Grenze International Journal of Engineering and Technology, Jan 2020

Estimating Software Project Development Effort using Simulation

Srinivasa Gopal[1] and Lakshmi Narasimhan[2]
[1-2]Ramanujan Society for Academic Research and Promotion of Science, Chennai, India
Email: srinivasa32@hotmail.com, narasimhan68@hotmail.com

***Abstract*—An assessment of the software engineering effort /schedule/expected no of defects etc., are required by Project Managers, Customers, Developers, Systems analysts amongst other stake holders. Poor Estimation during the early stages of the software development life cycle leads to schedule, cost overruns. There are several categories of estimation models ranging from Work Break Down Structure, Parametric Estimation and size-based estimation models which exist for predicting the software development effort. Estimates from many of these models do not converge to actual values. Moreover, many of these models provide an estimate of only the coding phase. Due to volatility of requirements and also due to varying problem and solution complexities , varying productivity ratios, varying size and complexity of code, varying testing scenarios the actual value of effort required to complete the requirements phase or effort required to complete the design phase vary considerably depending upon different conditions encountered during the execution of the project. Also, during project execution, one may encounter defects and would spend time removing these defects. Surplus effort may be encountered while removing the defect during the test phase instead of an inspection phase conducted during the Requirements development phase. Simulation models of expected software development effort have been developed in this paper. It is shown that the expected effort varies considerably and cannot be estimated using a single point estimate or even a estimate within a range.**

***Index Terms*— Software Engineering Effort Estimation, Simulation, Requirements Volatility, Problem Complexity, Solution Complexity, Defect occurrence, Defect Removal.**

I. INTRODUCTION

An estimate of the Software Engineering effort is essential to plan for Manpower, expected cost of the project, Project Scheduling amongst many other reasons as well. Many of the estimation models that are in vogue provide for point estimates. However, all requirements are not uniform, each one varies with a different level of problem /solution complexities. Also productivity ratio may vary between different team members , may also vary between different times of the day, also the requirements may be volatile may be frequently modified or added or deleted etc., There may also be a rapid occurrence of defects during the coding phase which may take additional effort to be removed during Test and UAT phases. Point estimates in such cases do not serve the purpose and one may hence see large deviation between actual and estimated effort using such models. In this paper we have developed many simulation models and we have considered many real time situations where each requirement that is taken up for further processing has a unique level of

Grenze ID: 01.GIJET.6.1.1

problem and solution complexity. We have also considered the effect of volatility of requirements and also the effect of removing defects via an Inspection process which occurs early in the life cycle or which occurs later in the life cycle during the testing phase. In all these cases it shown via Histograms the extent to which the effort can vary depending upon unique circumstances within the project. So, it is extremely essential to capture variability of the software development life cycle using Simulation Models.

II. Objective of this paper

Analytical models do not provide an accurate estimate when requirements are volatile or when the solution complexity varies between each unit requirement considerably due to dynamics of problem or solution complexity There is also the effectiveness of reviews and tests which go hand in hand in prolonging or shortening the overall effort required to complete a software work product. Many of these parametric estimation models or expert estimation models or size-based estimation models provide an effort for only the coding phase. Effort during the coding phase alone can vary depending upon several situations such as varying productivity ratios between different developers or during different times of the day or Defect removal efficiency. Extrapolating the effort required for other phases using a industry standard ratio is not effective when the requirements are volatile or when the testing /review is not effective. Hence it is essential to simulate the project life cycle from conception to delivery and User Acceptance Testing so that some key or critical factors of variation can be captured during the estimation process itself. The objective of this paper is to hence develop a variety of Simulation Models bearing in mind the dynamics of the software development process and compare it to estimates based on some of the widely used estimation models.

