A Survey on Software Cost and Risk Assessment Models

Poojanjali Karari
Computer Science Department
SVITS
Indore, India

Pooja Jain
Assistant Professor
Computer Science Department
SVITS
Indore, India

ABSTRACT
Software development is a work of significant effort and team work. A number of different phases are required to develop a bug free and assured quality product. In this context, the appropriate cost and effort estimation is a complex task where the various factors are affecting the development such as change management, expertise, development environment and others. On the other hand, the risk in all phases are also carried out for the technical and cost affecting scenarios. To deal with these issues in software development life cycle an effective solution is needed to be produce. In this context, this paper focused on analysis of various cost and risk analysis techniques that are traditionally available and frequently used in software development industries to maximize the production and reduction of development issues.

Keywords
Software development, SDLC, risk factors, cost affecting factors, survey.

1. INTRODUCTION
Set of instructions, program and applications which is used for controlling and managing various functions of a device such as computer is termed as software. Hardware categorizes into physical part and software as virtual in a device. Software product's development specifies a structure called as software development life cycle. Six phases depict software development life cycle model:

1.1.1 Requirement gathering and analysis: gathering of business requirements are done. Project managers and stake holders deal with this phase, then validation of requirement is done thereafter a requirement specification document is prepared for the guideline of next phase of model.

1.1.2 Design: requirement specification leads to system and software design which helps to specify system requirements, hardware and overall system architecture also system design is input to next phase.

1.1.3 Implementation / Coding: here work is divided into unit /modules, actual coding is done. Developer deals with this phase, the longest in sdlc.

1.1.4 Testing: here testing of code is done with respect to requirements in order to make sure that the product is able to solve the need according to requirements. Here both functional testing which includes unit testing, integration testing, system testing, and acceptance testing and non-functional testing is done.

1.1.5 Deployment: successful testing leads to delivery of product to customers. Here the customers will do beta testing. If bugs or change in requirement is found then it is reported to engineering team and after changes final deployment is done.

1.1.6 Maintenance: regular maintenance of the developed product needs to be done from time to time.

2. BACKGROUND
This section provides the overview of the different terminologies and their definition which is used in further paper.

A. Cost estimation

Iterative process of developing an approximation of resources needed to complete software project activities is termed as software cost estimation.it should be done throughout the life cycle, when cost estimation is done throughout the life cycle, it helps in refinement of the estimate as more data is available ,also progress of the project and whether deadlines will be met up .final assessment of the entire cost estimation will lead company to refine estimation in future as well as the developers will be able to review the development process.

B. Uses of risk analysis and cost estimation

a. Risk analysis is useful in following situations:

- To neutralize possible problems while planning projects.
- Deciding to move forward in the next phase or not.
- Improving safety and managing potential risks.
- Changes in government policy or environment or new competitors in the market.

b. Cost estimation is useful in following situations:

Continuous re estimation of cost helps to compare target against actual milestone and to identify necessary corrections to budget and schedule. Iteration as an important tool improves estimation quality. necessary feedback is also given to improve the estimation quality in future.

C. Cost estimation models

a. COCOMO Models

Constructive Cost Model (COCOMO) is one of the widely used algorithmic software. The basic or simple form of COCOMO model:

$$\text{MAN-MONTHS} = K1 * (\text{Delivered Source Instructions})^{K2}$$

Where K1 and K2 are two parameters, both are dependent on the application and development environment.

COCOMO model is a type of regression model which is based on the analysis of 63 selected projects. Here KDSI act as input. Major problems such as:

1. Size is estimated with great uncertainty value in the early phase of system life-cycle. So, accurate cost estimation cannot be done.
2. Since 63 selected project's analyses is done for the derivation of cost estimation equation. Recalibration is necessary to avoid problems outside its environment.

In order to overcome the difficulties of estimating cost of software developed for new life cycle such as rapid-development process model, reuse-driven approaches, object-oriented approaches and software process maturity initiative, newest version COCOMO 2.0 was developed. The major capabilities of COCOMO 2.0 are a tailor-able family of software size models, which includes object points, function points and source lines of code; nonlinear models for software reengineering; software diseconomies of scale can be modelled with this approach; and several additions, deletions, and updates to previous COCOMO effort-multiplier cost drivers. It can act as a framework for an extensive current data collection and analysis for the refinement and calibration of model's estimate capabilities.

b. Putnam model

Putnam model is a popular software cost estimation model. The form of this model is:

$$\text{Technical constant } C = \text{size} * B^{1/3} * T^{4/3}$$

$$\text{Total Person Months } B = 1/T^4 * (\text{size}/C)^3$$

T= Required Development Time in years

Size is estimated in LOC

Where, C is a parameter dependent on the development environment and it is determined on the basis of historical data of the past projects.

