ISSN 2394-739X

International Journal for Research in Science Engineering and Technology

ANALOGY OF SOFTWARE EFFORT AND COST ESTIMATION

[1]Dr. GurvinderKaur, [2]Ms. Arti Bajaj
[1]Director, [2]Research Scholar
[1]GNKITMS, [2]Mewar University

ABSTRACT- There is an increasing growth and demand of Web for completing business processes. The effort and cost estimation approach helps the companies to complete their web development projects with in time and budget. This paper examines the literature on Effort and Cost Estimation Techniques for Web applications. This paper provides a systematic review of previous web estimation studies particularly focusing on estimation, factors &methods of Estimation, Tools and models of Estimation. The paper reviews the progress and direct future research in this area.Main findings were that: (1) The estimation methods in most frequent use are expert judgement-based. (2) Structured review of typical software effort estimation terminology in software engineering textbooks and software estimation research papers. The review process provides the evidence that the term ‘effort estimate’ is frequently used without sufficient clarification of its meaning, and that estimation accuracy is often evaluated without ensuring that the estimated and the actual effort are comparable.

Keywords: Effort estimation, Cost estimation, software projects.

1. INTRODUCTION

The process of predicting the most realistic amount of effort required to develop or maintain software based on incomplete, uncertain and noisy input is known as Software development effort estimation. It may be used as input to project plans, iteration plans, budgets, and investment analyses, pricing processes and bidding rounds. The term “effort estimate” is used to denote as different concepts as most likely use of effort, the effort that corresponds to a probability of 50% of not exceeding the planned effort, or the effort used to propose a bid or price to the client.

Since 1960s the softwareresearchers and practitioners faced lot of problems of effort estimation for software development projects. Researchers emphasized on construction of formal software effort estimation models. The early models were typically based on regression analysis or mathematically derived from theories from other domains. Other approaches were also found based on case-based reasoning, classification and regression trees, simulation, neural networks, Bayesian statistics, lexical analysis of requirement specifications, genetic programming, linear programming, economic production models, soft computing, fuzzy logic modeling, statistical bootstrapping and combinations of two or more of these models. The most common estimation methods are the parametric estimation models COCOMO, SEER-SEM and SLIM. Then came the updated release of COCOMO II in the year 2000. The estimation approaches based on functionality-based size measures, e.g., function points, is also based on research conducted in the 1970s and 1980s, but are re-calibrated with modified size measures and different counting approaches, such as the use case points or object points in the 1990s and COSMIC in the 2000s.

1.1 Estimatesand Estimation

An estimate is a prediction of how long a project will take or how much it will cost. But estimation on software projects interplays with business targets, commitments, and control. Manybusinesses have important reasons to establish targets independent of software estimates.But the fact that a target is desirable or even mandatory does not necessarilymean that it is achievable.A target is a description of a desirable business objective; a commitment is apromise to deliver defined functionality at a specific level of quality by a certain date.A commitment can be the same as the estimate, or it can be more aggressive or moreconservative than the estimate. In other words, do not assume that the commitmenthas to be the same as the estimate.Software estimates should be continually tracked and updated throughout the life cycle of a project. Software estimates should be recalculated monthly and after any major redirection of the project by the customer. Each time an estimate is updated, the assumptions and inputs shall also be updated to reflect the most current information.

Software project estimation is one of the challenging and important activities in software development. Proper planning and control is not possible without a sound and reliable estimate. The software industry doesn't estimate projects well and doesn't use estimates appropriately, because of this all suffer far more than as a result and we need to focus some effort on improving the situation. Under-estimating a project leads to under-staffing it, under-scoping the quality assurance effort, and setting too short a schedule. Over-estimating a project can be just about as bad for the organization. The project is then likely to cost more than it should (i.e. a negative impact on the bottom line), take longer to deliver than necessary, and delay the use of your resources on the next project.

1.2 Relationship between Estimates and Plans

Estimation and planning are related to each other, but estimation is not planning, and planningis not estimation. Estimation should be treated as an unbiased, analytical process. Planning should be treated as a biased, goal-seeking process. With estimation it's difficult to define the estimate to come out to any particular answer. Its goal isaccuracy; the goal is not to seek a particular result. But the goal of planning is to seeka particular result. Itdeliberatelybiases the plans to achievespecific outcomes. Estimates form the foundation for the plans, but the plans don't have to be the sameas the estimates. If the estimates are different from the targets, theproject plans will need to recognize that gap and account for a high level of risk. If theestimates are close to the targets, then the plans can take less risk.Both are important, but the fundamental differences betweenthe two activities mean that combining the two trends to lead to poor estimates andpoor plans.

