Delhi Technological University

Bawana Rd, Shahbad Daulatpur Village,
Rohini, New Delhi
Delhi - 110042

DEPARTMENT OF SOFTWARE ENGINEERING

Software Project Management [SE-405]

PROJECT REPORT

Comparison of Software Project Cost Estimation Techniques and Models

Submitted to:

Ms. Kishwar Khan

Team Members:

Aryan Bansal - 2K18/SE/038

Ashish Kumar - 2K18/SE/041

Table of Contents

ABSTRACT

In the field of software engineering SDCE (Software Development Cost Estimation) is considered as one of the interesting fields. The success of a software project depends on whether it is aligned with the customer requirements and that it is completed within the deadline, within the given budget allocated for the project. Due to this very reason, finding a model that is accurate and reliable in determining the cost of a software project is of utmost importance.

This study describes SDCE by exploring its techniques and models and then compares all of them in one place. In this study, we've reviewed various software development effort and cost estimation models and techniques. After that, the techniques are divided into various categories based on their approach of finding the cost of a software project. Some of these categories are parametric models, expertise-based techniques, learning-oriented techniques, dynamics-based models, regression-based techniques, fuzzy logic-based methods, size-based estimation models, composite techniques, etc.

We have analyzed the results by enlisting the various strengths and limitations associated with every technique / model that is taken up in the study. In the conclusion part, we will be deciding the best model according to our study of the various models.

Based on our research, we recommend a hybrid approach for SDCE as not even a single technique is ideal and reliable to predicting the cost of a software project beforehand. The hybrid model can be chosen by prioritizing which pros are mandatorily needed in the software project and which cons can be compromised. This hybrid approach will significantly improve the existing approaches since one model's strengths can counter the other model's weaknesses, thus providing a near-to-perfect model based on the given design and implementation conditions of the project.

The later sections in this project report contains the following components in detail: Objective of the study, Motivation, Introduction to SDCE, Methodology, Results Analysis, Conclusion, Future Work, Contribution of the team members and References.

Objective of the Study

The study is divided into three phases. In the first phase, we will be gathering information about the various conventional as well as the newer software development cost estimation methods and classifying them based on their approaches. We will also list the pros and cons of each one of them. This step can also be seen as a comparison between different cost estimation techniques and models. In the last phase, based on the results of the above two phases, we will be finding out the best ideal technique of software cost estimation. The phases of the study are summarized below:

1. We will be categorizing software development cost estimation techniques and models into different categories and list down the strengths and weaknesses of each one of them.

2. The study will be useful in comparing different models, which will further aid the project managers in choosing the correct model according to given situations and environments.

3. The comparative study will include the conventional and the newer models, thus assisting in gathering information about the evolution of software project management. Moreover, this will lead to a better understanding of the recent trends in the cost estimation field.

4. Listing the pros and cons of every model and technique will assist the stakeholders and project managers in finding the best approach for their software project.

5. Based on the research results, we will be deciding which model is the ideal model in estimating the cost of a given software project.

MOTIVATION

- Software development cost estimation (SDCE) has always been an interesting and budding field in Software Engineering. This study will support the SDCE by exploring its techniques and models and collecting them in one place.

- This contribution in the literature will assist future researchers to get maximum knowledge about SDCE techniques and models and save their time. Listing the pros and cons of every model and technique will also assist the stakeholders in finding the best approach for their software project.

- Furthermore, this study will allow us to gain a deep understanding and insights into the evolution of cost estimation methods and increase our knowledge in this domain.

INTRODUCTION

SDCE is a primary activity in project management to manage resources in an effective manner by predicting the required number of resources to fulfill a given task.

The existing software cost estimation models & techniques are categorized into 6 major categories: parametric models, learning-oriented techniques, expertise-based techniques, regression-based models, dynamics-based models, and Composite-Bayesian techniques. A lot of research has been carried out on this and the vital issue which is closely related to the software projects financial aspects is to determine the accurate estimation of software cost. There exists a relationship between the software estimation and cost of software i.e., the primary factor for software cost is effort.

Several research studies were done for surveying effort and cost estimation techniques. They discuss only popular techniques i.e., COCOMO model, function points, SLIM, NN (Neural Networks), regression-based techniques, etc. Thus, there is a need to take maximum no. of classical as well as the latest techniques and to provide their overview. All techniques are mentioned category wise in the comparison table. This study is undertaken to study the maximum no. of SDCE techniques.

The remaining part of this introduction is organized as follows: Next Section describes all the techniques i.e., Parametric techniques for cost estimation, Expertise-based techniques, Learning-oriented techniques, Dynamics-based techniques, Regression-based techniques and fuzzy logic-based methods, Size-based estimation techniques and Composite techniques.

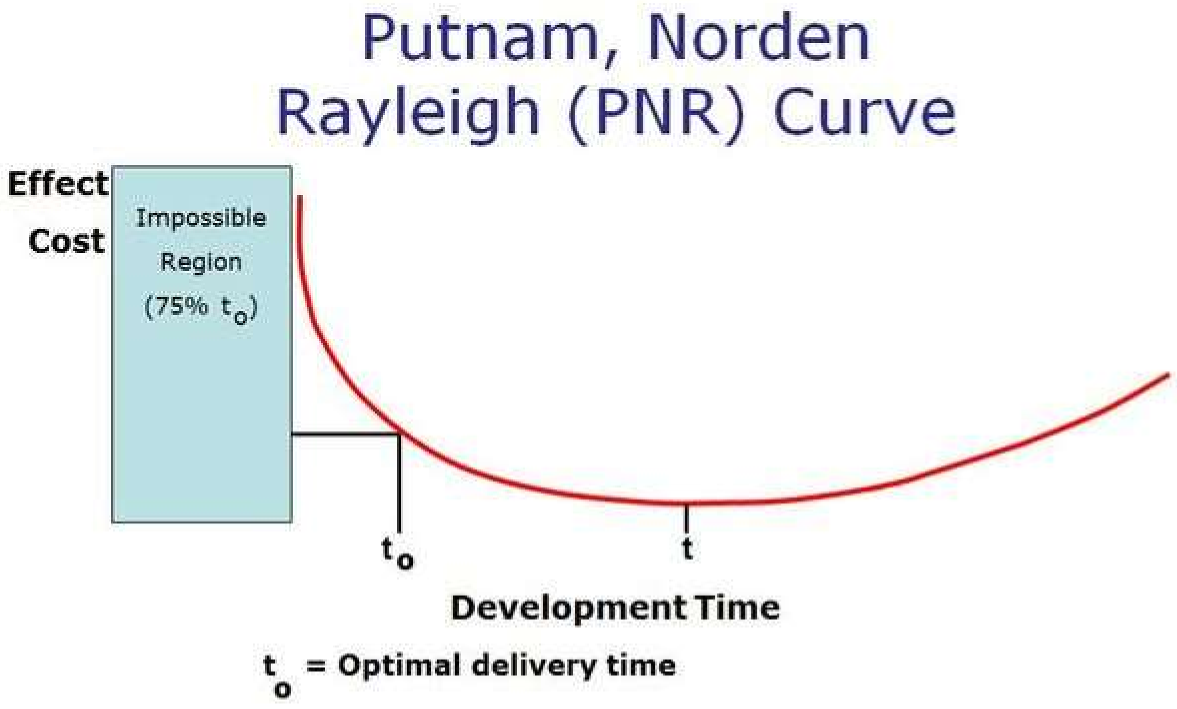
1. PARAMETRIC / ALGORITHMIC TECHNIQUES

For generating cost estimates as a function of major cost factors we used Algorithmic models. If E represents effort, and CF represents cost factors, then the algorithmic model will get the form as in the below equation:

$$E = f(CF_1, CF_2, CF_3, \cdots, CF_n)$$

a) **<u>Software Life-Cycle Model (SLIM)</u>:** A software lifecycle model, also known as the Putnam model, is an empirical software effort estimation model developed. SLIM takes LOC (Line of Code) as input to estimate software cost. Since this model assumes that resource consumption will vary with time, we can model it by using the well-known

Rayleigh distribution. To depict the interdependence of project's application effort and estimated delivery date, the PNR (Putnam Norden Rayleigh) curve formula is used.



The Rayleigh equation which is used to derive the software equation is written in below equation:

$$\frac{dy}{dt} = 2Kate^{at^2}$$

In the above equation, K represents the area of the curve, a = (1/2td^2), and td is the time at which dy/dt is at peak. Now, putting the value of a = (1/2td^2) in below equation, the Rayleigh equation will get the form of equation:

$$\frac{dy}{dt} = 2K(\frac{1}{2t_d^2})te^{-\left(\frac{1}{2t_d^2}\right)t^2}$$

b) **<u>COCOMO (Constructive Cost model)</u>:** The COCOMO model was developed in 1981. COCOMO is one of the most used software estimation models in the world. COCOMO calculates the efforts and schedule of a software product which is based on the software size. There are three sub-models of Cocomo: Basic Cocomo, Intermediate Cocomo, and Detailed Cocomo. Basic COCOMO calculates effort (person-months) and costs as a function of the program size in thousands of estimated delivered source instructions (KDSI). The basic COCOMO Equation is written below:

$$MM = a(KDSI)^b$$

The below equation is used to get development time:

$$t_{dev} = c(MM)^d$$

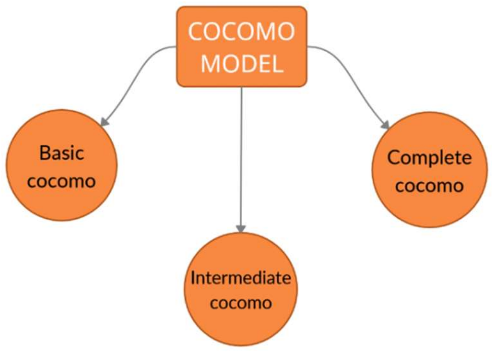
where, MM is man-month / person-month i.e. effort of one person in one month, KDSI is a measure of size, t is development time, the values of a, b, c, and d are dependent on the mode of development.

Intermediate COCOMO considers those cost factors that were missing in basic COCOMO. It calculates the effort as a function of program size and the four costs consist of a subjective assessment of products, personnel, hardware, and projects, each with several additional attributes, a total of 15 cost drivers/attributes. The intermediate COCOMO equation is written below:

$$MM = a(KDSI)^b C$$

where MM is man-month / person-month, KDSI is the number of thousand delivered source instructions, the coefficient a and exponent b depend upon the mode of development.

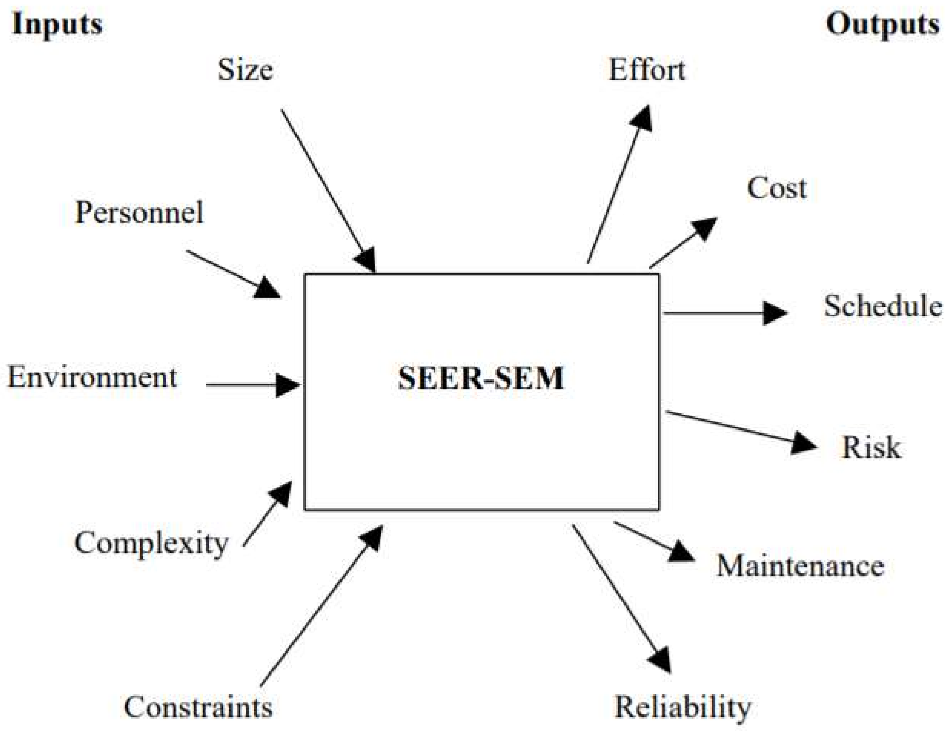
Detailed COCOMO considers all qualities of the standard version on each method of the software process. In detailed cocomo, the whole software is divided into several modules, and then we apply COCOMO in various modules to estimate effort and then sum the effort.



c) **<u>SEER-SEM:</u>** The Software estimation model (SEER-SEM) is a system evaluation and resource estimation product. It is a software project estimation model which is commercially available and widely used. SEER-SEM began with the Jensen model and it estimates the effort, cost, risk, and schedule of a project while covering all phases of SDLC. The effort is calculated using following Equation:

$$K = D^{0.4} \times \left(\frac{S_e}{C_{te}}\right)^{1.2}$$

where D is staffing complexity, S represents effective size, C indicates effective technology.



