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.

Our research describes SDCE by investigating various techniques & then we compared all techniques in the form of one table. We've have reviewed various SDCE 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:

1. Parametric techniques
2. Expertise-based techniques
3. Learning-oriented techniques
4. Dynamics-based techniques
5. Regression-based techniques
6. fuzzy logic-based methods
7. Size-based estimation models
8. Composite techniques.

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

This research is separated into 3 steps. In the first step, 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. 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

- This research describes SDCE by investigating various techniques & then comparing all the techniques in the form of a comparison table.

- Our research work will help the future researchers to get deep insights about SDCE techniques and models and make it easy to select a particular model. 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 major activity in software project management to handle resources in an efficient manner by guessing the requisite no. of resources to perform a given job.

"The estimation techniques are classified into seven main categories: parametric techniques, learning-based, expertise-based techniques, regression-based techniques, dynamics-based techniques, and Composite-Bayesian techniques." Lots of research was done on this and the main problem occurred is to determine the correct estimation of software cost.

Many researches were done for investigating cost estimation techniques, but they compare only the well-known techniques i.e., COCOMO model, FPs, Neural Networks 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.

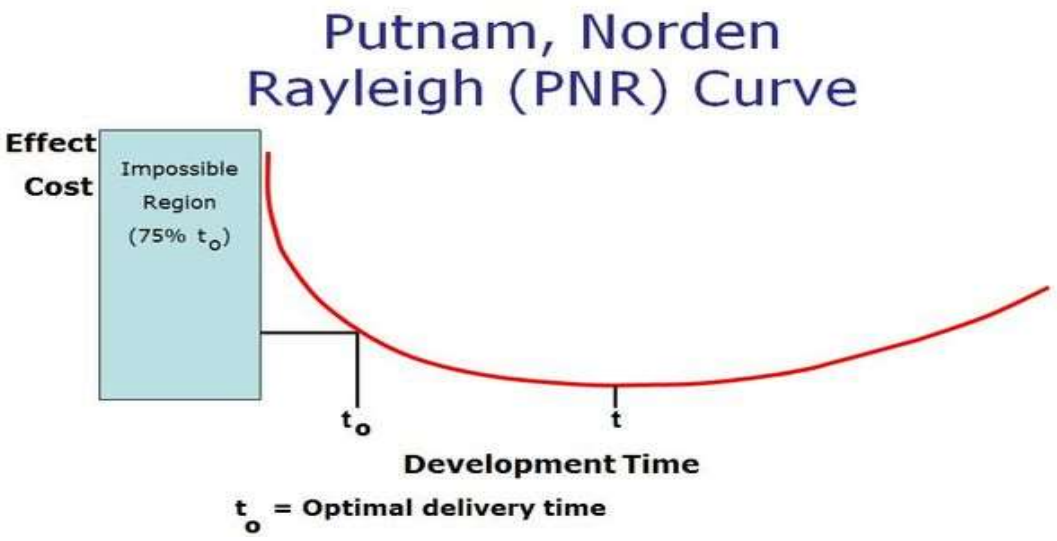
1. PARAMETRIC / ALGORITHMIC TECHNIQUES

We used Algorithmic techniques for generating cost estimates. The algorithmic model will represent in equation:

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

where E is effort, CF is cost factor.

a) **Software Life-Cycle Model (SLIM):** 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. As this model supposes resource utilization will change over time, we can modify it with the Rayleigh distribution. To describe the interdependence of project's effort, the PNR (Putnam Norden Rayleigh) curve is used.



The Rayleigh equation is written below:

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

where K is the area under the curve, a = (1/2td^2), td is time when dy/dt hit the highest point i.e. at peak.

b) **<u>COCOMO (Constructive Cost model)</u>:** The COCOMO model was developed in 1981. COCOMO model is used to calculate the effort needed to build a software product based on its size. There are three sub-models of Cocomo: Basic Cocomo, Intermediate Cocomo, and Detailed Cocomo. Basic COCOMO calculates effort (in person-months) and cost as software size in thousands. The basic COCOMO Equation is written below:

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

The below equation is used to get development time:

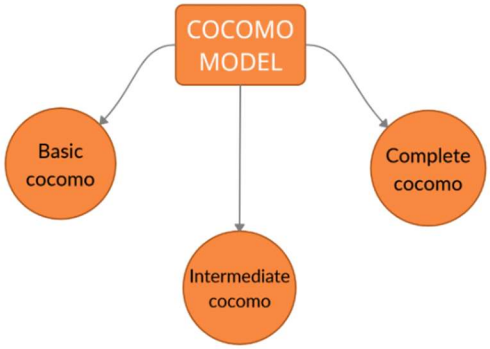
$$t_{dev} = c(MM)^d$$

The basic Cocomo model assumes that the effort is only a function of the number of lines of code and some constants evaluated according to the different software system. However, in reality, no system's effort and schedule can be solely calculated on the basis

of Lines of Code. For that, various other factors such as reliability, experience, Capability. These factors are known as Cost Drivers and the Intermediate Model utilizes 15 such drivers for cost estimation. The intermediate COCOMO equation is written below:

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

Detailed COCOMO considers all qualities of the standard version on each method of the software process and it divides the software into several parts, after that we apply COCOMO in these several parts to calculate effort.



c) **SEER-SEM:** The Software estimation model (SEER-SEM) is a system evaluation and resource estimation product. It is very famous and broadly used model. This model is used to predict the cost and the risk associated with the project. The effort is calculated using following Equation:

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

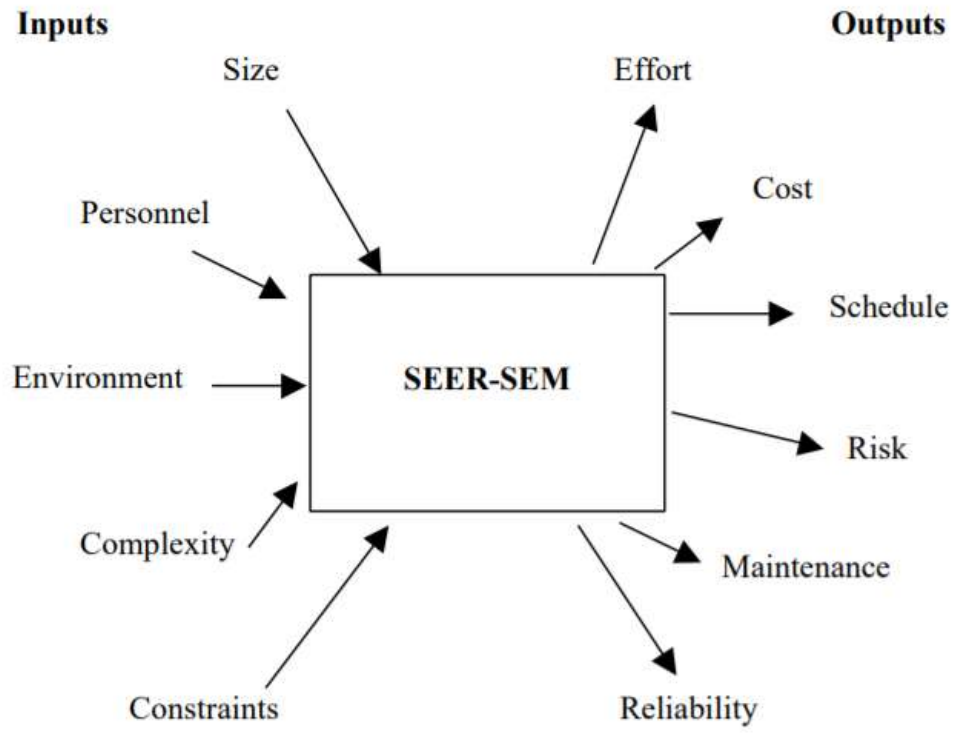


Fig. Inputs and Outputs for SEER-SEM Technique

d) Checkpoint: Checkpoint is a knowledge-based software project estimation tool. Software productivity research (SPR) developed checkpoints. It uses functional points as the measure of size. Checkpoint has 3 processes of SDLC:

a) Estimation
b) Measurement
c) Assessment

e) ESTIMACS: It was initially developed as QUEST (Quick Estimation System) and later on it had been converted into MACS which was called as ESTIMACS.
ESTIMAC includes 5 different models:

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

These models are used one after another so that the output from one model can be used as an input in the next model.

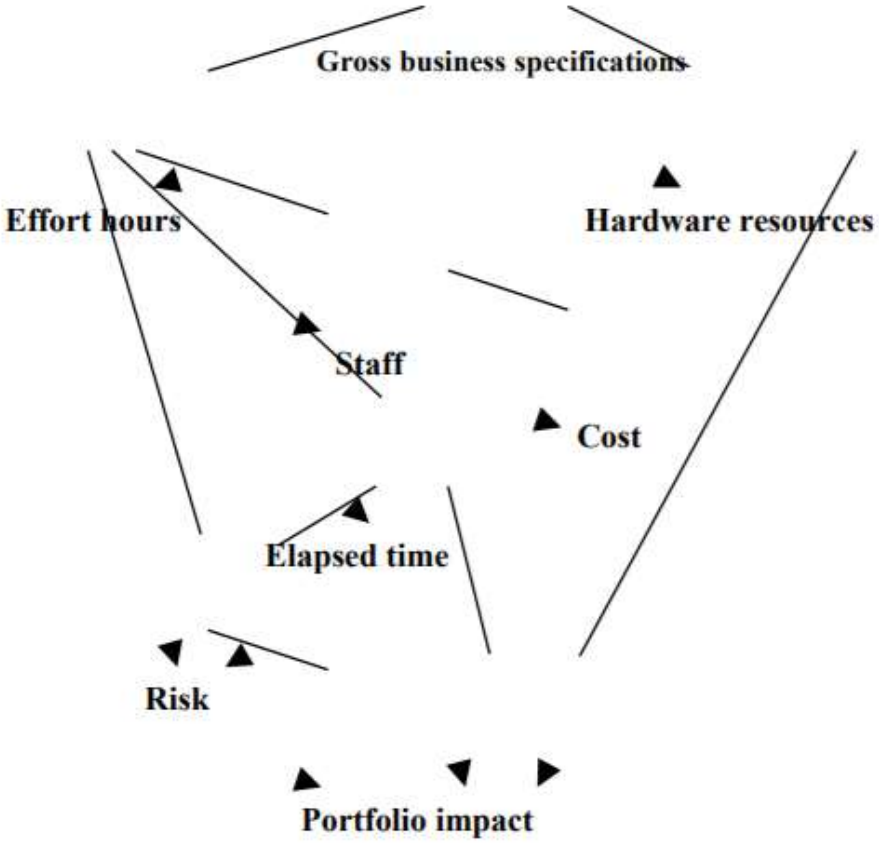


Fig. Rubin's Diagram of Interdependence of Estimation Dimensions

f) **PRICES-S** (Parametric Review of Information for Cost Accounting and Evaluation Software): It was developed by Radio Corporation of America (RCA). 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 schedule time.

It has 3 sub-models:

- Acquisition sub-model: It predicts cost & schedule of the s/w.
- Sizing sub-model: It estimates s/w size.
- Life cycle cost sub-model: It estimates maintenance cost.

g) **COSYSMO** (Constructive system engineering cost model (COSYSMO) is a parametric model. It calculates effort and time which is required in system engineering. It contains 14 cost drivers and 4 size drivers, a total of 18 parameters.

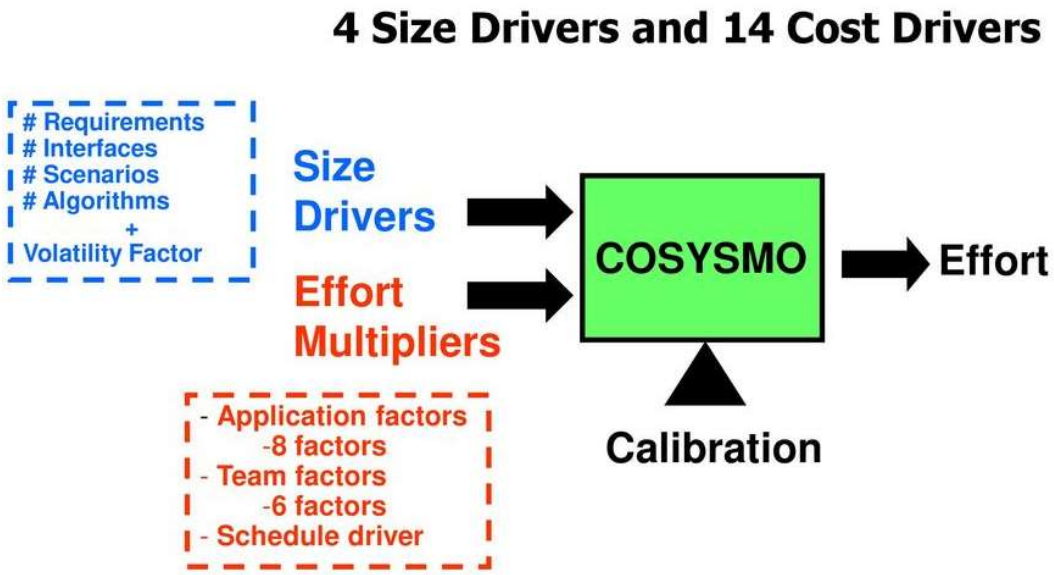


Fig. COSYSMO

A general form of the model is mentioned in the below equation:

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

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.