Fewsteps of planning that depend in part on accurateestimates: i) Creating a detailed schedule, ii) Identifying a project's critical path, iii) Creating a complete work breakdown structure, iv) Prioritizing functionality for delivery, v) Breaking a project into iterations and vi) Accurate estimates support better work in each of these areas.

2. STEPS OF ESTIMATION

The four basic steps in software project estimation are:

a) Estimate the size of the development product.

b) Estimate the effort in person-months or person-hours.

c) Estimate the schedule in calendar months.

d) Estimate the project cost.

a) Estimate the size of the development product: -Size isn't everything in a software project but it does influence most things (e.g. resources cost) so without accurate prediction of size it is difficult to plan. An actual estimate of the size of the software to be built is the first step to an effective estimate. Sources of information regarding the scope of the project should, wherever possible, start with formal descriptions of the requirements - for example, a customer's requirements specification or request for proposal, a system specification. If a project in later phases of the project's lifecycle, design documents can be used to provide additional detail. The level of risk and uncertainty in an estimate must communicate to all concerned and re-estimate the project as soon as more scope information is determined. There are two main ways for estimating product sizes are:

Wideband-Delphi Estimating - The Wideband-Delphi method is a way of attempting to get experts in predicting software size to come to a consensus on their predictions - important because experts often disagree. It is one of the widely used software testing estimation technique. It is based on surveys and basically collects the information from participants who are experts. In this estimation technique each task is assigned to each team member & over multiple rounds surveys are conduct unless & until a final estimation of task is not finalized. In each round the thought about task are gathered & feedback is provided. By using this method, quantitative and qualitative results are achieved. This technique gives good confidence in the estimation. This technique can be used with the combination of the other techniques.

Functional Point Method - Functional Point is measured from a functional, or user, point of view. It is independent of computer language, capability, and technology or development methodology of the team. It is based on available documents like SRS, Design etc. In this FP technique the weightage is defined to each functional point. Prior to start actual estimating tasks functional points are divided into three groups like Complex, Medium & Simple. Based on similar projects & Organization standards we have to define estimate per function points.

Total Effort Estimate = Total Function Points * Estimate defined per Functional Point

b) The effort in person-months or person-hours - After estimating the size of the product, the effort estimation is required. This conversion from software size to total project effort can only be done with the help of software development lifecycle and development process that you follow to specify, design, develop, and test the software. A software development project involves far more than simply coding the software – in fact, coding is often the smallest part of the overall effort. Writing and reviewing documentation, implementing prototypes, designing the deliverables, and reviewing and testing the code take up the larger portion of overall project effort. There are three main approaches for effort estimation: 1) Expert Estimation: An expert on the subject of effort gives judgment on this, 2) Formal Estimation Model: Using a proper model you feed the system with proper data to get some estimation, 3) Combination-based Model: The estimation arrives with a mixture of both expert and formal estimation procedures. There are different ways for each approach for estimating product effort:

1. **Work Break**-Down Structure: This seems to be the most common method. Using this method you break down the project to the small parts of works, tasks. Then, you estimate the effort for every task.This is an Expert Judgment method and it comes with two flavors: a) Three Point Estimation and b) Delphic Oracle. Using the Three Point method an expert gives 3 estimations for every task. Best Case, Most Probable, Worst Case. The effort for every task is the outcome of a weighted average of the three estimations where the most probable effort gets a higher weight.Delphic Oracle means that 3 different people estimate the task effort. The final task effort is the average.
2. **COCOMO II -** It is formal method that uses various parameters and a defined formula to estimate effort. Constructive Cost MOdel II (COCOMO II) is the latest major extension to the original COCOMO (COCOMO 81) model published in 1981. It accepts as input quantitative and qualitative weighted characteristics and produces effort estimation. It is an algorithmic approach to convert a size estimate into an effort estimate.