Inputs and Outputs for SEER-SEM Technique

d) <u>Checkpoint</u>: A checkpoint tool is a knowledge-based software project estimation tool. Software productivity research (SPR) developed checkpoints. It has its own large database of thousands of software projects. It uses functional points as the measure of size. Checkpoint supports three major capabilities of SDLC i.e., estimation, measurement, and assessment.

a) Estimation

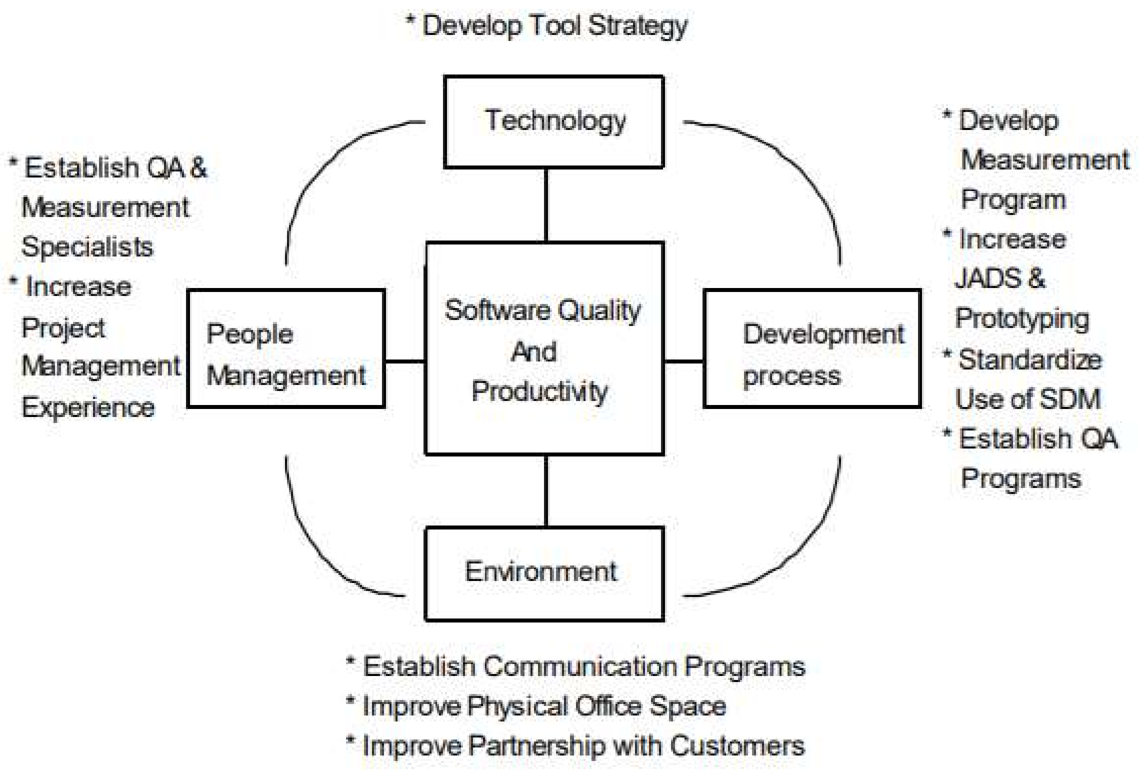
Checkpoint predicts effort at four levels of granularity: project, phase, activity, and task. Estimates also include resources, deliverables, defects, costs, and schedules.

b) Measurement

Checkpoint enables users to capture project metrics to perform benchmark analysis, identify best practices, and develop internal estimation knowledge bases (known as Templates).

c) Assessment

Checkpoint facilitates the comparison of actual and estimated performance to various industry standards included in the knowledge base. Checkpoint also evaluates the strengths and weaknesses of the software environment. Process improvement recommendations can be modeled to assess the costs and benefits of implementation.

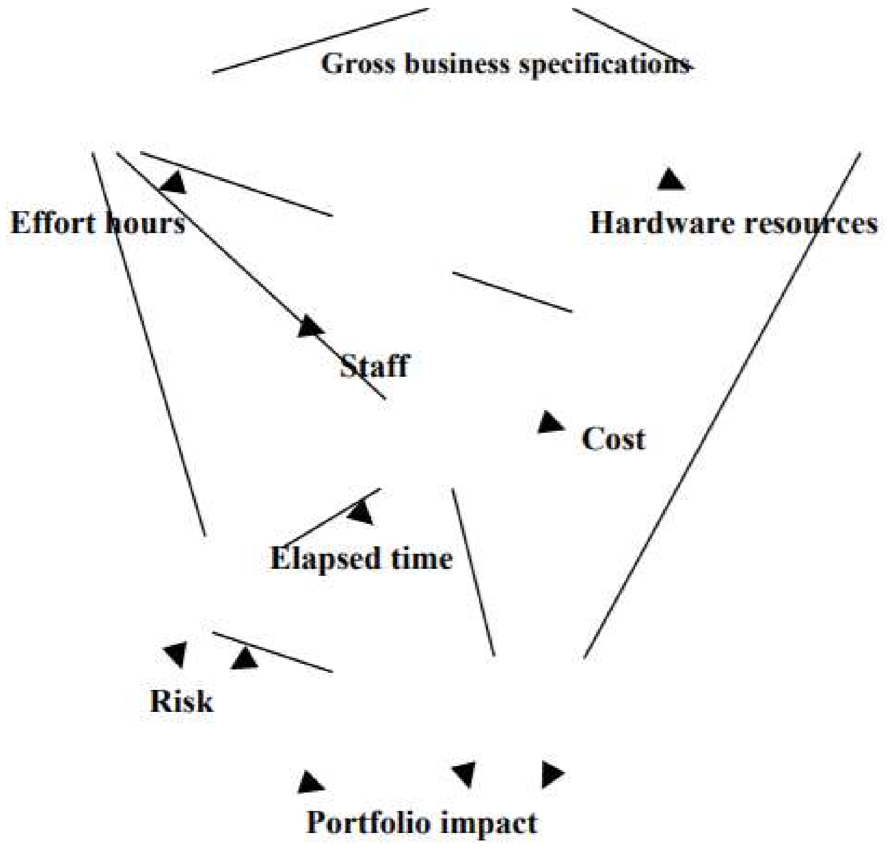

* Develop Tool Strategy
Technology
* Establish QA & Measurement Specialists
* Increase Project Management Experience
People Management
Software Quality And Productivity
Development process
* Develop Measurement Program
* Increase JADS & Prototyping
* Standardize Use of SDM
* Establish QA Programs
Environment
* Establish Communication Programs
* Improve Physical Office Space
* Improve Partnership with Customers

SPRs Summary of Responses

e) **<u>ESTIMACS</u>:** It was originally developed as QUEST (Quick Estimation System) and later it had been integrated into the product line of MACS as ESTIMACS. It focuses on the software development phase of SDLC. As ESTIMACS is a proprietary model that means internal details like expressions being used cannot be used.

This model consists of 5 different models: system development effort estimate, staffing and cost estimate, hardware configuration estimate, risk estimate, and portfolio analysis. These models are used sequentially so that the output from one model is often used as an input to the next model. The critical estimation dimensions are as follows:

- Effort hours
- Staff size and deployment
- Cost
- Hardware resource requirements
- Risk
- Portfolio impact

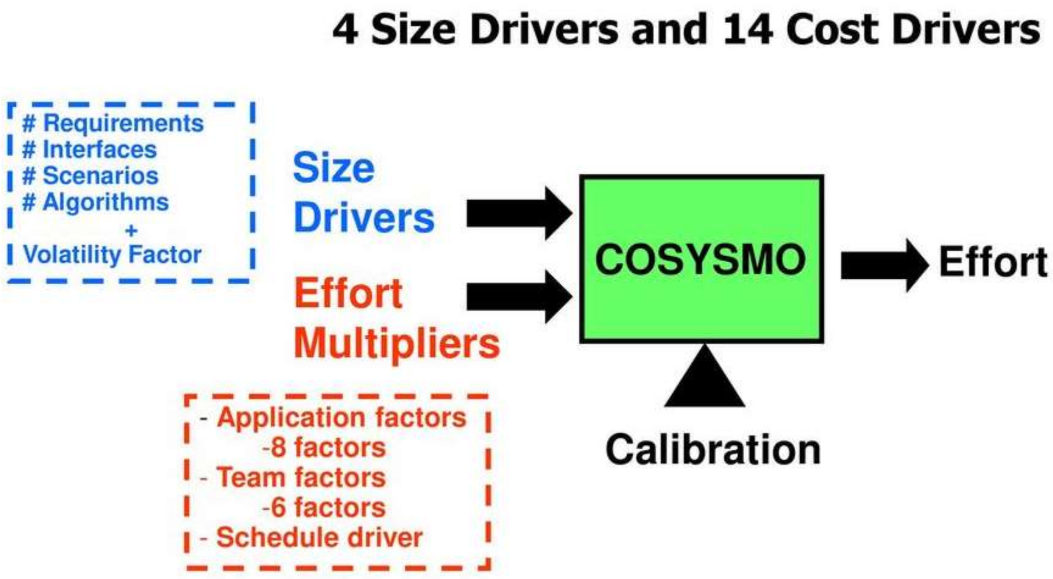


Rubin's Diagram of Interdependence of Estimation Dimensions

f) **PRICES-S (Parametric Review of Information for Cost Accounting and Evaluation Software)** : It was developed by Radio Corporation of America (RCA). After that it was released as a unique model in the 1970s. The PRICE-S tool is currently sold by private company PRICE - Systems. This model uses a 2-parameter distribution technique instead of a Rayleigh curve model in order to estimate the relationship between development effort distribution and calendar time.

The PRICE - S model has three sub-models, an acquisition sub-model, a sizing sub-model, and a life cycle cost sub-model. The acquisition sub-model predicts the cost and schedule of the software, the sizing sub-model helps in estimating the size of the software, the life cycle cost sub-model estimates the cost of the maintenance.

g) **COSYSMO** (Constructive system engineering cost model (COSYSMO) is a parametric model. COSYSMO is the advanced member of the COCOMO family of software cost estimation models. It calculates the effort and time which is required to perform system engineering tasks. This model contains 14 effort multipliers and 4 size factors, a total of 18 parameters.



The COSYSMO consists of three main processes i.e. Strawman COSYSMO, COSYSMO-IP, and COSYSMO. Strawman COSYSMO was the first major version of COSYSMO having 16 cost drivers. COSYSMO-IP was known as the second spiral of COSYSMO derivation consisting of a revised set of cost drivers. A general form of the model, mentioned in the below equation, was proposed which contains three parameters of different types i.e., additive, exponential, and multiplicative.

$$PM = A \times (\text{Size})^{E} \times (EM)$$

where PM is effort in person-months, A represents calibration factor, Size denotes the measure(s) of functional size of a system, and EM (multiplicative parameter) indicates effort multipliers which influence systems engineering effort. The third spiral of COSYSMO derivation is referred to simply as COSYSMO.