III. Literature Review

[1] The authors state that the actual cost of the project differs from the estimated planned cost due to Requirements volatility and effort spillage on defect correction. The project cost over runs and its variability are obtained using Simulation and fictional software project management data.
[2] The authors stress the importance of obtaining near accurate estimates for large sized projects. Also, different prediction techniques have been compared and have been rated with respect to different attributes of the project using simulation. It is hence concluded that there is a lot of variability depending upon the quality of the data set. It is also recommended to use an appropriate estimation technique depending upon the attributes of the project.
[3] In this paper the authors use a simulation method to generate different types of application scenarios which can occur during development which are then estimated using COCOMO II.
[4] Is a white paper which suggests that the software engineering community could exploit simulation to much greater advantage. Such areas in software engineering include requirements specification, process improvement, product line practices.
[5] In this paper an analogy-based estimation is combined with Simulation to identify the parameters for building effective similarity measures.
[6] In this paper a probability distribution of project related risks is arrived at using Monte Carlo Simulation. This is accomplished using Cobra a cost estimation method that applies Simulation.
[7] This paper describes the Software-Engineering Process Simulation (SEPS) model d SEPS is a dynamic simulation model of the software project-development process. It uses the feedback principles to simulate the dynamic interactions among various software life-cycle development activities and management decision-making processes. The model is designed to be a planning tool to examine trade-offs of cost, schedule, and functionality, and to test the implications of different managerial policies on a project's outcome. Furthermore, SEPS will enable software managers to gain a better understanding of the dynamics of software project development and perform post-mortem assessments.
[8] This paper attempts to integrate six sigma and simulation to define, analyse, measure and predict various elements of software development (such as cost, schedule, defects) that influence software quality, Simulation results provide qualitative and quantitative suggestions on the ways to change the software process to achieve six sigma quality.
[9] is a book chapter which aims to raise awareness about the usefulness and importance of simulation in support of software engineering. Simulation is applied in many critical engineering areas and enables one to address issues before they become problems. Simulation – in particular process simulation – is a state-of-the-art technology to analyse process behaviour, risks and complex systems with their inherent uncertainties.

Simulation provides insights into the designs of development processes and projects before significant time and cost has been invested and can be of great benefit in support of training. The systematic combination of simulation methods with empirical research has the potential for becoming a powerful tool in applied software engineering research. The creation of virtual software engineering laboratories helps to allocate available resources of both industry and academia more effectively.

IV. Simulating Volatility Or Dynamics Of The Software Development Process

The effort for a project of size 50 Function Points estimated using different estimation models is given below. It is assumed that the productivity ratio is 20 loc /day.
1FP = 50 LOC
50 FP = 2.5KLOC

TABLE **I**: EFFORT ESTIMATES USING DIFFERENT ESTIMATION TECHNIQUES

Estimation Technique/Model	**Estimate**
Function Point	125 hours
COCOMO	1800 hours
WBS	1200 hours

In the below sections different attributes of the software development life cycle will be varied, and the expected effort will be expressed as a histogram of values

A. Effort variation with respect to Problem and Solution Complexity

In the above example the project has many unit requirements. Each requirement is of a certain complexity level viz Problem and Solution. Problem complexity refers to how complex the specification is, how complex it is to infer or how complex it is to write the SRS based on the requirement provided. Solution complexity is defined as the level of complexity involved in constructing the solution or the difficulty of coding the software feature. Table 2 below shows different combinations of variations in requirements/associated coding phases.

TABLE II: DIFFERENT POSSIBILITIES OF SOLUTION /PROBLEM COMPLEXITY PER REQUIREMENT

Problem Complexity	High	High	Low	Low
Solution Complexity	High	Low	High	Low

Since a project consists of many requirements and during software development any of the combination of problem and solution complexities can occur during requirements development/analysis/design and coding phases, this aspect has been simulated using the input parameters shown in Table 3. Here the table shows that each requirement can have a problem complexity ranging between 1 and 5, solution complexity ranging between 1 and 5, estimated size is assumed to vary between 1 and 5 Function Points. The effort for Analysis depends upon the Problem complexity, effort for design is influenced predominantly by the size of the software, the effort for Coding depending upon both the size of the software and how complex the solution is. The effort for Testing has been approximated as 2/3rds the coding effort. The input to the simulation model is shown in the table below.

