ANALYSIS OF REVERSE AND FORWARD ENGINEERING BY USING FUNCTION POINT AND LINE OF CODE

Ayesha maqsood[1], Nayyar Iqbal[2], Iqra ayub[3]
Department of Computer Science, University of Agriculture, Faisalabad Pakistan
Corresponding Author: ayeshamaqsood75@yahoo.com[1]
nayyariqbal@uaf.edu.pk[2]

***ABSTRACT**: The purpose of this research was to conduct the analysis of reverse and forward engineering by calculating the function point and line of code of software. In this research the main purpose was to analyze which one is better approach forward engineering or reverse engineering. It was difficult for software engineers to decide whether reverse engineering or forward engineering approach is cost effective in the existing software in order to modify it. In this research forward and reverse engineering approaches were applied on same software. In this research complete estimation of number of developers required for development was determined. The analysis conducted also determines in which approach the number of error or defects percentage was more, time required to develop the software and cost estimation was more. This analysis also determines the numbers of developers required for software development. The results represented by graphs indicating the developers required, time required to develop the software, number of errors or defects occurrence.*

Keywords: Forward engineering**,** reverse engineering, function point, line of code, complexity

1. INTRODUCTION

Information technology involves all types of technologies that includes production, storage, installation and implementation of computer science and use all these information's in different forms. From this information can be retrieved and store. That can also transmit information [4]. Software engineering involves the construction and maintenance of products. It is also engineering profession involved in designing. The theories, methods and tools are applied. These methods are applied where these tools and theories are properly applied [1].

At higher levels, software engineering has two types of engineering, which is reverse engineering and forward engineering.The engineering play vital roles in the software industry. Forward engineering is the engineering which moves from high level to low level abstraction. In this the high level model or concepts are building to low level details [2].

It has various principles in software database. It is very important in IT because it describes the normal process. In this the models are formed from the codec set [7]. It is a traditional process to the physical execution of system. In some situation it is without any technical details like thermal and electrical properties [5].

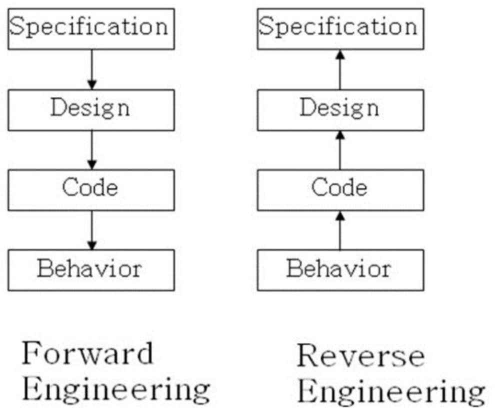

Fig. 1: cmaprison of Forward and Reverse engineering

Reverse engineering is another aspect of software engineering, it works in order to duplicate or enhance the object. Reverse engineering is the process of duplicate the existing objects. It moves from low level abstraction to high level. In this the codec set is developed from the existing model. It is backward engineering. It is the process of re-engineering. Reverse engineering compress the product development times. Reverse engineering is the contradiction of forward engineering [11].

The reverse engineering process can be understood by the example, assume you want to run the fan in your office by the two switches. You can switch on or off with any kind of switch. The logic of current is simple, that if one switch is up and the other one is down then the fan will run when both switches are up or both or down it will go off. So the logic is equal to the exclusive OR gate. You can understand the electrical circuits in this way easily and you can also easily design it. This is the reverse engineering [6].

In software systems, the function points play an important role. In early stages the software system could not thoroughly understand. With the passage of time the tools are developed in order to understand the systems. The function point plays a vital role to solve this problem. In this the code is divided into different smaller components and better understands and analyzed. Function point is the measurement for the software to understand it like an hour for measurement of time, kilometer for distances, etc. everyone understands their problems by dividing it into different pieces, Classes and categories. Function points measure the functional requirements of the system that are applied before development [12].

Line of code in a software program shows the complexity of software. If the less line of code in the program is less complex and if the more line of codes the program is more complex. Line of code shows the lines of a program code. LOC measures the volume of the program [9]. It is also uses to compare the program that use the same language and same coding standards. That measures the complexity of program input and output. It measures the size of the program and determines the effort that will require developing a program. Function point and line of code measures by developing the system using the platform of forward engineering and same system develop on the platform of reverse engineering[13].

2. METHODOLOGY

In this research, the software is developed and then on same software applied both forward and reverse engineering. Calculate the both function point and line of code of both engineering and then analyze that which one is better. Then decision will be taken on the basis of less and more line of code. Which one has less line of code and function points that is better and less complex than the other. It will help in the future to select the best and appropriate engineering in our research.

FP analysis technology is considered to be an early and effective sizing software count standard, but usually there's a complete and detailed look at behavior set be measured any software for the availability of documents the interplay of functional requirements for the application user refers to.