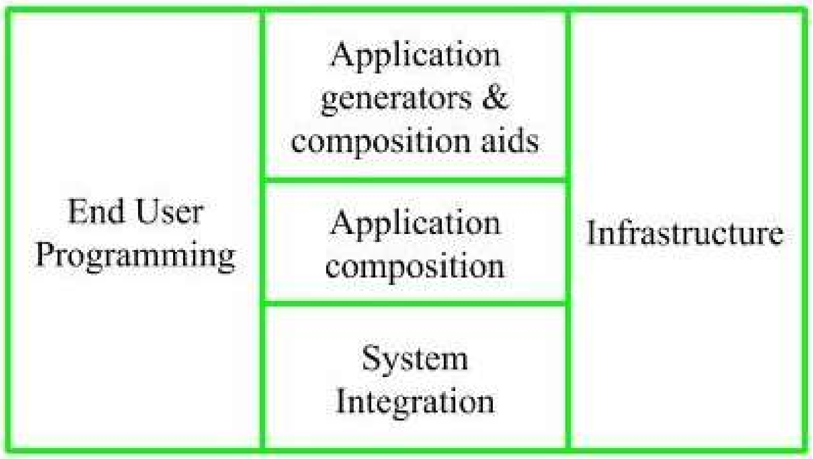
h) **COCOMO-II:** COCOMO-II is the advanced version of the original COCOMO (Constructive Cost Model) and was developed at the University of Southern California in 1995. It estimates the cost, effort and schedule when planning a new software development activity. It consists of three sub-models for estimation of software projects: Applications Composition sub-model, early design sub-model, and post-architecture sub-model.

The application composition model is based on the concept of Object Points. An Early design sub-model is a sub-model that is used to calculate estimates of the cost and duration of a project. Post-architecture sub-model is used after the entire construction of the project which has been developed and established. It used SLOC and FP as size metrics. The post architecture model provides more adequate cost estimates.

COCOMO II estimation model is summarized in the following equation used for both early design and post architecture model for Person-effort estimation.

$$PM = A \times \text{Size}^{E} \times \prod_{i=1}^{n} EM_i$$

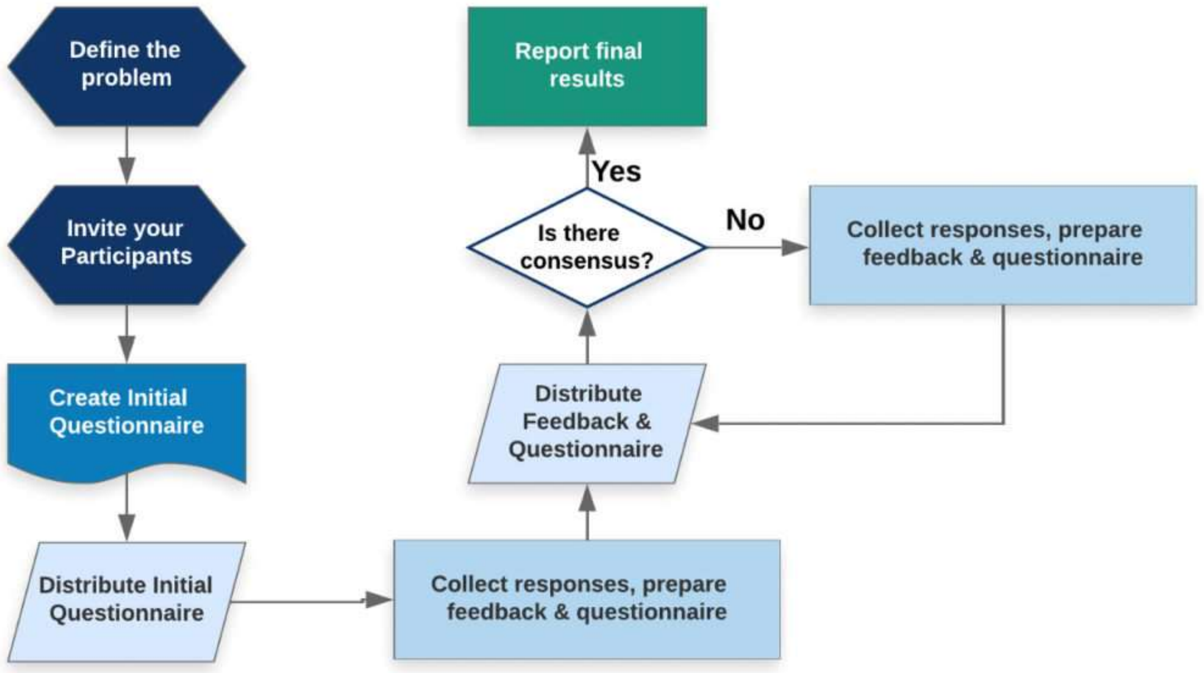
where PM is the effort in person-months, A is the constant and size is the size of software, an exponent E and several effort multipliers (EM), which depend on the model.



2. EXPERTISE-BASED / CONSENSUS BASED TECHNIQUES

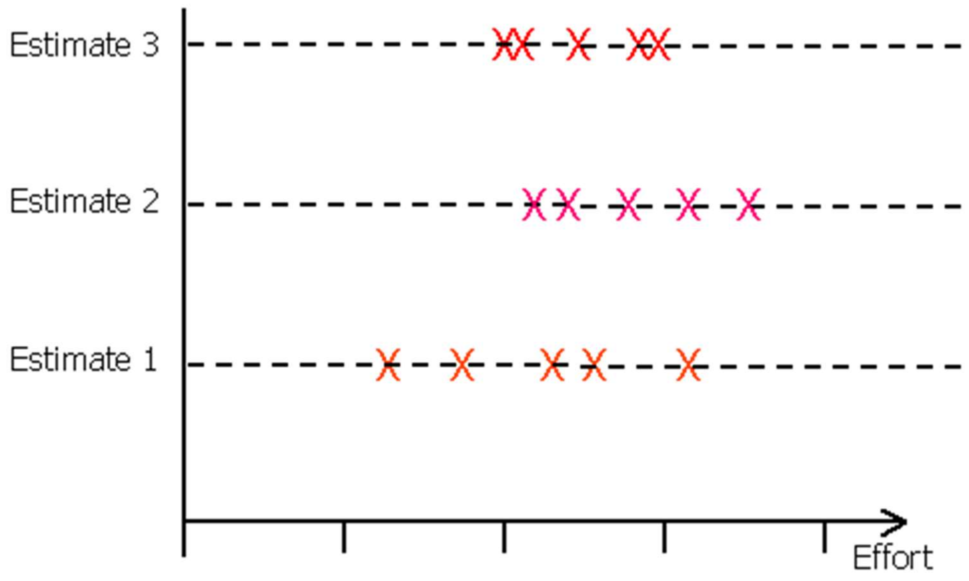
Expertise-based estimation techniques are based on the ability of one or more people, called experts in software development, to work to estimate software development efforts. The various techniques are:

a) **Delphi Technique:** Delphi's method is a very popular method developed in 1940 for predicting future events. It is used to guide groups of people on certain issues by combining opinions from experts. In this method, special meetings are held among project specialists to obtain true information. After each round of questionnaires, the experts are presented with a summary of the all rounds, allowing each expert to present their point of views according to the group response.

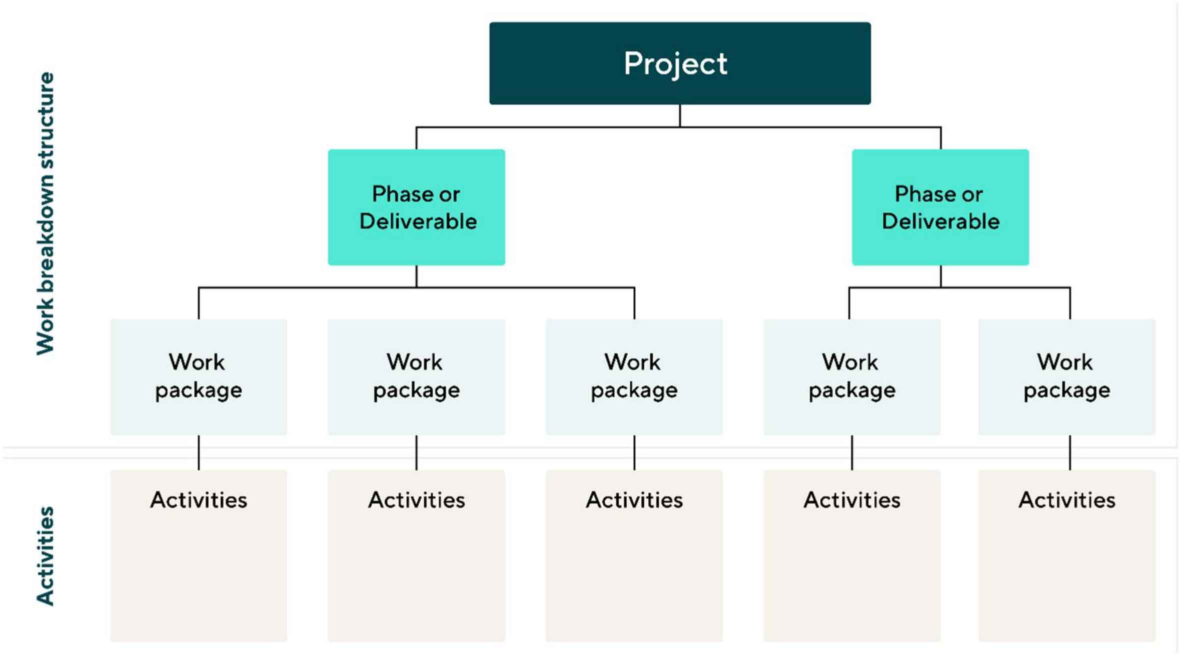

Define the problem
Invite your Participants
Create Initial Questionnaire
Distribute Initial Questionnaire
Collect responses, prepare feedback & questionnaire
Distribute Feedback & Questionnaire
Is there consensus?
Yes
Report final results
No
Collect responses, prepare feedback & questionnaire

Flowchart for Delphi method

b) **Wideband Delphi Technique:** Wideband Delphi is the improved form of the Delphi methods Technique. Compared to the original Delphi, the Wideband Delphi method includes a group discussion between evaluation rounds. This is a consensus-based estimation technique for estimating effort by a group of experts and functions.

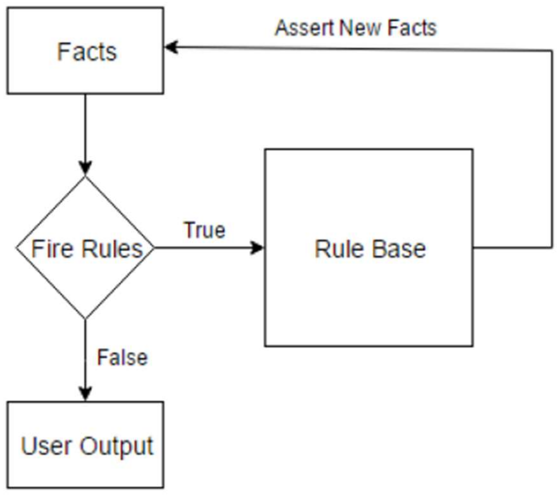


c) **<u>Work Breakdown Structure (WBS):</u>** It is a way to arrange project elements in a hierarchy. To identify the individual tasks, it divides whole work performed by the project team into smaller subsystems. The WBS method generates two kinds of hierarchies: The first one represents a software product and the other represents the activities required to build the product. The product hierarchy shows the basic structure of the software, i.e, how different software components characterized the entire system and the activity hierarchy specifies the activities related with a software component.



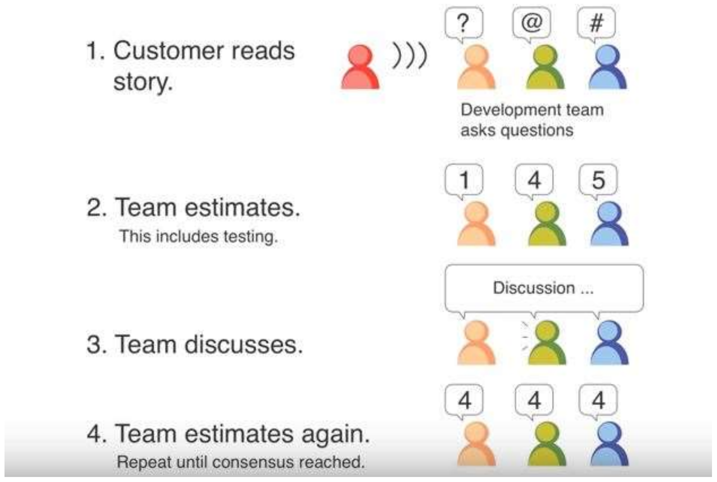
Work-Breakdown Structure Method

d) **<u>Rule-Based Systems:</u>** Rule-based systems use the knowledge of human experts to solve real-world problems which require the human brain. Rule-based systems are designed on a set of rules that exist in memory and are activated by facts that activate and assert new facts. Instead of representing knowledge in a static way, the rule-based system uses an "IF-THEN" statement. In a rule-based system, experts are represented by a set of rules that describe what to do or what can be concluded in a situation.