TABLE III: INPUT VARS FOR SIMULATION OF EFFORT VARIABILITY WRT PROBLEM AND SOLUTION COMPLEXITY

Parameter	**Range of Values**	**Distribution Modelled as**
Problem Complexity	1 to 5	Uniform
Solution Complexity	1 to 5	Uniform
Function Point	1 to 5	Uniform
Effort for Analysis	8 to 40	8 * Problem Complexity
Effort for Design	32	Taken as constant
Effort for Coding	16-400 hrs.	16*Size in FP *Solution Complexity
Effort for Testing	10.67 -267 hrs.	2/3rds of the coding effort

A sample of simulated data is shown below in Table 4. There were 1048575 observations totally generated using the input distributions/functions for different columns output parameters which are shown in Table 3.

TABLE IV : SAMPLE SIMULATED OUTPUT VARIATION PROBLEM AND SOLUTION COMPLEXITY

Problem Complexity	Solution Complexity	Function Point	Effort for Analysis	Effort for Design	Effort for Coding	Effort for Testing	Total Effort
5	2	5	40	32	160	106.667	350.67
4	4	2	32	32	128	85.3333	287.33
2	2	3	16	32	96	64	215
2	3	3	16	32	144	96	296
3	4	5	24	32	320	213.333	601.33
2	4	3	16	32	192	128	377
2	2	4	16	32	128	85.3333	269.33
3	5	5	24	32	400	266.667	735.67
5	4	1	40	32	64	42.6667	188.67
1	4	1	8	32	64	42.6667	152.67
3	5	5	24	32	400	266.667	735.67
4	1	2	32	32	32	21.3333	124.33
5	5	3	40	32	240	160	485
5	5	2	40	32	160	106.667	350.67
4	5	3	32	32	240	160	476
5	1	2	40	32	32	21.3333	133.33
4	2	5	32	32	160	106.667	341.67
5	3	4	40	32	192	128	404
3	4	3	24	32	192	128	386

Table 4 contains the simulated output showing the variation of effort with respect to different input parameters shown in Table 3. Table 4 shows that the estimated effort varies for each associated complexity of requirement//complexity of code /size of each requirement. Table 4 shows only a snapshot of the entire simulated output. The simulated output contains 1048575 records. The summary statistics /test of normality of the entire simulated output is presented in Table 5.

TABLE V : TEST OF NORMALITY OF SIMULATED OUTPUT

Input Parameter	Number of Observations	Mean	Standard Deviation	Skew	Kurtosis
Problem Complexity	**1048575**	2.998947143	1.413960266	0.000104453	-1.29907571
Solution Complexity	**1048575**	2.998864173	1.413829011	0.000969534	-1.29928212
Number of FP/Requirement	**1048575**	3.000444413	1.414575573	0.00044964	-1.30091386

Table 5 also shows a very small value for the skew indicating that the output distribution tends to normality. The entire output and the associated variations of the estimated effort is shown as a Histogram.
Figure 1 shows the histogram of all 1048575 observations simulated for different values of Problem /Solution Complexity and Size of each requirement. A snapshot of the simulated output of Figure 1 is shown in Table 7. In addition, the following summary statistics for the estimated effort is stated below.

