

Estimation of Cost and Efforts in Agile Methodologies - A Review

Venkata. Krishna Mohan.CH
Ph.D., Scholar, CSE- Dept.
GITAM University, Rishikonda, Vizag, India

Dr.G.V.S Raj kumar
Associate-Professor, CSE- Dept.
GITMA University, Rishikonda, Vizag, India

Abstract
The concept of Traditional software Development slowly started converting into a new methodology called as AGILE Methodology. Where in agile methodology the aim is to satisfy the customer, faster development times with less defects. Where as in traditional software development the effort and cost estimation methods are more when compared with Agile Methodology even though agile process is itself a software development process it has its own limitations generally used techniques or methods. In this paper we explain all the existing techniques which we discuss along with newly introduced methods.
Keywords - Agile methodology, FPA, COCOMO, EJ

1. Introduction

Software Engineering is one of the main objectives for developing good software applications all these Applications are developed using different types of process, models & techniques where we can ultimately give the customer better software by meeting all his requirements. There are number of methods for the cost estimation in the conventional process like SDLC where in this process the software development is done in a structured format like Requirements gathering, Analysis, design, code, testing and maintenance. Here the cost and effort estimation is done by using different methods where we have Algorithmic and non-Algorithmic methods. The main principle for cost estimation in Agile Methodology is to welcome the changing requirements estimate the cost and effort perfectly.

2. Techniques for Estimation

There are so many techniques for estimation of cost and effort in the software industry. These techniques were categorized into two Algorithmic Non-Algorithmic models.

2.1 Algorithmic Model

Cost estimation by using the algorithmic cost model is based on mathematical formulas. In this model to estimate the cost consider size of project, type of software, software team, and software attributes etc. There are different types of models which are used like Function points (FP) based model, Putnam Model, COCOMO Model all these methods uses mathematical formulae's

2.1.1 COCOMO model (Constructive Cost Model)

This is widely used cost estimation model for all the software projects. COCOMO 81 Model is proposed by Barry Boehm in 1981. Nowadays the COCOMO-II model is used in which effort estimation is based on Person-Month (PM) in software projects.

COCOMO model uses the function point or line of code as the size metrics and composes of 5 scale factor and 17 multipliers. The 5 scale factor are rated on a six-point scale from very low to extra high (5 to 0) [1]. After assigning the rating values add them, divide them by 100 and add the result to 1.01 to get the exponent that should be used [1].

This model uses the function point or line of code as the size metrics and composes of 5 scale factor and 17 multipliers. The 5 scale factors are rated on a six-point scale from very low to extra high (5 to 0) [1].After assigning the rating values add them, divide them by 100 and add the result to 1.01 to get the exponent that should be used [1].

Scale factor

Precedentendness	This Reflects the previous experience of the similar project. No previous experience means very low, organization is familiar with this type of project means very high.
Development Flexibility	Reflects the degree of flexibility in the development process. Rating very low means prescribed process is used. and client sets the only general goals in case of rating extra high.
Architecture /risk resolution	Reflects the extent of risk analysis carried out. Little analysis means very low and a complete and thorough risk analysis means extra-high
Team cohesion	Reflects how well known about team members & work together. Rating very low as very difficult interactions and rating extra-high when effective team and no any communication problems
Process Maturity	Rated nominal for some process control in place.

- Precedentedness: Rated low for new project for organization.(Rating : 4)
- Development Flexibility: Rated very high when no client involvement. (Rating: 1)
- Architecture/risk Resolution: Rated very low while no risk analysis carried out (Rating: 1)
- Team Cohesion: New team so no information. Hence rated as nominal. (Rating: 3)
- Process Maturity: Rated nominal for some process control in place.(Rating:3)
- The sum of above rating value is 16. So calculate the exponent by adding 0.16 to 1.01, getting a value of 1.17

The basic COCOMO equations take the form

Effort Applied (E) = $a_b(KLOC)^{b}_{b}$ **[man-months]**

Development Time (D) = $c_b(\text{Effort Applied})^{d}_{b}$ **[months]**

People required (P) = Effort Applied / Development Time **[count]**

The cost drivers that are used to adjust the initial estimates and create multiplier in the post-architecture model fall into four categories like :

Product Attributes: these attributes are concerned with required features for developing software product. Computer Attributes: these are the restriction enforced on the software by the hardware platform.Personal Attributes: these are multipliers that take the experience and capabilities of the people working on the project into account.

Project Attributes: these attributes are related to specific features of software development project

2.1.2 Putnam Model:

The Putnam model is an empirical software effort estimation model.. Putnam used his observations about productivity levels to derive the software equation:

Technical constant C = Size * B1/3 * T4/3

Total Person Months B = 1/T4 *(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 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 is very sensitive to the development time: decreasing the development time can greatly increase the person-months needed for development [2]. 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. There is often great uncertainty in the software size. It may result in the inaccuracy of cost estimation.

SLIM (Software Life Cycle Management) is a tool that acts according to the Putnam's model.