Structure of Rule-based Systems

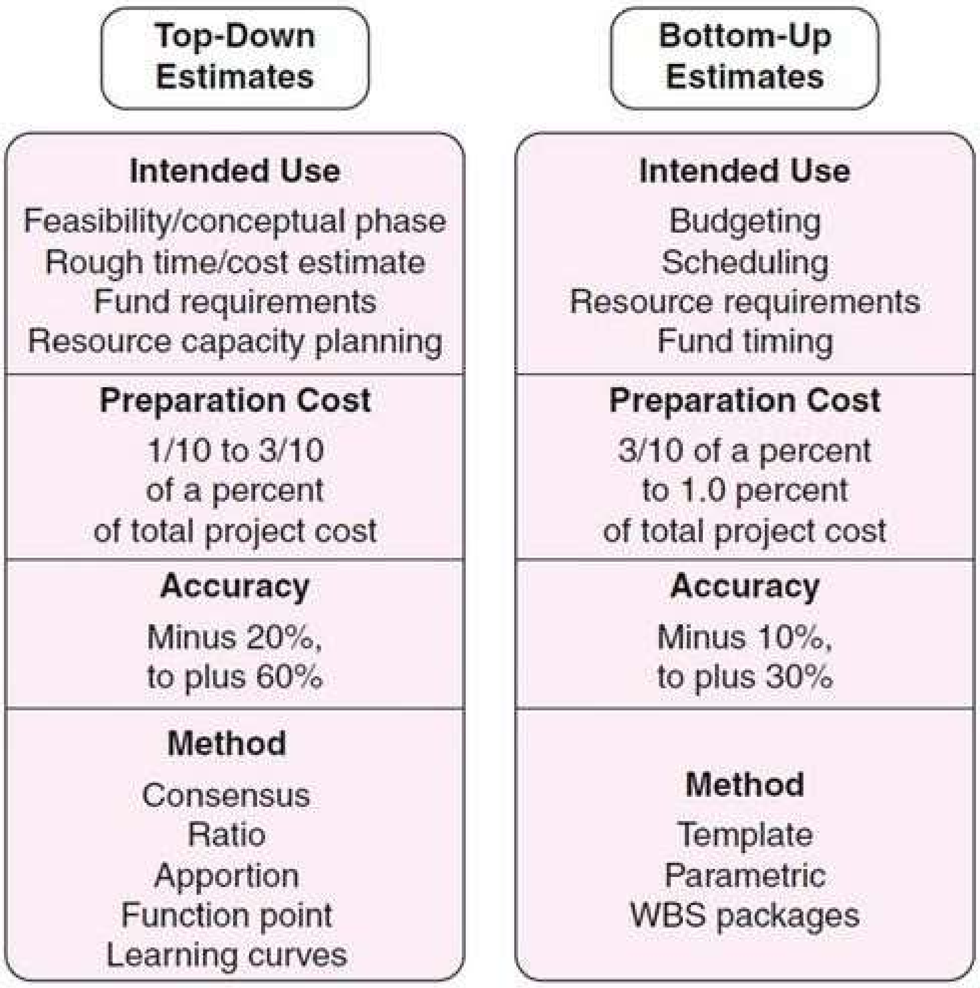
e) **<u>Planning Poker</u>:** It is an expert judgment-based estimation technique. It was first defined by James Grenning in 2002. This technique creates an estimate by taking the opinion of multiple experts and then combining it. Members of planning poker are all the developers on the project team i.e., programmers, analysts, testers, product owners, etc.



Working of Planning Poker Technique

f) **<u>Top-Down Approach:</u>** The Top-Down approach was promoted by IBM researchers Harlan Mills and Niklaus Wirth in the 1970s. In this approach, the total cost estimate of a project is taken from the software product in either an algorithmic or non-algorithmic way. Then the project is divided into different components and subcomponents. Once a total cost of the project is found, a proportion of that cost is assigned to each component.

g) **<u>Bottom-Up Approach:</u>** In the Bottom-Up approach, the cost estimation process initiates with the lowest level components of the software system. The cost of each component is estimated separately by the person who is developing the component. After that these individual estimated costs are aggregated and bind up to the highest level to determine an estimate for the overall software product. This approach looks at the costs from a more granular viewpoint that is why these estimates are normally more accurate than the other methods or techniques.



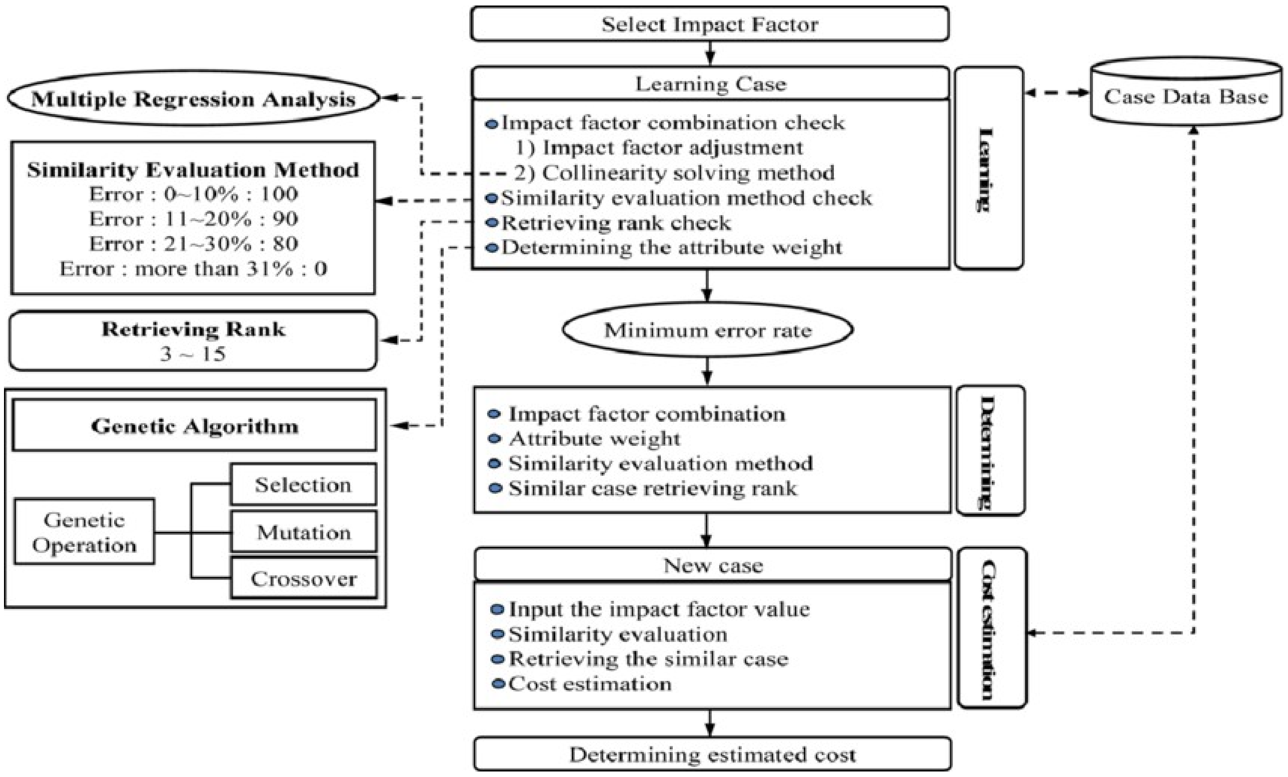
Top-Down Estimates	Bottom-Up Estimates
Intended Use Feasibility/conceptual phase Rough time/cost estimate Fund requirements Resource capacity planning	**Intended Use** Budgeting Scheduling Resource requirements Fund timing
Preparation Cost 1/10 to 3/10 of a percent of total project cost	**Preparation Cost** 3/10 of a percent to 1.0 percent of total project cost
Accuracy Minus 20%, to plus 60%	**Accuracy** Minus 10%, to plus 30%
Method Consensus Ratio Apportion Function point Learning curves	**Method** Template Parametric WBS packages

Top-Down Vs Bottom-Up Estimates

3. LEARNING-ORIENTED TECHNIQUES

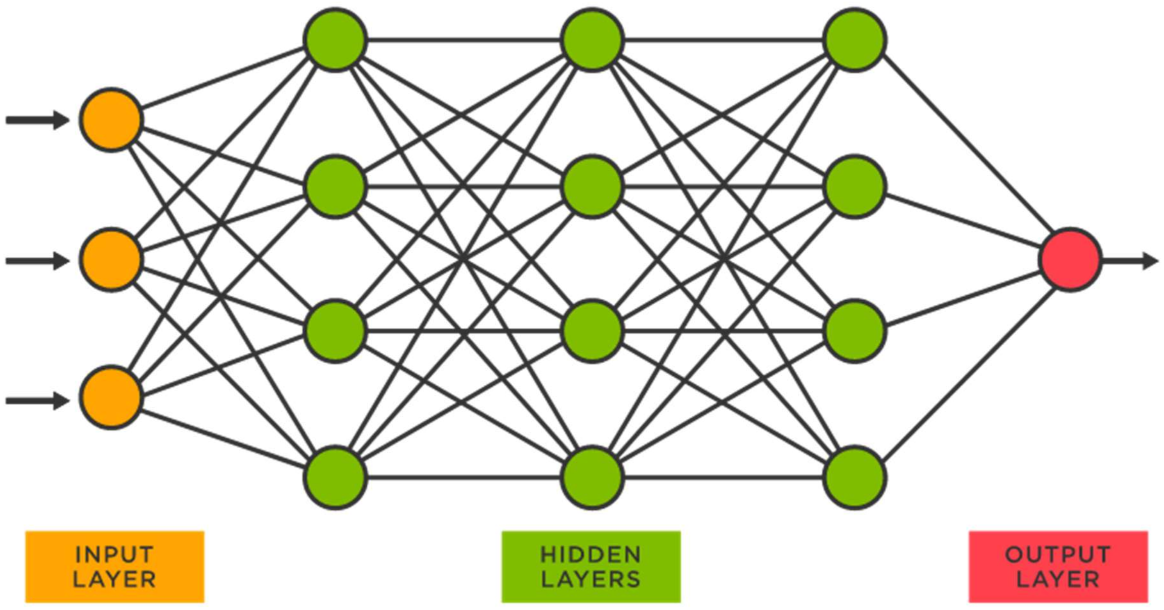
Learning-oriented techniques use prior knowledge as well as current information to develop software cost-estimation models. These techniques take reference from previous experiences and build a model to automate the estimation process.

a) **Case-Based Reasoning:** The CBR model assumes that the same cases have the same solutions. The system continuously learns without taking the consideration of experts, the learning process gathers the resolved case (solution) in the database and makes it accessible to solve new cases in the future. The CBR system consists of a preprocessor that organizes the input data for processing, a similarity function used to search for the same cases, a predictor to generate a prediction, a predictor for estimating the output value of the subject case.