c) Estimate the schedule in calendar months - After estimating the effort the next step in estimating a software development project is to determine the project schedule. It involves estimating the number of people who will work on the project, what they will work on (the Work Breakdown Structure), when they will start working on the project and when they will finish (this is the “staffing profile”). After collecting all this information the next step is to lay it out into a calendar schedule. Historical data from any organization’s past projects or industry data models can be used to predict the number of people required for a project of a given size and how work can be broken down into a schedule. If no information is present, a schedule estimation rule of thumb [McConnell 1996] can be used to get a rough idea of the total calendar time required:

Schedule in months = 3.0 * (effort-months) 1/3

Opinions vary as to whether 2.0 or 2.5 or even 4.0 should be used in place of the 3.0 value.

d) Estimate the project cost - There are many factors to consider when estimating the total cost of a project. These include labor, hardware and software purchases or rentals, travel for meeting or testing purposes telecommunications (e.g., long distance phone calls, video-conferences, dedicated lines for testing etc.), training courses, office space, and so on. Exactly how to estimate total project cost will depend on how organization's allocates costs. Some costs may not be allocated to individual projects and may be taken care of by adding an overhead value to labor rates ($ per hour). Often, a software development project manager will only estimate the labor cost and identify any additional project costs not considered "overhead" by the organization..

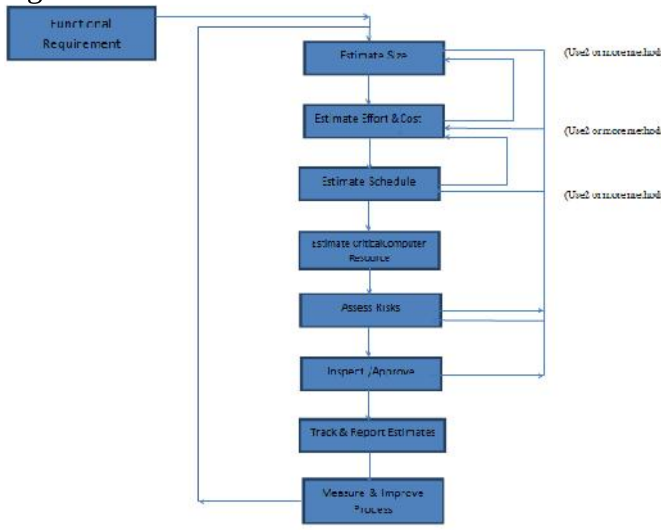

Figure 1.Estimation Process Flow

3. FACTORS FOR ESTIMATION

For any software testing estimation, technique it is highly recommended that following factors should be taken into account:

1. Domain Knowledge and core requirements
2. Risks and complexity of the application
3. Team Knowledge on the subject/skills
4. Historical data for the previous estimation for improvement and accuracy
5. Estimation should include buffer time
6. Bug cycles for the project
7. Resources availability (Like vacations, holidays, and sick days can have a great impact on your estimates)

However the techniques just provide the means for estimating but rely heavily on the team productivity variations, individual skills, complexity of the unknown factors like system environment and downtime.

4. RECENT ESTIMATION TECHNIQUES

Many different effort and cost estimation techniques have been 'proposed and used over the last 30 years. There are various techniques used in Cost and Effort estimation, they are:

1. 3-Point Software Testing Estimation Technique
2. Use – Case Point Method
3. Work Breakdown Structure
4. Wideband Delphi technique
5. Function Point/Testing Point Analysis
6. Percentage of development effort method
7. Percentage distribution
8. Best Guess
9. Ad-hoc method
10. Experience Based

3-Point Software Testing Estimation Technique:

It is based on statistical methods in which each testing task is broken down into sub tasks and then three types on estimation are done on each tasks. The formula used by this technique is:

Test Estimate = P + (4*N) + E / 6

Whereas P = Positive Scenarios or Optimistic Estimate (Best case scenario in which nothing goes wrong and all conditions are optimal.)

N = Negative Scenarios or Most Likely Estimate (most likely duration and there may be some problem but most of the things will go right.)

E = Exceptional Scenarios or Pessimistic Estimate (worst case scenario which everything goes wrong.)