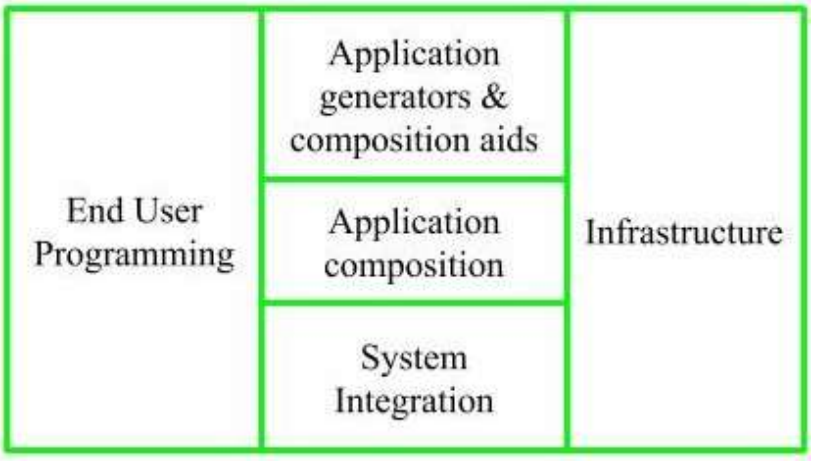
h) **COCOMO-II:** It is the advanced version of COCOMO. It estimates cost, effort and schedule of a software development activity. It includes 3 sub-models for estimation of software projects:

- Applications Composition sub-model: It based on the concept of Object Points.
- Early design sub-model: It calculates cost and duration of a project.
- Post-architecture sub-model: It used SLOC and FP as size metrics and it provides more accurate cost estimates.

COCOMO-II estimation model is summarized in the following equation:

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

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 depends on the model.



2. EXPERTISE-BASED / CONSENSUS BASED TECHNIQUES

An expertise-based technique is the ability of one or more people known as "experts" to work together to calculate 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. In this method, meetings were organized among project specialists so called experts to obtain some conclusion. After each round of meetings, the expert's opinions were presented with a review of each round.

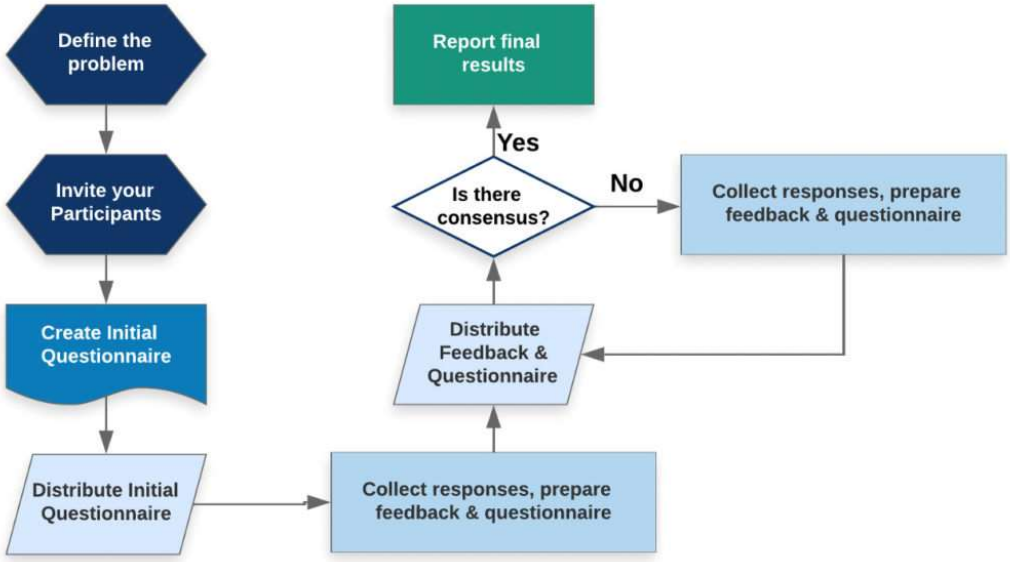
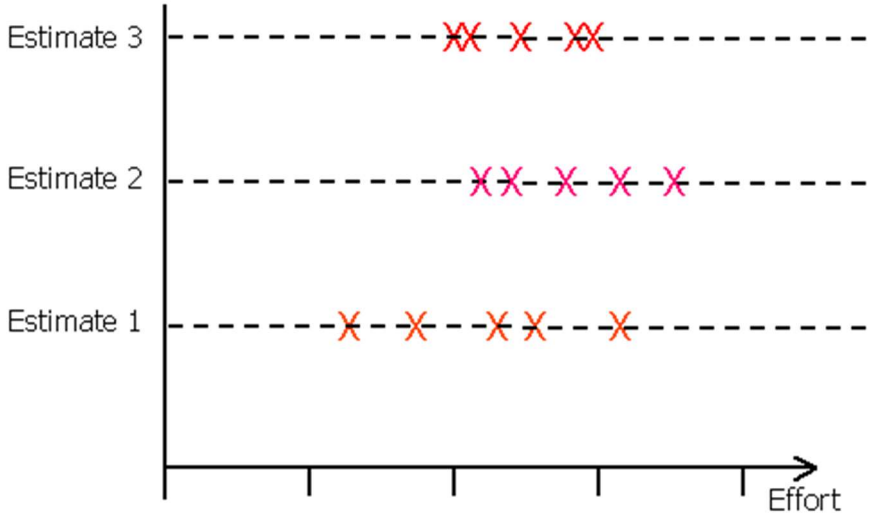


Fig. Flowchart for Delphi method

b) **Wideband Delphi Technique:** Wideband Delphi is the improved form of the Delphi methods Technique. In contrast to Delphi technique, the Wideband Delphi technique consists a group conversation in different rounds. This is an estimation technique for estimating effort by a group of experts.



c) **WBS (Work Breakdown Structure):** It arranges project elements in a hierarchical manner. To identify individual tasks by the project team, it divides the whole work into

subsystems. The WBS method produces 2 types of hierarchies: The 1st one indicates a software product & the 2nd one indicates activities required to develop the product.