Select Impact Factor
Learning Case
Impact factor combination check
1) Impact factor adjustment
2) Collinearity solving method
Similarity evaluation method check
Retrieving rank check
Determining the attribute weight
Learning
Case Data Base
Multiple Regression Analysis
Similarity Evaluation Method
Error : 0~10% : 100
Error : 11~20% : 90
Error : 21~30% : 80
Error : more than 31% : 0
Retrieving Rank
3 ~ 15
Minimum error rate
Genetic Algorithm
Genetic Operation
Selection
Mutation
Crossover
Impact factor combination
Attribute weight
Similarity evaluation method
Similar case retrieving rank
Determining
New case
Input the impact factor value
Similarity evaluation
Retrieving the similar case
Cost estimation
Cost estimation
Determining estimated cost

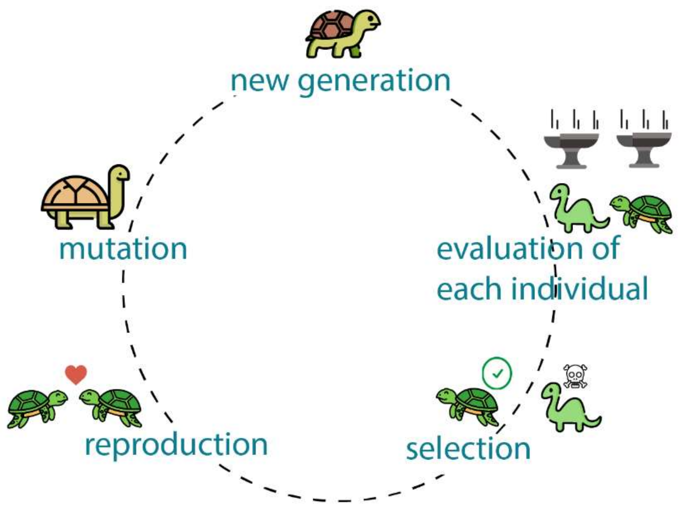
Work Flow for CBR Model

b) **Neural Networks:** Neural networks (NN) were introduced by Warren McCulloch and Walter Pitts research who created a neural networks calculation model. These estimation models are trained using historical data or datasets and it automatically adjusts their algorithmic parameter values to reduce error. In software estimates, backpropagation networks are the most common form of neural networks.



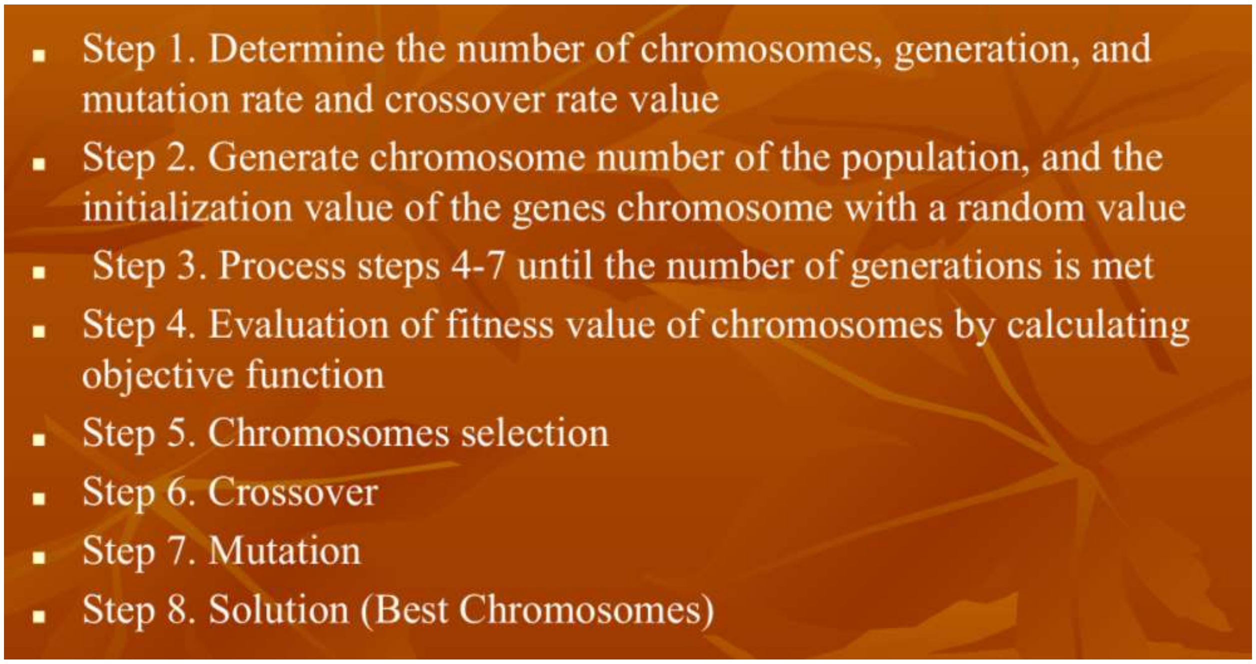
Neural Network Architecture

c) **Genetic Algorithm**: The genetic algorithm (GA) was invented by John Henry Holland in the 1970s. The main elements of GA are chromosome populations, selection by fitness values, crosses to make new descendants, and random mutations of new offspring. The solutions to the problems are represented by fixed-length binary strings called chromosomes. Different methods, such as selection and mutation, are performed on selected chromosomes having higher fitness values. As a result, a new population is created, and then an optimal solution is produced.



Five Stages of Genetic Algorithm

The process for genetic algorithm is as follows:



- Step 1. Determine the number of chromosomes, generation, and mutation rate and crossover rate value
- Step 2. Generate chromosome number of the population, and the initialization value of the genes chromosome with a random value
- Step 3. Process steps 4-7 until the number of generations is met
- Step 4. Evaluation of fitness value of chromosomes by calculating objective function
- Step 5. Chromosomes selection
- Step 6. Crossover
- Step 7. Mutation
- Step 8. Solution (Best Chromosomes)

d) **<u>Genetic Programming:</u>** Genetic programming (GP) is an extension of GA and is not limited to chromosomes of fixed-length binary strings. The modern GP which was based on tree structure was given by Nichael L. Cramer in 1985, after which this study was extended by John R. Koza. In GP, chromosomes are programs that are run to get the necessary results.

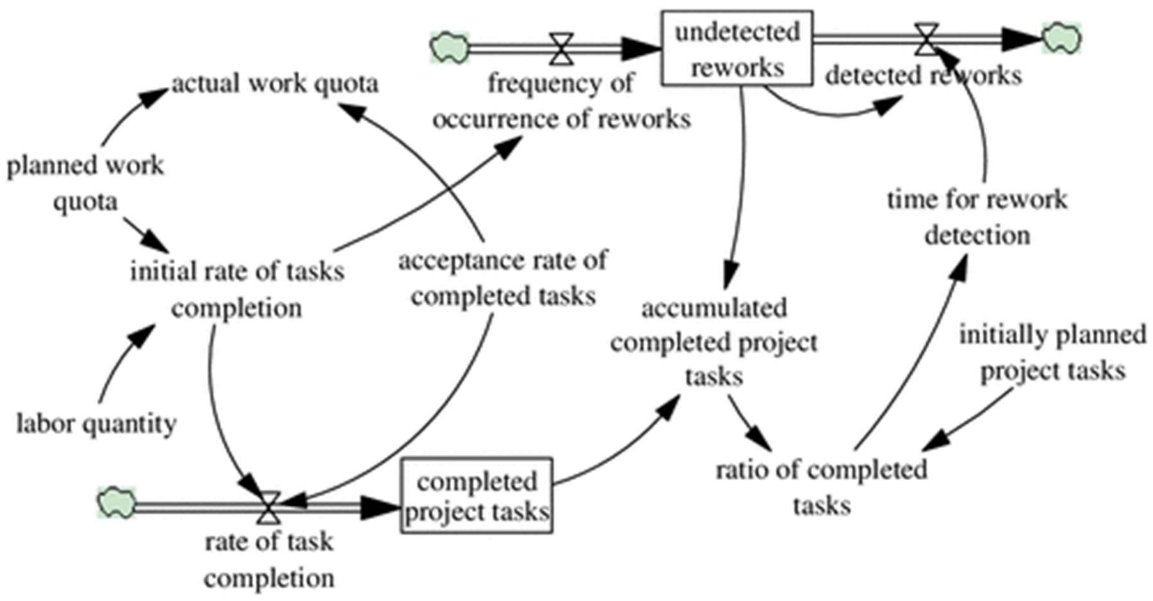
4. DYNAMICS-BASED TECHNIQUES

As compared to other techniques, dynamics-based techniques consider effort and cost factors to be dynamic as they change over the period of the system development process. Changes in software factors such as design requirements, budget, project time etc. will change the productivity of the project. This type of technique is suitable for planning & management.

a) **<u>System Dynamics Approach</u>:** The system dynamic approach was designed by Massachusetts Institute of Technology's Jay Forrester in 1961 to analyze and understand the dynamic behavior of complex systems. This is a simulation modeling methodology that displays results and behavior as a graph that varies with time. In this approach, the model is represented as a modified network with positive and negative feedback loops. The system dynamics simulation model is represented by the set of first-order differential equation:

$$x'(t) = f(x, p)$$

where x is a vector that describes states in the model, t is time, f is a vector function, which is nonlinear, and p is a set of model parameters.


undetected reworks
frequency of occurrence of reworks
detected reworks
actual work quota
planned work quota
time for rework detection
initial rate of tasks completion
acceptance rate of completed tasks
accumulated completed project tasks
initially planned project tasks
labor quantity
completed project tasks
ratio of completed tasks
rate of task completion

System Dynamics Approach

5. REGRESSION-BASED TECHNIQUES

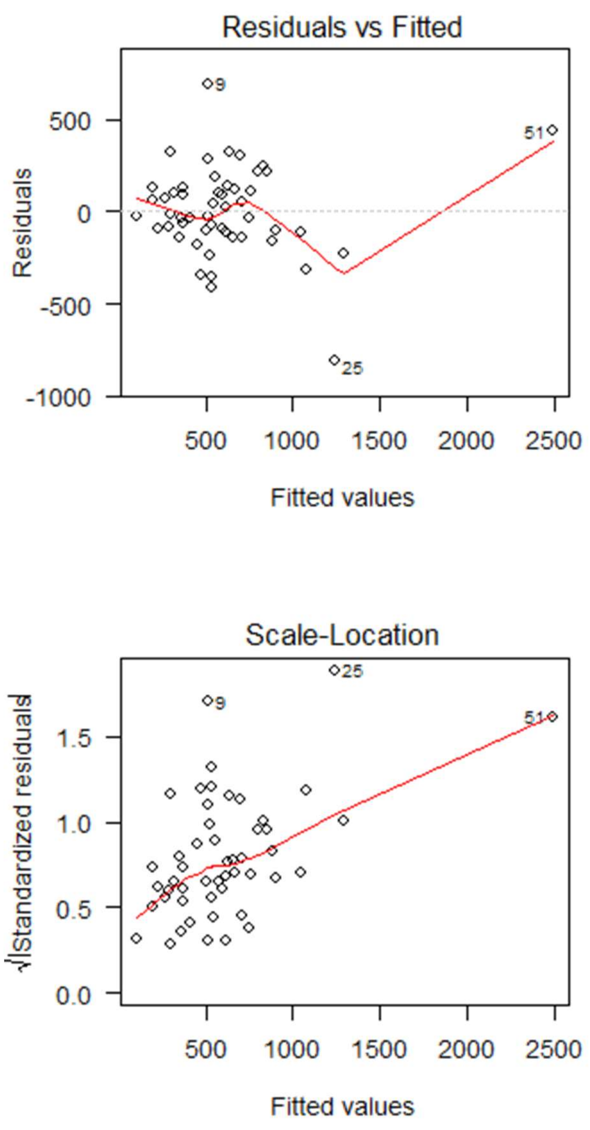
Regression-based techniques are very famous in model building and are used in combination with model-based methods. They estimate software costs as a function of key cost factors by using mathematical algorithms.

a) **<u>Standard Regression</u>:** Standard regression is used for estimating unknown parameters and it is based on the ordinary least squares method (OLS) by using the linear regression model. This technique is simple and easily accessible from many statistical software packages. The model using the OLS method can be written as in following Equation:

$$y_t = \beta_1 + \beta_2 x_{t2} + \cdots + \beta_k x_{tk} + e_t$$

where y_t represents the response variable for the observation, β1 represents an intercept parameter, $\beta_2 . . . \beta_k$ are response coefficients, $x_{t2} . . . x_{tk}$ are predictor (or regressor) variables, and the error term, et is a random variable with a probability distribution.

