ASSIGNMENT

SOFTWARE METRICS

COCOMO MODEL

Submitted by:

Sana Iqbal **2019/comp/MS (SE)/377**

January 2020

DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

JINNAH UNIVERSITY FOR WOMEN

5-C NAZIMABAD, KARACHI 74600

Table of Content

1. Economics

It is the study of production, distribution and consumption of goods and services. In short it is how people make decisions in resource-limited situations.

2. Software Engineering Economics

Software Engineering Economics is concerned with making decisions within the business context to align technical decisions with the business goals of an organization.

3. COCOMO Model (Constructive Cost Model)

It is a cost estimation model proposed by Berry Boehm in 1981. COCOMO predicts the effort and schedule for a software product development based on inputs relating to the size of the software and a number of cost drivers that affect productivity. COCOMO has three different models that reflect the complexity:

3.1 The Basic Model
3.2 The Intermediate Model
3.3 The Detailed Model

3.1 Basic COCOMO Model

Basic COCOMO estimates the software development effort using LOC (Lines of Codes). It is based on size of the project. The size of the project may vary depending upon the function points. It is good for quick, early, rough order of magnitude estimates of software costs. It does not account for differences in hardware constraints, personnel quality and experience, use of modern tools and techniques, and other project attributes known to have a significant influence on software costs, which limits its accuracy.

3.1.1 Basic COCOMO Equations

The basic COCOMO equations are:

$$E=a^{b} (KLOC \text{ or } KDSI)^{b_b}$$

$$D=c_b (E)^{d_b}$$

$$P=E/D$$

Where,

E- Effort applied in person-months,
D- Development time in months,
P- Number of people required,
KLOC/KDSI- estimated delivered lines of code for the projects,
a_b, b_b, c_b and **d_b**- Coefficients

Software projects	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Figure 1: Coefficients

Figure 1 shows the constant values that have been calculated from various projects.The basic cocomo gives the magnitude of cost of the project. It varies depending upon size of the project. The various classes of software projects are :

3.1.2 The Development Modes: Project Characteristics

COCOMO applies to the three classes of software projects:

i. Organic Projects

It is related to small projects with small development team. Team members should have an experience to work with less rigid requirements. Such as Payroll project.

ii. Semi-detached Projects

It is related to the projects of intermediate size and complexity with mixed experience team. Such projects may have mix of rigid and less than rigid requirements. Example: Banking System

iii. Embedded Projects

It is concerned with the projects that must be developed under tight hardware, software and operational constraints.

3.1.3 Example

Consider a software project using semi-detached mode with 30,000 lines of code. We will obtain estimation for this project as follows

i. Effort Estimation

$E=a_b (KLOC)^{b}_{b}$

$E_= 3.0(30)^{1.12}$ where LOC= 30000= 30KLOC

E= 135 person-month

ii. Duration Estimation

$D=C_b (E)^{d}_{b}$

$D= 2.5(135)^{0.35}$

D= 14 months

iii. Persons Estimation

P=E/D

P=135/14

P=10 persons approx.

The accuracy of this model is limited because it does not consider certain factors for software cost estimation. These factors are hardware constraints, personal quality and experience, modern techniques and tools. The estimates are within a factor of 1.3 only 29% of the time and within the factor of 2 only 60 % of time.

3.2 Intermediate Model

It is an extension of Basic COCOMO. This make use of set of cost driver attributes to compute software cost. It includes subjective assessments of product, hardware, personnel and project attributes. It can be applied to entire software product for easy and rough cost estimation during early stage and for obtaining more accurate cost estimation.

3.2.1 Classification of Cost Drivers and their attributes:

(i) Product attributes

- Required software reliability extent (RELY)
- Size of the application database (DATA)
- The complexity of the product (CPLX)

(ii) Hardware/ Computer attributes

- Run-time performance constraints (TIME)
- Memory constraints (STOR)
- The volatility of the virtual machine environment (VIRT)
- Required turnaround time (TURN)

(iii) Personnel attributes

- Analyst capability (ACAP)
- Software engineering capability (PCAP)
- Applications experience (AEXP)
- Virtual machine experience (VEXP)
- Programming language experience (LEXP)

