CGC International Journal of Contemporary Technology and Research
ISSN: 2582-0486 (online) Vol.-6, Issue-1; DOI: 10.46860/cgcijctr.2024.04.10.370

A statistical model for estimating size and productivity ratio in SDLC

Anuj Kumar Gupta*
Department of Computer Science and Engineering, Chandigarh Group of Colleges, Landran, Mohali, PB, India
*Email ID: cgccoe.cse.anuj@gmail.com

Abstract: **During a software development life cycle, one has to estimate the effort and schedule required to produce a software unit. Estimates for The effort and schedule are derived from other measurements such as size and productivity Ratio. Size may be estimated using methods such as Function Point, WBS, LOC, etc., and the Productivity Ratio is expressed as Size / Time to complete the Unit of Software. Such estimates are static and do not consider real-time Parameters which cause variations from the estimate. In this paper, a statistical model for Size and Productivity Ratios is derived using Historical values after considering several factors influencing the size and the productivity ratio of produced software. Typically, an estimate of the effort and schedule is required to complete. A software work product an enhancement request or a software fix. We use Function Points or WBS to estimate the size of the software in consideration. We use baseline values for estimating productivity Ratios. Real-time values however differ from Baseline values as they are influenced by several factors. Here are some factors which would influence the Actual Size of the software Later on we will use Statistical techniques of Multivariable Parameter estimation and logistic Regression to derive run time equations of Size and Productivity Ratio.**

Indexed Terms*: SDLC, statistical model, size, productivity ratio*

.

I. INTRODUCTION TO SDLC

SDLC, or Software Development Life Cycle, is a structured approach to developing software applications. It encompasses a series of phases that guide the entire process from conception to deployment and maintenance. The typical phases include planning, analysis, design, implementation, testing, deployment, and maintenance. Each phase has its specific objectives, activities, and deliverables, ensuring that the software development process progresses systematically and efficiently. Adhering to SDLC helps in managing resources effectively, controlling costs, ensuring quality, and delivering software that meets stakeholders' requirements. It provides a framework for collaboration among developers, testers, project managers, and stakeholders, fostering communication and alignment throughout the development lifecycle. Overall, SDLC serves as a roadmap for delivering high-quality software solutions on time and within budget (See Figure 1).

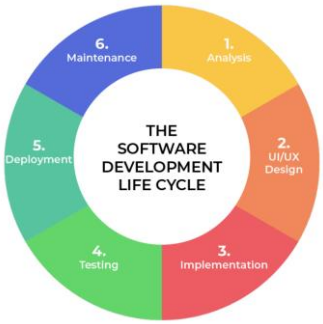

Fig 1: SDLC

There are several types of Software Development Life Cycle (SDLC) models, each with its unique approach to managing the software development process. Some common types of SDLC models include the waterfall Model, Agile Model, Iterative Model, Spiral Model, Incremental Model & Rapid Application Development Model. These are just a few examples of SDLC models, and there are variations and hybrids tailored to specific project requirements and organizational preferences. (See Figure 2).

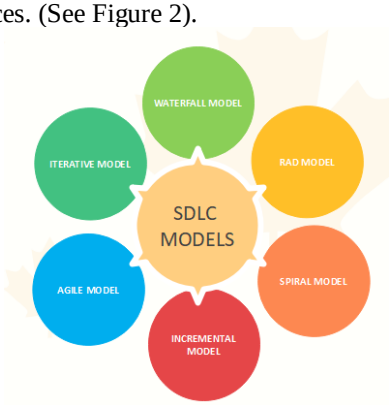

Fig 2: SDLC Models

Choosing the right SDLC model depends on factors such as project size, complexity, timeline, and stakeholder preferences.

II. FACTORS THAT INFLUENCE SIZE

The following are the key factors that influence the size of software [1, 2]:

A. Scope Creep: If there is a scope creep in the project, i.e., If the requirements phase has a deviation or defects, this would affect the actual size.