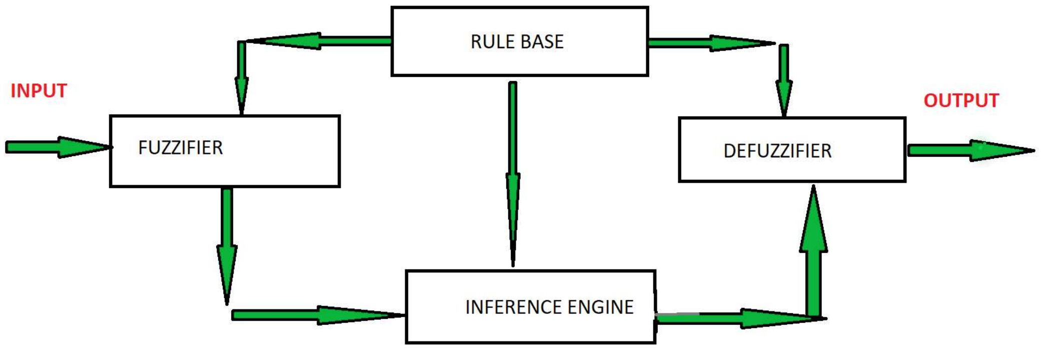
b) **<u>Robust Regression:</u>** Robust regression is an advancement of the OLS approach. There are many outliers in the software development dataset like software development processes and quantitative data. Robust regression has been applied to screen outliers for software metric models. Many parametric cost estimation models have adopted regression-based approach because it is simple and widely accepted. The disadvantage of this technique is that we can discard outliers without direct reasoning.



c) **Fuzzy Logic-Based Methods:** The term fuzzy logic was introduced in 1965 in fuzzy set theory. In Boolean logic the truth value of a variable can be either 0 or 1 but fuzzy logic deals with the concept of partial truth i.e., the truth value of a variable can be a real number that exists between 0 and 1.

d) **Fuzzy Systems:** A fuzzy system is a process of mapping semantic terms. The input and output of the fuzzy framework can be either numerical or etymological and it has 3 parts:

enrollment works, a standard base, and a yield joining capacity. Fuzzy frameworks have been used for programming improvement models.

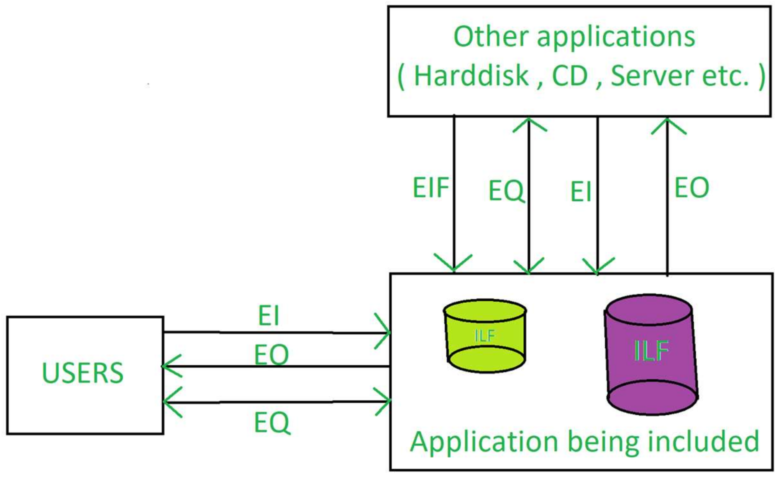


Fuzzy Logic Architecture

6. SIZE-BASED ESTIMATION TECHNIQUES

Software estimation techniques are used to predict the software size for software development projects and some of them are discussed below.

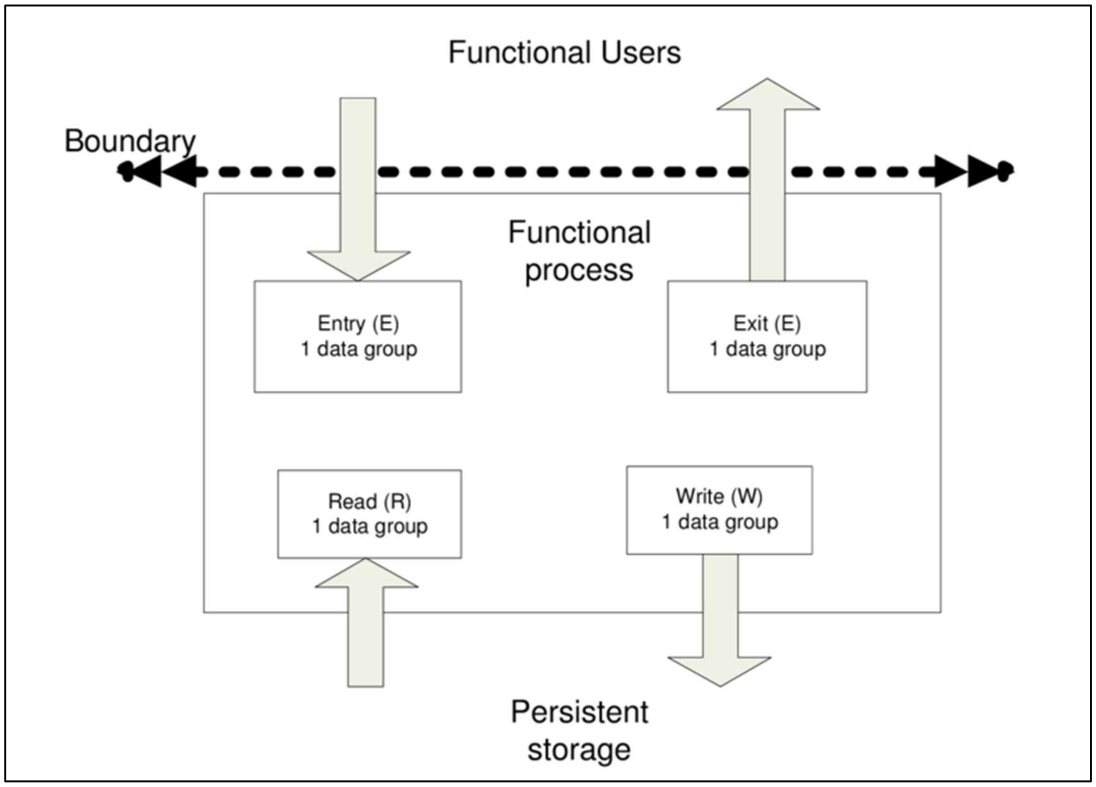
a) **<u>Function Points:</u>** Function points (FPs) were introduced by Allan Albrecht in 1979. This strategy calculates the size estimation of programming and the expense is determined from past activities. Function Point Analysis (FPA) measures the logical view of an application and discards the physically implemented view. This technique is used to analyze the functionality delivered by software and Unadjusted Function Point (UFP) which is the unit of measurement.



Function Point Analysis

- External input (EI) generates data that comes from outside of the application.
- External output (EO) generates data that is sent to the outside of the application. External
- Inquiry (EQ) is a combination of input-output that results in data retrieval.
- Internal logical file (ILF) is a process of gathering information that is utilized and shared inside the application’s boundary.
- External interface file (EIF) is gathering information which is utilized for reference reasons.
- EI, EO, and EQ are transactional function types while ILF and EIF are data function types.

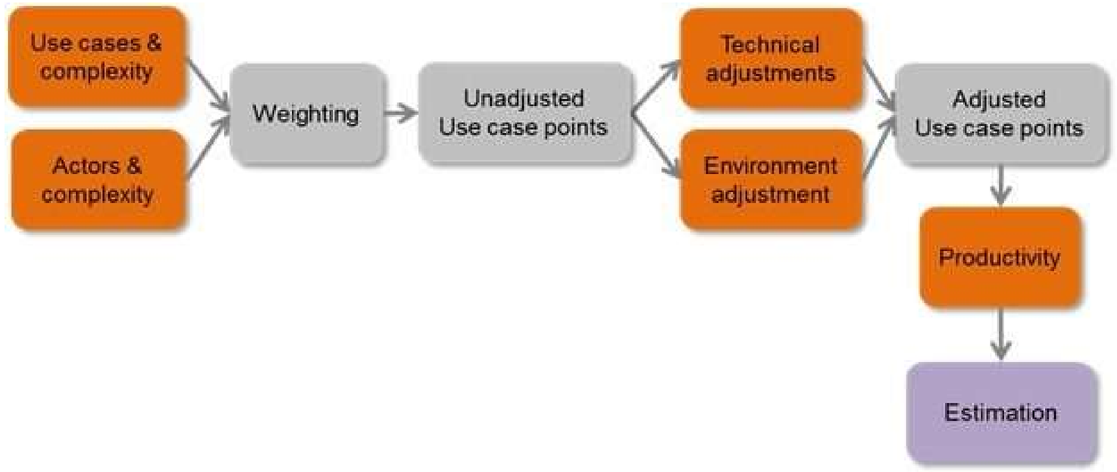
b) **<u>Full Function Points (FFPs):</u>** FFP is an expansion of the standard Function Point Analysis (FPA) strategy. It presents two extra control information types and four new control value-based capacity types. “Since function points is believed to be more useful in the MIS domain and problematic in the real-time software domain, another recent effort, in functionality-based estimation, is the Full Function Points (FFP) which is a measure specifically adapted to real-time and embedded software.”



Cosmic Function Point Concept

c) <u>**Use Case Points:**</u> Use Case Points (UCPs) were created by Gustav Karner in 1993 and it depends on FP estimation procedure. It was used for the specific arranged frameworks being composed of use cases, which is a part of the Unified Modeling Language (UML) systems. Based on components of the framework use cases, the UCP is determined to quantify the product measure, which is then used to evaluate the task exertion. The UCP condition consists of 3 factors i.e., Unadjusted Use Case Points (UUCP), Technical Complexity Factor (TCF), and Environment Complexity Factor (ECF). UCP is calculated by using the formula:

$$UCP = UUCP \times TCF \times ECF \times PF$$



Work Flow for Use Case Point Method

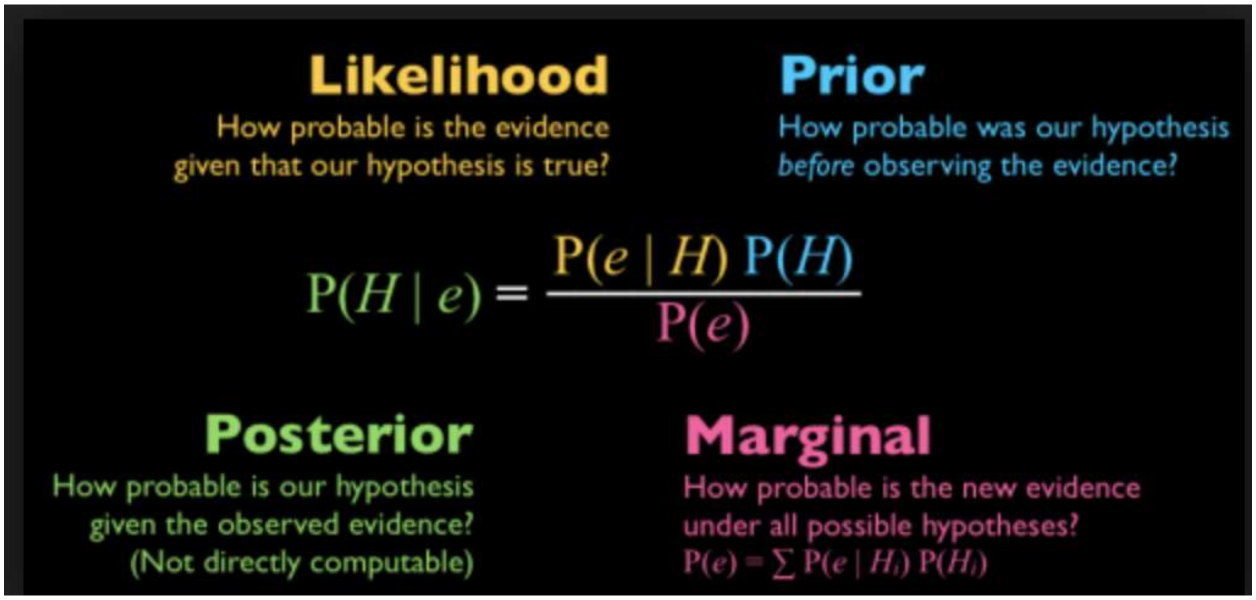
7. COMPOSITE TECHNIQUES

Composite techniques incorporate at least two procedures to define the most reasonably useful frame for estimation through the cons of a technique.

a) <u>**Bayesian Approach:**</u> Bayesian approach is an evaluating approach which is the advancement of the COCOMO-II model. The Bayesian methodology has benefits; however, it incorporates earlier learning of specialists. The two data sources can be joined utilizing Bayes' hypothesis as pursued in the below Equation.

$$f(\beta/Y) = \frac{f(Y/\beta)f(\beta)}{f(Y)}$$

where β is the parameter of the vector, Y is the vector of test perceptions, f (β/Y) means the work for β which outlines all the data about β, f(Y/β) represents the example data, and f (β) is for the earlier data that abridges the data about β.