Standard deviation for the technique is calculated as: Standard Deviation (SD) = (N – E)/6

Use – Case Point Method: It is based on the use cases where we calculate the unadjusted actor weights and unadjusted use case weights to determine the software testing estimation.Use case is a document which well specifies different users, systems or other stakeholders interacting with the concerned application. They are named as ‘Actors’. The interactions accomplish some defined goals protecting the interest of all stakeholders through different behaviour or flow termed as scenarios.

The formula used for this technique is:

- Unadjusted actor weights = total no. of actors (positive, negative and exceptional)
- Unadjusted use case weight = total no. of use cases.
- Unadjusted use case point = Unadjusted actor weights + Unadjusted use case weight
- Determine the technical/environmental factor (TEF) (if not available take as 0.50)
- Adjusted use case point = Unadjusted use case point * [0.65+ (0.01 * TEF]
- Total Effort = Adjusted use case point * 2

4.1 Work Breakdown Structure:

It is created by breaking down the test project into small pieces. Modules are divided into sub-modules. Sub modules are further divided into functionalities and functionalities are divided in sub-functionalities.Review all the requirements from Requirement Document to make sure they are added in WBS. Now you figure out the number of tasks your team needs to complete. Estimate the duration of each task.

4.2 Wideband Delphi technique:

In Wideband Delphi Method, work breakdown structure is decomposed for each task and is distributed to a team comprising of 3-7 members for re-estimating the task. The final estimate is the result of the summarized estimates based on the team consensus. This method speaks more on experience rather than any statistical formula. This method waspopularized by Barry Boehm to emphasize on the group iteration to reach to a consensus where the team visualized on the different aspects of the problems while estimating the test effort.

4.3 Function Point/Testing Point Analysis:

The FP technique is a direct indicator of the functionality of software application from the user's perspective. This is the most accepted technique used to estimate the size of a software project. This technique is a part of TMap. Base of this technique is function point technique. Here we convert function points into test points. In Test Point analysis, we usually carry out the following:

õ Dynamic Test Points
õ Static Test Points
õ Environmental Factor
õ Productivity Factor
õ Primary Test Hours
õ Control Factor
õ Total Test Hours

In Testing, This estimation is based on requirement specification document, or a previously created prototype of the application. To calculate FP for a project, some major components are required.

The major components are:

1. Unadjusted Data Function Points: i) Internal Files, ii) External Interfaces
2. Unadjusted Transaction Function Points: i) User Inputs, ii) User Outputs & iii) User Inquiries
3. Capers Jones basic formula:
4. Number of Test cases = [Number of Function Points] x 1.2
5. Total Actual Effort, TAE = (Number of Test cases) * (Percentage of development effort /100)

This method is done in a case when a detailed low level design document or requirement document is available (i.e measure of function point is available) & Previous data for development and testing is available. But now a days, when we are using agile and iterative methodologies to deliver projects, so most of the times all this documentation is not available.

4.4 Percentage of development effort method:

Here the assumption is that a more complex business application may require more

testing effort. The test effort required is a direct proportionate or percentage of the development effort.
If a previous project with 500 FPs required 50 man hours for testing, the percentage of testing effort is calculated as:
P = (50 / 500) * 100 =10%
For the current project with a development effort, say 1500 FPs, the testing effort is:
Total Actual Effort, TAE = 1500 * (P/100) = 1500 * (10/100) = 150 man hours.

4.5 Percentage distribution:

Here all the phases of SDLC are divided in parts and assigned effort in %. Like –
Project management 7%
Requirements 9%
Design 16%
Coding 26%
Test (all test phases) 27%
Documentation 9%
Installation and training 6%
Now testing % is further distributed into all testing phases:

All phases	**%**
Component testing	16
Independent testing	84
Total	**100**
Independent testing	**%**
Integration testing	24
System testing	52
Acceptance testing	24
Total	100
System testing	**%**
Functional system testing	65
Non-functional system testing	35
Total	100
Test Planning and Design Architecture	50%
Review phase	50%

4.6 Best Guess:

This technique is purely guesswork and based on the some sort of experience. The method is very common, but since it isbased on your gut feeling, its uncertainty contingency is probably around 200% or even higher.

4.7 Ad-hoc method:

The test efforts are based on tentative timeframe. The timeline set by managerial or marketing personnel or by client without any guess/experience. Alternatively, it is done until the budgeted finances run out. This is very common practice in extremely immature organizations and has error margins of over 100% at times.

