ISSN No. 0976-5697

Volume 2, No. 6, Nov-Dec 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Simulator for Identifying the Importance of Software Reuse in Cost Estimation Model

Mr. Aman Kaushik*
Deptt. Computer Science Engineering
University Institute of Engineering & Technology, KUK
Kurukshetra, India
er.amankaushik@gmail.com

Abstract: Software reuse is the process of implementing or updating software systems using existing software assets. Assets may be designs, requirements, test cases, architectures etc. Idea behind code reuse is that a partial or complete computer program written at one time can be written into another program at later time. In this paper we use the concept of reuseability and implement with the help of simulator, which calculate the efforts of different type of project with and without reuseability. Simulator also calculate the mean relative error between original effort and calculated efforts. To achieve all these goals,we implement the simulator in high level language. Purpose of simulation is to shed light on the underlying mechanisms that control the behavior of a system. The simulator will calculate total efforts for the projects. On the basis of this estimation reuse manager will decide how to allocate resources for different phases, how to decide work within a phase

Keywords: Random number, Reusable Assets, Simulator , Box Muller transformation, simulation.

I. INTRODUCTION

Reuse is to use an item more than once. Reuse is the application of existing solution to new problems. Reuse can reduce the time spent in creating solutions by avoiding duplicated efforts. In software engineering the concept of reuse has been explored and has been reported to be very beneficial. This includes conventional reuse where the item is used again for the same function and new-life reuse where it is used for a new function. Software reuse is the process of implementing or updating software systems using existing software assets. Software asset is simply another term for source code. Software assets, or components, include all software products, from requirements and proposals, to specifications and designs, to user manuals and test suites.

Anything that is produced from a software development effort can potentially be reused [1]. Organizations which face the difficulties and costs associated with the development of software have turned to the reuse of existing software or using commercial off-the-shelf (COTS) software as an option. Reuse, whether involving home-grown or COTS components, certainly promises lower cost, better quality, a decrease in risk, and the potential for a less stressful development process. Many such efforts succeed, but the promises of decreased cost and risk are not always realized. Requirements, algorithms, functions, business rules, architecture, source code, test cases, input data, and scripts can all be reused. Architecture is a key for reuse.

II. REUSE BENEFITS

A. Increased Dependability:

Reused software that has been tried and tested in working system should be more dependable than new software. The initial use of the software reveals any design and implementation faults. These are then fixed, thus reducing the number of failures when the software is reused [2].

B. Reduced Process Risk:

If software exists, there is less uncertainty in the costs of reusing that software than in the cost of development. This is an important factor for project management as it reduces the margin of error in project cost estimation. This is particularly true when relatively large software components such as sub system reused.

C. Effective Use Specialists:

Instead of application specialists doing the same work on different project, these specialists can develop reusable software that encapsulate their knowledge.

D. Accelerated Development:

Bringing system to market as early as possible is often more important than overall development costs. Reusing software can speed up system production because both development and validation time should be reduced.

Software products are expensive. Software project managers are worried about the high cost of software development and are desperately look for ways to cut development cost. A possible way to reduce development cost is to reuse parts from previously developed software. In addition to reduced development cost and time, reuse also leads to higher quality of the developed products since the reusable components are ensured to have high quality [3].

III. SIMULATION

Simulation is defined as the process of designing a model of a real system and conducting experiments with this model for the purpose either of understanding the behavior of the system or evaluating various strategies within the limits imposed by a criterion or a set of criteria for the operation of the system.Once a simulation is in the use, running it on new data or with new parameters is usually just a matter of few keystrokes or dragging and dropping a different life. Depending upon the variables being deterministic or random, the simulation models can be classified as

a) Deterministic Simulation
b) Stochastic Simulation

a. ***In deterministic simulation***, a system is simulated under well determined conditions. This kind of simulation is useful to observe the behavior of system in certain particular cases, to discover errors in the design or in the implementations, to build examples, etc. In this kind of simulations, only one run is needed and there is no truly random variable involved.

b. **In Stochastic simulation**, system performance is measured. This is useful to see if the system has good response time under average conditions, to compare different implementations of the same system, or totally different systems that have the same output. It is useful to classify the system being simulated into separate categories depending upon the degree of randomness associated with behavior of the system in its simulated environment. A system that relies heavily upon random behavior is referred to as a stochastic system [4].