Likelihood
How probable is the evidence given that our hypothesis is true?
Prior
How probable was our hypothesis *before* observing the evidence?
$P(H \mid e) = \frac{P(e \mid H)\, P(H)}{P(e)}$
Posterior
How probable is our hypothesis given the observed evidence? (Not directly computable)
Marginal
How probable is the new evidence under all possible hypotheses?
$P(e) = \sum P(e \mid H_i)\, P(H_i)$

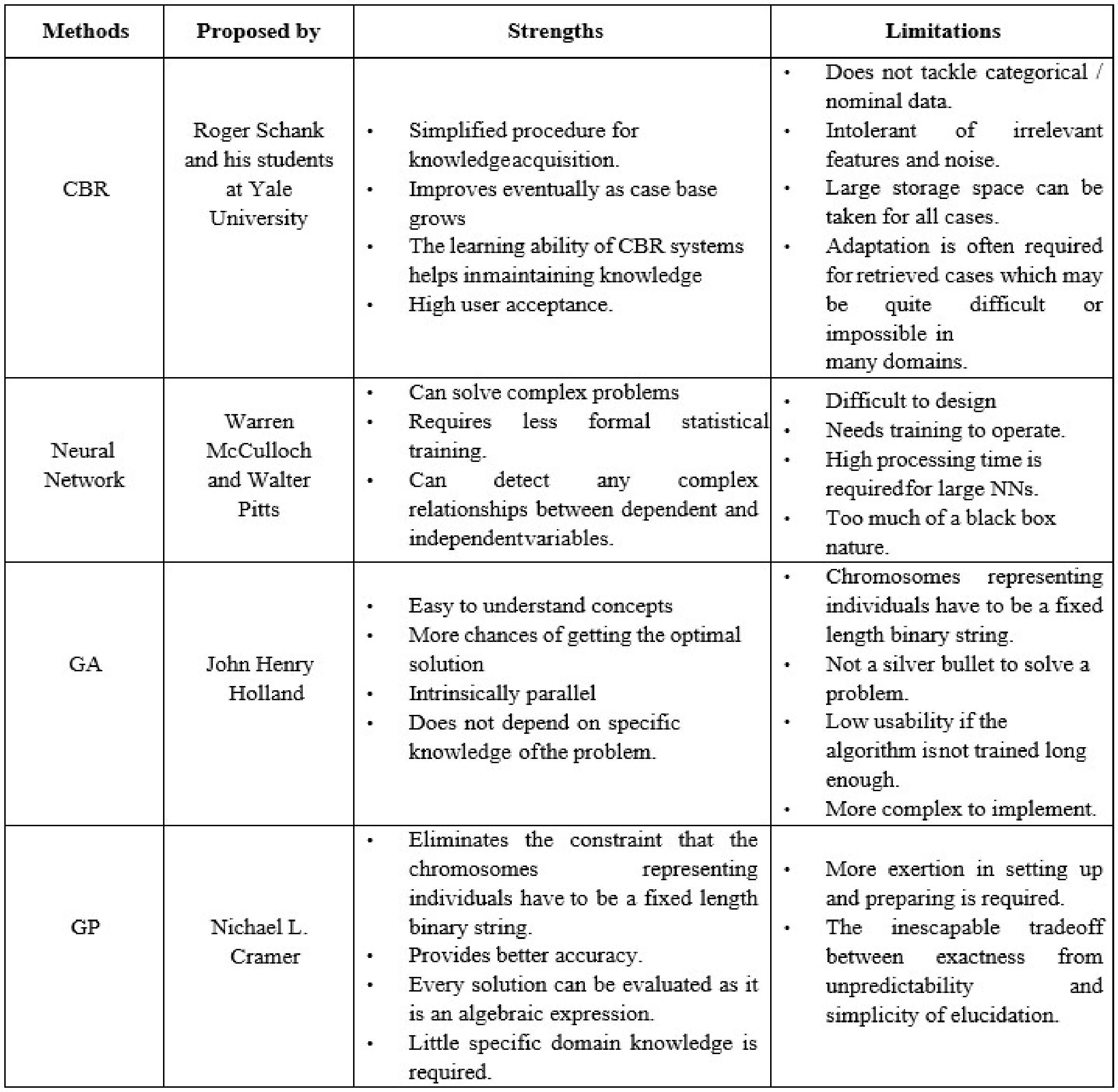
Bayes Formula for Likelihood Prediction

RESULT AND ANALYSIS

In this section, we have summarized the techniques mentioned above based on the strengths and weaknesses of each model. This comparison will not only provide a gist of all the methods but will also help the project managers to choose the appropriate method for their software project.

The table below provides a summary of Learning Oriented Techniques. The techniques covered are:

1. **Case-Based Reasoning (CBR)**
2. **Neural Networks (NN)**
3. **Genetic Algorithms (GA)**
4. **Genetic Programming (GP)**

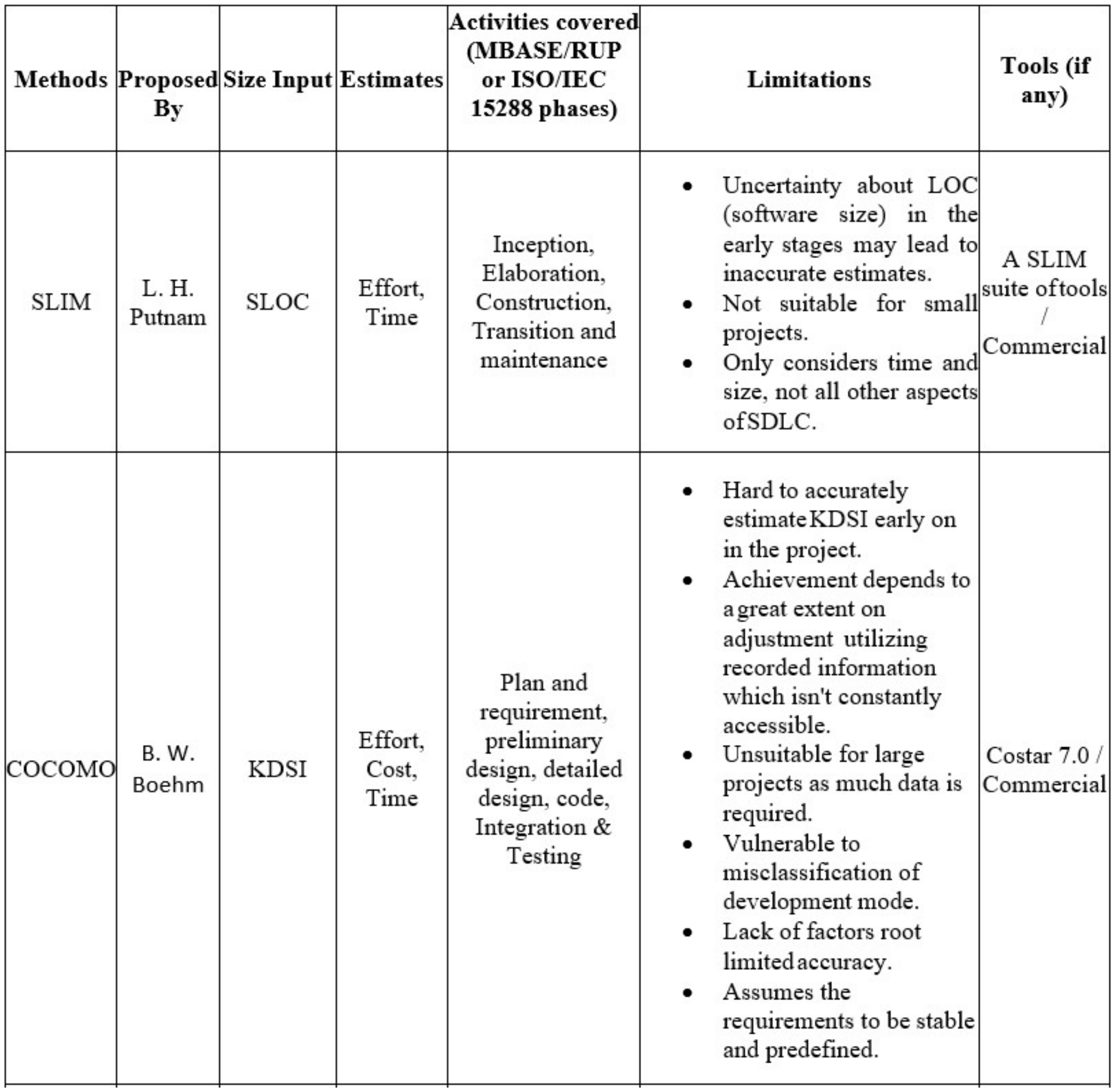


Methods	Proposed by	Strengths	Limitations
CBR	Roger Schank and his students at Yale University	• Simplified procedure for knowledgeacquisition. • Improves eventually as case base grows • The learning ability of CBR systems helps inmaintaining knowledge • High user acceptance.	• Does not tackle categorical / nominal data. • Intolerant of irrelevant features and noise. • Large storage space can be taken for all cases. • Adaptation is often required for retrieved cases which may be quite difficult or impossible in many domains.
Neural Network	Warren McCulloch and Walter Pitts	• Can solve complex problems • Requires less formal statistical training. • Can detect any complex relationships between dependent and independentvariables.	• Difficult to design • Needs training to operate. • High processing time is requiredfor large NNs. • Too much of a black box nature.
GA	John Henry Holland	• Easy to understand concepts • More chances of getting the optimal solution • Intrinsically parallel • Does not depend on specific knowledge ofthe problem.	• Chromosomes representing individuals have to be a fixed length binary string. • Not a silver bullet to solve a problem. • Low usability if the algorithm isnot trained long enough. • More complex to implement.
GP	Nichael L. Cramer	• Eliminates the constraint that the chromosomes representing individuals have to be a fixed length binary string. • Provides better accuracy. • Every solution can be evaluated as it is an algebraic expression. • Little specific domain knowledge is required.	• More exertion in setting up and preparing is required. • The inescapable tradeoff between exactness from unpredictability and simplicity of elucidation.

The table below provides a summary of Parametric Models.

The techniques chosen were:

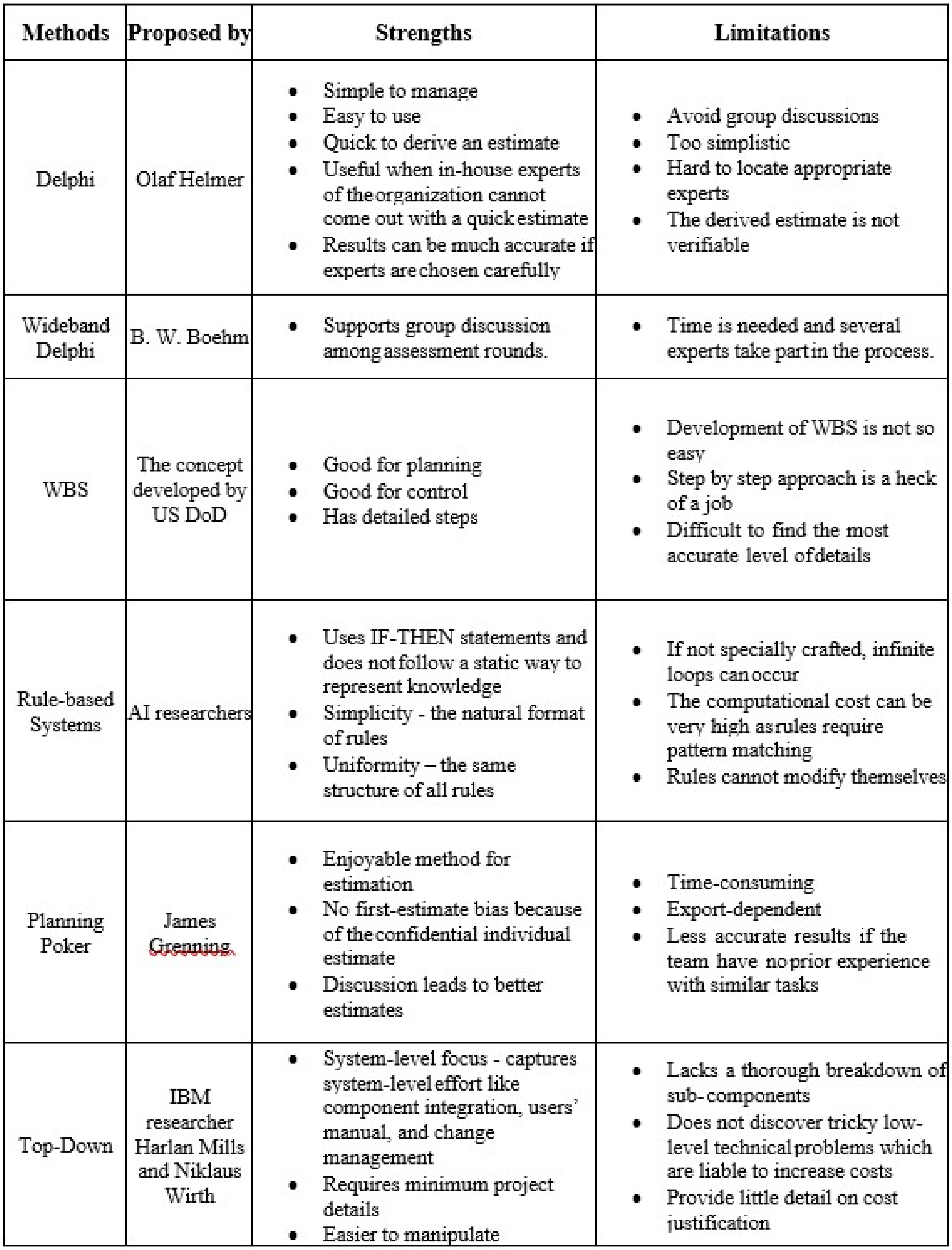
1. **SLIM**
2. **COCOMO**
3. **SEER-SEM**
4. **CheckPoint**
5. **Estimacs**
6. **Price-S**
7. **COSYSMO**
8. **COCOMO-II**



