

**Kingdom of Saudi Arabia
King Saud University**

**College of Computer and Information Sciences
Computer science Department**

Software Cost Estimation Model

**For
Object Oriented Software
Console Applications**

**Submitted by
Amel Ali Al- hossan**

**Supervisor
Dr. Ghazy Assassa**

Master project submitted in partial fulfilment of the requirements for the Masters degree in the Department of Computer science at the College of Computer and Information Sciences King Saud University.

2005

Abstract

Software cost estimation is an important activity during its development. There are many sophisticated parametric models for estimating the size, cost, and schedule of Object Oriented (OO) software projects. Several authors assert that a model's predictive accuracy can be improved by calibrating its default parameters to a specific environment [4][5][10]. The main aim of this paper is to improve the cost estimation accuracy of Object Oriented console applications* at design phase.

This research adds to previous researches the development of accurate estimation formula by calibrating COCOMOII early design model with a specific organization using Object Oriented Function Point (OOFP) as a size measure instead of standard Function Point (FP) that is used currently in COCOMOII models. This methodology will benefit from the model calibration and consider all OO concepts such as inheritance, aggregation, association and polymorphism. Linear regression will be investigated to relate LOC to software OOFP based on the historical projects of the organization. The result regression model will be used to estimate LOC as a step towards estimating effort and cost. We developed a software tool to implement the proposed estimation formula. It performs automatic cost estimation for any OO console applications and generates the results as printed reports. To get more accurate results, the organization should feed the tool with at least 10 historical software projects.

* Visual C++ application = Visual part + Programming language part

Here we are only concern about the programming language part, which can be put together by just using the **console-based** applications. These types of applications are character based in that they do not use any internal visual components provided by Visual C++.

Acknowledgment

*A special thank to **Dr. Ghazy Assassa**, the supervisor of this research who manages this research and provides all necessarily guidance and consultations throughout the project phases.*

*Thanks to **Dr. Hassan Mathkour** who gave the idea of this research and who teaches the concepts of building scientific researches and writing literature reviews.*

*I would like to acknowledge **Al Ejtiaz House Establishment** for providing the information that helps to test the proposed formula and helps to perform the calibration process.*

*Thanks to **my family** for their encouragement and support.*

Table of Contents

ABSTRACT	2
ACKNOWLEDGMENT	3
TABLE OF CONTENTS	4
LIST OF FIGURES	6
LIST OF TABLES	7
INTRODUCTION	8
LITERATURE REVIEW	9
1. PHASE 1: PROJECT PLANNING	14
1.1. PROJECT OVERVIEW	14
1.1.1. PROBLEM DEFINITION	14
1.1.2. THE SCOPE OF THE PROBLEM (PROJECT)	15
1.1.3. PROJECT OBJECTIVES	15
1.2. PROJECT RESOURCES	16
1.2.1. SOFTWARE	16
1.2.2. HARDWARE	17
1.2.3. MAN POWER	17
1.2.4. BOOKS	17
1.2.5. INTERNET	18
1.3. PROJECT SCHEDULING	18
1.3.1. PROJECT TASKS	18
1.3.2. TASKS DEPENDENCIES	19
1.3.3. PROJECT TIME LINE	19
2. PHASE 2: REQUIREMENTS ANALYSIS	23
2.1. FEASIBILITY STUDY	23
2.2. REQUIREMENTS ELICITATION AND ANALYSIS	24
2.2.1. SYSTEM DOMAIN UNDERSTANDING:	24
2.2.2. REQUIREMENTS COLLECTION AND CLASSIFICATION:	25
2.2.2.1. Organization Information	25
2.2.2.2. COCOMOII Early Design Model Calibration requirements:	25
2.2.2.3. New Project estimation requirements:	26
2.2.2.4. Software Cost Estimation Tool phases	28

2.2.3. DATA COLLECTION:	32
2.2.3.1. COCOMOII definition	32
2.2.3.2. COCOMOII Models	32
2.2.3.3. Development Effort Estimates	33
2.2.3.4. Scaling Drivers	33
2.2.3.5. Post-Architecture Model Cost drivers	34
2.2.3.6. Early Design Model Cost drivers	35
2.2.3.7. Object Oriented Project sizing:	36
2.2.3.8. Object Oriented Function Point Counting Procedure:	37
2.2.3.9. Example:	39
2.2.3.10. Line Of Code (LOC) estimation:	42
2.3. REQUIREMENTS SPECIFICATION:	43
2.3.1 Requirement Document	43
3. PHASE 3: SOFTWARE DESIGN	69
3.1. ARCHITECTURAL DESIGN	69
3.2. SOFTWARE DESIGN SPECIFICATION	70
3.3. COMPONENT DESIGN	76
3.3.1. DATA DESIGN	76
3.3.1.1.Database	76
3.3.2. SCREENS DESIGN	89
3.3.2.1. Screens Layout and components	89
3.3.2.2. Screens Flow Diagram	94
3.4. FUNCTIONS ALGORITHM DESIGN	97
3.4.1. CALIBRATION FUNCTION ALGORITHM	97
3.4.2. OBJECT ORIENTED FUNCTION POINT FUNCTION ALGORITHM:	97
3.4.3. LINEAR REGRESSION FUNCTION ALGORITHM:	99
3.4.4. NEW PROJECT ESTIMATION FUNCTION ALGORITHM:	99
4. PHASE 4: SOFTWARE CONSTRUCTION	100
4.1. DATABASE CONSTRUCTION	100
4.2. SCREENS CONSTRUCTION	100
4.3. FUNCTIONS CONSTRUCTION	103
4.3.1. CALIBRATION FUNCTION CODE TO CALCULATE A VALUE FOR THE ORGANIZATION	103
4.3.2. OBJECT ORIENTED FUNCTION POINT OOFPP CALCULATION CODE:	105
4.3.3. LINEAR REGRESSION FUNCTION CODE:	107
4.3.4. NEW PROJECT ESTIMATION CODE	107
5. PHASE 5: EXPERIMENTAL RESULTS	108
5.1. EXPERIMENTAL DATA AND RESULTS	108
5.2 TEST CASES	119
5.3 RESULTS COMMENTS	120
CONCLUSION & FUTURE WORK	121
REFERENCES	123
APPENDICES	127

List of Figures

Figure 1: Project Time Line.....	22
Figure 2: Flow diagram of New Project estimation.....	27
Figure 3:Flow Diagram of COCOMOII early design model calibration.....	28
Figure 4: Flow Diagram of Linear regression to relate project size in OOF to LOC.....	29
Figure 5: Flow Diagram (estimate the effort for new project)	30
Figure 6: Over View of the Cost Estimation System Phases	31
Figure 7: Perspectives and measures in the software development process.....	36
Figure 8: Example to count the OOF.....	39
Figure 9: The Architectural Design of proposed system.....	69
Figure 10:Database Relationship.....	78
Figure 11:Flow diagram for system screen (first time usage).....	95
Figure 12: Flow diagram for system screen after more than or equal 10 old project stored in database and the model has been calibrated.....	96