4.8 Experience Based:

Analogies and experts:

õ Metrics collected from previous tests.
õ You already tested similar application in previous project.
õ Inputs are taken from Subject Matter experts who know the application (as well as testing) very well.

5. STANDARD METHODS OF ESTIMATING SIZE, EFFORT AND COST

5.1 Estimating Size:

(a)Wideband Delphi Technique: There are various forms of Delphi technique just as there are various forms of Expert Judgment techniques. The Wideband Delphi Technique is one in which the participants are encouraged to discuss the problem with each other. This techniquerequires participation from a group of participants with a diversity of software related experience:

Step 1: Coordinator presents each expert with the project's specification and an estimation form.
Step 2: Coordinator calls a group meeting in which the experts discuss product issues related to size.
Step 3: Each expert fills out the form anonymously.
Step 4: The coordinator prepares a summary of the estimates on an Iteration Form and returns them to the experts.
Step 5: The coordinator calls a group meeting, primarily to discuss the most widely-varied estimates.
Step 6: The experts review the summary and submit another anonymous estimate on the Iteration Form.

Step 7: Steps 4 through 6 are repeated until a consensus of the lowest and highest possible estimates are reached.

(b)Pert Sizing: This method involves deriving three estimates: an expected size of the product, a lowest possible estimate, and a highest possible estimate. These three estimates are used to arrive at a pert statistical estimate for the expected size of the product and a standard deviation.

For example, for a new communications routine:a = the lowest possible size, e.g. 10 KSLOC, b = the expected size, e.g. 12 KSLOC and c =the highest possible size, e.g. 15 KSLOC

(c)Function Points: "Function point metrics" is a method of estimating size during the requirements phase based on the functionality to be built into the system. Initial application requirements statements are examined to determine the number and complexity of the various inputs, outputs, calculations and databases required. Points based on established values are assigned to each of these counts and then added to arrive at an overall function point rating for the product. The general approach is:

Step 1 Count the number of inputs, outputs, inquiries, master files, and interfaces required.

Step 2 Multiply these counts by the following factors: Inputs (4), Outputs (5), Inquiries (4), Master Files (10), and Interfaces (10).

Step 3 Adjust the total of these products +25 percent, 0, or -25 percent, depending on the estimator's judgment of the program's complexity.

Function points have been found to be helpful in estimating size very early in a software product's development. However, after more is known about the product, function points can be converted to SLOCs which is the software size metric more widely used.

(d)Sizing by Analogy: This approach involves relating the proposed project to previously completed projects of similar application, environment and complexity. The organization's software process database can be used to compare size data from similar projects. The basic steps of sizing by analogy are shown below:

Step 1: Develop a list of functions and the number of lines of code to implement each function,

Step 2: Identify similarities and differences between previously developed database items and those database items to be developed,

Step 3: From the data developed in Steps 1 and 2, select those items which are applicable to serve as a basis for the estimate,

Step 4: Generate a size estimate.

The accuracy of the derived estimate will, obviously, depend on the completeness and the accuracy of the data used from the previous projects.

5.2 Estimating Effort and Cost

(a) Manual Method: The manual estimate of software effort should be based on a combination of the Top-Down and Bottom-Up approach and should be based primarily on the size estimates and schedule requirements. Two or more software engineers with experience with the specific application under development should develop a top down/bottom up estimate based on the size and schedule estimates as follows:

1. Top-Down - A derivation of an estimate of the total effort based on the estimated size of each major function. This can be accomplished manually or with an automated estimation tool. As in estimating size, the effort estimate should also be based on experience with a similar application.

(i) Each estimate should consist of a nominal or most probable estimate plus lowest and highest possible estimates to reflect the uncertainty of the size estimates. The spread between the low and high estimates may be as much as 30-50% in the early phases of a project, e.g., the Concept Phase. Functions for which experience is scarce or for which there is high technical risk should be given an even wider range.

2. Bottom-Up - Derive an estimate for the project by summing up the effort associated with each low level task. This is best accomplished by developing a work breakdown structure (WBS) which includes not only the details of the software architecture hierarchy but details on the software development organization.

(i) The WBS should include activities such as integration, documentation, and software quality assurance and configuration management. While all of these costs may not be known early in the project, at least the identification of these activities will ensure that they are considered.