Methods	**Proposed By**	**Size Input**	**Estimates**	**Activities covered (MBASE/RUP or ISO/IEC 15288 phases)**	**Limitations**	**Tools (if any)**
SLIM	L. H. Putnam	SLOC	Effort, Time	Inception, Elaboration, Construction, Transition and maintenance	• Uncertainty about LOC (software size) in the early stages may lead to inaccurate estimates. • Not suitable for small projects. • Only considers time and size, not all other aspects of SDLC.	A SLIM suite of tools / Commercial
COCOMO	B. W. Boehm	KDSI	Effort, Cost, Time	Plan and requirement, preliminary design, detailed design, code, Integration & Testing	• Hard to accurately estimate KDSI early on in the project. • Achievement depends to a great extent on adjustment utilizing recorded information which isn't constantly accessible. • Unsuitable for large projects as much data is required. • Vulnerable to misclassification of development mode. • Lack of factors root limited accuracy. • Assumes the requirements to be stable and predefined.	Costar 7.0 / Commercial

SEER-SEM	Galorath Inc.	SLOC, FPs, UCs	Effort, Cost, Risk, Duration	Inception, Elaboration, Construction, Transition and maintenance	It takes many parameters as input which increases the complexity and uncertainty. The exact size of the project is a key concern in this model.	SEER for Software
Checkpoint	Capers Jones	FPs	Effort, Cost, Schedule, Defect	Inception, Elaboration, Construction, Transition and maintenance	Estimation is bit complex since it is done at activity-level and task-level.	Checkpoint/ Commercial
ESTIMACS	Howard Rubin	Function Point	Effort, Cost, Risk	Inception, Elaboration, Construction	Each stage does not clearly translate the effort. The results of the package are not totally explainable.	Estimacs / Commercial
PRICE-S	RCA	SLOC, FPs, POPs, UCCP	Cost, Schedule	Inception, Elaboration, Construction, Transition and maintenance	Model is presented as a black box to the users because its core concepts and ideas are not publicly defined.	TruePlanning/ Commercial
COSYSMO	Ricardo Valerdi	Requirement, Interfaces, Algorithms, Operational, Scenarios	Effort, Time	Conceptualize, Develop, Activity, Test, and Evaluation, Change to Operation, Work Maintain or Enhance, and Supplant	It overlaps with the COCOMO II model causing needless double-counting of effort; as in most organizations, software engineering and systems engineering are highly coupled.	SystemStar / Commercial
COCOMO II	B. W. Boehm *et al.*	KSLOC, FPs, Application Points	Cost, Effort, Schedule	Inception, Elaboration, Construction, Transition and maintenance	Its 'heart' is still based on a waterfall process model. Duration calculation for small projects is not reasonable.	USC COCOMO II / Free

The table below provides a summary of Expertise / Consensus based methods, their strength and limitations.



Methods	Proposed by	Strengths	Limitations
Delphi	Olaf Helmer	• Simple to manage • Easy to use • Quick to derive an estimate • Useful when in-house experts of the organization cannot come out with a quick estimate • Results can be much accurate if experts are chosen carefully	• Avoid group discussions • Too simplistic • Hard to locate appropriate experts • The derived estimate is not verifiable
Wideband Delphi	B. W. Boehm	• Supports group discussion among assessment rounds.	• Time is needed and several experts take part in the process.
WBS	The concept developed by US DoD	• Good for planning • Good for control • Has detailed steps	• Development of WBS is not so easy • Step by step approach is a heck of a job • Difficult to find the most accurate level of details
Rule-based Systems	AI researchers	• Uses IF-THEN statements and does not follow a static way to represent knowledge • Simplicity - the natural format of rules • Uniformity – the same structure of all rules	• If not specially crafted, infinite loops can occur • The computational cost can be very high as rules require pattern matching • Rules cannot modify themselves
Planning Poker	James Grenning	• Enjoyable method for estimation • No first-estimate bias because of the confidential individual estimate • Discussion leads to better estimates	• Time-consuming • Export-dependent • Less accurate results if the team have no prior experience with similar tasks
Top-Down	IBM researcher Harlan Mills and Niklaus Wirth	• System-level focus - captures system-level effort like component integration, users' manual, and change management • Requires minimum project details • Easier to manipulate	• Lacks a thorough breakdown of sub- components • Does not discover tricky low-level technical problems which are liable to increase costs • Provide little detail on cost justification

The table below provides a summary of size-based methods, their strength and limitations.

The techniques chosen were:

1. **Function Point (FP)**
2. **Full Function Points (FFP)**
3. **Use Case Points (UCP)**

Methods	Proposed by	Strengths	Limitations
FP	Allan Albrecht	• Make estimation possible early in theproject lifecycle. • Independent of how the requirements ofthe software were expressed. • Does not depend on a specific technology or programming language.	• Not capable of dealing with hybrid systems. • No availability of enough research data as compared to LOC. • Time-consuming method.
FFP	Denis St-Pierre et al.	• Can cope with real-time software domain. • Can cope with embedded software. • Retains the actual FPA quality characteristics.	• Restricted range of software (specifications) that can be sized is covered. • Time-consuming method.
UCP	Gustav Karner	• The process can be automated. • Can be measured early in the projectlifecycle. • Easy to use. • When estimation is performed by • skilled people, estimates would be closeto the actuals.	• Only applicable for those software projects whose specification can be expressed by use cases. • UCP is less useful in iteration tasks in the team.

CONCLUSION

This study discusses various software development cost estimation techniques and models along with their strengths and weaknesses. During our research, we realized that software development effort and cost estimation is an interesting area in software project management field.

There have been ample researches that propose various software development cost estimation models but there was a lack of research in comparing the different models available and listing their pros and cons. Our study discusses the various techniques and models available, categorizes them based on eight different classes and enlist the scenarios in which the technique can be utilized along with the scenarios where using the technique will not generate optimal results.

We were able to find out situations pertaining to every technique and model in which the technique opted will not generate satisfactory results and will hence be an overhead on the software project undertaken. Therefore, we concluded that no cost estimation technique is 100% ideal for all scenarios. Since the software industry is rapidly changing with newer models like Agile coming in the industry, no technique is considered to be ideal.

A trustworthy effort estimation technique or model is yet needed to be discovered or explored that provides accurate estimates for a software project irrespective of the nature of the project, the environmental factors, or the situation in which the project is meant to be implemented.

According to the results mentioned above, we strongly recommend the usage of Hybrid Techniques that are formed by combining two or more of the existing models. The models can be selected based on the strengths mentioned in our study. The model can be chosen by prioritizing which pros are mandatorily needed in the software project and which cons can be compromised. This hybrid approach will significantly improve the existing approaches and one model's strengths can counter the other model's weaknesses. In this way, a near-to-perfect hybrid approach could be applied as per the nature of the software project.

FUTURE WORK

Currently, we have divided the existing and the most popular software cost-estimation models and techniques. In the future, even further techniques can be analyzed and appended to the existing list. Moreover, there may be models that require a different classification class altogether.

With software industry ever-changing nature, newer cost-estimation techniques will be proposed that would need to be classified after analyzing their strengths and weaknesses. The research will be declared complete only when the objective of finding a perfect algorithm is achieved.

CONTRIBUTION

Aryan Bansal (2K18/SE/038) and Ashish Kumar (2K18/SE/041)

- **Literature Review** – We both read around 15-20 research papers on the selected topic to know about the different cost estimation techniques and models. It was after this that we selected the ones that we found were most widely used in the software industry.

- **Brainstorming on the strengths and weaknesses of each algorithm** – We divided the algorithms equally among ourselves and worked on our part to create the required comparison tables.

- **Concluding the best algorithm –** We both were of the opinion that since no algorithm performed ideal in all the scenarios, the conclusion should be a hybrid algorithm consisting of the features of two or more existing techniques to cost estimation.

- **Report Writing –** We made the report using Google Docs so that we could work simultaneously on the report. Each member did the analysis of half of the techniques covered in this study and presented the report for that.

- **Presentation Creation –** The same format was followed in making the presentation as it was done in the Report Writing task.

REFERENCES

1. Bob Hughes and Mike Cotterell “Software Project Management Second edition” School of Information management, University of Brighton 1999.
2. Mendes, E., Watson, I., Triggs, C., Mosley, N., & Counsell, S., "A comparative study of cost estimation models for web hypermedia applications", Empirical Software Engineering, Vol. 8, No. 2,pp 163-196, 2003.
3. Kumar, S., Rastogi, R. and Nag, R., "Function Point Analysis in Multimedia Software/Application Estimation", In Software Engineering, Springer, pp. 383-392, 2019.
4. Kemerer, C.F.," An empirical validation of software cost estimation models", Communications of the ACM, Vol. 30, No 5, pp.416-429,1987.
5. Heemstra, F.J., "Software cost estimation", Information and software technology, Vol. 34, No 10, pp.627-639, 1992.
6. Boehm, B., Clark, B. Horowitz, E., Westland, C., Madachy, R. and Selby, R., "Cost models for future software life cycle processes: COCOMO 2.0", Annals of software engineering, Vol. 1, No 1, pp.57- 94,1995.
7. Valerdi, R. and Boehm, B.W., "COSYSMO: A systems engineering cost model",2010.
8. Dalkey, N., "An experimental study of group opinion: the Delphi method", Futures, Vol. 1, No 5, pp.408-426,1969.
9. Pospieszny, P., Czarnacka-Chrobot, B. and Kobylinski, A., " An effective approach for software project effort and duration estimation with machine learning algorithms", Journal of Systems and Software, Volume 137, pp.184-196.2018.
10. Mitchell, M., "An introduction to genetic algorithms", MIT press,1998.
11. Miyazaki, Y., Terakado, M., Ozaki, K. and Nozaki, H., "Robust regression for developing software estimation models", Journal of Systems and Software, Vol. 27, No 1, pp.3-16,1994.
12. Yager, R.R., "Connectives and quantifiers in fuzzy sets", Fuzzy sets and systems, Vol. 40, No 1, pp.39-75,1965.
13. Clemmons, R.K., "Project estimation with use case points", The Journal of Defense Software Engineering, Volume 19, No 2, pp.18-22,2006.
14. Nasir, M., "A survey of software estimation techniques and project planning practices", In Software Engineering, Artificial Intelligence, Computing,pp. 305- 310,2006.
15. https://www.geeksforgeeks.org/
16. https://www.javatpoint.com/
17. Dian Pratiwi, Indonesia, “Implementation of FPA in Measuring the Volume Estimation of Software System in Object Oriented and Structural Model of Academic System”, International Journal of Computer Applications, 2013.
18. Chirra, S. and Reza, H. (2019) A Survey on Software Cost Estimation Techniques. Journal of Software Engineering and Applications, 12, 226-248. doi: 10.4236/jsea.2019.126014.