A. Need for Simulation:

Simulation is used to observe the dynamic behavior of a model of real or imaginary system. Indeed by simulating a complex system one is able to understand its behavior at low cost. To simulate the behavior of complex systems, simulators are designed and developed. A simulator is a collection of hardware and software systems which are used to mimic the behavior of some entity or phenomenon. Simulators may also be used to analyze and verify theoretical models which may be too difficult to grasp from a purely conceptual level. As such, simulators provide a crucial role in both industry and academia.

IV. PROPOSED MODEL

The simulator will calculate the effort for the project over several simulation runs. After calculated efforts there will be clear idea to reuse managers how to allocate resources.

A. Introduction:

In computer science and software Engineering, reusability is the likelihood a segment of source code that can be used again to add new functionalities with slight or no modification. Reusable modules and classes reduce implementation time, increase the likelihood that prior testing and use has eliminated bugs and localize code modification when a change in implementation is required.

The issues relating to managing the development process of a project are handled through project management, because a development process does not specify how to allocate resources to the different activities, how to divide work within a phase. Software project management is an interpolation of project planning, project monitoring and termination. Planning is necessary for quality software. It is essential the knowledge about the effort estimation foe a good planning. On the basis of effort estimation cost of a project can be determined. Software cost estimation is necessary for resources allocation and bidding [5]. There are different models available.

Simulator will calculate efforts for the different type of projects over several simulation runs. The simulator will calculate efforts and mean relative error for different type of projects. We have administered the historical data to COCOMO 81 model and identified that no cost model gives the exact estimate of a software project. This is due to the fact that a lot of productivity factors are not contemplated in estimation process. Software reuse is being eclipsed although most of the contemporary software projects are based on object oriented development where no component is made from scratch (Inheritance). After using the concept of reuse size of software should be considered. And after calculated optimal size, we can measure the optimal efforts for the software. On the basis of efforts estimation reuse managers can decide how to allocate resources to the different activities, how to divide work within a phase. We can modify the parameter size. The concept of reuse plays very important role in the industries. [6].

Effective size of existing software can be calculated as

Effective size = existing size * (0.4 * redesign% + 0.25 * reimplementation% + 0.35 * retest %)

Effective size of new software can be calculated as

Effective size1 = new code + existing size * (0.4 * redesign% + 0.25 * reimplementation% + 0.35 * retest %)

Reuse% = (RSI/Total Statement) * 100

RSI (Reused Source Instruction)

Formula used for calculation the efforts for the project is

$EFFORT = a * (\text{Effective size1})^b$

Effective Size of a project measures in KLOC (Kilo Lines of Code).

EAF is effort adjustment factor, which will be calculated using cost drives

A cost estimation model calculates efforts using a function of program size and a set of cost drivers attributes.

Value of a, b depends on the complexity of software.

EAF which will be calculated using cost drivers factors.

Ex. required reliability, database size, application exp. Etc.

EFFORT for project will be determined in person-month using formula.

The projects are categorized into three types:

a. Organic
b. Semidetached
c. Embedded

These categories roughly characterize the complexity of the project with organic projects being those that are relatively straight forward and developed by a small team, and embedded are those that are ambitious and high requirements for such aspects as interfacing and reliability.

The constant a, b for different system are:

Table 1 Constants value for different system

System	A	B
Organic	3.2	1.05
Semi detached	3.0	1.12
Embedded	2.8	1.20

For cost driver factors there can be different scale. The rating scale for RELY can be very low, low, high, very high. Ex 0.75, 0.88, 1.00, 1.5.

To calculate EAF multiply all these cost drive attributes.

