Global Journal of Computer Science and Technology
Volume 11 Issue 20 Version 1.0 December 2011
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Cost Model for Reengineering an Object Oriented Software System

By Dr. Ashok Kumar, Bakhshsish Singh Gill
Kurukshetra University, Kurukshetra

Abstract - The cost of reengineering of object-oriented software is often significantly less than the cost of developing new software. Object oriented software systems are more reusable. Reengineering of software systems rather than developing new software will save precious time and resources. Reengineering reduces the cost of maintenance by increasing the software quality and reducing complexity.To justify reengineering, the cost of reengineering software must be estimated and compared with the cost of new software. The cost of reengineering depends upon many factors but major factors are the portion of the software (number of objects) to be reengineered and complexity (interrelationship between objects) of the software. In this paper efforts are done to present a reengineering cost estimation model. On the basis of this model, software managers can take a decision whether to maintain, reengineer or retire the software.

Keywords : *Objects, reengineering, complexity, fine object, faulty object.*

GJCST Classification : *D.1.5*

Strictly as per the compliance and regulations of:

Cost Model for Reengineering an Object Oriented Software System

Dr. Ashok Kumar [α], Bakhshsish Singh Gill [Ω]

Abstract - The cost of reengineering of object-oriented software is often significantly less than the cost of developing new software. Object oriented software systems are more reusable. Reengineering of software systems rather than developing new software will save precious time and resources. Reengineering reduces the cost of maintenance by increasing the software quality and reducing complexity.To justify reengineering, the cost of reengineering software must be estimated and compared with the cost of new software. The cost of reengineering depends upon many factors but major factors are the portion of the software (number of objects) to be reengineered and complexity (interrelationship between objects) of the software. In this paper efforts are done to present a reengineering cost estimation model. On the basis of this model, software managers can take a decision whether to maintain, reengineer or retire the software.

Keyterms : Objects, reengineering, complexity, fine object, faulty object.

I. Introduction

The ability to accurately estimate the time and cost of reengineering software is the key factor for successful of reengineering project. Cost estimation is needed before reengineering is initiated. The primary objective is to enable the client and software engineer to perform a cost benefit analysis. The estimate can be in terms of person-month (PM), which can be translated into actual rupees cost. Cost estimation is not easy task; many factors in estimation are not quantifiable. Also reengineering area is young and needs much maturity and improvement. Quality of software design matters in the process of estimation. Easiness in software understanding, maintenance and reengineering depends upon the decomposition of system.

In successive generation of languages decomposition is supported differently. The advancement in the software technology, from machine language to assembly to procedural to object-oriented languages, helps programmers to estimate time and efforts for software development. Object - oriented technology is more helpful in measuring reengineering cost as it uses objects and not algorithms as its fundamental building blocks. In this context, object is not object of some object oriented language but it is a conceptual module than can be plugged in and plugged out from the software system. Reengineering identifies reusable components (objects) and analyzes the changes that would be needed to regenerate them for reuse within new software architecture. The use of a repeatable, clearly defined and well understood software objects, has make reengineering more effective and reduced the cost of reengineering.

II. What is an Object?

I thought of an object in the context of object-oriented technology as independent component that can be pulled out and plugged in the software system. Object is taken as a private, separable independent module of software. If we pull out an object from a software system, it is working system without much affecting the whole system except the job done by that particular object. As in the other physical systems, a component is plugged out, repaired and plugged in the system again. Additional screws, nuts and bolts are required for this purpose. We must develop a universal language of such type. We must have a set of additional instructions as nuts and bolts to plug-out and plug-in the objects in the whole system. Following figure shows an object which is an independent unit of software that can be interfaced with the software system.

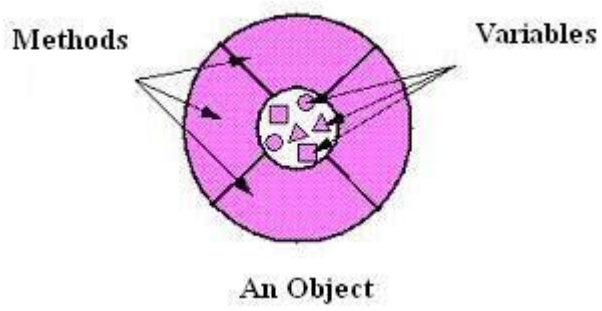

Fig. 1

The object oriented approach attempts to manage the complexity inherent in the real world problems by abstracting out knowledge and encapsulating it [1]. Object is an instance of a class and has an identity and stores attribute values [2].