Rating:

- C=2,000 (poor),
- C=8000 (good)
- C=12,000 (excellent).

The Putnam model has a high involvement of development time: development time is inversely proportional to the person-months needed for development. One significant problem with the PUTNAM model is that it is based on knowing, or being able to estimate accurately, the size (in lines of code) of the software to be developed. Software size can be uncertain resulting in the inaccuracy of cost estimation. According to Kemerer's research, the error percentage of SLIM, a Putnam model based method is 77.287%.

c. Function Point Analysis Based Methods

Estimators are required to estimate the number of SLOC in order to get man-months and duration estimates in case of the above two algorithmic models. In terms of the functions that the systems deliver to the user Function Point Analysis is another method of quantifying the size and complexity of a software system. ESTIMACS and SPQR/20 are some of the proprietary models for cost estimation which have adopted a function point type of approach. Allan Albrecht at IBM developed the function point measurement method which was published in 1979. Function points have offered several significant advantages over SLOC counts of size measurement. Two steps of counting function points:

- **Counting the user functions**. Function counts comprises of five basic software components: external inputs, external outputs, external inquiries, logic internal files, and external interfaces, there are three complexity levels: simple, average or complex. The summation of these numbers, in accordance with the complexity level, is the number of function counts (FC).
- **Adjusting for environmental processing complexity.** The final function points is gained by multiplying FC by an adjustment factor that is determined by considering 14 aspects of processing complexity. This adjustment factor allows modification of FC by at most 35% or -35%.

Following are some of the advantages of function point analysis based model are:

1. Function points can be estimated from requirements specifications or design specifications, thus making it possible to estimate development cost in the early phases of development.
2. Function points are independent of the language, tools, or methodologies used for implementation.
3. Non-technical users have a better understanding of what function points are measuring since function points are based on the system user's external view of the system.

D. Risk assessment models

The process used to identify and assess factors that may risk the success of a project or become a hurdle in achieving a goal is termed as risk analysis. Another term for this process is project impact analysis (PIA). In-depth cost-benefit analysis needs to be conducted. It helps the organization management to examine and assess a proposal before it becomes a live project. The risk analysis process is organized and carries out in the analysis phase of the SDLC. Here the interested parties are charged with building their case and presenting the proposal to the management review committee for approval and initial funding. In [1] authors had depicted some of the famous risk assessment models which are as follows:

a. Software Risk Assessment Model (SRAM)

SRAM makes use of comprehensive Questionnaire Test whose results show that using the risk indicator; it is possible to predict the possible outcome of software projects with good accuracy. Considering the following nine critical risk elements:

1. Complexity of software
2. Staff involved in the project
3. Targeted reliability
4. Product requirements
5. Method of estimation
6. Method of monitoring
7. Development process adopted
8. Usability of software and Tools used for development
9. Finally, a set of questions is carefully chosen for each of these elements with three choices of answers each.

Following which the answers are arranged in increasing order of risk. The method of prioritization is used as a single step of risk assessment but do not specify how prioritization would be done.

b. Software Risk Assessment and Estimation Model (SRAEM)

Risk exposure and software metrics are two parameters of risk management based on Mission Critical Requirements Stability Risk Metrics (MCRSRM) which is used in this model.in this model not only evaluation but estimation of the risk is also done. Initially the model estimates the sources of uncertainty using different paradigms such as measurement error, model error and assumption error. This model also uses the concept of function points to explain these errors. This model is different from other existing models because the other models do not consider the issues related to requirement analysis.

c. Risk Identification, Mitigation and Avoidance Model for Handling Software Risk (RIMAM)

The model RIMAM is used to provide prototype to handle various risk factors which comes from nature requirement, delivery deadline, and staff turnover experience. It identifies and avoids the risk impact by minimizing the risk. This model can be implemented easily with minimum cost. The easy and less costly processing may ensure that there are less chances of risk or further delay in case of already affected or risky area. The best part of the model can be customized with respect to the environment in which it is being used.