List of Tables

Table 0: Project Time Line	21
Table 1: Scale Factors for COCOMO II Early Design and Post-Architecture Models	33
Table 2: Scale Factors Explanation	34
Table 3: Post-Architecture cost drivers rating values	35
Table 4: Early Design cost drivers	35
Table 5: IFPUG Complexity matrix for ILFs and ELFs	37
Table 6: IFPUG Unadjusted Function Point Table	38
Table 7: TFPCP's Complexity matrix for external input (EIs)	39
Table 8:IFPUG Complexity matrix for ILFs and ELFs	71
Table 9: IFPUG Unadjusted Function Point Table	71
Table 10: TFPCP's Complexity matrix for external input (EIs)	73
Table 11: IFPUG Unadjusted Function Point Table	73
Table 12:Data items description for Answers table	82
Table 13:Table Indexes for Answers table	82
Table 14:Data items description for AnswersTXT table	83
Table 15:Table Indexes for AnswersTXT table	83
Table 16:Data items description for New_Class_Data table	83
Table 17:Table Indexes for New_Class_Data table	83
Table 18:Data items description for New_Class_Methods table	83
Table 19:Table Indexes for New_Class_Methods table	84
Table 20:Data items description for New_projects table	84
Table 21: Table Indexes for New_projects table	85
Table 22:Data items description for Old_Class_Data table	85
Table 23:Table Indexes for Old_Class_Data table	85
Table 24:Data items description for Old_Class_Methods table	85
Table 25:Table Indexes for Old_Class_Methods table	85
Table 26:Data items description for Old_Projects table	87
Table 27:Table Indexes for Old_Projects table	87
Table 28:Data items description for Questions table	87
Table 29:Table Indexes for Questions table	87
Table 30:Data items description for QuestionsWizard table	87
Table 31:Table Indexes for QuestionsWizard table	87
Table 32:Data items description for SW_House_info table	88
Table 33:Table Indexes for SW_House_info table	88
Table34: Model calibration For 5 completed projects	109
Table35: Model calibration For 10 completed projects	110
Table36: Linear regression For 5 completed projects	117
Table37: Linear regression For 10 completed projects	118
Table38: 10 test cases with the model built from 5 historical software projects	119
Table39: 10 test cases with the model built from 10 historical software projects	119

Introduction

In recent years Object Oriented (OO) technologies have emerged as a dominant software engineering practice. As happens with many new technologies, the growth of OO practices has required software developers and their managers to rethink the way they have been estimating the size, effort and cost of their development projects. To better understand and control these costs, the software organization often use parametric cost models for software development cost and schedule estimation. However, the accuracy of these models is poor when the default values embedded in the models are used [4]. The purpose of this project is to construct a software cost estimation tool used to calibrate selected software cost estimation model (COCOMOII- Early design model) using the historical data of specific software house (in this project a designed form used to collect the historical data of software hose – refer to appendix B). Software effort models work better when calibrated with local data [5, 28]. Most cost Models require an estimate of the size of the software. Accurate estimation of size is vital at early stage of software development. The ideal is to identify size measure that can be used to estimate effort directly such as line of code (LOC) measure. The problem was this measure is not available at early stage of development. To solve this problem we use other measures (Object Oriented Function Point - OOFPP) as a basis for estimating length in LOC and then use calibrated cost model to estimate the effort. Since COCOMOII Early design model uses function point to estimate the size of project at this stage and this measure is not suitable to estimate the size of Object Oriented projects, because some aspects of (OO) system (e.g. inheritance, aggregation, association and polymorphism) are not included in the classical function point count [10]. *This project proposes to use Object Oriented Function Point (OOFPP) as a size measure for OO software.* A main advantage in defining OOFPPs was that the method could be automated and this makes it easy to recalculate OOFPP throughout a project, supporting re-estimation. This project will use the procedure mentioned in [4] to automate OOFPPs counting. Then OOFPPs used to estimate LOC as a step to estimate effort and cost. Also the proposed tool will employ the linear regression to find the regression model used to relate computed OOFPPs and Actual size in LOC. The result transformation function used to estimate the size in LOC for a new project from calculated OOFPPs. To test this proposed tool, the measurements (Cost drivers) were collected for ten completed OO projects form a specific software house, for which both an OO design model and the final code were available. All were developed in the same environment, using the same

language (C++ - console application). The result formula that relate computed OOFPs to LOC will be used to estimate the size in LOC for a new designed project then the tool will estimate the effort and the cost of the project. This report presents the project in 6 chapters. Chapter1 consist of introduction and literature review. Chapter 2 is a project planning. Chapter 3 is a Requirements Analysis. Chapter 4 is a software design. Chapter 5 is a software construction (implementation). Chapter 6 is a software Testing. The last part of this report consists of project conclusion and recommendation for future work then the references and Appendices of the project.

Literature review

The formal study of software estimating technology did not begin until the 1960s. In the 1960s, Frank Freiman developed the concept of parametric estimating, and this led to the development of the PRICE model for hardware. This was the first generally available computerized estimating tool. It was extended to handle software in the 1970s[18].

Nowadays there are many **cost estimation tools** have been developed and a lot of studies that gives general guidelines to develop accurate software cost estimation tool. But most of the previous cost estimation methods and tools suffer from many weakness and limitations and it still no one tool, method, or technique that works well in all situations so more studies and researches required for more improvements.

In software cost estimation models, it is important to take into account the software technologies to get more accurate estimation. That is why many models are developed for different applications such as Web-application projects [26], Component-Based-Development (CBD) projects [12][29] and Object Oriented Projects [26][23] [13] [9][7].

Software cost estimation supports planning and tracking of software projects, but the estimation process is difficult because the projects often must satisfy conflicting goals, and must provide specified functionality within specified performance criteria and within a specified cost and schedule and with some desired level of quality. As the factors (or driver of cost estimation) increase the cost estimation will be more difficult. Another challenge in the software cost estimation is that the estimation is required before the product is well defined. Also a complication arises because the way software is being built is changing [37]. The well understanding of all aspects of software cost estimation and all methods and models that are used

in different cases lead to more accurate estimation. There are a lot of studies provide the reader with the important information needed to understand the main concepts of software cost estimation and present useful guidelines to develop a new cost estimation method. The most famous leader in the cost estimation field is Barry Boehm. He [6] explains why he thinks that software costs are important. He stated first that costs are big and growing; therefore any percentage savings will also be big and growing. Second, Many useful products are not being developed; therefore reducing costs will provide more time to attack this backlog. Third; understanding and controlling costs can give us better software, not just more software.

There are many contributions to clarify the picture of the cost estimation. Briand et.al. [8] present overview of estimation methods and project sizing as important issues in the cost estimation, they also define an evaluation framework that allows the reader to compare these methods and they present guidelines regarding the selection of appropriate estimation methods and typical scenarios for using them. Many of studies present useful and general guidelines such as the cost estimation handbook, which is evaluated and validated by NASA Cost Estimating Community (CEC) [21], this handbook is limited to NASA projects, but it could be benefit for any cost estimation researcher.

The construction of cost estimation model is not a trivial job. It needs to know how to build and validate this model as well as full understanding of the cost estimation itself. Vijayakumar [35] propose the construction of a model based on the user's own database, which could provide clear evidence of the rational behind the estimation. Also the author explains how the developers of the model can validate their models more effectively. He proposes to create a database to store the gathering data, validate them and develop a cost model to estimate software costs with greater accuracy.

Estimating the effort, time, and cost of object-oriented software projects is a difficult task. Generating Reliable, repeatable, and accurate estimates can only be achieved through the use of appropriate estimating models that reflect the iterative and incremental nature of object technology. Several authors assert that a model's predictive accuracy can be improved by calibrating (adjusting) its default parameters to a specific environment [19],[18] ,[2]. There are significant studies of cost models calibration. Thibodeau [33] calibrated nine models using three databases. The significance of this study is: Results greatly improved with calibration. Also the IITRI [16] study was significant because it analyzed the results of seven cost models to eight

Ada specific programs. one of the models was COCOMO. The results of this study, like other studies, showed estimating accuracy improved with calibration. In 1997 COCOMO II Post-Architecture model was calibrated by Bernheisel [3] as A Thesis Submitted to the Faculty In Partial Fulfillment of the Requirements for the Degree of Master of Science in Cost Analysis. The model was calibrated using the procedure described in Chapter 3 “ Methodology” Model calibration part. Since the data sets were relatively small, only the coefficient of the effort equation was calibrated. The exponent was set to an “average” value of 1.153.

One of the purposes of this Project is to automate the calibration procedure described in [3] for Object Oriented software –consol application to calibrate COCOMOII early design model for a specific environment.

The obvious challenge in the cost estimation, that it appears impossible to identify features early in the life cycle that define the size of the project. It is important in any cost estimation model to take into accounts the ability to estimate the size and effort in the early stage of the project development life cycle.

If function points or object points are used the software cost estimation can be done at an early stage in the development process [20]. Estimates of these parameters can be made as soon as the external interactions of the system have been designed. At this stage, it is very difficult to produce an accurate estimate of the size of program in lines of source code. Early estimates are essential when using the algorithmic cost estimation models.