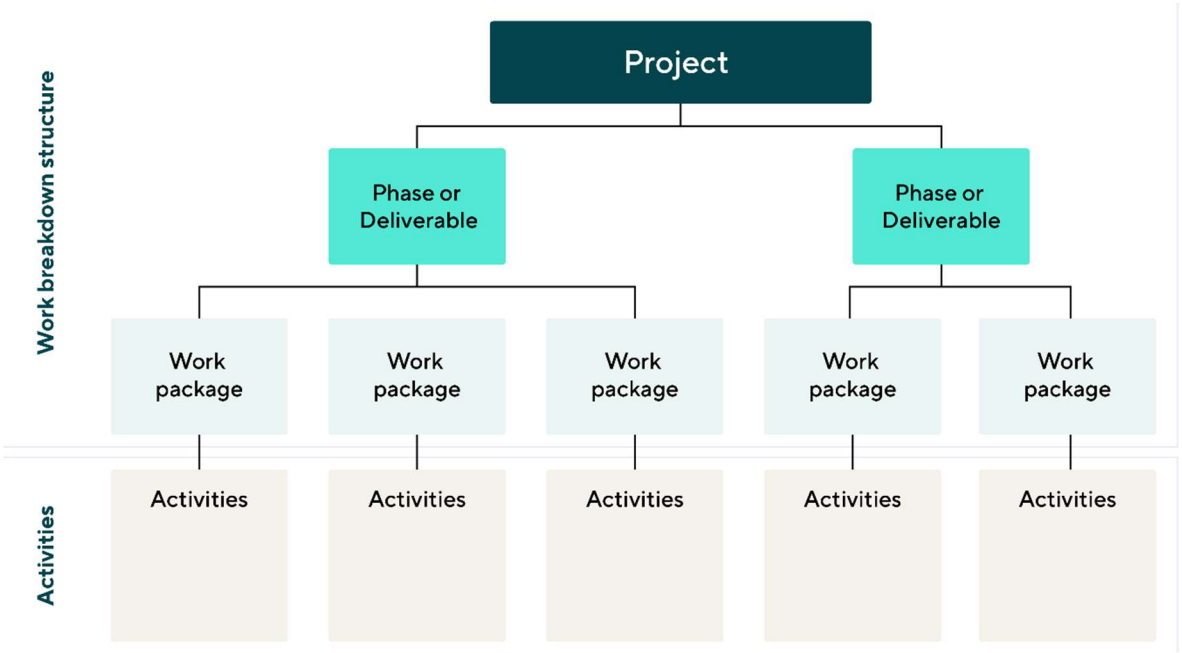


Fig. Work-Breakdown Structure Method

d) **Rule-Based Systems:** It uses the understanding of human experts in resolving real-world problems. Instead of representing knowledge of a human in a static manner, the rule-based system uses an "IF-THEN" condition. In this system, the human experts are indicated by a set of rules that illustrate what to do or what can be done in a situation.

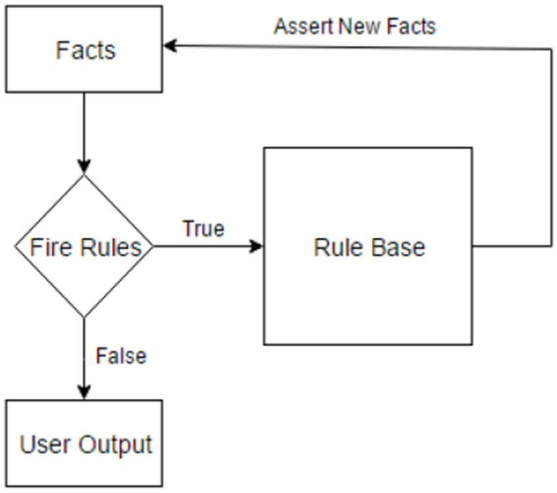


Fig. Structure of Rule-based Systems

e) **Planning Poker:** It is an expert judgment-based estimation technique. This technique creates an estimate by taking the opinion of multiple experts and then combining it. The people taking part in this method can be developers, analysts, project managers, testing people, etc.

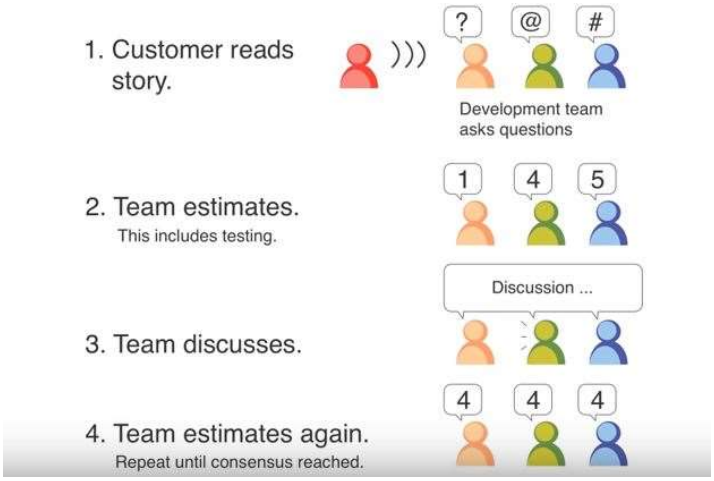


Fig. Working of Planning Poker Technique

f) **Top-Down Approach:** In this approach, total cost estimate of a project is taken either in algorithmic or non-algorithmic way from the software product. After that the project is divided into different components and subcomponents. Once the total cost of project is determined, a part of that cost is allotted to each component.

g) **Bottom-Up Approach:** In this approach, the process of cost estimation of the software system starts with the low level of it. The cost of each component is determined independently by the person who is developing the software. After that estimated costs of each component are combined to the highest level to calculate estimate of the overall software product.

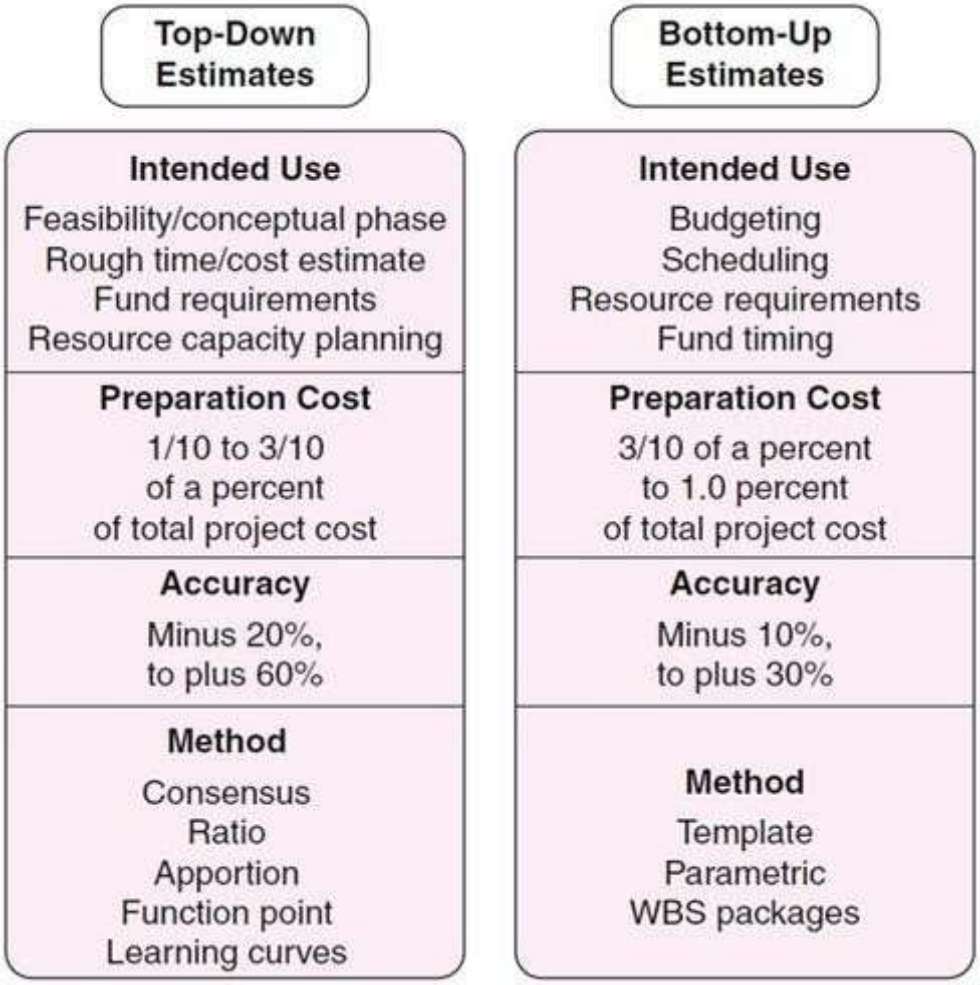


Fig. Top-Down Vs Bottom-Up Estimates

3. LEARNING-ORIENTED TECHNIQUES

These types of techniques use previous knowledge and current knowledge to build up software cost-estimation models. These techniques take reference from prior experiences & develop a model to calculate estimation cost.

a) **Case-Based Reasoning:** The CBR model assumes that the same cases have the same solutions. The system gains understanding w/o taking the consideration of human experts, the learning process takes reference from the determined solution which resides in DB & uses it as a reference to resolve new problems.