d. Software Risk Assessment and Evaluation Process using Model Based Approach

A better technique of risk estimation and risk prioritization. In SRAEP, software fault tree approach (SFTA) is used to identify and analyze the risk. In SFTA, a high-level threat is further decomposed into intermediate objectives which can be further decomposed into individual attacker actions. Attacker actions are connected by AND or OR relationship. Generally, the exchange of an OR node with an AND node is expected to increase the safety of software development project. After the Identification of risk, several countermeasures come across for risks. An appropriate countermeasure, a new technique known as Risk Reduction Leverage (RRL) is introduced which is a simple calculation that gives a numeric value to a countermeasure enabling different countermeasures to be compared. Mathematically it can be written as: RRL= Reduction in Risk exposure / Cost of Countermeasure After computation of risk and value of RRL and its prioritization the next step is team review and action planning.

3. LITERATURE REVIEW

A. Previous research and implementation on cost estimation models includes:

In [2] authors had proposed a new model based on COCOMO II which has two input's group from COCOMO II cost drivers and other effective input parameters, Complexity, Size, Experience and Mode, and one output, effort estimation. It shows in Fig. 1 in below.

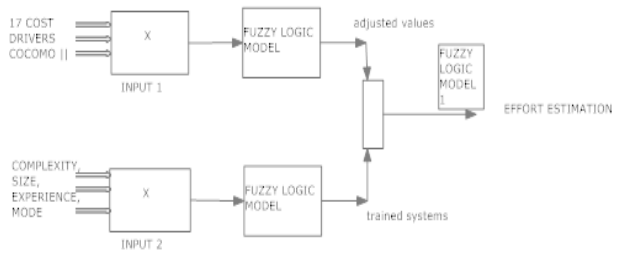

Fig 1. Model used [2]

Input group 1: COCOMO II cost drivers,

Input group 2: Effective parameters (Complexity, Size, Experience, Mode),

Output: Effort estimation

The new fuzzy model that proposed in this paper has 21 inputs and one output. The inputs are the complexity to modify the modules, the experience of developers in software project, COCOMO II cost drivers, and the size of the alteration. The output from the inference system is the effort time estimated in days.

The MATLAB fuzzy inference system (FIS) was used in the fuzzy calculations, in addition to the Max-Min composition operator, the Mandani implication operator, and the Maximum operator for aggregation. As the resulting values were more appropriate when compared to the other evaluated techniques (Centre of Area (COA) and First of Maximum (FOM)), the defuzzification of the output "effort" used the Mean of Maximum (COM) technique in this work. Once the new fuzzy model was developed, and the pertinence function and linguistic rules were included and adjusted, an artificial dataset based on COCOMO II model ware selected. The algorithm for fuzzy set learning in a Mamdani-type fuzzy system is following this four-step procedure:

- Choose a training sample and propagate the input vector across the network to get the output.
- Determine the error in output, and the error gradient in all the other layers.
- Determine the parameter changes for the fuzzy weights and update the fuzzy weights.
- Repeat until the fuzzy error is sufficiently small after an epoch is complete.

This paper proposed a model for handling imprecision and uncertainty by using the fuzzy logic systems. Software effort estimation based on fuzzy logic is an attempt in the area of software project estimation. This model aims to provide a technique for software cost estimation that is better than other techniques in terms of the accuracy of effort estimation. This model has tried by applying fuzzy logic on the algorithmic and non-algorithmic software effort estimation models accurate estimation is achievable.

Another cost estimation model based on use case points, Use Case Points (UCP) is very important method to estimate the total effort in software development projects. While the technique of Activity-based Costing (ABC) serves as the calculation of costs in each of the activities, especially the allocation of project resources. ABC technique consists of five stages of estimates based on the allocation of its resources, by having the total of UCP first. The result of this research is a proposed model of integration between UCP and ABC method or also called UCPabc [3].

There are six main phases to integrate two methods (UCP and ABC) for counting software cost estimation (also called as an integration model UCPabc). The explanations of each phase are:

- Analyse the problem for constructing the proposed models
- Gather Data from Several Software Development Projects
- Count Effort (UCP's Result)
- Analyse Resource, Activity, and Cost Rate (ABC's Result)

- Distribute UCP Total Effort Estimation Based on Product Relative Weight (PRW)
- Determine Cost Object

The proposed model to integrate the UCP method and ABC techniques shown in Fig 2 The total effort estimation which is an output of the UCP method was added to the three main components in the ABC technique, such as: the identification of resources and activities, cost rate for resource and activity, and the product relative weight. In the end, UCPabc generates cost object which can estimate cost of each software development project.