Allan Albrecht first proposed original function point analysis [1]. Albrecht's function point is computed by counting the following software characteristics: (1) External inputs and outputs, (2) User interactions, (3) External interfaces and (4) Files used by the system. Each of them is then individually assessed for complexity and given a weighting value, which varies from 3 (simple) to15 (complex). Albrecht's function point has been widely used but it has some weakness. Thus, many kinds of function point, such as TFPCP version [15], 3D Function Points version[13], Feature Points version[17] have been proposed. In recent years, OO technology has emerged. This forced software developers and their managers to rethink the way they have been estimating the size of their development projects [32]. The TFPCP [15] is not suitable for measuring the functionality of OO software. To measure OO software, the main components to

be considered are raw functionality of the software, communication among objects and inheritance [32].

A lot of researchers have proposed methods for adapting function point (FP) to object oriented software. Some researchers retain a focus on traditional function point as the output from a count. They relate OO concepts to FP elements; for example, Whitmire [36] considers each class as an internal file. Messages sent across the system boundary are treated as transaction. Also Schooneveldt [27] treats classes as files, and considers services delivered as transaction. Other researchers develop a new measures, tailored to OO software but analogous to FPs. Sneed [30] proposed Object Point as a measure of size for OO software. Predictive Object Point (POPs) are proposed by mehler et.al.[22]. It based on counts of classed and weighted methods per class, with adjustment for the average depth of the inheritance tree and the average number of children per class. Methods are weighted by considering their type and complexity, giving a number of POPs in a way analogous to FPs. Caldiera et.al. [10] Proposed Object Oriented Functoion Point (OOFPs) as a measure of size for OO software. Some aspects of (OO) system (e.g. inheritance, aggregation, association and polymorphism) are not included in the classical function point count. Nevertheless, they contribute to the final size of the system. If the objective of measuring functionality is to estimate the final size of an implementation of a system, and from that the effort and duration of software project, theses aspects should be taken into account. OOFFP solve this problem and it would be possible to refine a size estimate repeatedly by recalculate OOFPs of the software. The authors construct a tool to automate OOFPs counting procedure mentioned in their paper. Ram et.al.[4] proposed a new Object Oriented Design Function Points OODFP counting procedure which is considers all the basic concepts of OO systems such as inheritance, aggregation, association and polymorphism. *The Cost estimation tool that will be constructed in this project will automate OODFP counting procedure mentioned in this study [24]. The new contribution of this study is to propose using counted OOFPs with calibrated COCOMOII – early design model as the size measure in the intermediate phase to predict the effort and cost.* This OOFFP will be used instead of traditional FPs that was used in COCOMOII – early design model in order to get more accurate estimation for the cost of object oriented software in a specific environment. Unfortunately, estimating LOC has proved to be just as difficult as estimating effort, especially early in development when the estimates are of most use. Several regression models were developed, to relate LOC to software

metrics computed at the design stage such as (OOFPs). Caldiera et al [10] compared robust regression techniques with the more traditional least squares line fitting. They found that the robust models do not gain much in explaining the data better. They found “ The best predictive accuracy (NMSE=0.337) was achieved by the rreg-logistic-G model with tuning parameter $u=8$, corresponding to the linear predictor $LOC = 7183.4 + 25.6 \text{ OOF}$ (This model is very close to the basic linear model $lm-G$, whose equation is $LOC = 7435.1 + 25.2 \text{ OOF}$)”. So *the Cost estimation tool that will be constructed in this project will apply linear regression to find the regression model (transformation function) of the software house from the historical OO software information (actual LOC and Computed OOF). The result regression model will be used to estimate LOC of new OO software at design phase. As a result the estimated size in LOC used to as a size measure in the calibrated cost model to estimate the effort and the cost of the software.*

1. Phase 1: Project planning

1.1. Project Overview

Software Cost Estimation can be defined as the approximate judgment of the costs for a software project. Cost estimation will never be an exact science because there are too many variables involved in the calculation for a cost estimate, such as human, technical, environmental, and political. The four basic steps in software project estimation are:

1. Estimate the **size** of the development product.
2. Estimate the **effort** in person-months or person-hours.(The effort is the amount of time for one person to work for a certain period of time).
3. Estimate the project **cost** in dollars (or local currency)

This project aims to collect a historical cost estimation data for Object Oriented software projects (written in C++ language – console application) from a software house, then use these data to calibrate the COCOMOII cost estimation model to produce a new cost model. Then the estimating cost for the software can be used for pricing purpose.

1.1.1. Problem Definition

Effective software project estimation is one of the most challenging and important activities in software development. Proper project planning and control is not possible without a sound and reliable estimate. Under-estimating a project leads to under-staffing it (resulting in staff burnout), under-scooping the quality assurance effort (running the risk of low quality deliverables), and setting too short a schedule (resulting in loss of credibility as deadlines are missed). Also over-estimating gives a project more resources than it really needs. The project is then likely to cost more than it should (a negative impact on the bottom line), take longer to deliver than necessary (resulting in lost opportunities), and delay the use of your resources on the next project. A robust software cost estimation is required to overcome these problems, and produce an accurate estimation. The existence software cost estimation models suffers from many problems, and needs to be calibrated before it used by a specific software house to give accurate estimation. Also most of software house needs to price the software to the costumer in early stage of software life cycle. The problem in this project stated in the following questions:

- What is the most accurate cost estimation model to be calibrated?
- What type of data we need to collect to calibrate the cost estimation model?
- What are the calibration processes and statistical analysis used with the collected data?

- What are the main steps of software cost estimation?
- How can we measure the accuracy of the result of software cost estimation model?
- How can we use the cost estimation to price the software package?

1.1.2. The Scope of the Problem (project)

There are many different systems where the cost of software is vital. These include, Embedded systems, reusable systems, new development system and Maintenance of legacy systems. In other hand there are large scale, medium scale and small-scale systems. The proposed software cost estimation model that are calibrated to produce effort and schedule estimates for new development projects assume everything is created from scratch and ranges from small to medium scale size of Object Oriented projects .

1.1.3. Project Objectives

There are many-sophisticated parametric software estimating models that use multiple parameters to compute software costs and effort. These models cannot be used directly by any software house. It needs to be calibrated by using the historical data of actual cost of the software house projects. The comprehensive calibration method adjusts all cost driver parameters in the cost model. This level of analysis requires a lot of data to get more accurate model. “Since the number of data points will be less than 100, only the A coefficient (multiplicative calibration variable) will be calibrated. There will be no attempt at calibrating the Effort Multipliers (EM) since there will not be enough data points to justify it”. In this project I aims to use the most popular cost estimation model named COCOMOII to construct a software cost estimation and pricing tool to be used by any software house to estimate the cost of Object Oriented project. This tool guide the user to do the following tasks:

- Feed the system with actual cost of the finished Object Oriented software projects.
- Calibrate the COCOMOII early desin model to adjust the model to be suitable to estimate the cost of Object Oriented projects for the software house that feed the system with their data.
- At design phase the user feeds the system with new Object Oriented project properties (inheritance, aggregation, association and polymorphism).

- The system will estimate the size of the project as Object Oriented Function Point (OOFP)
- The system estimates the size of the project as Line Of Code (LOC) using proposed transformation (inference linear regression model) function to transform (OOFP) to (LOC).
- The system Estimate the effort in person-months
- The system Estimate the project cost
- Re_Estimate when more details come up until the final cost is reached.

All the previous functions offered by the system need to be easy to use by inexperienced user. As the user feeds the system with more accurate data, more accurate cost estimation he gets.

1.2. Project Resources

1.2.1. Software

In this Project I need to use the following software:

- Operating System: Microsoft Windows XP
- Programming language:
Delphi 6
- DBMS:
Borland Database Management System (BDE)
Using Access tables.
- Microsoft Project
Project planning and scheduling
- Microsoft PowerPoint:
Diagrams
- Microsoft Word:

Documentations

- Install shield:
For CD installation

1.2.2. Hardware

The hardware required in this project are:

- PC: Pentum4 intel
Speed: 3.2 MHZ
Memory: 528 MB
HD: 20 GB
- Printer: print Document
- Scanner: Scan Figures

1.2.3. Man Power

This software project will be designed and implemented by one person (my self) including software packaging and on-line help system.