B. Code Review: This would have an effect on the size as a Peer review of the code can ensure that the minimum size is applied to the functions required to generate the code [3]

C. Developer Skill Level: The skill level of the developers would affect the size.

D. Usage of Model: If a model is used for estimating size it makes a lot of assumptions. For example, the

Function Point methodology is platform independent, but, sometimes there may be a loss of accuracy in predicting the actual Size when this factor is used for conversion from Function Point to LOC [4]

E. Object code: Use of Object code to nest functionality. If the size estimation Model uses linear supposition to add size to functions then Using Object code in the form of Dynamic Link Libraries etc., will reduce the size of the software [5].

Let us also take into account some factors, which do not have an impact and measure the co-relation with the size and the productivity ratio:

1. Highest degree obtained
2. Gender of the developer

III. REGRESSION ANALYSIS

Regression analysis is a statistical method used to study the relationship between a dependent variable and one or more independent variables. Its primary goal is to understand how the dependent variable changes as the independent variables vary. Regression analysis can be used for prediction, inference, and hypothesis testing. Here is a sample of tabulated values for the different factors taken into consideration. The analysis has been computed in Table 1, where the code size is in LOC, and the other values are in percentage [6].

TABLE 1: ACTUAL LOC AS A FUNCTION OF THE DIFFERENT FACTORS WHICH WOULD INFLUENCE THE SIZE

Actual Size	Skill Level	RSI	Code Review	Machine Automation
4,000.00	1.00	100.00	0.00	0.00
3,400.00	2.00	100.00	0.00	0.00
3,300.00	3.00	100.00	0.00	0.00
3,200.00	4.00	100.00	0.00	0.00
3,000.00	5.00	50.00	0.00	0.00
6,000.00	5.00	0.00	0.00	0.00
3,500.00	5.00	100.00	0.00	0.00
3,750.00	5.00	75.00	100.00	0.00
3,200.00	5.00	100.00	50.00	0.00
3,217.00	5.00	100.00	100.00	0.00
3,200.00	5.00	100.00	100.00	10.00
3,100.00	5.00	100.00	100.00	20.00
3,000.00	5.00	100.00	100.00	50.00

IV. REGRESSION EQUATION

The generation steps of a regression model equation typically involve several key steps:

i. Define the problem
ii. Collect data
iii. Explore data
iv. Choose a model
v. Specify the model
vi. Estimate coefficients
vii. Evaluate the model
viii. Interpret the results
ix. Use the model for prediction
x. Validate the model

The regression was run using an Excel tool assuming linear relations [7, 8] and the result obtained was:

*Actual Size = -4.39*Skill Level + 21.64*RSI + -0.38*Code Review + -11.53*Machine Automation + 1378.40 (+/- 1569.33)* ---(i)

For the same example, the Predicted Size using a model such as FP is 3500 LOC.

Equation Parameters

R Square	0.9970
Adjusted R Square	0.9951
Standard Error	2.4460
F - Statistic	506.8929

Multiple Regression Equation

	Coefficients	Standard Error
Intercept	-20.128	7.056
Skill Level	20.000	1.730
Reusable Code	14.231	0.480

Independent Analysis

R Squared	Gradient	Intercept
13.15%	20.00	8.33
86.55%	14.23	59.87

Auto Correlation Statistics

Dl = 1.08
Du = 1.36
DW-Stat
2.25
2.53

V. SOME FACTORS THAT INFLUENCE THE PRODUCTIVITY RATIO ARE

A. REUSABLE CODE: If the entire application can be automatically engineered this would make the application skill-independent and would enhance productivity [9, 10].

B. SKILL LEVEL: An experienced developer would produce better than an experienced developer.

C. SHIFT TIMINGS: Daytime work is more productive than nighttime work.

Some sample values for the different parameters under which the real-time Productivity Ratio would vary were taken into consideration [11, 12, 13].