(iv) Project attributes

- Use of software tools (TOOL)
- Modern programming practices (MODP)
- Required development schedule (SCED)

The intermediate COCOMO equation is:

EAF (*effort adjustment factor*)

$$E=aKLOC^{b} \times EAF \text{ person-month}$$

Mode	*a*	*b*
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

Figure 2: Effort Parameters for three modes of Intermediate COCOMO

The duration and person estimate is same as in basic COCOMO model i.e.

$$D=C_b (E)^{d}_{b} \quad \text{months}$$

$$P=E/D \quad \text{persons}$$

This model estimation is limited within 20% of actual and 68% of the time. Its effort multiplies are not dependent on phases and a product with many components is difficult to estimate.

3.2.2 Example

Consider a project having 30,000 lines of code which is an embedded software with critical area hence reliability is high. The estimation can be

$$E=a_i(KLOC)^{b}_{i} \times EAF$$

As reliability is high, EAF=1.15 (product attribute)

$a_i=2.8$, $b_i= 1.20$ (for embedded software)

$$E = 2.8(30)^{1.20} \times 1.15$$

$$= 191 \text{ person-month}$$

$$D = c_b(E)^{d}_{b} = 2.5(191)^{0.32}$$

$$= 13 \text{ months}$$

$$P=E/D \Rightarrow 191/13 \Rightarrow 15 \text{ persons approx.}$$

3.3 The Detailed/Advanced Model

The detail model uses the same equations for estimation as the intermediate model. It can estimate the effort (E), duration (D) and persons (P) of each development phases, subsystems and modules. This model allows the experimentation with different development strategies. Four phases used in it are:

i. Requirements planning and product design (RPD)
ii. Detailed design (DD)
iii. Code and unit test (CUT)
iv. Integrate and Test (IT)

Phases	Very low	Low	Nominal	High	Very high
RPD	1.80	0.85	1.00	0.75	0.55
DD	1.35	0.85	1.00	0.90	0.75
CUT	1.35	0.85	1.00	0.90	0.75
IT	1.50	1.20	1.00	0.85	0.70

Figure 3: Effort Multipliers for Detailed COCOMO

4. Function Point (FP)

Function point analysis (FPA) is a methodology for measuring software productivity and the cost associated with the development and maintenance. One function point (FP) is one end-user requested business function. The following defines the five characteristics of function points:

- **External Inputs**: these are end-user actions such as putting in a login or executing a mouse click.
- **External Outputs**: the system provides the end-user output or interface such as a GUI display or items in a report.

- **Logical Internal Files**: these files are the master or transaction files that the system interacts with during its session.

- **External Interface Files**: unlike logical internal files, where the application uses solely for its purpose, these files are or databases are shared with other applications or systems.

- **External Inquiries**: this function is initiated by the end-user. For example, the end-user wishes to submit a query to a database or requests on-line help. In any case the developer provides a means for the end-user to "search" for answers.

4.1Counting Function Point (FP):

- **Step-1:**

  F = 14 * scale

  Scale varies from 0 to 5 according to character of Complexity Adjustment Factor (CAF). Below table shows scale:

  0 - No Influence

  1 - Incidental

  2 - Moderate

  3 - Average

  4 - Significant

  5 - Essential

- **Step-2:** Calculate Complexity Adjustment Factor (CAF).

  CAF = 0.65 + (0.01 * F)

- **Step-3:** Calculate Unadjusted Function Point (UFP).

Table 1: Function Point Table

Function Units	Low	Average	High
EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EIF	5	7	10

Multiply each individual function point to corresponding values in TABLE.

- **Step-4:** Calculate Function Point.

  FP = UFP * CAF

4.2Example

Given the following values, compute function point when all complexity adjustment factor (CAF) and weighting factors are average.

User Input = 50

User Output = 40

User Inquiries = 35

User Files = 6

External Interface = 4

Explanation:

- **Step-1:** As complexity adjustment factor is average (given in question), hence,
- Scale = 3.