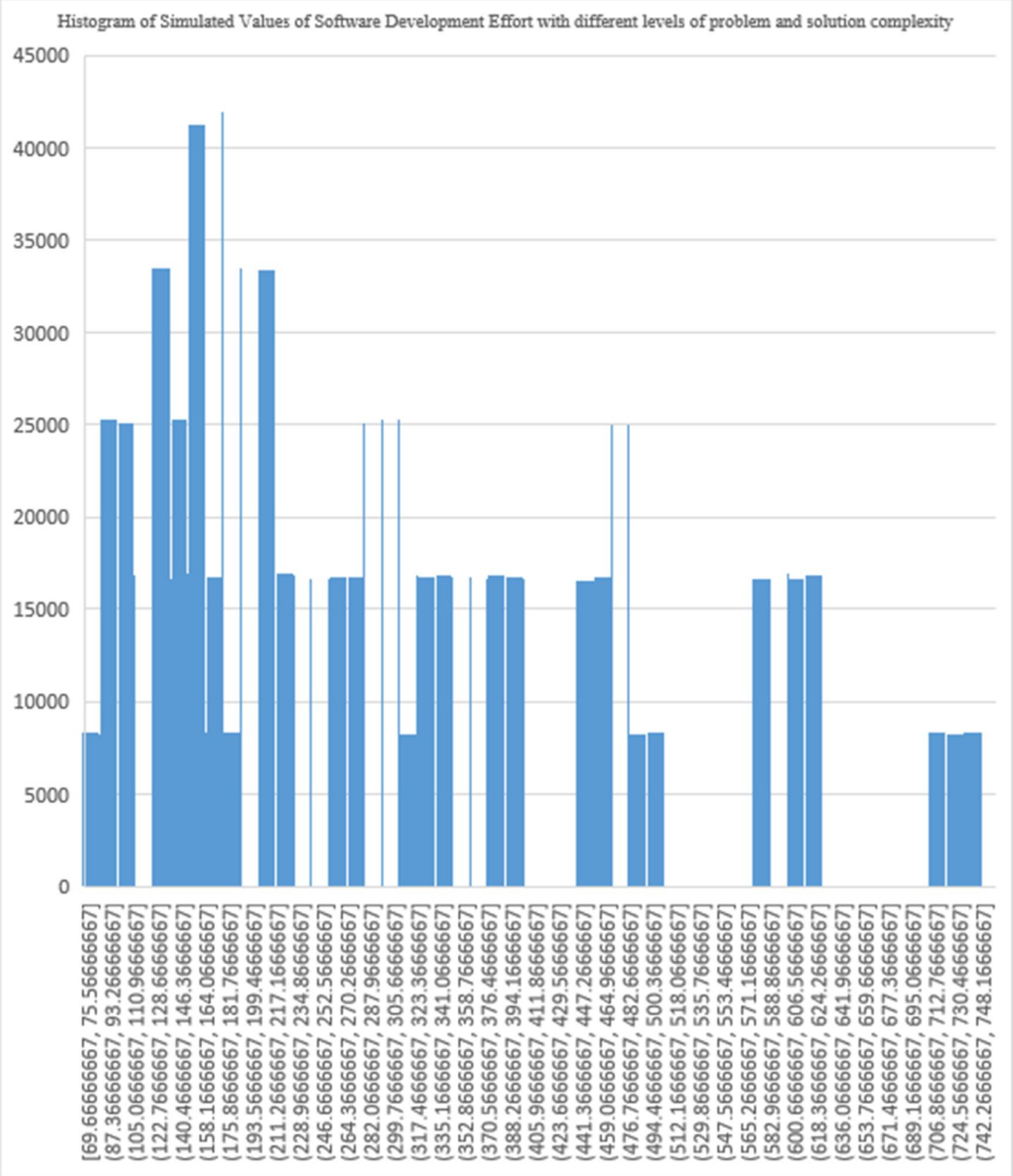

Figure 1: Histogram of Simulated Values wrt varying levels of Problem/Solution Complexity per requirement

Mode: 224 hours, Mean: 304.8886924, Maximum Number of Observations (42,004) occurs between 176-182 hours. As each unit requirement was simulated for unit effort and as the mean size of all the requirements is 3 as per Table 4, the total effort for the project of 50 FP was obtained by multiplying the expected value or the median or the weighted mean of the range of values shown in the histogram by a factor of (50/3). The estimated effort can hence vary between 2933 hours to 3033 hours for a majority of observations depending upon different problem and solution complexities with expected value around 1800 hours. Simulation hence provides a very wide range of values compared to estimates obtained using other estimation techniques as shown in Table 1. It is to be noted and inferred here that the other estimation models provide estimates that do not account for such situations when each requirement can vary considerably from the point of complexity of the specification or and the complexity of the solution.