1.2.4. Books

In This project I will refer to the following books:

1. B. Boehm, E. Horowitz, R. Madachy, D. Reifer, B. K. Clark, B. Steece, A. W. Brown, S. Chulani, and C. Abts. **Software Cost Estimation with Cocomo II**. Prentice Hall, 2000.
2. David G., D., Herron, ” **Function Point Analysis, Measurement Practices for successful Software Projects**”, Addison-Wesley, 4th printing, june 2004.
3. **IT Project Estimation : A Practical Guide to the Costing of Software**
Paul Coombs, Cambridge University Press, 2003

4. Sommerville, I. (2001). “**Software Engineering**”, Addison wesley; 6th edition.
5. **Software Engineering Fundamentals**- Ali Behforooz and Frederick J.Hudson
,1996 by Oxford University Press, Inc.
6. **Structured Computer Project Management**- William H.Roetzheim

1.2.5. Internet

In this project I refer to all sites and papers mentioned in the reference part.

1.3. Project Scheduling

1.3.1. Project Tasks

1. Phase 1: Project planning
 - 1.1: Project Overview
 - 1.1.1: Problem Definition
 - 1.1.2: The Scope of the Problem
 - 1.1.3: Project Objectives
 - 1.2: Project Resources
 - 1.2.1: Software
 - 1.2.2: Hardware
 - 1.2.3: Man Power
 - 1.2.4: Books
 - 1.2.5: Internet
 - 1.3: Project Scheduling
 - 1.3.1: Project Tasks
 - 1.3.2: Tasks Dependencies
 - 1.3.3: Project Time Line
2. Phase 2: Requirements Analysis
 - 2.1: Feasibility Study
 - 2.2: Requirements Elicitation and Analysis
 - 2.3: Requirements Specification

- 2.4: Requirements Validation
- 3. Phase 3: Software Design
 - 3.1: Architectural Design
 - 3.2: Software Design Specification
 - 3.3: Component Design
 - 3.3.1: Data Design
 - 3.3.1.1: Database
 - 3.3.1.2: Data items
 - 3.3.2: Screens Design
 - 3.3.2.1: Screens Layout & components
 - 3.3.2.2: Screens Flow Diagram
 - 3.4: Algorithm Design
- 4. Phase 4: Software Construction
 - 4.1: Database Construction
 - 4.2: Screens Construction
 - 4.3: Functions Construction
- 5. Phase 5: Experimental Result
 - 5.1: Experimental data and results
 - 5.2: Test Case
 - 5.3: Results Comments

1.3.2. Tasks Dependencies

Task dependencies appear through the time line sheet.

1.3.3. Project Time Line

Project Parameters	
Project Name	Software Cost Estimation
Project Supervisor	Dr. Ghazy Assassa
Project Start Date	Searching Actual Planning 18-sep-2004
Project Deadline	-Dec-2004
Scheduling frequency	Days

Task Description		Dependencies & Deliverables		Timing		
ID	Phase	Description	Dep. ID	Deliverables	Start Date	End Date
1)	0- Searching and reading				Before 18-Sep	15-Dec-2004
2)	1-Project planning	Describe a clear plan for the project	-	Project outline Project plan Document	18-Sep-2004	15-Oct-2004
3)	1.1- Project Overview				18-Sep-2004	18-Sep-2004
4)	1.1.1: Problem Definition				19-Sep-2004	19-Sep-2004
5)	1.1.2: The Scope of the Problem				20-Sep-2004	20-Sep-2004
6)	1.1.3: Project Objectives				21-Sep-2004	21-Sep-2004
7)	1.2: Project Resources	All resources needed in the project			22-Sep-2004	25-Sep-2004
8)	1.2.1: Software				22-Sep-2004	22-Sep-2004
9)	1.2.2: Hardware				23-Sep-2004	23-Sep-2004
10)	1.2.3: Man Power				24-Sep-2004	24-Sep-2004
11)	1.2.4: Books				25-Sep-2004	25-Sep-2004
12)	1.2.5: Internet				26-Sep-2004	26-Sep-2004
13)	1.3: Project Scheduling	Timing concentrate			27-Sep-2004	15-Oct-2004
14)	1.3.1: Project Tasks				27-Sep-2004	29-Sep-2004
15)	1.3.2: Tasks Dependencies				1-Oct-2004	6-Oct-2004
16)	1.3.3: Project Time Line				7-Oct-2004	15-Oct-2004
17)	2: Requirements Analysis		2	Requirement Document	16-Oct-2004	14-Nov-2004
18)	2.1: Feasibility Study		2	Feasibility report	16-Oct-2004	20-Oct-2004
19)	2.2: Requirements Elicitation and Analysis		18	System models	21-Oct-2004	29-Oct-2004
20)	2.3: Requirements Specification		19	User and system requirements	1-Nov-2004	7-Nov-2004
21)	2.4: Requirements Validation		19-20	Requirement document	8-Nov-2004	14-Nov-2004
22)	3: Software Design		17		15-Nov-2004	29-Dec-2004
23)	3.1: Architectural Design		20	System Architecture	15-Nov-2004	21-Nov-2004
24)	3.2: Abstract Specification		20-23	Software Specification	22-Nov-2004	29-Nov-2004

Task Description		Dependencies & Deliverables		Timing		
ID	Phase	Description	Dep. ID	Deliverables	Start Date	End Date
25)	3.3: Component Design			Component Specification	-Nov-2004	22-Nov-2004
26)	3.3.1: Data Design			Data store Specification	1-Dec-2004	9-Dec-2004
27)	3.3.1.1: Data items				1-Dec-2004	4-Dec-2004
28)	3.3.1.2: Database				5-Dec-2004	9-Dec-2004
29)	3.3.2: Screens Design		24	Interface Specification	10-Dec-2004	19-Dec-2004
30)	3.3.2.1: Screens Layout & components				10-Dec-2004	17-Dec-2004
31)	3.3.2.2: Screens Flow Diagram				18-Dec-2004	20-Dec-2004
32)	3.4: Data Structure Design		25	Data Structure Specification	21-Dec-2004	25-Dec-2004
33)	3.5: Algorithm Design		33	Algorithm Specification	26-Dec-2004	29-Dec-2004
34)	4: Software Construction		22	Executable system	1-jan-2005	22-Apr-2005
35)	4.1: Database Construction				1-jan-2005	15-jan-2005
36)	4.2: Screens Construction				11-Feb-2005	3-Mar -2005
37)	4.3: Functions Construction				4-Mar-2005	22-Apr-2005
38)	5: Experimental Results				4-May -2005	20-May-2005
39)	5.1: Experimental Data and Results				4-May -2005	20-May -2005
40)	5.2: Test Case				21-May -2005	27-May -2005
41)	5.3: Results Comments				27-May -2005	29-May -2005
42)	6: Preparing for presentation				21-May-2005	Presentation Date