There are circumstances in which estimation method of at least two compatible, but the alternative could be decisive for FP standard rules. This software or development of such a plan in the early stages, it is only one FP count (feasibility study), it is not possible to perform according to the quality when the first case. As a standard, in fact, are not identifiable at the start of a project is always based on the reviews, however, size, time and price predictions need to be indeed stronger than functional details at the end of the stage, when there is a need to identify elements [19].

It's so easy to measure time. It is very easy to measure distance, too. Temperature, speed and etc. Miles, for hours, degree it's pretty understandable matrix. When it comes to estimating a software project or an application for measuring the problem occurs. How to measure this application [17]. There are countless theories-worse than each other though. This problem is a growing complexity, spare no direct methods can be used. So much time, number of lines of code, the application will be most meaningful, but I thought that it should not at all the way [3]. Numerous programming languages basically can be used to reach a goal. Each of these languages and the quantity of data and code lines is different- as a result-the need for a different number. Due to a complex application, written in C, Visual Basic, for example letters designed to consist of only two hundred thousands of lines [8].

Software size may be estimated either in terms of KLOC (Kilo Line of Code) or by calculating thenumber of function points in the software. Lines of code depend upon coding practices and Function points vary according to the user or software requirement [14].

The managers estimate efforts in terms of personnel requirement and man-hour required to produce the software. For effort estimation software size should be known. This can either be derived by managers' experience, organization's historical data or software size can be converted into efforts by using some standard formulae [16].

Function point analysis (FPA) is a methodology for measuring software productivity and the cost associated with the development and maintenance. One function point (FP) is one end-user requested business function [18].

In this initially the project of BATA was developed in C# in which average complexity was considered because some component were already present in developed systems already installed in working condition in the market and some were totally new.

Table. 1: Complexity Value Table

	Estimated Count	**average Complexity**
No. of Inputs	42	4
No. of outputs	15	5
No. of Inquiries	3	4
No. of files	26	10
No. of Interfaces	1	7

The fourteen question values for this case $\sum F_1 - F_{14} = 53$ **FP Calculation [3] FP=** UCF * TCF

UCF= Estimated count * Complexity Factor

Table. 2 : Calculated UCF

(42) * (4) =	168
(15) * (5) =	75
(3) * (4) =	12
(26) * (10) =	260
(1) * (7) =	7
Total UCF=	**522**

TCF= 0.65 + 0.01 * ∑ F1 – F14

=0.65 + 0.01 * 53

$\boxed{\textbf{TCF = 1.18}}$

So the **FP**= 522 * 1.18

= 616

Line of Code Calculation [3]

Line of Codes of BATA software developed in C# language application is **1667**

So in initial software, the FP are 616 and LOC 1667 are for C#. Labor rate per month is $1000 and Productivity is 20FP/month

Cost/FP = 1000/2

= $50

Total Bata project cost = 50 * 616

=$30,798

Effort in person Month = total project cost / labor rate/month

= 30798/1000

Effort in person Month= 31

Using Reverse engineering

In this applied reverse engineering on the initially developed application and add purchase components in C # BATA software and considered its complexity average because in this application developed two components add purchase item and purchase status which are not included in the application in C # and then Calculate its FP and LOC as follows

Table. 3: Complexity Value Table

	Estimated Count	**average Complexity**
No. of Inputs	56	4
No. of outputs	20	5
No. of Inquiries	3	4
No. of files	29	10
No. of Interfaces	1	7

The fourteen question values for this case$\sum F_1 - F_{14}$= 55 **FP Calculation [3] FP=** UCF * TCF
UCF= Estimated count * Complexity Factor

Table. 4: Calculated UCF

(56) * (4) =	224
(20) * (5) =	100
(3) * (4) =	12
(29) * (10) =	290
(1) * (7) =	7
Total UCF=	**633**

TCF= 0.65 + 0.01 * ∑ F1 – F14
TCF=0.65 + 0.01 * 55
TCF = 1.2
So the **FP**= 633 * 1.22
FP=759

Line of Code Calculation [3]

Line of Codes for C# language application are **1821**
So the FP are 759 and LOC are 1821 for C# Labor rate per month is $1000 and Productivity is 20FP/month
Cost/FP = 1000/20
= $50
Total Bata project cost = 50*759
=$37950
Effort in person Month = total project cost / labor rate/month
= 379500/1000
Effort in person Month= 37.95

Table. 5 Comparison of FP, LOC and Cost

Case 1	**Initial application**	**After reverse engineering**
Function point	616	=initial project FP+ new FP developed = 616+143 =759
Line of code	1667	=initial project LOC+ new LOC developed =1667+154 =1821
Total cost	30798	=initial project cost + new components developed cost =30798+7950 =38,748
Number of developer	31	38

In this case function point, line of code and cost of reverse engineering case is greater than the initially developed application because new components are added.