B. Simulating the occurrence of Volatility of Requirements

TABLE VI: INPUT PARAMETERS FOR SIMULATION MODEL - REQUIREMENTS VOLATILITY

Scenario of Volatility	Phase of Occurrence	Impact
Requirement Added or Modified or Deleted	Requirements Development	No Impact
Requirement Added or Modified or Deleted	Analysis	No Impact
Requirement Added or Modified or Deleted	Coding Phase	Existing code has to be Modified or freshly added or code has to be deleted.
Requirement Added or Modified or Deleted	Testing Phase	Existing code has to be Modified or freshly added or code has to be deleted.
Requirement Added or Modified or Deleted	User Acceptance Phase	Existing code has to be Modified or freshly added or code has to be deleted.

Table 6 shows the input parameters for Simulating the occurrence of volatility of requirements. It is assumed that if volatility of requirements occurs in the Requirements Development Phase or Analysis phase it does not have any impact in the effort increase. But if it occurs during the Coding /Testing or UAT phases then an increase in effort will occur.

TABLE VII: SIMULATED OUTPUT FOR TABLE 6(SNAPSHOT)

No of Requirements	Avg No of FP/Req	Effort/FP	Estimated Effort	Volatility in Requirements Phase	Additional Effort	Volatility of Design Phase	Additional Effort
21	1	21	441	1.90476	59.85	1.525	34.72875
41	2	11	902	1.80488	108.9	1	0
30	5	11	1650	1.73333	181.5	1	0
42	5	10	2100	1.66667	210	1	0
31	3	22	2046	1.74194	227.7	1.57407	176.1833
47	5	17	3995	1.57447	344.25	1.63514	380.6047
44	4	14	2464	1.61364	226.8	1	0
28	4	20	2240	1.17857	60	1.84848	285.0909
48	1	14	672	1.1875	18.9	1	0

TABLE VIII: SIMULATION TABLE OF EFFORT OVERLAY DURING OCCURRENCE OF VOLATILITY OF CODING AND TEST PHASES

Volatility of Coding Phase	Additional Effort3	Volatility of Test Phase	Additional Effort4	RSI	%age Effort Increased
1.6557377	115.672131	1.6039604	79.90396	216.9403	65.794749
1	0	2	270.6	114.7049	42.073171
1	0	1	0	186.2333	11
2	840	2	630	1056.667	80
1	0	2	613.8	410.1993	49.740143
1	0	1	0	730.0643	18.144048
1	0	2	739.2	232.4136	39.204545
1.54098361	484.721311	1.6489362	436.08511	836.0292	56.513274
1	0	2	201.6	24.0875	32.8125
1.66129032	99.9870968	1	0	189.1544	48.420254

Table 7 shows that Volatility occurring in all the different phases. It also shows that the simulation has been done varying the No of Requirements, Avg No of FP/Req, Effort/FP (Productivity ratio). Table 7 and Table 9

also show that different extents of Volatility have been taken into consideration in each phase. The additional effort that is required to be expended has been stated in columns next to the volatility metrics.
In Table 8 the Total RSI (Requirements Stability Index) and the percentage increased effort due to the conditions causing the volatility have been shown.

TABLE IX: INPUT VARS - SUMMARY STATISTICS AND TEST OF NORMALITY

Input/Simulation Stats	RSI of Requirements Phase	No of Req	Size/Req	Estimated Effort	RSI of Req Phase	RSI of Design Phase	RSI of Coding Phase	RSI of Testing Phase	%Age increased Effort
No of Runs	1048575	1048575	1048575	1048575	1048575	1048575	1048575	1048575	1048575
Mean	1.499775071	34.99	16.0081	1680.304	1.499775	1.35506	1.40302	1.403020	42.57160292
Std Dev	0.297300177	8.942	4.89831	1094.912	0.297300	0.36703	0.42565	0.425657	20.88054649
Skew	-6.65305E-05	0.000	-0.00234	1.020838	-6.65305E-05	0.23476	0.31062	0.310628	-0.194958921
Kurt	-1.2017	-1.201	-1.20986	0.719459	-1.201786	-1.62055	-1.59907	-1.599071	-0.657177901

Table 9 shows the summary statistics for the Input parameters of the Simulation Model.