Table 0: Project Time Line

Project Time Line Diagram

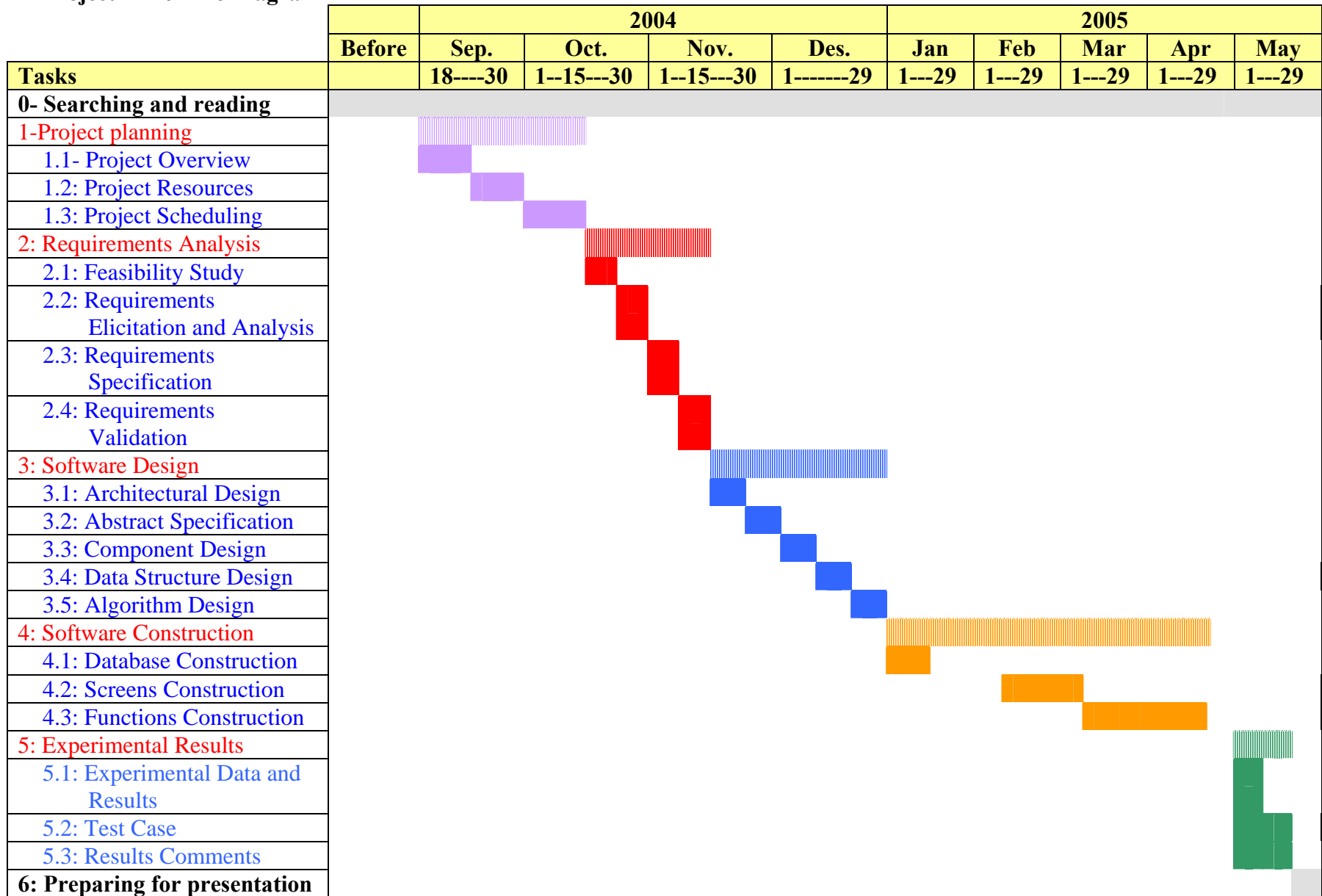


Figure 1: Project Time Line

2. Phase 2: Requirements Analysis

2.1. Feasibility Study

In recent years Object Oriented (OO) technologies have emerged as a dominant software engineering practice. As happens with many new technologies, the growth of OO practices has required software developers and their managers to rethink the way they have been estimating the size of their development projects. Traditional software measurement techniques have proven unsatisfactory for measuring productivity and predicting effort. The Source Lines of Code (SLOC) metric and the Function Point metric were both conceived in an era when programming required dividing the solution space into data and procedures. This notion conflicts with the object-oriented paradigm. Traditional design techniques separate data and procedures while object-oriented designs combine them. There are multiple dimensions that an OO metric must have if it is to provide accurate effort prediction or productivity tracking. It is important to measure the amount of raw functionality the software delivers, but it is equally important to include information about communication between objects and reuse through inheritance in the 'size' as well.

Most of software houses need to estimate the cost of the projects that they are developed at early stage of software life cycle. Estimating the effort, time, and cost of object-oriented software projects is a difficult task. Most cost estimation tools in the market are general purpose, difficult to use and do not address object oriented projects properties such as (inheritance, aggregation, association and polymorphism). Generating reliable, repeatable, and accurate estimates can only be achieved through the use of appropriate estimating models that reflect the iterative and incremental nature of object technology. The COCOMO II is one of the most popular software cost estimation models. The COCOMO II provides a tailorable set of three sub-models, Early Design, Post Architecture, Application Compositions models. This project tends to construct a software cost estimation tool for object-oriented projects written in C++ (console application) with no automatic generation of code. This project satisfies the needs of any software house needs to calibrate COCOMOII Early Design model. The calibrated model Used for estimating the size, and consequently effort and duration, of object oriented software development projects. Different estimates may be made in different phases of

the development process, according to the available information of the classes used in the project. We define an adaptation of traditional function points, called "object oriented function points" (OOFP); to enable the measurement of object oriented analysis and design specifications. Tools have been constructed to automate the counting method as described in [5].

The reasons for choosing this particular model (COCOMOII) are two. First, it represents the newest developments in the field of software cost estimation. Second, it uses large number of parameters.

In this project the constructed object oriented cost estimation tool applied in the data (completed object oriented projects) collected from software house called "Al-Ejtiaz House Establishment" to calibrate COCOMOII early design cost model. The calibrated model used to estimate the size of a new object oriented projects in OOFP according to counting method described in [5]. The calculated OOFP is used to estimate the equivalent KLOC according to proposed transformation function that is deductive by applying linear regression as the best technique to relate LOC to software metrics computed at the design stage (OOFPs) [10]. Finally the estimated LOC used to Estimate the effort, time, and cost of the new project. In this project the limitation of the data collected (10 projects) is due to the cost and schedule constraints. The proposed cost estimation tool need to be easy to use, and the cost estimation accuracy depends on the number of historical data used to calibrate the model and the design details of project classes.

2.2. Requirements Elicitation and Analysis

This section presents the services that the system should provide. The Requirements Elicitation and Analysis accomplished through the following process:

2.2.1. System Domain Understanding:

The software cost estimation system in this project restricted by the following aspects:

- 1- The cost estimation model used in the calibration is COCOMOII early design model.
- 2- Since the number of completed projects data will be less than 100 projects, So in the cost estimation model only the A coefficient will be calibrated. There will be no attempt at calibrating the Effort Multipliers (EM) since there will not be

- enough data points to justify it and not enough references describe how the Effort Multipliers rating values could be adjusted [23][3].
- 3- The calibration process applied using only Object Oriented projects.
 - 4- The projects used in the calibration and estimation process are small to medium scale of size object oriented projects.
 - 5- The object oriented projects should be written C++ language as consol application (not Visual object oriented projects)
 - 6- The estimation of the project size will be in Object Oriented Function Points (OOFP) instead of traditional Function point (FP) used in COCOMOII model.
 - 7- This proposed system can be used to estimate the size, effort and cost for object oriented projects at only design phase (when the project classes are determined)

2.2.2. Requirements Collection and Classification:

There are a lot of requirements needs to be organized into coherent clusters.

2.2.2.1.Organization Information

The software house that will use this system should feed it with their profile information (Company Name, Telephone, Fax, Labor Rate, Hours in Person Month and the maturity level of the procedures followed in the company), so these information can be used in the calibration process.

2.2.2.2.COCOMOII Early Design Model Calibration requirements:

- 1- Understanding COCOMOII Models
- 2- Collecting completed Object Oriented projects data (actual size, effort, cost drivers) from specific software house.
- 3- Compute Unadjusted object oriented Function Points (UOOFP) for each completed project according to counting procedure mentioned in [5] .
- 4- Compute adjusted object oriented Function Points (OOFP) from (UOOFP) after computing the adjustment factor.
- 5- Apply linear regression on the data collected form the previous projects to estimate the regression function to use it as transformation function for project size. This transformation function transform (OOFP) of the project to estimated Line Of Code (LOC).