(ii) Estimate the effort related to the amount of time to prepare for and attend formal project reviews. The cost of reviews is often significantly higher than anticipated.

(iii) The estimates should be reviewed by software engineers who have worked on similar applications.

(b) Software Estimation Tools

A number of automated software estimation tools are available that allow a user to quickly derive effort and schedule estimates based on size estimates and cost driver attributes. Some tools have unique attributes, e.g., requirements volatility, and number of organizations involved.Generally, a tool should be used after an estimate has been manually derived. Automated tools provide a good method to cross-check manually-derived estimates.Manual estimates are usually low because of several reasons. The most prevalent reason is optimism on the part of the software engineer who has forgotten all of the effort that went into design, test, documentation, configuration management, and quality assurance. People do not remember all of the time spent debugging, preparing for project reviews, or how often the requirements were modified. Another common reason for underestimating is unclear or misunderstood requirements.There are various Estimation tools available like Costar, SoftEst, REVIC. The details are:

MODEL	DEVELOPER	PLATFORM	STATUS
REVIC	Major Ray Kile, Air Force Reserves	PC	Public Domain, REVIC Users' Group
SoftEst	Air Force Cost Analysis Agency	PC	Public Domain
COSTAR	SoftStar Inc.	PC	Single user license

Table:1 Overview of Software Estimation Models Available

The use of an automated tool requires the user to consider all of the above plus project attributes such as personnel experience, security complications, requirements volatility and number of sites, hardware constraints, and schedule requirements. However, caution must be exercised when using tools. Because of their ease of use, these tools can give a wide range of estimates by varying just a few parameters. Furthermore, most of the tools currently available are calibrated to Air Force applications and data. Thus, users of these tools should not rely solely on a tool to develop a credible estimate.

6. CONCLUSION

The results of the review have identified several research gaps. This paper has presented the body of research on effort and cost estimation models for web applications by examining techniques, were used to build models. After the conduction of many studies on effort estimation, till now there are no proven methods for estimating the effort and cost of web application. All the techniques are taken from traditional software engineering are customized only. No significantly new techniques have been proposed. More new size metrics are being developed and customized from existing methods for e.g. Web Objects, more or less variations of the Function Points for which the reason not always apparent. Future work includes the extension of this review by including other sources.In order to improve effort estimation accuracy, a more precise terminology for software effort estimation is needed. We provide two simple guidelines for this purpose: (1) Do not mix estimation of most likely effort with planning, budgeting or pricing, and (2) When assessing estimation accuracy, ensure that the estimate and the actual effort are comparable. Although these guidelines are not innovative and might seem obvious, they are nevertheless worth stressing. As this review points out, they are frequently violated.

7. REFERENCES

[1] B. Boehm, R. Fairley, Software estimation perspectives, IEEE Software 17 (6) (2000) 22–26.

[2] E.J. Barry, T. Mukhopadhyay, S.A. Slaughter, Software project duration and effort: an empirical study, Information Technology and Management 3 (1–2) (2002) 113–136.

[3] Heemstra, F.J., Software Cost Estimation. Information and Software Technology, 1992. 34(10): p. 627-639.

[4] Jorgensen, M. and D.I.K. Sjoberg, The impact of customer expectation on software development effort estimates. To appear in Journal of Project Management, 2004. 22(4).

[5] Kumari, Sweta&Pushkar, Shashank. Performance Analysis of the Software Cost Estimation Methods: A Review. International Journal of Advanced Research in Computer Science and Software Engineering,3(7), July - 2013, p. 229-238.

[6] Kishore, Swapna&Naik, Rajesh (5 June 2001). Software Requirements and Estimation: McGraw Hill Education (India) Private Limited.

[7] Litoriya, Ratnesh& Kothari, Abhay. An Efficient Approach for Agile Web Based Project Estimation: Agile MOW. Journal of Software Engineering and Applications, 2013, 6, 297-303.

[8] Niazi, Adnan & Dai, Jian S. Product Cost Estimation: Technique Classification and Methodology Review, Journal of Manufacturing Science and Engineering MAY 2006, 128: 563-575.

[9] Molokken Kjetil&Jorgensen Magne, “A Review of Surveys on Software Effort Estimation”, 2003IEEE.

[10] Software Estimation Process, August 31, 1999, Version 2.2.