Case#2

In this I developed a VB application for BATA, by converting C# application of BATA. Its complexity factor was considered complex because in the market no project was available in the VB in the market and calculate its FP and LOC as follows:

Table. 6: Complexity Value Table

	Estimated Count	**Complex Complexity**
No. of Inputs	56	6
No. of outputs	20	7
No. of Inquiries	3	6
No. of files	21	15
No. of Interfaces	1	10

The fourteen question values for this case $\sum F_1 - F_{14}$= 55 **FP Calculation [3] FP=** UCF * TCF
UCF= Estimated count * Complexity Factor

Tab. 7: Calculated UCF

(56) * (6) =	336
(20) * (7) =	270
(3) * (6) =	18
(21) * (15) =	315
(1) * (10) =	10
Total UCF=	**949**

TCF= 0.65 + 0.01 * ∑ F1 – F14
=0.65 + 0.01 * 55
TCF = 1.15
So the **FP**= 1049 * 1.15
FP=1091

Line of Code Calculation [1]

Line of Codes for VB language application are **1926** Labor rate per month is $1000 Productivity is 200FP/month
Cost/FP = 1000/20
= $50
Total Bata project cost = 50 * 1091
= $54,550
Effort in person Month = total project cost / labor rate/month
= 54550/1000
Effort in person Month= 54.550

Tab. 8: Comparison of FP, LOC and cost

Case 2	**Reverse Engineering**	**Forward Engineering**
Function point	759	1091
Line of code	1821	1926
Total cost	38,388	54,550
Number of Developers	38	54
Number of Errors & defects	15	7

RESULT AND DISCUSSION

A Calculating the result of two cases, in this analysis compared th both cases of Forward engineering and Reverse engineering. In Foforrward engineering application is developed in VB and in Rereverse engineering developed same interface application in C# an and Reverse engineering is better because function point and

linline of code and cost is less than the forward engineering and more error and defects was occurred in forward engineering that's why Reverse engineering g is better than the forward engineering and its shows the difference between these two Engineering.
FP evaluation result shown in form of pie chart and shown the ratio of percentage of FP effecting the FE and RE as follows:

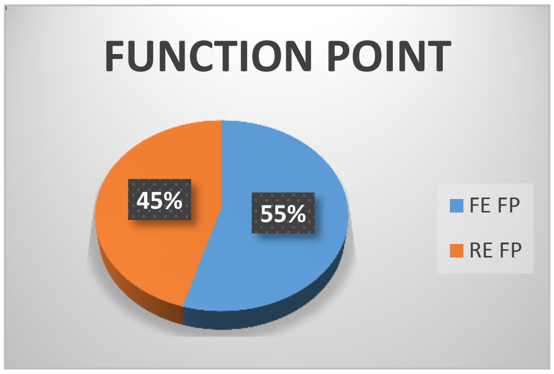

Fig. 2: FP evaluation Ratio

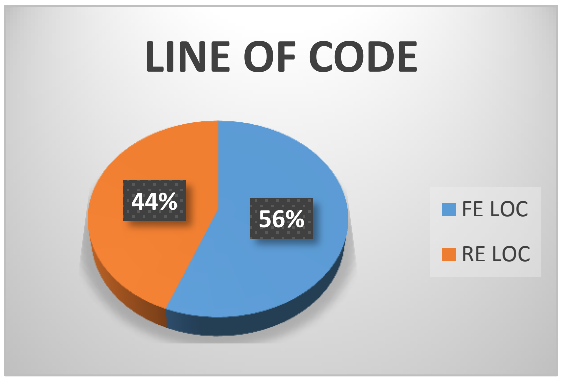

Fig. 3: LOC evaluation Ratio

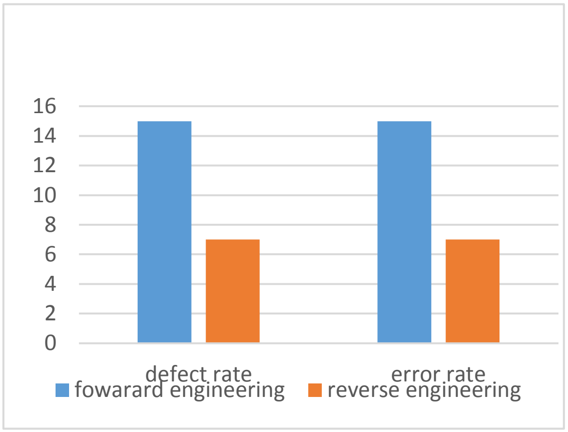

Fig. 4 : error and defect rate in F.E and R.E

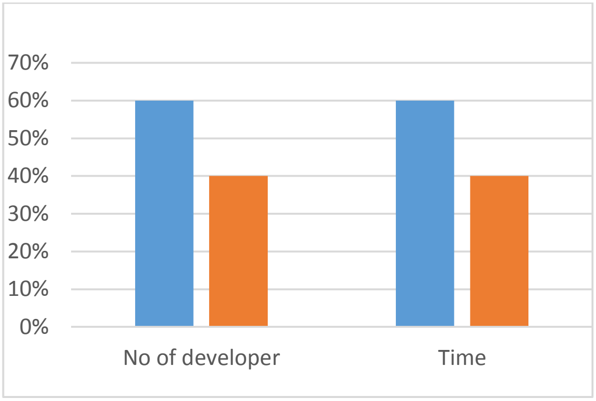

Fig. 5 No. of developer and Time in F.E nad R.E

3. CONCLUSION

In this study after seeing all these results we can say that the Reverse Engineering is better than Forward Engineering. In any organization we can calculate the project time, cost, person and all other things which effect the project productivity. Without a reliable sizing metric relative changes in productivity (Function Points per Work Month) or relative changes in quality (Defects per Function Point) cannot be calculated. If relative changes in productivity and quality can be calculated and plotted over time, then focus can be put upon an organizations strengths and weaknesses. Most important, any attempt to correct weaknesses can be measured for effectiveness.

4. REFERENCES

[1] Lee, N. Y. L. (2013). A theory of reverse engineering and its application to Boolean systems, (April), 37–41.
[2] Prof, A. (2013). A Review of Reverse ngineering Theories and Tools Ramandeep Singh, *2*(1), 35–38.
[3] Ying, W. (2013). Protocol reverse engineering through dynamic and static binary analysis, *20*(December), 75–79.
[4] Brunelière, H., Cabot, J., Dupé, G., & Madiot, F. (2014). MoDisco: A model driven reverse engineering framework. *Information and Software Technology*, *56*(8), 1012–1032.
[5] Jin, D., Cordy, J. R., & Dean, T. R. (2003). Transparent Reverse Engineering Tool Integration Using a Conceptual Transaction Adapter, (March), 399–408.
[6] Conejero, J. M., Rodríguez-Echeverría, R., Sánchez-Figueroa, F., Linaje, M., Preciado, J. C., & Clemente, P. J. (2013). Re-engineering legacy Web applications into RIAs by aligning modernization requirements, patterns and RIA features. *Journal of Systems and Software*, *86*(12), 2981–2994.
[7] Deursen, A. Van, & Burd, E. (2005). Software reverse engineering. *Journal of Systems and Software*, *77*(3), 209–211.
[8] Felfernig, a, & Salbrechter, a. (2004). Applying function point analysis to effort estimation in configurator development. *International Conference on Economic, Technical and Organisational Aspects of Product Configuration Systems, Kopenhagen, Denmark*, 109–119.

[9] Jeffery, D. R., Low, G. C., & Barnes, M. (1993). Comparison of function point counting techniques. *IEEE Transactions on Software Engineering*, *19*(5), 529–532.

[10] Kumar, V., & Pandey, S. (2013). Accurate Software Size Estimation Using the Updated Function Point Analysis Model. *International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 6, June 2013*, *2*(6), 1–3

[11] Lavazza, L., & Tosi, D. (2014). Using Function Point Analysis and COSMIC for Measuring the Functional Size of Real-Time and Embedded Software : a Comparison, *7*(1), 330–340.

[12] Orr, G., & Reeves, T. E. (2000). Function point counting: One program's experience. *Journal of Systems and Software*, *53*(3), 239–244.

[13] Sheetz, S. D., Henderson, D., & Wallace, L. (2009). Understanding developer and manager perceptions of function points and source lines of code. *Journal of Systems and Software*, *82*(9), 1540–1549.

[14] You, S., & Chen, L. (n.d.). Reverse and Forward Engineering of Frequency Control in Power Networks.

[15] Lee, J., Avgerinos, T., & Brumley, D. (2011). TIE: Principled Reverse Engineering of Types in Binary Programs. *Network and Distributed System Security*.

[16] Sandhu, R. (2013info Reverse Engineering Of Web Applications : An Emerging Trend, *4*(10), 569.

[17] Raja, V. (2008). Introduction to reverse engineering. *Reverse Engineering*, 1–10.

[18] Boehm, B., Abts, C., & Chulani, S. (2000). Software development cost estimation approaches. A survey. *Annals of Software Engineering*, *10*, 177–205

[19] Borade, J. G., & Khalkar, V. R. (2013). Software Project Effort and Cost Estimation Techniques. *International Journal of Advanced Research in Computer Science and Software Engineering*, *3*(8), 730–739.