2.2.2.3. New Project estimation requirements:

- 1- Feed the system with the properties of the new Object Oriented project (# of classes, # of simple and complex data in each class, # of inherited simple and complex data in each class, reference parameters in the method, single valued and multi valued association between methods)
- 2- Compute the size of the project in Unadjusted object oriented Function Points (UOOFPP) according to counting procedure mentioned in [5].
- 3- Converting Unadjusted object oriented Function Points (UOOFPP) to (OOFPP) by computing the adjustment factor.
- 4- Estimate the size of the project in Line Of Code (LOC) using the inference transformation function to relate LOC to (OOFPP) .
- 5- Feed the system with the seven Effort Multipliers EM1-7 (used in design phase), which affect the total cost of the project.
- 6- Estimate the effort required to finish this project using calibrated COCOMOII early design model

$$\text{Effort PM} = A * (\text{Size})^B * EM_1 * EM_2 * \dots * EM_7$$

A: is the constant (calibrated according to historical data of the information)

Size: the estimated thousand Line Of Code (KLOC)

B: Exponent value calculated for each project by considering five scale factors described in COCOMOII.

EM: is the effort multiplier according to COCOMOII values

- 7- Estimate the cost of the project in SR by the equation
- 8- Estimated Software Cost = Estimated Effort * Labor Rate of the organization.

Flow diagram of New Project estimation requirements:

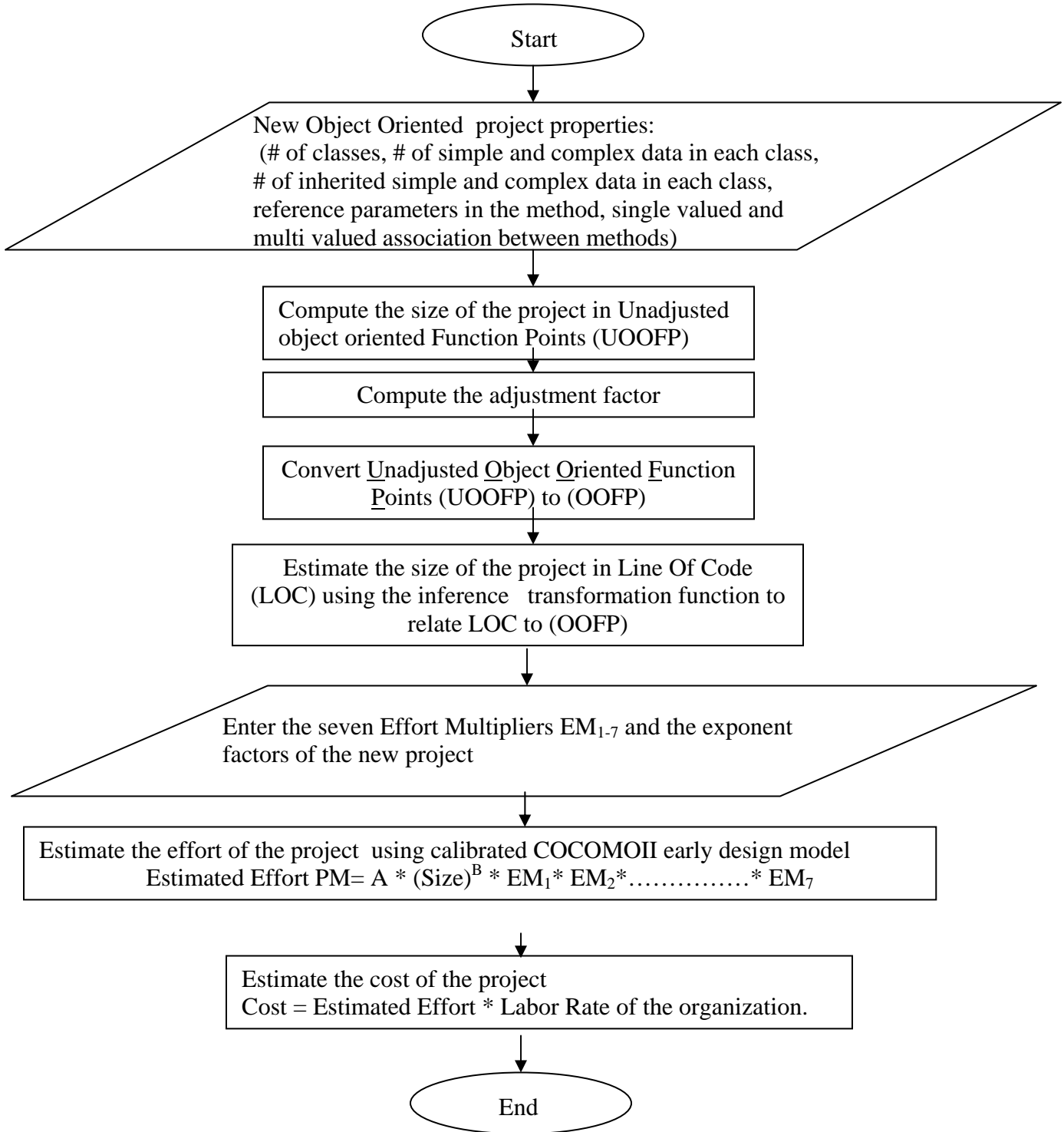


Figure 2: Flow diagram of New Project estimation

2.2.2.4. Software Cost Estimation Tool phases

The effort (PM) estimated for any new project from the following COCOMOII equation

$$PM = A * (Size)^B * \prod EM$$

A: is the constant (calibrated according to historical data of the information)

Size: the estimated thousand Line Of Code (KLOC)

B: Exponent value calculated for each project by considering five scale factors (Exp1.....Exp5) described in COCOMOII.

EM: is the effort multiplier according to COCOMOII values (EM1....EM17) **Phase 1: COCOMOII Early design Model Calibration sub system**