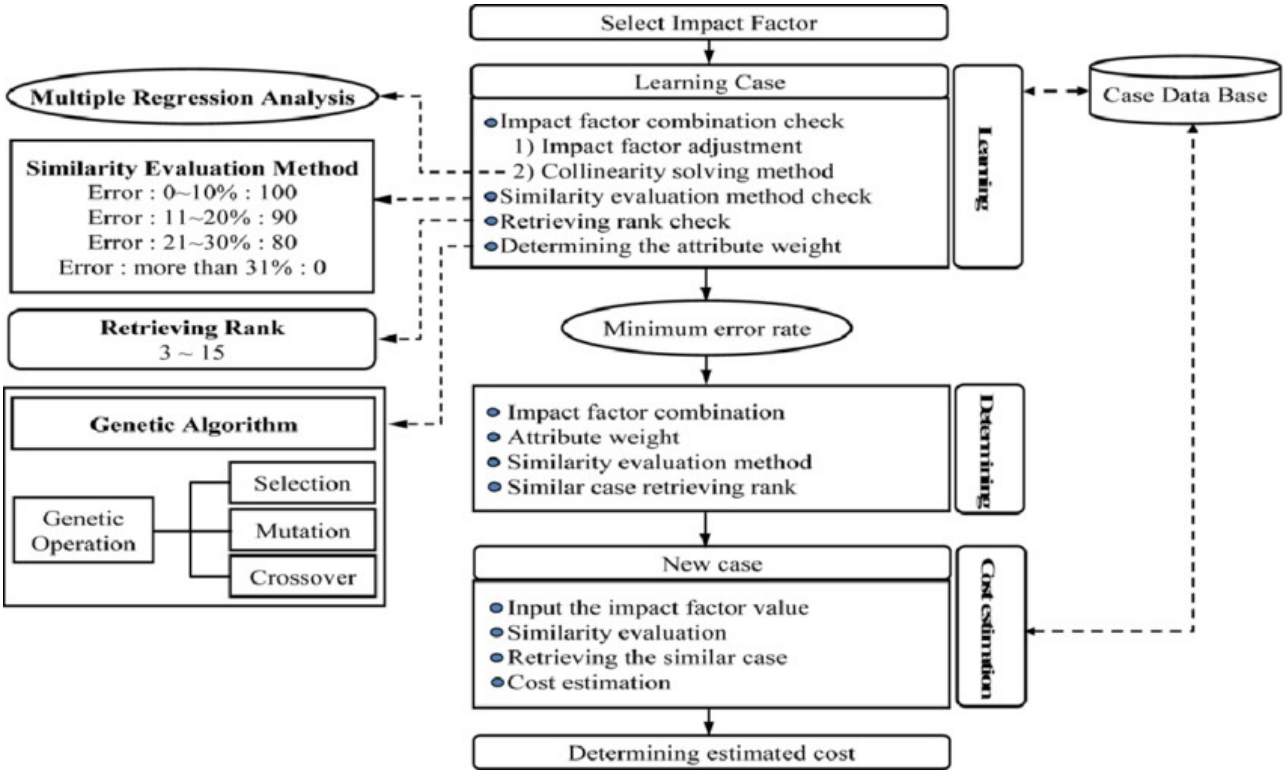


Fig. Work Flow for CBR Model

b) <u>Neural Networks</u>: These estimation models are trained using different kind of datasets & it automatically changes their algorithmic parameter values for reduction of errors.

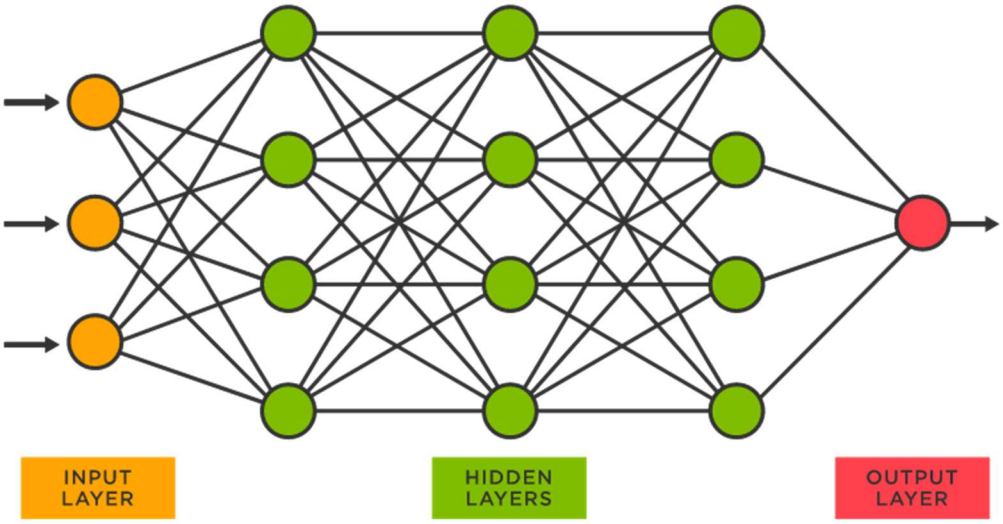
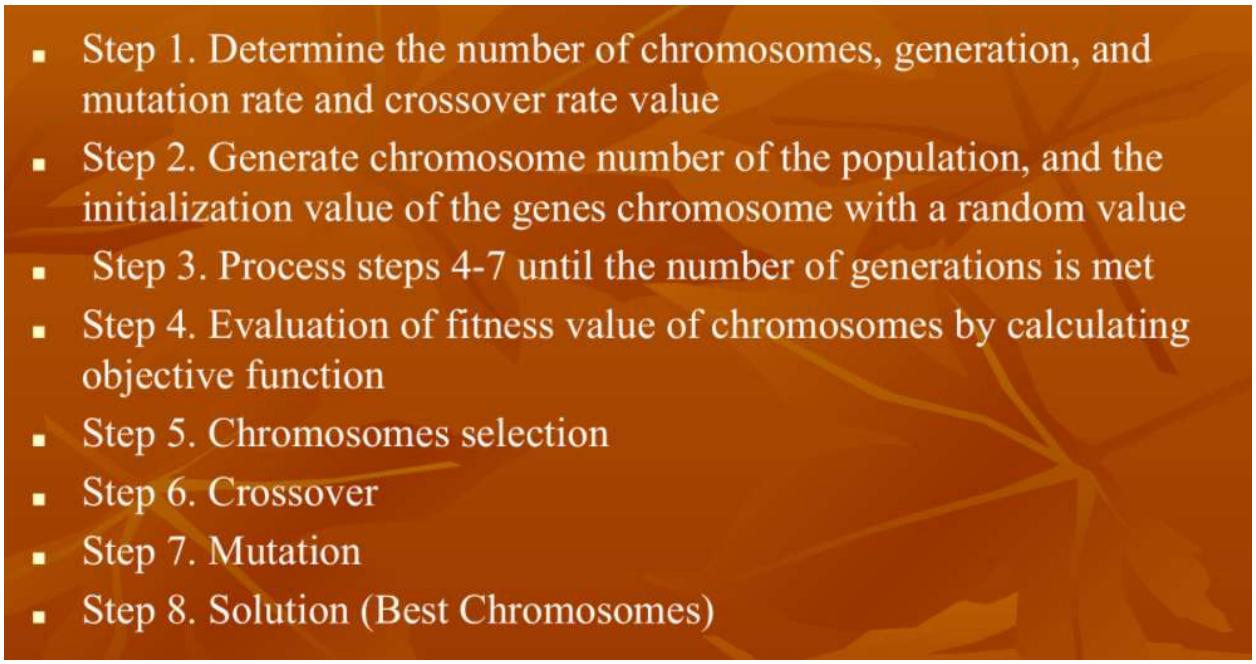