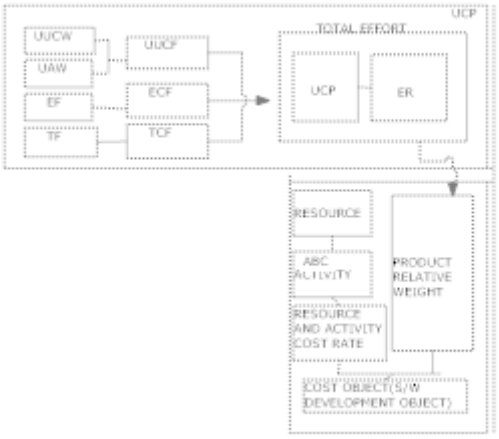

Fig 2. The Integration Model of UCP and ABC for Software Development Cost Estimation

Hybrid model:

Another proposed methodology [4] is based on algorithmic & non-algorithmic methods such as a FP size estimate, COCOMO II & ANN. The combinations of all these methods help in estimating the cost of the software. Following is the objective of the given method:

2.1.1 The first objective is to choose accurate method for calculating the size as it plays a vital role in calculating cost and effort.

2.1.2 To improve the performance of existing methods we need to create a hybrid tool for software cost estimation that helps in a software development organization.

Given system follows specific steps in which the flow is Maintained.

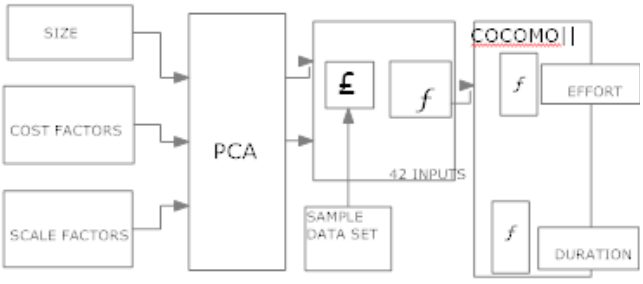

Fig 3. Hybrid model

Input

1) SIZE: Calculate the TCF, FP and size using the following equations:

$$TCF=0.65+0.01\left(\sum F_{P_{-var\ i(i=1to\ 14)}}\right)$$

FP= UFP*TCF

SIZE (inKLOC) = (FP*select language)/1000

2) Cost factors

3) Scale factors

B. Classification using PCA and Artificial neural network

Output using COCOMO II

$Effort=A*[SIZE]^{E}*\prod_{i=1}^{17} EM_i$

$Where\ E=B+0.01*\sum_{j=1}^{5} SF_j A=2.94,\ B=0.91$

$Time_{Months}= C*(Effort)^{F}$

$Where\ F=D+0.2*0.01\sum_{j=1}^{5} S\ F_J C=3.67,\ D=0.28$

$People=\frac{Effort}{Time}$

COCOMO II uses function point size estimation method for calculating the size of the software.it is composed of 17 effort multipliers and five scale factors.

Cost = (Effort *$\text{Payment}_{\text{Month}}$)

In the Hybrid model, estimated effort is very close to the actual effort. Thus, Hybrid model helps in improving the accuracy of software cost estimation. Performance measure depends on the following equation:

$$MRE=\frac{|\ \text{ActualEffort}-\text{EstimatedEFFORT}|}{\text{ActualEffort}}$$

B. Previous research and implementations of software risk assessment model

The proposed risk assessment model takes the following nine critical risk elements: complexity of software, staff involved in the project, targeted reliability, product requirements, method of estimation, method of monitoring, development process adopted, usability of software and tools used for development into considerations. Each element considers a **set** of questions with three choices of answers each. The answers are arranged such that risk is in increasing order [5]. For example, Software Complexity is one of the risk elements of software projects. The higher the complexity of the software, the higher is the risk. The overall risk level **R** may then be normalized as follows:

Normalized R = R' = (R - Rmin/) (Rmax- Rmin)

The risk levels of individual risk element may be computed. The model also allows levels **of** risk associated with Quality, Schedule and Cost of a project to be calculated separately. The model is tested using past projects of known outcome. It is concluded that the model can assess the level of risk of a software project and predict the possible outcome **of** the project with reasonable accuracy. However, it is to be noted that the success of a software product not only depends on the development risk but also on the marketing risk. The SRAM only addresses the development risk and does not assess the marketing risk.