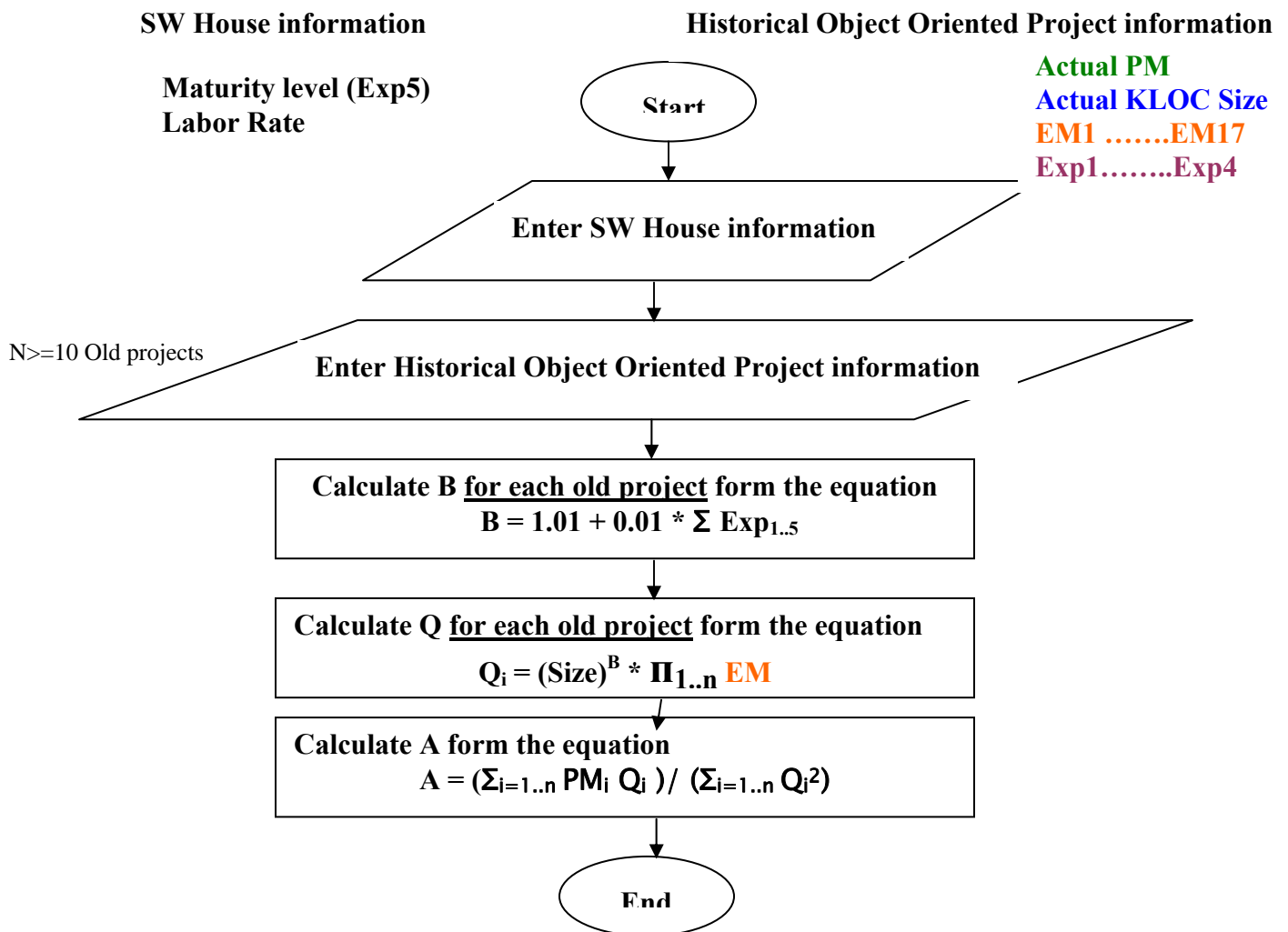


Figure 3: Flow Diagram of COCOMOII early design model calibration

Phase 2: Use Linear regression sub system to relate project size in OOF to LOC

Estimated LOC = $b_0 + b_1$ OOF

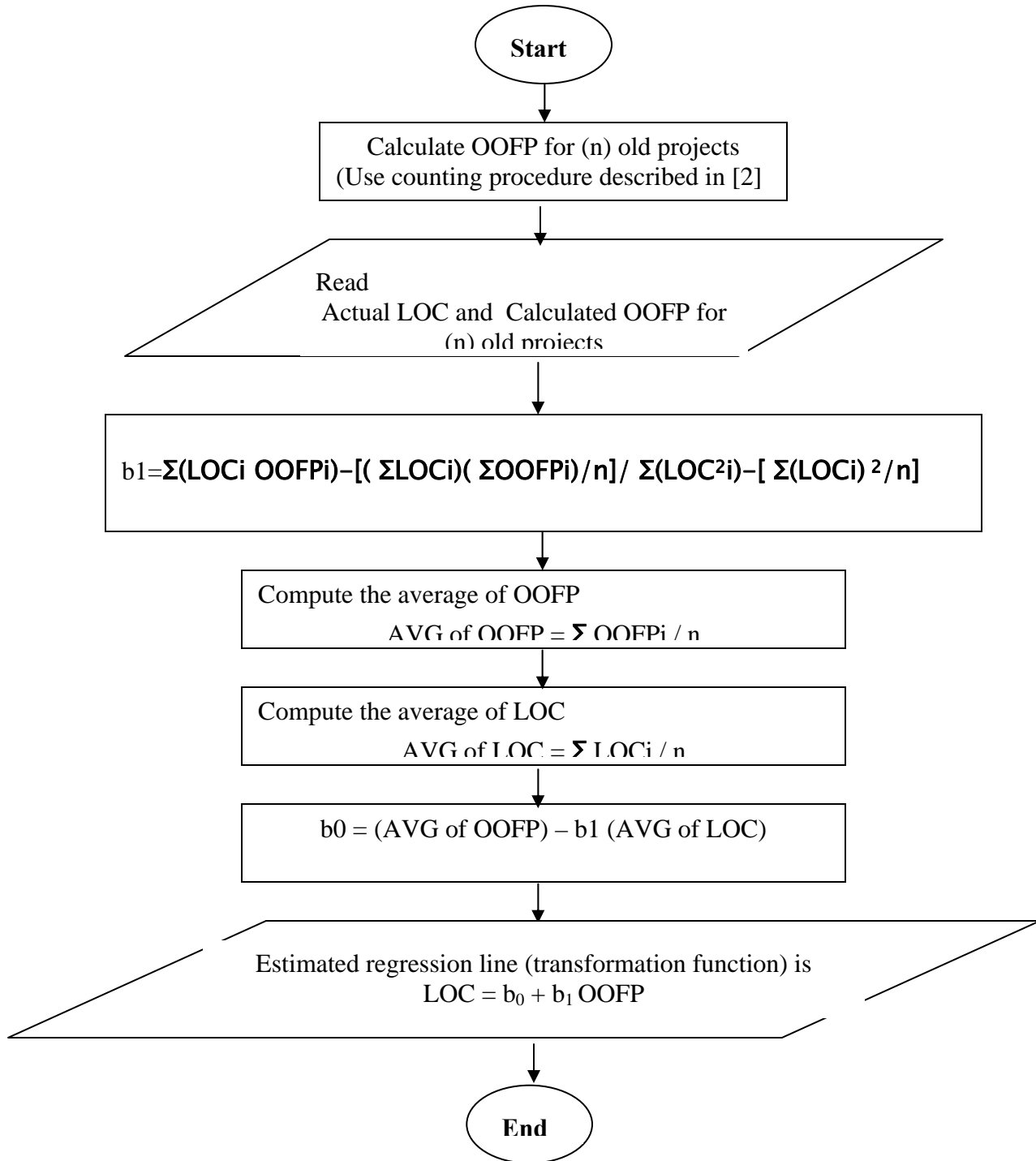


Figure 4: Flow Diagram of Linear regression to relate project size in OOF to LOC

Phase 3: Use Calibrated COCOMOII Early Design Model and regression model to estimate the effort for new project.