Fig. Neural Network Architecture

c) **Genetic Algorithm (GA)**: (GA) was created by John Henry Holland.

The process for genetic algorithm is as follows:



The main process of GA includes chromosome populations, selection & mutation on chromosomes having higher fitness values. Chromosomes are the fixed-length binary strings that represent solutions to the problems. Thus a new population of chromosomes is created & an optimal solution is produced.

d) **Genetic Programming:** Genetic programming (GP) is an extension of GA and is not limited to chromosomes. The modern GP which was based on tree type structure was given by Nichael L. Cramer, after which this study was expanded by John R. Koza. In GP chromosomes are programs that execute for getting optimal solution.

4. DYNAMICS-BASED TECHNIQUES

These types of techniques view effort and cost as dynamic because it changes with the span of system development process. This type of technique is suitable for planning & management.

a) **System Dynamics Approach:** It was designed to analyze and understand the dynamic behavior of complex systems. It is a simulation modeling technique which produces outputs in a graph that changes with period of time.

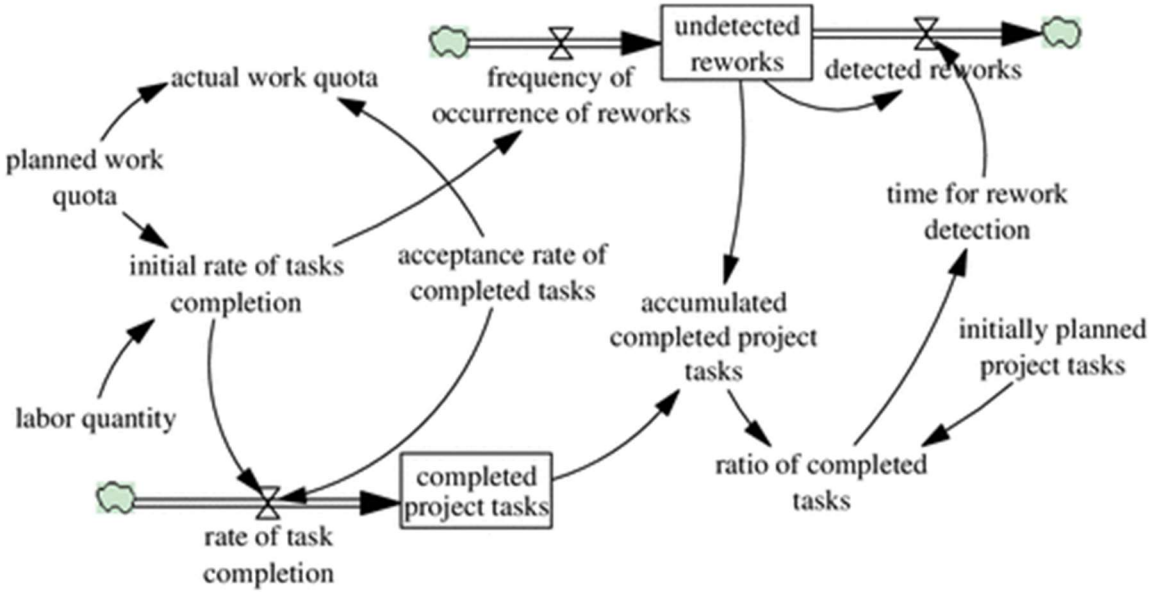


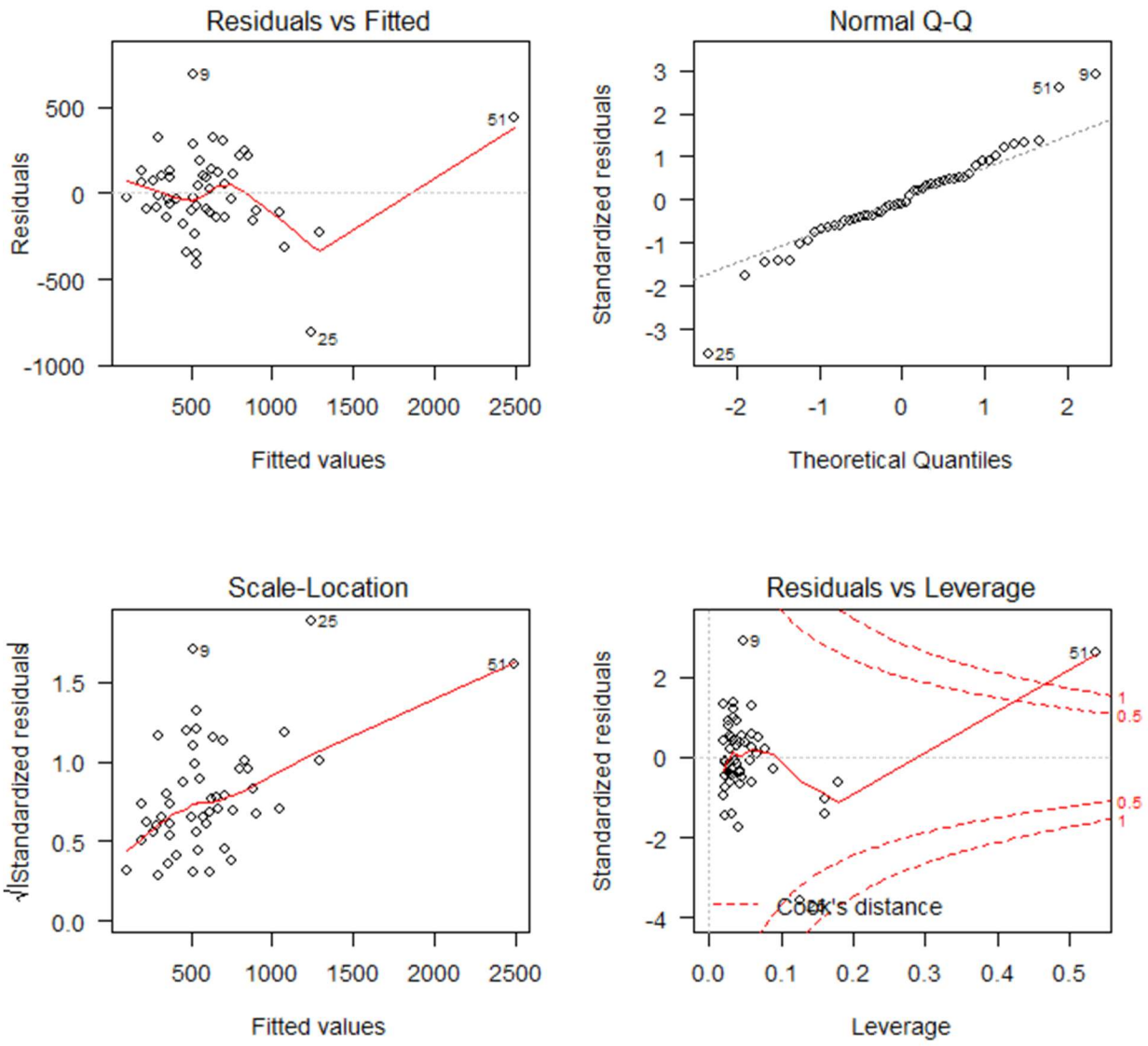
Fig. System Dynamics Approach

5. REGRESSION-BASED TECHNIQUES

These types of techniques are famous in the field of model building. They estimate software costs factors by using mathematical algorithms.

a) **Standard Regression:** 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.