Author [α]: Professor, Department of Computer Applications, Kurukshetra University, Kurukshetra, Phone: 01744-238195(Office), 01744-239231(Residence)
Author [Ω] : Sr. Programmer, Computer Centre, Guru Nanak Dev University, Amritsar-143001, E-mail bbssgill@yahoo.com, Phone 0183-2258802-09 Ext. 3297(Office), 0183-2502813(Residence) , Mobile 9988112620

Here in this piece of work, Object is seen at a higher level of abstraction and is taken as independent module or unit that can be plugged in or plugged out of the software system. As software ages some objects become faulty (new term coined). Faulty objects are indentified and modified. Faulty object is that which hang without responding or if responding to some operation, its response is incorrect. Candidate software system is disbanded; all objects of the system are separated. Faulty objects are identified and collected for reengineering. Old design of the software is examined (Reverse Engineering). Then redesigning of the structure of the system to improve the quality of software system is done (transformation of the architecture). According to new quality and modern design objects are integrated (Forward Engineering).

Following is the example of object oriented software system with eight objects like real world objects. Circles are objects and lines represent communications to send messages between objects. The object in the system is characterized with three properties Identity, State and Behavior. Identity distinguishes it from others, state is the data stored in it and behavior describes the methods by which the objects can be used.

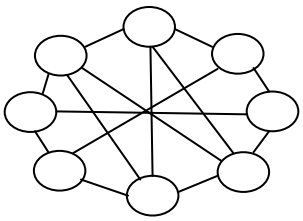

Fig. 2 : object oriented software system

Abstraction is good tool for reengineering object oriented design as it helps in reducing complexity. Large systems are complex having more objects as each additional object increases the complexity of the system [3].

III. Need for Reengineering

Software maintenance starts after delivery of the software to correct faults, to improve performance and other attributes of the software. Maintenance plays an important role in the life cycle of a software system. Maintenance is the last stage of the software life cycle. After the product has been released, the maintenance phase keeps the software up to date with environment changes and user requirements changes. With recurring maintenance, complexity increases and software quality decreases. As the software is maintained, errors are introduced. Many studies have shown that each time an attempt is made to decrease the failure rate of a system, the failure rate got worse. On average, more than one error is introduced for every repaired error. In this way maintenance cost goes on increasing with time. Software maintenance can account for 60 to 80 percent of the total life cycle of software product. More than 90 % of the total cost of software goes to maintenance and evolution of the software product [4].

After a certain period, there is a crucial point when it is difficult to maintain the system or maintenance cost is too much high. Maintenance problems are a driving force behind re-engineering. Reengineering is the only way to avoid development cost. But what is the cost of reengineering software? If we do not know how can we go for reengineering? If the cost of reengineering is known then it can be compared with the cost of the new system.

In the following figure, maintenance cost is raising high from red point D onward. At this time we think of reengineering or retiring the software because maintenance cost increases rapidly. System can be maintained well if this situation does not arise.

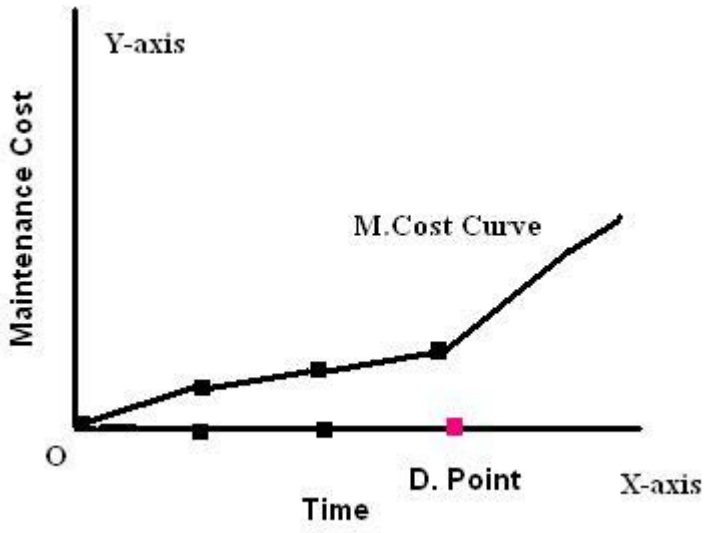

Fig. 3 : Decision point D

Now software managers are worried for making decision for reengineering at point D. Name D is given as it is a decision point for the software managers. If we retire the system then we have to bear the cost of new software and naturally it will be high cost.

At this time, the cost model will help the software managers to compare the quoted cost of new software with cost of reengineering. Certainly reengineering will attract the software managers. Reengineering can not be escaped. Reengineering is the only way to utilize the resources fully and the problem of backlog of software will also be solved.