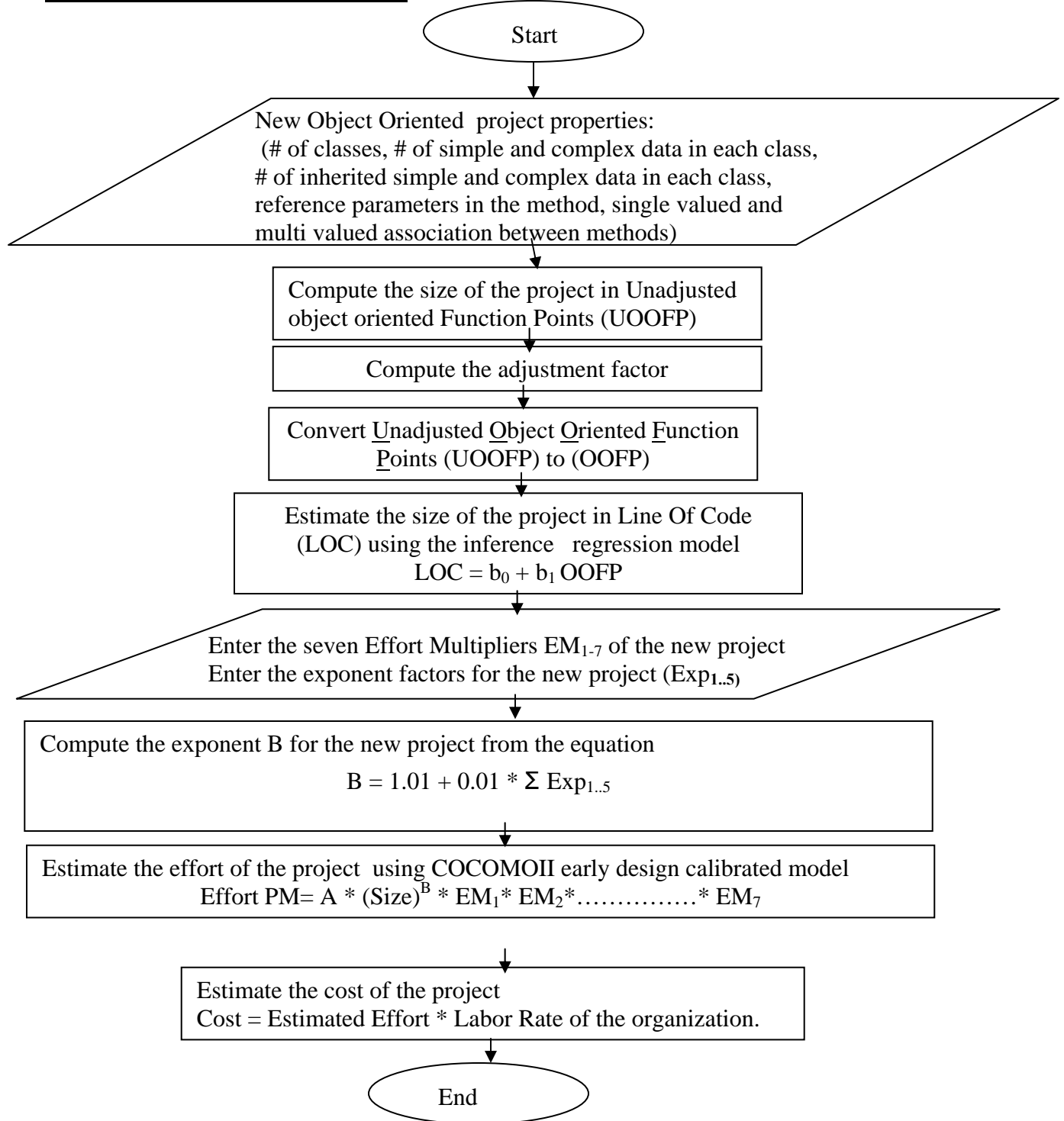
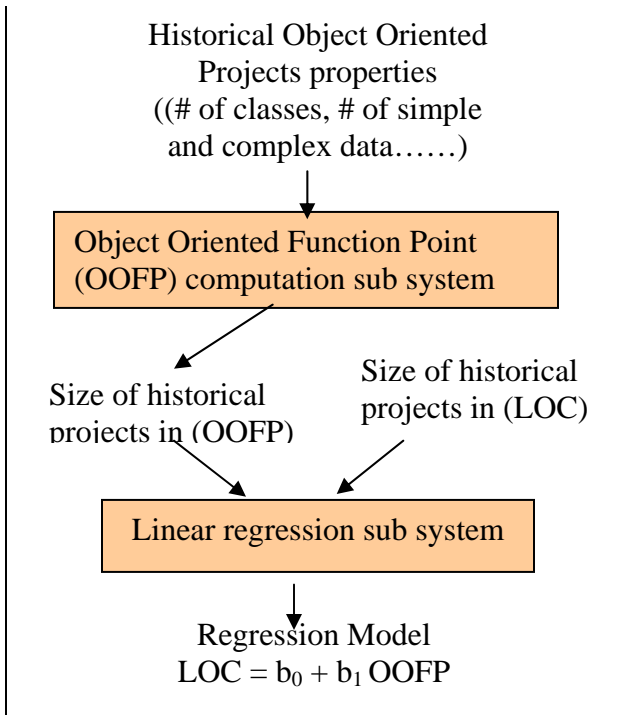


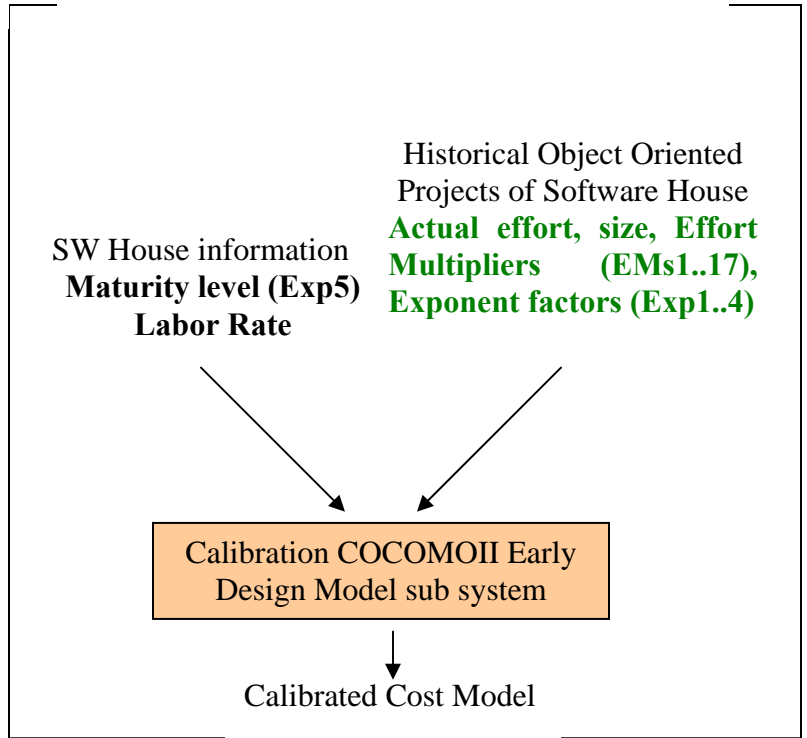
Figure 5: Flow Diagram (estimate the effort for new project)

Over View of the Cost Estimation System Phases.

Regression Model to Estimate the Project Size in LOC



Calibrated COCOMOII Early Design Model



New Object Oriented Project

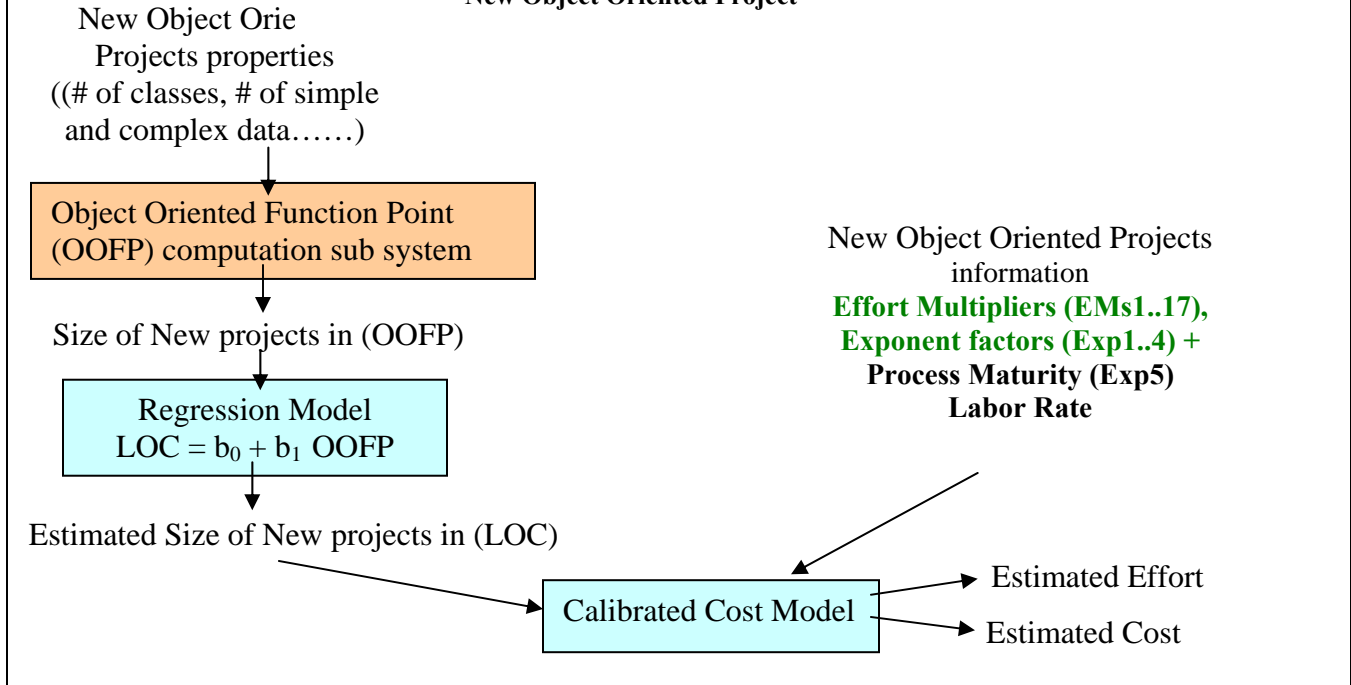


Figure 6: Over View of the Cost Estimation System Phases