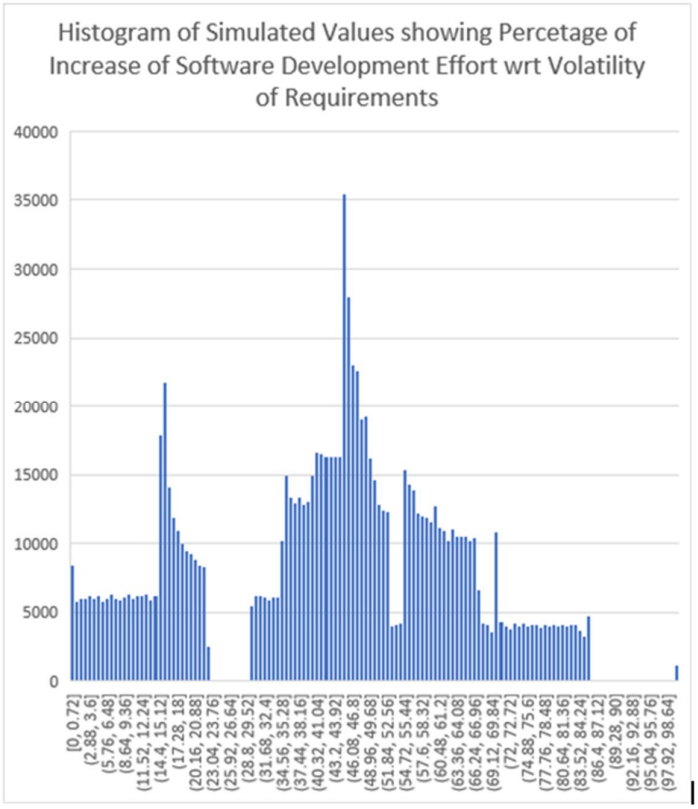

Figure 2: Percentage of Increased Effort for each simulation run of variable input parameters as Table 7 and Table 8

Figure 2 shows a distribution of values of Percent Increased Effort per Simulation Run with respect to varying input conditions. The interval with the highest number of occurrences (35000 times out of 1048575

trials) is between 43.2 and 43.92 percent increase of Effort due to Volatility of requirements. The mean of all the observations in Table 10 is 42.57. The mode is 70.

C. Simulating Effort due to variable rate of occurrence of defects and removal of defects

TABLE X: DEFECT OCCURRENCE AND REMOVAL

Phase	**Probability of defect occurrence**	**Defect Type (Major, Minor, Trivial)**
Requirements	0.6	Major, Minor, Trivial
Design	0.3	Major, Minor, Trivial
Testing	0.1	Major, Minor, Trivial

When defects occur, they are first detected, later on removed. Very simply defects are identified during requirements phase itself via an Inspection process or systematically during a Test process in the testing phase. There can be other modes of defect removal, but in order to model the variability of the system only these two modes are being considered. Table 10 shows that Defect Types can be Major, Minor or Trivial and can occur in the Requirements/Design or Test Phase with a probability of occurrence as shown above.

TABLE XI: PROBABILITY OF OCCURRENCE OF DIFFERENT TYPES OF DEFECTS

Defect Type	Probability of Occurrence
Major	0.2
Minor	0.5
Trivial	0.3

Table 11 shows the probability of Occurrence of Different types of Defects. Defects are known to have a higher probability of occurrence during the Requirements phase. Here the defect can be detected via an Inspection process when it occurs. But many times, the inspection process is not conducted, and the defect is detected during the Test Phase. The defect seeping in during the requirements phase can be detected via an Inspection process if the problem is sufficiently easy to understand or also if the problem is hard to understand but there is sufficient human /automated expertise available to detect and remove the defect.
Alternately the defect can be detected and removed during the Testing phase in which case the product feature has already been coded. In this case when the defect is detected during the test phase after coding is complete, it can be resolved via a combination of factors involving human skill and also how complex the code used is.
This is captured in the Simulation Model as shown in Table 12.