Risk Assessment Methodology

In this model [6], the authors had used methodology based on UML and VModel development lifecycle. As we know, UML

is the most widely used for software development, and it is understandable by developers and designers. UML diagram also helps to manage complexity of system and architectural problems, and enhances comprehensibility. On other hand, V-Model is widely accepted development lifecycle for software development. It is best model for test driven software development. Each phase of V –Model have two objectives. One is Validation, and second is Verification as defamed by IEEE. Validation refers to requirements analysis and verification refers to evaluation of requirements at each phase with respect to validation.

In this methodology, it is assumed validation as identification of risks and verification as monitoring or evaluation of risks. They have seven phases of risk assessment in this model as Shown in Fig 4. Each phase of risk assessment, V -Model shows identification and evaluation of risks at start and end respectively. First phase is about requirements mapping as risks are common in requirements phase. In S. Li and S. Duo describe hazards caused by risks in software requirements in models and process. Second phase is about detail process analysis focuses to identify risks within processes that could be severe in future.

Third phase is to identify the risk elements in list form based on possibility in phase two. Risk Prioritization is one of modem ways of risk assessment in software development. X. Lin and D. Mingrong in focuses to indexing of risks to gain more accuracy in risk assessment. In fourth phase, they assume to prioritize risks. Solving risk items is also a critical part of software development. Phase five focuses on difficulty of risk solution by means of risk occurrence probability. We can identify solution difficulty by applying weights. Phase six focuses towards use of matrices to be applied to measure risks. As described by X. Lin and D. Mingrong in, there are different methods to measure risks but accuracy is not possible from all of them. Therefore, matrices must be selected basis on type of risk. The seven and last phase is to calculate-risks and takes measures to solve them.

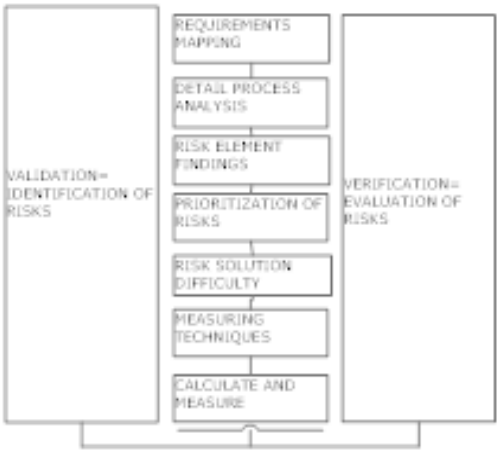

Fig 4. Phases of Risk Assessment V-Model

SRAEP:

In this paper, we have proposed a Software Risk Assessment and Evaluation Process (SRAEP) using model based approach. We have used model based approach because it requires correct description of the target system, its context and all security features. In SRAEP, we have used the software fault tree (SFT) to identify the risk [7].

Identify Context

In this model, the first activity is to identify context. This activity consists of identification of areas of concern, identification and evaluation of assets, and identification of security requirements. For instance, use case diagram with accompanying use case descriptions specify requirements, sequence diagram illustrate potential scenarios in the target of evaluation. Finally, in this category we have the security requirements identification.

Identify Risk using SFTA

In this paper, we have used the concept of software fault tree approach (SFTA) to identify the risk and analyze the risk. SFT A is the main component of the model based approach once a high-level threat has been determined, a method to decompose this threat into intermediate objectives is used.

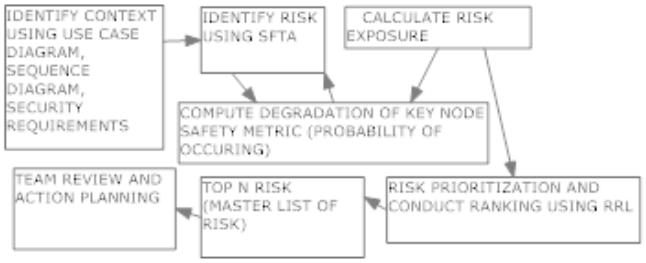

Fig 5. Proposed model of SRAEF

Compute Risk Exposure

Risk Prioritization

Action Planning

4. PROPOSED WORK

Comparison of various cost and risk estimation models:

A. Cost estimation models:

1. Model: New high-performance model

Input: Two inputs

1. 17 cost drivers from COCOMO II
2. Effective i/p parameters.
3. Complexity, size, experience, mode

Output: Effort estimation

Advantages and limitations: Accuracy of effort estimation, good performance, Determination of fuzzy rule set plays an important role.