EAF can be different for different projects and there is possibility that rating scale for cost driver attributes can be changed. On generation of random numbers and using Box Muller Transformation for different rating scale for cost driver attributes are calculated

The μ is the mean value and the σ the standard deviation

According to standardized normal distribution function and generating random numbers from the standardized normal distribution function the following relation called the Box-Muller transformation.

S= $((-2 * \log_e r1)^{1/2}$ *** cos (2 * π * r2))**

Where r1 and r2 are two uniform random numbers in the range (0.1) and s is the desired sample from the standardized normal distribution.

Now the value of cost driver attributes P[i] based on the Box Muller Transformation is

P[i] = σ *s + μ

Thus a number of samples of P[i] will be generated.

Relative error = ((calculated effort-original effort) / (original effort)))*100

B. Uncertainties in Effort Estimation:

Effort estimation can perform at any point in software life cycle. As the effort of the project depends on the nature and characteristics of the project, the accuracy of the estimation will depend on the amount of reliable information we have about final product. There is a great deal of uncertainty represent a range of possible final products, not one precisely defined product. Hence, the effort estimation based on this type of information cannot be accurate. Estimate at this phase of project can be off by as much as a factor of four from the actual final effort

C. Algorithms to Calculates the Effort for the Project:

This algorithm calculates the efforts for a project by simulating it for a large no. of times.

It uses two functions namely:

a. EFFORT(), to calculate efforts for the different type of projects.
b. PRODUCE(), to generate random samples of cost driver attributes.

D. EFFORT (a,b,,eaf,e,n,m,size, μ, σ*, nlen, relerr, oref):*

a. len ← 1
b. size1← exsize2 * (0.4 * redesign% + 0.25 * reimp% + 0.35 * retest%)
c. esize1 ← new code + size1
d. e ← a * (esize1)^b
e. while len < nlen
f. eaf ← 1
g. do produce(μ, σ ,p,m)
h. for k ← 1 to m
i. do eaf ← eaf * p[m]
j. do e1 ← e * eaf
k. relerr ← ((e1– oref) / oref) * 100

PRODUCE (μ, σ ,p,m)

a. for i ← 1 to m
b. do r1 ← random(dum1)
c. r2 ← random(dum2)
d. v ← (sqrt(-2 * $\log_e$(r1))*cos(r2))
e. p[i] ← (σ [i] * v) + μ [i]

E. Genereation of random numbers using Box Muller Transformation:

Table 2 Notation and terms used

Sr.No	Term	Meaning
1	n	No. of project
2	nlen	No. of simulation runs
3	σ [i]	Standard Deviation of Cost Driver
4	e	Initial Effort
5	i	Variable
6	x	Random values
7	EAF	Effort adjustment factor
8	P	Cost driver attribute
9	size	Size
10	a	Constant for different Projects
11	b	Constant for different Projects
12	exsize2	size of existing software
13	size1	effective size of existing software
14	esize1	effective size of new software Using existing software
15	Relerr	relative errors
16	Oref	original efforts
17	μ [i]	mean value for cost driver attribute
18	e1	Final effort

V. IMPLEMENTATION AND EXPERIMENTAL RESULTS

In order to implement our proposed model, we developed an Application which identifying importance of software reuse in cost estimation. We have implemented the Application using C++ programming language