IV. Reengineering Cost Model

Object-oriented paradigm has changed the scene for reengineering. Object-oriented software system is all about objects. Object-Oriented software system is being more reusable and hence more suitable for reengineering. Reengineering of software systems rather than developing new software will save precious time of skilled engineers and legacy resources. Reengineering reduces the cost of maintenance and to escape the new software development. Maintenance cost increases as software ages. Maintenance problems are a driving force behind reengineering. To justify reengineering, the cost of reengineering software must be calculated and it should be less than the cost of purchasing new software with a great difference. The cost of reengineering depends upon many factors but major factors are the portion of the software (number of objects) to be reengineered and complexity (interrelationship between objects) of the software. Cost model will help organizations whether to reengineering the software system or to buy new software. This is a major decision faced by the software managers. In this paper efforts are done to present a cost model for reengineering legacy object oriented software system.

To calculate the cost of reengineering the following factors are taken into consideration.

1. Number of objects to be reengineered.
2. Size of the object (Number of attributes)

Each object has its own attributes, but attributes are taken into consideration according to requirement specification and business process. For example attributes of object employee are name, employee identification code, address, mobile phone, landline phone, age, height, color, basic pay, grade pay and many more. If software is required for payroll of the employee, the attributes like height, age, color etc is not required. Number of attributes depends upon the size of the problem.

Reengineering cost depends upon the number of objects to be reengineered. It means reengineering cost of an object is to be calculated to calculate the reengineering cost for the software system. The candidate object is called faulty object. On average half of the objects could be candidate objects. Following is the figure of software system with some 7 faulty objects. This system can be maintained as faulty objects are less than half.

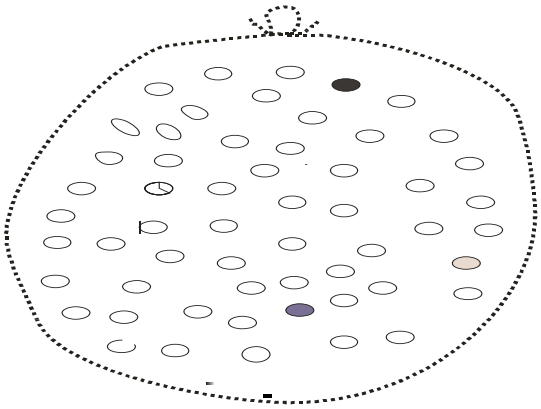

Fig. 4 : Application System with faulty objects

Application software system (in figure 3) can be maintained by modifying the faulty objects and need not reengineering. Number of faulty objects increases with the age of software. System is candidate for reengineering when the faulty objects number reaches to half. These faulty objects can be reengineered and can be plugged to enhance the functionality of the application.

V. Reengineering Cost of the Candidate Object

Traditional software measurement techniques are not satisfactory for measuring productivity and predicting efforts for object oriented software systems. The Source Lines of Code (SLOC) metric and the Function Point metric both were for programming environment putting the data and procedures separate. In object oriented paradigm data and procedures are combined. In object oriented approach the role of UML is supreme. It was designed to provide a standard for software modeling languages. It is a graphical notation for object-oriented analysis and design. UML provides a framework for describing a set of models that capture the functional and structural semantics of any complex information system. UML constructs in object oriented software can be used for estimation of resources like efforts & cost etc. While calculating the efforts of reengineering it is important to include information about communication between objects and reuse through inheritance in the size of the object (Lines of code) as well. An object is small piece of source code that can be maintained or reengineered.

Common and simple approach for measuring efforts for developing small software with single variable is as under

Efforts = a*(SIZE)b where a and b are constants determined by regression analysis applied to historical data [5]. SIZE is a variable and the value of this variable depends upon the size of the object. The SIZE is the

number of lines of code. This model is for the structured software systems. This model measures the efforts for developing software from scratch. Reengineering the legacy software object will take the efforts to half.

Efforts for reengineering small piece of source code as an object can be calibrated as under

Efforts == [a*(number of attributes)b] / 2 where a, b are constants and can be determined from historical data(from past experience) of reengineered software systems. If we denote number of attributes of an object involved in computations by A, then the above model will be as under

Efforts = E1= [a*(A)b] / 2 Person Months for an object say O1.

This model estimates the total efforts for reengineering an object, a small piece of code that can be plugged in with the object oriented application. Cost of reengineering all the faulty objects will be estimated by adding all above such Ei's. Let us suppose there are n faulty objects then cost of reengineering of all the objects will be $E_1+E_2+E_3+\ldots\ldots\ldots E_n$.

