing Machinery, New York, NY, USA, 81–87.

[19] Uemura, Takuya & Kusumoto, Shinji & Inoue, Katsuro. (2001). Function-point analysis using design specifications based on the Unified Modelling Language. Journal of Software Maintenance. 13. 223-243. 10.1002/smr.231.

[20] Akca, A. A., & Tarhan, A. (2012). Run-time Measurement of COSMIC Functional Size for Java Business Applications: Initial Results. 2012 Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement. doi: 10.1109/iwsm-mensura.2012.40

[21] Java Simple Hotel Management Available at: https://github.com/faysal515/Java-Simple-Hotel-Management

[22] ZenUML Available at: https://app.zenuml.com/

[23] Visual Paradigm Available at: https://www.visual-paradigm.com/

[24] ObjectAid UML Explorer Available at: https://www.objectaid.com/home

[25] AutoCorrectionConsoleApp Available at: https://github.com/berkaykarakoc/AutoCorrectionConsoleApp

[26] Addressbook Available at: https://github.com/vaadin/addressbook

[27] Simple Employee Payroll Management System Available at: https://github.com/didarulcseiubat17/Simple_Employee_Payroll_Management_System