Technology: MATLAB fuzzy inference system (based on fuzzy logic)

2. Model: Integration model UCPabc

Input: Effort, resource activity and cost rate

Output: Cost object

Advantage and limitation: Financing technique activity based costing act as an advantage to this model since ucp alone has some weakness.

Model limits to especially using resource sharing system, not validated for enterprise scale system.

Technology: Based on use case points.

3. Model: Hybrid model

Input: Size, cost factors, scale factors

Output: o/p using COCOMO II (effort duration)

Advantage and limitation: Proposed technology increases the correctness of the estimate without worsening the variability which improves the accuracy, turnaround time and performance of system

Technology: Based on FP size estimate, COCOMO II and ANN

B. Risk estimation models:

1. Model: SRAEF

Input: Use case diagram, sequence diagram, security requirement

Output: Risk estimation and risk prioritization

Advantage and limitation: Support the issues related to requirement and sources of estimate uncertainty.

Calculation of rrl.

Technology: Model based approach Concept of software fault tree (SFTA).

2. Model: SRAM

Input: Nine critical risk elements.

Output: Output is based on the ranking of choices in the questionnaire

Advantage and limitation: Addresses the development risk. Does not assess the marketing risk.

Secondly, no technology is used fully paper based which is not recommended in today's internet revolution

Technology: Paper based questionnaire work

3. Model: Risk assessment v-model

Input: Use case diagram-model

Output: Identification of risk and calculate the measure.

Advantage and limitation: Facilitate its usability and extendibility according to their own software assessment criteria.

Technology: Methodology based on uml and v-model life cycle.

Microsoft excel tool is used.

5. CONCLUSION

The paper provides the study of risk and cost estimation models. Both the techniques are used to minimize the costing of software development and to verify the feasibility of the developed product. But both the techniques are utilized in different manners to verify the upcoming issues and finding the solutions before it occurred. In this context, recently some models are developed that helps to investigate about the proposes involved in the software development cost and risk analysis. In near future, a case study of the software development cost and risk modeling technique with a tool is presented which used to estimate the cost of development and the risk of development.

6. REFERENCES

[1] Yogini Bazaz, Shashank Gupta, Om PrakashRishi, LalitSen Sharma, "Comparative Study of Risk Assessment Models Corresponding to Risk Elements", IEEE-International Conference on Advances in Engineering, Science and Management (ICAESM -2012) March 30, 31, 2012, pp. 61-66

[2] (Iman Attarzadeh, Siew Hock Ow, Proposing a New High Performance Model for Software Cost Estimation, 2009 Second International Conference on Computer and Electrical Engineering, pp 112-116)

[3] Renny Sari Dewi, Grandys Frieska Prassida, Sholiq, Apol Pribadi Subriadi," UCPabc as an Integration Model for Software Cost Estimation", 2016 2nd International Conference on Science in Information Technology (ICSITech), pp 187-192

[4] Lalit V. Patil, Rina M. Waghmode, S. D. Joshi, V. Khanna ,"GENERIC MODEL OF SOFTWARE COST ESTIMATION: A HYBRID APPROACH", 2014 IEEE International Advance Computing Conference (IACC), pp 1379-1384

[5] Say-Wei Foo and Arumugam Muruganantham, SOFTWARE RISK ASSESSMENT MODEL, IEEE 2000, pp 536-544

[6] Muhammad Rashid Naeem WeihuaZhu, Adeel Akbar Memon Adeel Khalid "Using V-Model Methodology, UML Process-Based Risk Assessment of Software and Visualization", 2014 International Conference on Cloud Computing and Internet of Things (CCTOT 2014),pp 197-202

[7] Mohd. Sadiql, Mohd. Wazih Ahmad2, Md. Khalid Imam Rahmani, SherJung," Software Risk Assessment and Evaluation Process (SRAEP) using Model Based Approach", 201O International Conference on Networking and Information Technology, pp 171-177.