VI. Reengineering Cost of Software System

In the beginning, behavior of Legacy software system (Object-Oriented) is examined. The system is disbanded; objects are identified and separated for reengineering. Cost model for reengineering an object is presented above as efforts for reengineering an object O_1 will be E_1. With this model, reengineering cost of all the faulty objects is calculated separately. Let us suppose our candidate system is with n faulty objects say $O_1, O_2, O_3,\ldots\ldots\ldots\ldots O_n$ Reengineering cost of all the n objects will be added and is equal to $E_1+E_2+E_3+\ldots\ldots\ldots E_n$. Now all the objects are fine and we need to integrate them into a system. At this stage software architecture will also be changed (improved). There will be additional efforts (cost) for identifying the faulty objects, transformation of the architecture and integrating them into the new design. We denote it by C_n; the constant is to be determined after verifying the results by empirical data available from the past reengineered systems. Then the total efforts for reengineering the object oriented software system will be as under

$$E = E_1+E_2+E_3+\ldots\ldots\ldots E_n. + C_n$$

E is total Person Months of reengineering of software system with n faulty objects. Efforts $E_1, E_2, E_3,\ldots\ldots\ldots E_n$ are person months of reengineering n faulty objects $O_1, O_2, O_3,\ldots\ldots\ldots\ldots O_n$ respectively. Person Months can be multiplied by the current rupees rate of Person Month work.

Estimated reengineering cost = Estimated Effort * current Rate of Person-Month. If we denote current rate in rupees for person months by R then cost model will be as under

Cost = E*R

VII. Results and Conclusions

In this piece of work cost model for reengineering object oriented software system is presented which will be valuable to both the community's software managers and software engineers. Firstly cost model for reengineering an individual object is presented as Efforts = [a*(A)b] / 2 Person Months where constants a and b are to be determined, A is the number of attributes of the object. Cost model for reengineering software system is 'Cost = E * R'

This model is indispensable to organizations as they can settle the deal for reengineering software and can escape buying the new software.

VIII. Future Work

In this paper, software system is viewed as system of real world objects. Object is seen as packed separable module that can be plugged in and plugged out of the software system. System is disbanded (split up/break up) and faulty objects are identified for reengineering. These objects are renovated and arranged as to improve the design structure, packed to work as a system. On this concept reengineering cost model is calibrated.

The future work is to test it for suitability to fit in on the basis of analysis of past data. The constants 'a' 'b' and 'Cn' are unknown and to be determined. These constants can be found out from the past experience by collecting empirical data from reengineered object-oriented software projects. Near about 50-80 projects can be judged to fit this model. With suitable values of above constants in discussion, the reengineering cost model for object oriented software systems is ready. This model will facilitate both the communities the software engineers and the software managers.

References References Referencias

1. Brock R.W.,Wilkerson B., Wiener L. (2007) "Designing Object-Oriented Software", Prentice-Hall of India, New Delhi p. 5.
2. Bernd Bruegge, Dutoit Allen H., "Object-Oriented Software Engineering Using UML, Patterns, and Java", Pearson Education (Singapore), p.724.
3. Halladay S., Wiebel M., "Object-Oriented Software Engineering", BPB Publications, New Delhi. P. 35.
4. Erlikh, L. (2000). "Leveraging legacy system dollars for E-business". (IEEE) IT Pro, May/June 2000, 17-23. Down loaded on 24-02-2011 from the site: http://users.jyu.fi/~koskinen/smcosts.htm
5. P. Jalote (1996). "An Integrated Approach to Software Engineering", Narosa Publishing House , New Delhi. P. 88.

Books

1. Sal Valenti, "Successful Software Reengineering", IRM Press, 1331 E., Chocolate Avenue, Hershey, 2002.
2. Robert S. Arnold, "Software Reengineering", IEEE Computer Society Press Los Alamitos,California
3. Roger S. Pressman, "Software engineering", 3rd ed., McGraw-Hill, New York, 1992.
4. Grady Booch, "Object-Oriented Analysis and Design with Applications", Pearson Education, Singapore, 2003.
5. IAN Sommerville, "Software Engineering", Addison-Wesley Publishing Company, Singapore, 1994.
6. K.K.Aggarwal and Yogesh singh, " Software engineering", New age International (P) Ltd., Publishers, New Delhi, 2002.
7. Nasib Singh Gil, " Software Engineering: software reliability, Testing and Quality Assurance", Khanna Book Publishing Co.(P) Ltd., New Delhi, 2002.

Web Sites

http://journals.ecs.soton.ac.uk/java/tutorial/java/objects/object.html dated 1/8/2011
http://softwaredesign.com/objects.html dated 13/8/2011