2.1.3. Function-Point based Model:

Function point metrics, developed by Alan Albrecht of IBM, were first published in 1979 and In 1984, the International Function Point Users Group (IFPUG) was set up to clarify the rules, set standards, and promote their use and evolution.

In this models the estimation can be done by using the following five factors

User Inputs, User Outputs, Logic Files, Inquiries, Interfaces

Function point metrics provide a standardized method for measuring the various functions of a software application it measure functionality from the users point of view, that is, on the basis of what the user requests and receives in return

An Organization can apply function point analysis as [3]:

A tool to determine the size of a purchased application package by counting all the functions included in the package.

A tool to help users determine the benefit of an application package to their organization by counting functions

that specifically match their requirements.
A tool to measure the units of a software product to support quality and productivity analysis.
A vehicle to estimate cost and resources required for software development and maintenance A normalization factor for software comparison
On the whole:
FP = UFP * VAF
The constant values in the equation and the weighting factors are determined empirically.

2.1.4 Parkinson's Law:

Parkinson's Law states that work expands to fill the time allotted for its completion. Time management is all psychological. We naturally pace ourselves to finish a project in the nick of time. The same task can take one hour or one week depending on how much time we give ourselves to complete it. Ever pull off a big presentation where your only prep was during your commute on the way over? The law is true!

Track your time:

Time tracking encourages you to be hyper-aware of a project's progress. Using data from time tracking, allot 25% less project time to your next project to employ Parkinson's Law for maximum efficiency.My personal favorite! Parkinson's Law tells us we can accomplish things in much less time than we think, which is excellent news because our brains work best if we take a small break after 90 minutes of work.

2.1.5 Price–To-Win estimating:

The price-to-win technique has won a large number of software contracts for a large number of software companies. Almost all of them are out of business today[13]. The inevitable result is that the money or schedule runs out before the job is done, everybody gets mad at each other, a lot of compromises are made about the software to be delivered, and a lot of programmers work long hours just trying to keep the Job from becoming a complete disaster. The main reason that the price-to-win technique continues to be used is because the technology of software cost estimation has not provided powerful enough techniques to enable software customers or software developers to convincingly differentiate between a legitimate estimate and a price-to-win estimate. One of the primary objectives of the COCOMO model is to begin to provide a way for people to make the differentiations. It is possible to make the COCOMO model give you a lower cost estimate but only by changing some objectively defined cost driver rating, whose validity can be checked by someone other than the estimator.

2.2 Non-Algorithmic Model:

In non-algorithmic model, the estimation can be done by using the previous projects experience which is similar to the under estimate project.

2.2.1 Expert Judgment (EJ):

EJ is used extensively during the generation of cost estimates. Cost estimators have to make numerous assumptions and judgments about what they think a new product will cost. However, the use of EJ is often is not well accepted or understood by non-cost estimators within a concurrent engineering environment. Computerized cost models, in many ways, have reduced the need for EJ but by no means have they, or can they, replace it. Very little research tackles the issues of capturing and integrating EJ and rationale into the cost process.
EJ is examined in terms of what thought processes are used when a judgment is made.

2.2.2 Estimation based Analogy costing:

Analogy costing method requires one or more completed projects that are similar to the new project and derives the estimation through reasoning by analogy using the actual costs of previous projects. Estimation by analogy can be done either at the total project level or at subsystem level. The total project level has the advantage that all cost components of the system will be considered while the subsystem level has the advantage of providing a more detailed assessment of the similarities and differences between the new project and the completed projects. The strength of this method is that the estimate is based on actual project experience. However, it is not clear to what extend the previous project is actually representative of the constraints, environment and functions to be performed by the new system. Positive results and a definition of project similarity in term of features were reported in.

Advantages:

- Depends on the values and data of previous projects.
- Estimators experience can be used which helps in arriving at a better cost estimate.
- We get to know the minute distinction between the previous completed projects and our current projects and this in a way also helps in knowing their impacts.

Disadvantages:

Using this method requires estimators to find out the attributes through which a project can be described best also we need to provide weightage to these to get a better analogy. We cannot use this technique for every project.

2.2.3 Top-down estimation:
In this technique we derive total cost from global properties using either of algorithmic or non-algorithmic technique. Then this cost is spitted to various components of the system. Top-down Estimation is more beneficial in the early stages of software development because detailed information is not available during this stage [9],[10]. Putnam's Model is an example of this technique.

2.2.4 Bottom-up estimation:
Bottom-up estimation is opposite of Top-down estimation method. In this method we derive cost of each software component and then the result is combined to achieve the overall cost of the software. Goal is to derive system estimate from the accumulated estimate of the small component.

2.2.5 Planning Poker:
Planning Poker is an agile estimating and planning technique that is consensus based. To start a poker planning session, the product owner or customer reads an agile user story or describes a feature to the estimators. Each estimator is holding a deck of Planning Poker cards with values like 0, 1, 2, 3, 5, 8, 13, 20, 40 and 100, which is the sequence werecommend. The values represent the number of story points, ideal days, or other units in which the team estimates [12].