TABLE XII: PARAMETERS TO SIMULATE REMOVAL OF DEFECTS

Defect Type	**Defect Removed via Inspection**	**Defect Removed via Testing**	**Effort Required – Modelled as**
Trivial	YES		Uniform random number between 0 and 2
Minor	YES		Uniform random number between 1 and 8
Major	YES		Uniform random number between 1 and 16
Trivial		YES	Uniform random number between 0 and 4
Minor		YES	Uniform random number between 2 and 16
Major		YES	Uniform random number between 2 and 32

Table 13 below shows the outcome after simulating for effort required for removing all the defects via Inspection, Testing or a combination of Inspection and Testing.

TABLE XIII : MEAN DEFECT REMOVAL EFFORT UNDER DIFFERENT INSPECTION/TESTING RATIOS

Percent Defect removed via Inspection /Testing	Effort
100 percent removed via Inspection	4.30381
60 percent removed via Inspection	6.014812
40 percent removed via Inspection	6.879689
20 percent removed via Inspection	7.736111
0 percent Inspection, 100 percent Test	8.594696

D. Phase Wise Estimation of Effort

Estimation models provide a direct estimate of only the coding phase. The Effort for other phases is derived as a prorated ratio of the estimated effort in the coding phase. This is shown in Figure 8. (Ratio Source www.softwaremetrics.com).

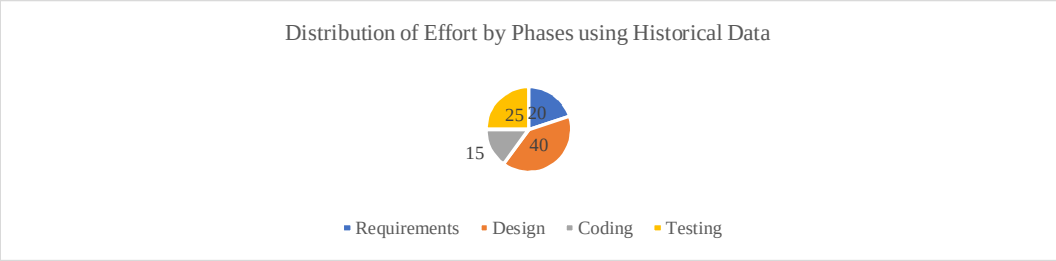

Figure 3 : Distribution of Effort by Phases - Historical Data

Table 14 shows that the percent of effort expended during different phases varies considerably due to the dynamics of the software development life cycle. The values of percent effort by different phases has been constructed using Simulation data of Table 14 instead of a fixed ratio. This is shown in Figure 4.

TABLE XIV: PHASE WISE DISTRIBUTION OF EFFORT DERIVED FROM DATA TABLE OF FIGURE6

Percent of Analysis	Percent of Design	Percent of Coding	Percent of Testing
19.48051948	15.58441558	38.96103896	25.97402597
10.34482759	20.68965517	41.37931034	27.5862069
8.571428571	34.28571429	34.28571429	22.85714286
21.42857143	42.85714286	21.42857143	14.28571429
15.78947368	31.57894737	31.57894737	21.05263158
17.24137931	13.79310345	41.37931034	27.5862069
5.084745763	10.16949153	50.84745763	33.89830508
14.01869159	11.21495327	44.85981308	29.90654206
22.3880597	17.91044776	35.82089552	23.88059701
5.454545455	21.81818182	43.63636364	29.09090909
10.34482759	20.68965517	41.37931034	27.5862069
1.818181818	7.272727273	54.54545455	36.36363636
6.607929515	5.286343612	52.86343612	35.24229075
35.29411765	35.29411765	17.64705882	11.76470588
8.021390374	6.417112299	51.3368984	34.22459893
4	16	48	32

16.21621622	16.21621622	40.54054054	27.02702703
22.3880597	17.91044776	35.82089552	23.88059701
22.3880597	17.91044776	35.82089552	23.88059701
2.608695652	10.43478261	52.17391304	34.7826087
7.692307692	15.38461538	46.15384615	30.76923077