  F = 14 * 3 = 42

- **Step-2:**
  CAF = 0.65 + (0.01 * 42) = 1.07
- **Step-3:** As weighting factors are also average (given in question) hence we will multiply each individual function point to corresponding values in TABLE.
  UFP = (50*4) + (40*5) + (35*4) + (6*10) + (4*7) = 628
- **Step-4:**
  Function Point = 628 * 1.07 = 671.96

5. Object Point

Object Point is similar to FP. It computes the number of screens and classify them as simple, medium, complex. It computes the number of reports and classify them as simple, medium, complex. It counts the number of modules that have to be developed. It uses weight matrices to sum the values above, taking into account reused code.

Table 2: Object Point Data

Object Type	**Complexity Weights**		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL Components	-	-	10

5.1Example

Screen = 15 Screens

Reports = 10 Reports

3GL Components = 15GL components

Evaluating the above values in Table 2

Object Type	**Complexity Weights**			
	Simple	Medium	Difficult	Total
Screen	1 x 5 +	2 x 5 +	3 x 5	30
Report	2 x 0 +	5x 10 +	8x 0	50
3GL Components	-	-	10 x 15	150
			Object Point =	**230**

5.2New Object Point

New object point (NOP) = (Object Point) x (100 – Reuse Percentage/ 100)

Having Reuse Percentage: 33%, Therefore:

New object point (NOP) = (230) x (100-33% / 100)

New object point (NOP) = (230) x (0.67)

New object point (NOP) = 154.1

6. Story Point

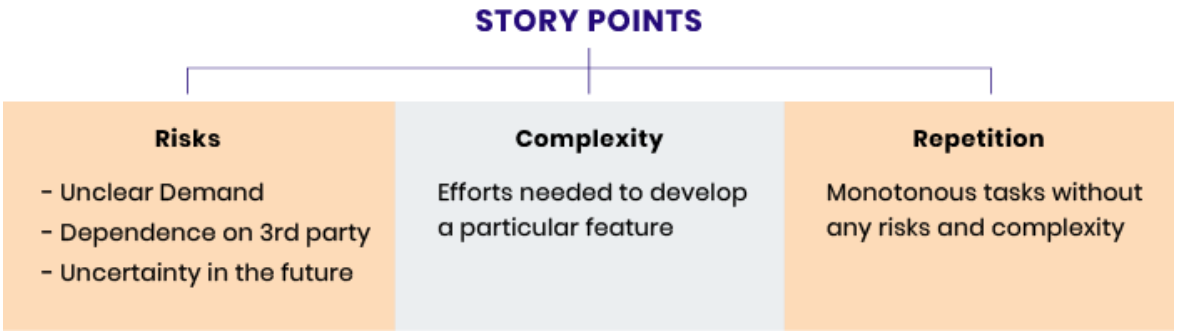
A story point is a metric used in agile project management and development to estimate the difficulty of implementing a given user story, which is an abstract measure of effort required to implement it. In simple terms, a story point is a number that tells the team about the difficulty level of the story. Difficulty could be related to complexities, risks, and efforts involved. The Story Points approach uses historical data to compare features of one project to features of a previous similar project to generate a precise estimate.

The general characteristics of story points include:

- Story points represent the total amount of work required to fully implement a user story.
- The stories are estimated independently by team members and the team drives toward a consensus opinion.
- If a user story is too large to be implemented in one iteration it needs to be broken down into two or more smaller stories.
- Many of the estimating models are designed as games that are interesting and engaging for the project team.

Step 1 — Identify a Base Story

Story Points in agile are a complex unit that includes three elements: risk, complexity and repetition.



To find our Base Story, we search for one elementary task that corresponds to internal standards of Definition of Done for User Stories and assign it one Story Point. This will be our Base Story.

Step 2 — Create a Matrix for Estimation

There are two types of scales used for creating estimation matrices: the linear scale (1,2,3,4,5,6,7…) and Fibonacci sequence numbers (0.5, 1, 2, 3, 5, 8, 13 …), T-Shirt Sizing (X-Small, Small, Medium, Large, Extra-Large).

Base Story is already in this matrix in the first row with a value of one Story Point as shown in Figure 4.

StoryPoints	Story
1	As a site visitor, I want to visit the About page to know more about services
2	
3	
5	
8	

Figure 4: Story Point Metric

There can be several stories in one row as shown in Figure 5.