Table 3 Input data for simulation

Project Type	A	B
Organic	3.2	1.05
Semi detached	3.0	1.12
Embedded	2.8	1.20

Table 4 Input data for the project

Project no.	Size (KLOC)	Original effort
1	50	47
2	40	66
3	22	60
4	2.5	312

Input value of μ =0.567, σ =0.007

Table 5 Calculated Effort for the Organic project

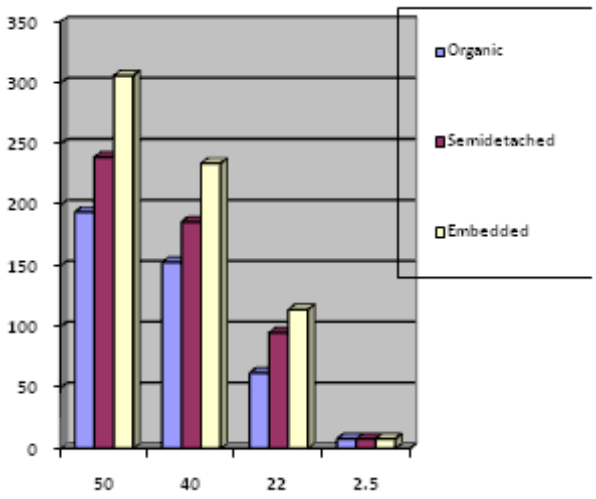
Project no	Size (KLOC)	Original Effort	Calculated effort	Relative Error%
1	50	47	194	313
2	40	66	153	133
3	22	60	82	36
4	2.5	312	6.6	-99

Table 6 Calculated Effort for the Semi detached project

Project no	Size (KLOC)	Original Effort	Calculated effort	Relative Error%
1	50	47	239	410
2	40	66	186	183
3	22	60	95	59
4	2.5	312	8	-97

Table 7 Calculated Effort for the Embedded project

Project no	Size (KLOC)	Original Effort	Calculated effort	Relative Error%
1	50	47	306	551
2	40	66	234	254
3	22	60	114	90
4	2.5	312	8	-97



Graph. 1. shows COCOMO81 Intermediate model chart with given values.

In given chart X-axis shows Size of the project and Y-axis shows Efforts. This chart shows calculated effort of different type of projects as Organic, Semidetached, Embedded projects. As described in the chart when size of project is 50 then calculated efforts for the Organic Type of project is 194, for the Semidetached Type of project is 239 and for the Embedded Type of project is 306

VI. CALCULATE SIZE OF SOFTWARE WITH REUSING

Calculation of size when we are using the existing software in development of new software:
Size of existing software = 20.
Effective size of new developed software:
New code=40
Effective size= new code + size of existing software* (0.4*25% +.25 *10% + 0.35 * 18%)

= 40 + 20*(0.148)

= 42.96

Table 8 Calculated efforts for the Organic project using existing software

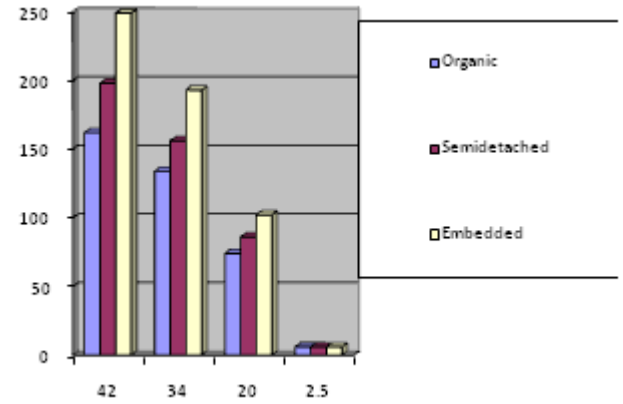
Project no	Old Size	New Size(KLOC)	Original Effort	Calculated effort	Relative Error%
1	50	42	47	162	244
2	40	34	66	134	103
3	22	20	60	74	23
4	2.5	2.5	312	6	-98

Table 9 Calculated Effort for the Semi detached project using existing software

Project no	Old Size	New Size(KLOC)	Original Effort	Calculated effort	Relative Error%
1	50	42	47	198	319
2	40	34	66	156	136
3	22	20	60	86	43
4	2.5	2.5	312	6	-98

Table 10 Calculated Effort for the Embedded project using existing software

Project no	Old Size	New Size(KLOC)	Original Effort	Calculated effort	Relative Error%
1	50	42	47	249	429
2	40	34	66	193	192
3	22	20	60	102	71
4	2.5	2.5	312	6	-98



Graph 2: Shows COCOMO81 Intermediate model chart with given values

In given chart X-axis shows Size of the project and Y-axis shows Efforts. This chart shows calculated effort of different type of projects as Organic, Semidetached, Embedded projects. As described in the chart when size of project is 42 then calculated efforts for the Organic Type of project is 162, for the Semidetached Type of project is 198 and for the Embedded Type of project is 249.