b) **Robust Regression:** Robust regression is an advancement of the OLS approach. Robust regression is used in screen outliers for software metric models. Various parametric estimation techniques use regression-based approach because it is simple & widely used.



b) **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.

c) **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. Fuzzy frameworks have been used for programming improvement models.

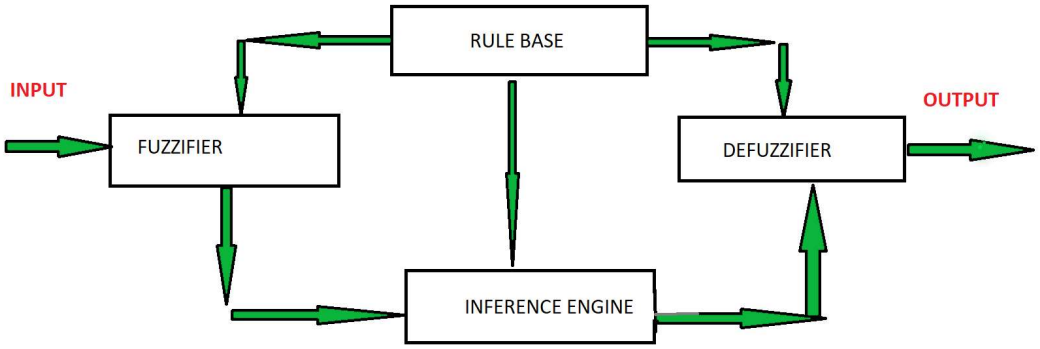


Fig. Fuzzy Logic Architecture

6. SIZE-BASED ESTIMATION TECHNIQUES

Size based estimation techniques are utilized for estimating the size for software development projects and some of them are discussed below.

a) **Function Points:** Function points (FPs) were introduced by Allan Albrecht in 1979. This approach calculates the size estimation of software & cost. FPs is used to evaluate the performance of software and Unadjusted Function Point (UFP) which is the unit of measurement.

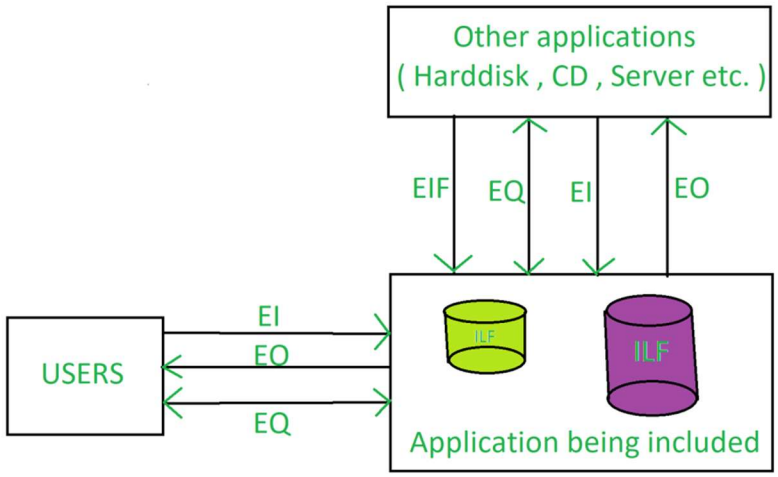


Fig. Function Point Analysis

- 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 extension of the FPA approach. It gives 2 extra information types & 4 new value-based types.

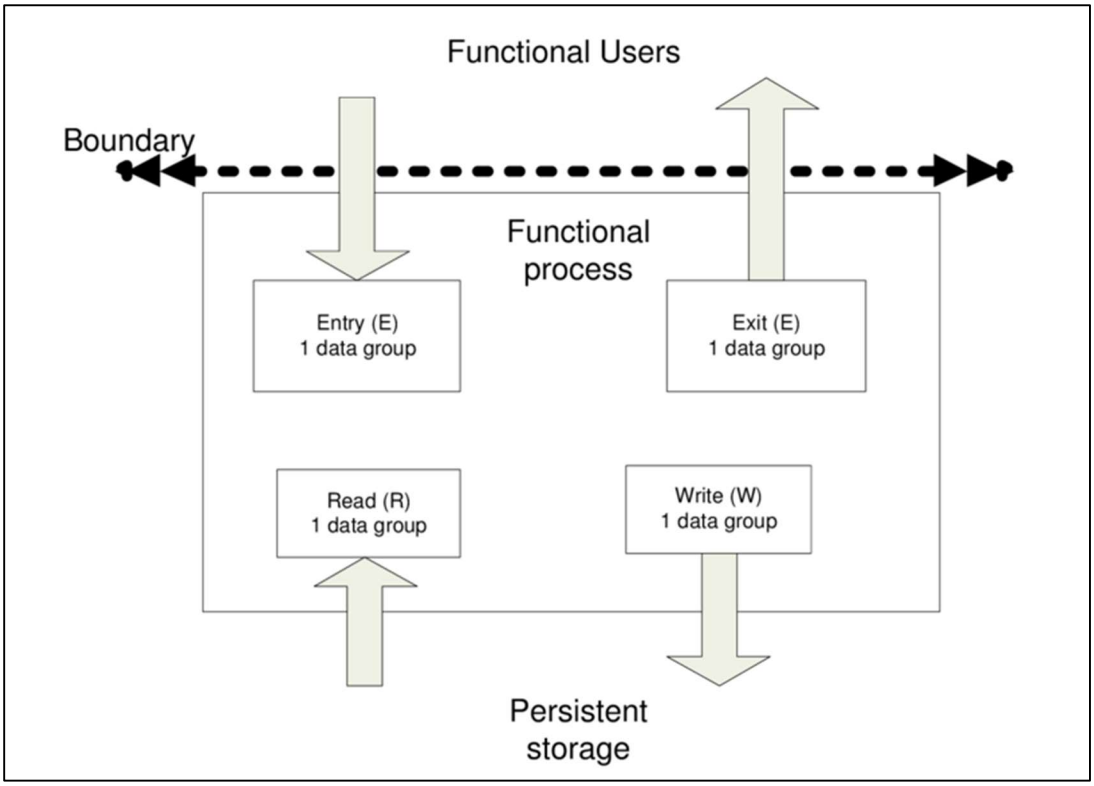


Fig. Function Point Concept

c) **<u>Use Case Points (UCPs):</u>** UCP is used for the specific frameworks which is a collection of use cases. Based upon the types of use cases, the UCP is calculated & then TCF is evaluated on the basis of UCP. UCP is calculated by using the formula:

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

where TCF is Technical Complexity Factor, UUCP is Unadjusted Use Case Points and ECF is Environment Complexity Factor.

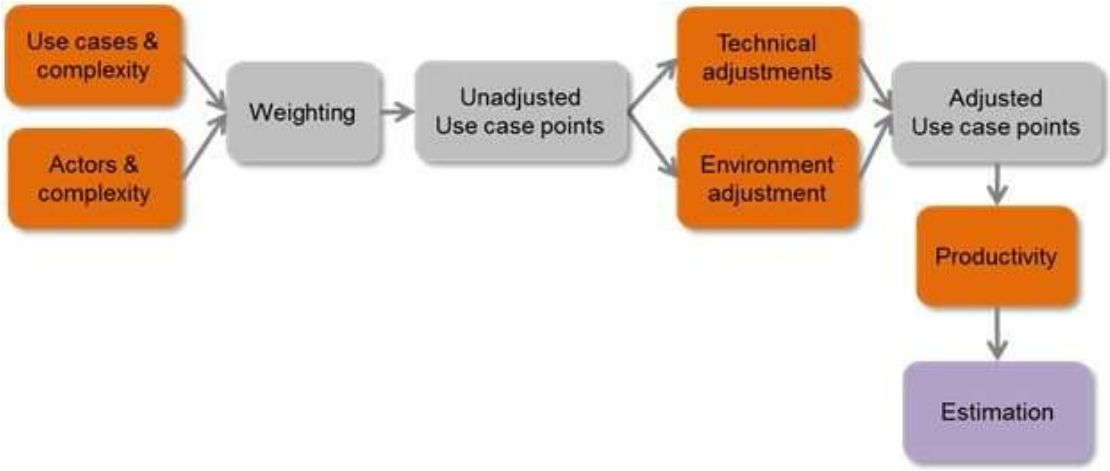


Fig. Work Flow for Use Case Point Method

7. COMPOSITE TECHNIQUES

Composite techniques include two processes to describe the most useful structure for estimation.

a) **<u>Bayesian Approach:</u>** Bayesian approach is an evaluating approach which is the advancement of the COCOMO-II model. The Bayesian approach has advantages; although, it includes initial learning of experts. Bayes formula is described 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.

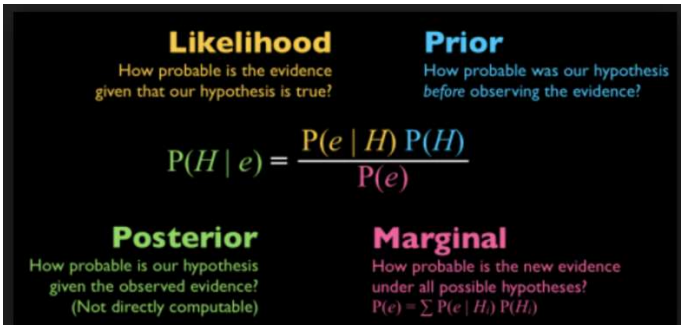
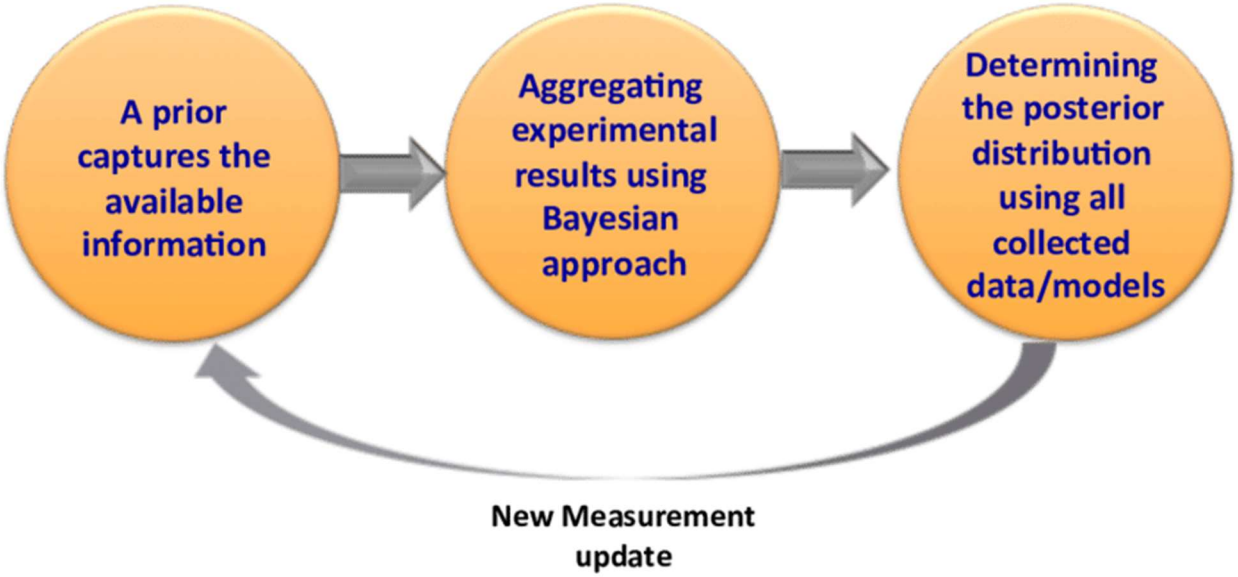


Fig. Bayes Formula for Likelihood Prediction



Work Flow Diagram for Naïve Bayes

The next section discusses the results and analysis of the study.

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 knowledge acquisition. • Improves eventually as case base grows • The learning ability of CBR systems helps in maintaining 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 independent variables.	• Difficult to design • Needs training to operate. • High processing time is required for 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 of the 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 is not 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 the project lifecycle. • Independent of how the requirements of the 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 project lifecycle. • Easy to use. • When estimation is performed by • skilled people, estimates would be close to 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

Our project research describes numerous cost estimation techniques with their advantages and disadvantages in different work environments. During our research, we realized that cost estimation is an interesting area in software project management field.

There have been ample researches that propose numerous project expense estimation techniques 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.