Skill Level	Reusability	Ratio
4.00	0.00	60.00
3.00	0.00	40.00
4.00	3.00	100.00
4.00	4.00	120.00
4.00	5.00	130.00

The equation obtained due to multi-variable regression analysis using an Excel tool.

*Productivity Ratio = 20.00*Skill Level + 14.23*Reusable code + -20.13 (+/- 2.45) ---(ii)*

The baseline value which was taken in this company was 80 tested lines per day. But in reality, due to factors of influence and positive observed correlation, the Actual productivity ratio varies as the linear equation suggests [14, 15].

VI. CONCLUSION

Projects derive estimates using baseline values. But frequently these baseline values are correlated with other measures. In this paper, two measurements Size and Productivity ratio, which are used to derive measurements such as Effort and Schedule, have been taken into consideration. Using sample values, a real-time regression equation was derived to show that the actual values vary concerning some parameters.

REFERENCES

[1]. Sudhakar G.P, Farooq A, Patnaik S (2012) Measuring productivity of software development team. Serbian Journal of Management 7: 65–75.

[2]. Nwelih E, Amadin I.F (2008) Modeling software reuse in traditional productivity model. Asian Journal of Information Technology 7: 484–488.

[3]. M. Azzeh, A. B. Nassif, and S. Banitaan, “Comparative analysis of soft computing techniques for predicting software effort based use case points,” IET Software, vol. 12, no. 1, pp. 19–29, 2018.

[4]. M. Hosni, A. Idri, A. Abran, and A. B. Nassif, “On the value of parameter tuning in heterogeneous ensembles effort estimation,” Soft Computing, vol. 22, no. 18, pp. 5977–6010, 2017.

[5]. Bluemke, Ilona, and Agnieszka Malanowska. "Software testing effort estimation and related problems: A systematic literature review." ACM Computing Surveys (CSUR) 54, no. 3 (2021): 1-38.

[6]. Mustafa, Emtinan, and Rasha Osman. "An Analysis of the Inclusion of Environmental Cost Factors in Software Cost Estimation Datasets." In 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), pp. 623-630. IEEE, 2018.

[7]. Boehm, B.W.; Abts, C.; Chulani, S. Software development cost estimation approaches—A survey.Ann. Softw. Eng. 2000, 10, 177–205.

[8]. Shanky Goyal, Navleen Kaur, Sachin Majithia, Software Security: Role in SDLC, CGC International Journal of Contemporary Technology and Research, ISSN: 2582-0486 (online) Vol.-3, Issue-2, pp. 205-210, 2021.

[9]. Vera, T.; Ochoa, S.F.; Peroich, D. Survey of Software Development Effort Estimation Taxonomies; Computer Science Department, University of Chile: Santiago, Chile, 2017.

[10]. Basha, S.; Dhavachelvan, P. Analysis of Empirical Software Effort Estimation Models. Int. J. Comput. Sci. Inf. Secur. 2010, 7, 69–77

[11]. Booch G., “The History of Software Engineering,” IEEE Softw., vol. 35, no. 5, pp. 108–114, 2018,

[12]. Kneuper R., “Sixty years of software development life cycle models,” IEEE Ann. Hist. Comput., vol. 39, no. 3, pp. 41–54, 2017

[13]. Kaur, R., Gupta, A. K., “Online Banking Customers Information Security Awareness Model”, Proceedings of 9th International Conference on Engineering and Technology ICET – 2016, pp: 1-5, ISBN: 978-81-925978-4-3, Melbourne, Australia, 19 March 2016

[14]. Arora R. and Arora N., “Analysis of SDLC Models,” Int. J. Curr. Eng. Technol., vol. 6, no. 1, pp. 2277–4106, 2016.

[15]. Sarker I. H., Faruque F., Hossen U., and Rahman A., “A survey of software development process models in software engineering,” Int. J. Softw. Eng. its Appl., vol. 9, no. 11, pp. 55–70, 2015.