The estimators discuss the feature, asking questions of the product owner as needed. When the feature has been fully discussed, each estimator privately selects one card to represent his or her estimate. All cards are then revealed at the same time. If all estimators selected the same value, that becomes the estimate [12]. If not, the estimators discuss their estimates. The high and low estimators should especially share their reasons. After further discussion, each estimator reselects an estimate card, and all cards are again revealed at the same time. The poker planning process is repeated until consensus is achieved or until the estimators decide that agile estimating and planning of a particular item needs to be deferred until additional information can be acquired.

2.2.6 Throwing fingers:
This is a simple variation on the poker theme. Instead of flipping cards or revealing Fibonacci numbers on a phone app, each person just raises their hand with fingers raised, from one to five[13]. This won't give you Fibonacci numbers but that's ok. If you really want to convert them, you can (e.g. you can make four fingers into an estimate of 8 points, and five fingers an estimate of 13 points).

This isn't too different from poker but it's a slight change and is a bit faster because people don't waste time playing with or searching through their cards. (I find the damn cards always go missing as well, so it solves that problem too!).If you want to take it a bit further, you can have half-points by people holding up half a finger.

2.2.7 T-shirt Sizing:
Instead of numbers (Fibonacci or otherwise), you can do T-shirt sizes. Small, medium, large, xtra-large[13]. This gives you a smaller range of possible estimates, which means you will get fewer disagreements and less variation. You might think that the estimates will be less accurate, but I don't think they will be. I find rough t-shirt sizes to be good enough. You will need some baselines for these sizes; just use previous stories or features. (I actually like to do t-shirt size feature estimates and no story estimation, more on that later). You could also just use a smaller set of numbers (1, 5, 8, 20) or something as proxies for t-shirt sizes. But I like the fact that numbers aren't used here. It makes it clearer that these are not measurements and are not accurate.

2.2.8 Affinity Mapping:
You might be familiar with Affinity Mapping from Sprint Retrospectives. It is a technique for grouping similar items together. Start by creating a series of "tags" or "buckets" on the table: these could be Fibonacci numbers, or t-shirt sizes, or categories, or anything [13]. Then you lay the stories down on the table as cards or something similar. Next, the team collectively moves the cards into the buckets to represent that as an estimate.

Next, each person is randomly assigned a set of stories to estimate (maybe deal the stories out as if they were a deck of cards)[13]. Then, taking turns, each person silently estimates by placing a card on one of the buckets alternatively, a person can move a card from one bucket to another, if they strongly feel it is an incorrect estimate. Keep doing this until all the cards are estimated. If a card is moved twice, take it off the table – it will need a separate discussion after the meeting since there is wide disagreement on its estimate.

The advantage of this technique is that a team can estimate a lot of stories in a very short amount of time. I would use it if you are asked to estimate 100 stories or similar (though I think if you are, that is a sign of a bigger problem – you should only estimate one or two sprints' worth of stories in advance). The disadvantage of it is you miss out on what is often the valuable part: the discussion around the stories.

3. CONCLUSION
An attempt is made in this paper to bring all the methods which are used for calculating effort and cost estimation in agile methodology so as to anticipate the new methods in future. Different methods used by the industry were examined and it was noted that the accuracy feature is not much to be seen in any of them. Though agility give a chance for the team members to discuss and execute new methods.

This study aims to go much further and bring up new methods in algorithmic and non-algorithmic category to deduce how much more methods with good accuracy.

3.1 Scope for Future Study

The main aim of this research work was to develop Different types of methods for estimating effort and cost using latest methods in the Agile Methodology. A number of areas of future research have arisen from the experimental work and the most significant of them are outlined below. Further studies can be extended to the development of effort calculation and cost estimations of the projects. If a suitable method was found to calculate the effort estimation the possibility for developing with High- end was used.

The efficiency of Throwing finger can be increased appreciably by raising their hands but Fibonacci series cannot be identified instead an alternate way was we have make the four fingers into 8 divisions. Although there are many methods based on throwing fingers in the literature, the author believes that they are very difficult to develop. Hence developing different types of methodologies gives great importance for Agile Estimations.

References

[1]. Software Engineering 8th Edition – Lan Sommerville
[2]. R.S. Pressman, Software Engineering: A Practitioner's Approach, McGraw-Hill, 2000
[3]. IFPUG: Function Point Counting Practices Manual, Release 4.1.1
[4]. G. Eason, B. Noble, and I.N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529-551, April 1955.
[5]. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.
[6] I.S. Jacobs and C.P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G.T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271-350.
[7]. K. Elissa, “Title of paper if known,” unpublished.
[8]. R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
[9] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[10]. M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989
[11] https://www.mountaingoatsoftware.com/blog/why-i-dont-use-story-ints-for-sprint-planning
[12] https://www.extremeuncertainty.com/alternatives-planning-poker
[13]. http://sunset.usc.edu/classes/cs510_2012/EPs_2012/EP25_Chapter_22.pdf
[14]. International Journal of Electronics Communication and Computer Engineering Volume 6, Issue 6, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209