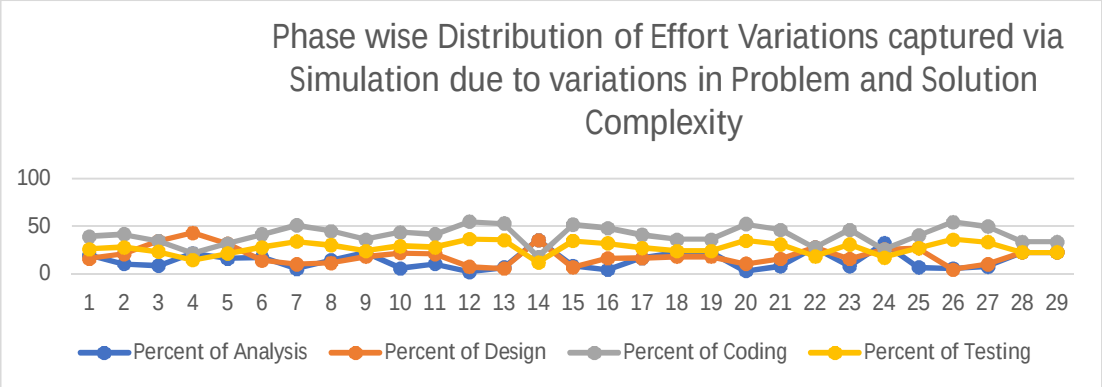

Figure 4:Phase wise distribution of Effort derived from data table of Figure 2

V. About us

Ramanujan Society for Academic research and promotion of science is a non-profit registered society for promotion of academic research, registered under section 10 of the Tamil Nadu societies act with registration no SRG/Chennai South/102/2019.

References

[1] Nonaka M., Zhu L., Babar M.A., Staples M. (2007) Project Cost Overrun Simulation in Software Product Line Development. In: Münch J., Abrahamsson P. (eds) Product-Focused Software Process Improvement. PROFES 2007. Lecture Notes in Computer Science, vol 4589. Springer, Berlin, Heidelberg.

[2] M. Shepperd and G. Kadoda, "Comparing software prediction techniques using simulation," in *IEEE Transactions on Software Engineering*, vol. 27, no.1, pp. 1014-1022, Nov. 2001.doi: 10.1109/32.965341kData engineering},URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=965341&isnumber=20846K. Elissa, "Title of paper if known," unpublished.

[3] Chukwunwike, Chiamaka & Onyesolu, Moses & E Osuagwu, O & G Onwurah, C & Uduiguomen, Usifoh. (2011). SIMULATING SOFTWARE COST ESTIMATION USING CONSTRUCTIVE COST MODEL (COCOMO) II.

[4] Simulation: An Enabling Technology in Software Engineering Alan M. Christie Software Engineering Institute

[5] Angelis, L. & Stamelos, I. Empirical Software Engineering (2000) https://doi.org/10.1023/A:1009897800559

[6] Michael Kläs, Adam Trendowicz, Axel Wickenkamp, Jürgen Münch, Nahomi Kikuchi, Yasushi Ishigai, Chapter 4 The Use of Simulation Techniques for Hybrid Software Cost Estimation and Risk Analysis, Advances in Computers, Elsevier, Volume 74,2008, Pages 115-174, ISSN 0065-2458, ISBN 9780123744265, https://doi.org/10.1016/S0065-2458(08)00604-9.

[7] Chi Y. Lin, Tarek Abdel-Hamid, Joseph S. Sherif,Software-Engineering Process Simulation model (SEPS), Journal of Systems and Software,Volume 38, Issue 3,1997,Pages 263-277,ISSN 0164-1212,https://doi.org/10.1016/S0164-1212(96)00156 (http://www.sciencedirect.com/science/article/pii/S0164121296001562)

[8] Rupa Mahanti, Jiju Antony, (2005) "Confluence of six sigma, simulation and software development", Managerial Auditing Journal, Vol. 20 Issue: 7, pp.739-762,https://doi.org/10.1108/02686900510611267

[9] Müller M., Pfahl D. (2008) Simulation Methods. In: Shull F., Singer J., Sjøberg D.I.K. (eds) Guide to Advanced Empirical Software Engineering. Springer, London.