VII. CONCLUSION

In this case study, simulator calculated efforts for the projects by taking the value of a, b, size and cost driver attributes. Cost driver attributes are calculated by Box Muller Transformation after generating random number between 0 and 1. After simulation for 1000 simulation runs, the efforts for the project were calculated. In this relative error between original effort and calculated effort were calculated. So we identify total efforts to make a software component reusable by using Box Mulller Transformation for various cost driver attributes. Reusable components can be produced and re-engineered effectively in a large scale if we can formulate objective and reusable guidelines and apply them systematically.

Simulation has been applied in diverse fields, ranging from aerospace to energy production and commendable work has been done in these fields. Simulators approximate the real phenomenon in a better way compared to the repeated testing process in software development house. Software Reuse is the application of existing solutions to new problems. Reuse can reduce the time spent in creating solutions by avoiding duplicate efforts. In computer science and software engineering, reusability is the likelihood a segment of source code that can be used again to add new functionalities with sight or no modification. Reusable modules and classes reduce implementation time, increase the likelihood that prior testing and use has eliminated bugs and localize code modifications. In order to diverse a measurement model or qualitative guidelines for evaluating reusable components, the factors that are known to influence reuse must be identified. The factors include coupling, cohesion and complexity. In software reuse, many components are available which can be modified to make them reusable

VIII. REFERENCES

[1]. Benbasat, I., Goldstein, D.K. and Mead, M., "The case Research strategy in studies of IJOPM information systems", MIS Quarterly, September 1987, pp. 369-86. 15,12

[2]. Benediktsson, O., Dalcher, D.: "Developing a new Understanding of Effort Estimation in Incremental Software Development Projects. Proc. Intl. Conf. Software & Systems Engineering and their Applications" (ICSSEA'03), December 2-4, 2003, Paris, France. Volume 3, Session 13, ISSN 1637-5033, 10 p.

[3]. Anda, B.: "Comparing Effort Estimates Based on Use Cases with Expert Estimates. Proc. Empirical Assessment in Software Engineering" (EASE 2002), Keele, UK, April 8-10, 2002, 13p.

[4]. Jørgensen, M., Moløkken, K.: "Situational and Task Characteristics Systematically Associated With Accuracy of Software Development Effort Estimates". Proc. Information Resources Management Association Conference (IRMA 2003), pp. 824-826.

[5]. CH.V.M.K.Hari, Prof. Prasad Reddy P.V.G.D, J.N.V.R Swarup Kumar, G.SriRamGanesh." Identifying the Importance of Software Reuse in COCOMO81, COCOMOII". International Journal on Computer Science and Engineering Vol.1(3), 2009,pp.3-6

[6]. Moløkken, K., Lien, A.C., Jørgensen, M., Tanilkan, S.S., Gallis, H., Hove, S.E.: "Does Use of Development Model Affect Estimation Accuracy and Bias"? Proc. Product Focused Software Process Improvement: 5th International Conference, PROFES 2004, Kansai Science City, Japan, April 5-8, 2004. Springer-Verlag, ISBN: 3-540-21421-6. pp. 17-29.

[7]. Barry W. Boehm. Software risk management: Principles and practice. IEEE Software, pages 32{41, January 1991.

[8]. Barry Boehm, Bradford Clark, Ellis Horowitz, Chris Westland, Ray Madachy, Richard Selby. "Cost models for future software life cycle processes:" COCOMO 2.0 [2005].

[9]. Verner, J.M., Evanco, W.M.: State of the Practice: "Effect of Effort Estimation on Project Success". Proc. of the Intl. Conf. On Software & Systems Engineering and their Applications (ICSSEA'03), Vol. 3, Session 13,10p.

[10]. Jørgensen, M.: "Top-down and Bottom-up expert Estimation of Software Development Effort. Information and Software Technology", vol.46 (2004), pp. 3-16.

[11]. P.K. Suri, Neeraj Garg. "Simulator for evaluating Reliability of Reusable Components in a Domain Interconnection Network". In IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.3, 2008.pp.4-6.