StoryPoints	Story
1	As a site visitor, I want to visit the About page to know more about services
	As a site visitor, I want to be able to reset my password in case I forget it
2	As a logged in user, I want to be able to see a history of my payments on the settings page
	As a site visitor, I want to be able to send feedback or report a problem using the Contact Us form
3	As a site visitor, I want to log in / sign up using my email / password
	As a logged in user, I want to be able to add comments to content on the website
5	As a site visitor, I want to use the Search form with filters to search for specific content
	As a site visitor, I want to see detailed information about content
8	As a logged in user, I want to be able to add content on the website: title, description, media content (images, video, audio), geolocation
	As a logged in user, I want to be able to communicate via messages with other users

Figure 5: Story Points with Several Stories

7. COCOMO-II

It is the revised version of the basic Cocomo (Constructive Cost Model) and is developed at University of Southern California. It is the model that allows one to estimate the cost, effort and schedule when planning a new software development activity. It consists of three sub-models:

7.1 Sub-Models

- **End User Programming**

Application generators are used in this sub-model. End user write the code by using these application generators.

Example – Spreadsheets, report generator, etc.

- **Intermediate Sector:**

  It consists of:

  (a). Application Generators and Composition Aids

  This category will create largely prepackaged capabilities for user programming. Their product will have many reusable components. Typical firms operating in this sector are Microsoft, Lotus, Oracle, IBM, Borland, Novell.

  (b). Application Composition Sector

  This category is too diversified and to be handled by prepackaged solutions. It includes GUI, Databases, and domain specific components such as financial, medical or industrial process control packages.

  (c). System Integration

  This category deals with large scale and highly embedded systems.

- **Infrastructure Sector**

  This category provides infrastructure for the software development like Operating System, Database Management System, User Interface Management System, Networking System, etc.

7.2 Stages of COCOMO II:

Stage-I:
It supports estimation of prototyping. For this it uses *Application Composition Estimation Model*. This model is used for the prototyping stage of application generator and system integration.

Stage-II:
It supports estimation in the early design stage of the project, when we less know about it. For this it uses *Early Design Estimation Model*. This model is used in early design stage of application generators, infrastructure, and system integration.

Stage-III:
It supports estimation in the post architecture stage of a project. For this it uses *Post Architecture Estimation Model*. This model is used after the completion of the detailed architecture of application generator, infrastructure, and system integration.

7.3 Cost Drivers

COCOMO II has 17 cost drivers used to assess project, development environment, and team to set each cost driver. The cost drivers are multiplicative factors that determine the effort required to complete your software project. For example, if your project will develop software that controls an airplane's flight, you would set the Required Software Reliability (RELY) cost driver to Very High. That rating corresponds to an effort multiplier of 1.26, meaning that your project will require 26% more effort than a typical software project.

Table 3 COCOMO II Cost Drivers

Attribute	Type	Description
RELY	Product	Required system reliability
CPLX	Product	Complexity of system modules
DOCU	Product	Extent of documentation required
DATA	Product	Size of database used
RUSE	Product	Required percentage of reusable components
TIME	Computer	Execution time constraint
PVOL	Computer	Volatility of development platform
STOR	Computer	Memory constraints
ACAP	Personnel	Capability of project analysts
PCON	Personnel	Personnel continuity
PCAP	Personnel	Programmer capability
PEXP	Personnel	Programmer experience in project domain
AEXP	Personnel	Analyst experience in project domain
LTEX	Personnel	Language and tool experience
TOOL	Project	Use of software tools
SCED	Project	Development schedule compression
SITE	Project	Extent of multisite working and quality of inter-site communications

7.4 Scale Drivers

In the COCOMO II model, some of the most important factors contributing to a project's duration and cost are the Scale Drivers. You set each Scale Driver to describe your project; these Scale Drivers determine the exponent used in the Effort Equation. The 5 Scale Drivers are:

- Precedentedness
- Development Flexibility
- Architecture / Risk Resolution
- Team Cohesion
- Process Maturity

References

[1] A. A. Puntambekar, *Software Engineering*, Third Revi. Technical Publications Pune, 2009.

[2] P. Sharma, *Software Engineering*. APH Pubishing Corporation, 2004.

[3] B. W. Boehm, *Software Engineering Economics*.

[4] COCOMO Tool. http://groups.umd.umich.edu/cis/course.des/cis525/js/f00/kutcher/kutcher.html