Integrated Intelligent Research (IIR)
International Journal of Computing Algorithm
Volume: 04 Issue: 01 June 2015 Pages:22-25
ISSN: 2278-2397

Analysis of Effort Estimation Model in Traditional and Agile

(Using Metrics to Improve Agile Methodology)

R.Manjula[1], R.Thirumalai Selvi[2]
[1]Research Scholar, Department of Computer Science, Bharathiar University,India,
[2]Assistant Professor, Department of Computer Science,Govt. Arts College, Nandanam, Chennai, India
Email: sarasselvi@gmail.com, cpmanjula76@yahoo.com

Abstract-Agile software development has been gaining popularity and replacing the traditional methods of developing software. However, estimating the size and effort in Agile software development still remains a challenge. Measurement practices in agile methods are more important than traditional methods, because lack of appropriate an effective measurement practices will increase the risk of project. This paper discuss about traditional and agile effort estimation model, and analysis done on how the metrics are used in estimation process. The paper also suggeststo use object point and use case point to improve accuracy of effort in agile software development.

Keywords- Effort Estimation, Metrics,Function point, Story point, Object point

I. INTRODUCTION

Effort estimation process in any software project is essential and it is very critical component. In software engineering effort is used to denote measure of use of workforce and defined as total time that takes members of development team to perform a given task. To estimate effort some of the conventional metrics are used in traditional and agile methodologies. A metric is a standard for measuring or evaluating something. A measure is a quantity, a proportion or a qualitative comparison of some kind. Good metrics should enable the development of models that are efficient of predicting process or product spectrum. The optimal metric should be simple, objective, easily obtainable, valid and robust. The first method uses self-learning algorithm to obtain decision-making tree

II. TRADITIONAL EFFORT ESTIMATION MODEL

The accuracy of effort estimation is as current issue for researchers today as it was 25 years ago when it was launched by Brooks[20] in his work “The Mythical Man Month”. Even today these estimations are mainly unreliable, with no proof of significant progress that has been made in their improvement, despite considerable funds and activities that have been invested to that purpose. Different authors classify effort estimation methods differently. They are empirical parametric estimation models; Empirical non-parametric estimation models; expert estimation; analogue estimation models; downward estimation; upward estimation. Comparisons are made between the suggested project and similar projects for which data in respect of cost, time and effort are known

A. Empirical Parametric Estimation Models

These models rely on the experience gained on previous software projects in the sense that they connect size and effort value by means of the explicit function forms, by applying regression analysis method. In doing so, most widely used are linear and exponential dependence. Good sides of these models are: objectivity, formalism, efficiency and the fact that they have been based on experience drawn from engineering practice. Its bad sides are: necessity for calibration before application in the concrete environment, subjectivity of input values, that they have founding in the past instead of future. Following are some known empirical parametric model.

Effort= aLOCb(hitherto form)

Effort = a + b FP [19]

LOC = $C_K K^{1/3} t_d^{4/3}$

B. Empirical Non-parametric Estimation Model

It is characteristic for empirical Non-parametric models that they use data on projects realized earlier. However the estimation is not done by applying given mathematic formula but by means of other approaches. Out of these models mentioned herein will be optimized set reduction technique (OSR), decision-making trunk and neural networks. [16].OSR selectssubset of projects based on which it estimates productivity of the new project. Productivity is defined as effort in man-months divided by the number of code lines. Projects grouped in optimum subset should have similar cost factors, like the new project. The other method relies on neural networks. The neural networks model shows smaller mean error than the decision-making tree model. However, training of neural networks is often strenuous. Accuracy of these models is similar to that of the OSR. In order that these models can be applied in practice, calibration should be done on a great number of data, since these models have a great number of independently variable values.

C. Expert Estimates

These models are based on consultation of one or more people considered to be experts in software development. For coordination of differing opinions among estimators, often used in one of formal techniques like Delphi. There exist a number of Delphi technique forms. Widebrand Delphi ( ) encourages those involved to discuss the problem among themselves.

D. AnalogueEstimation Models

These models are require as much data as possible concerning implemented projects.Two best known analogue models are

ESTOR [18] and ANGEL [15]. ESTOR is case-based reasoning model. This case-base reasoning form consists of 5 basic processes:

- Target case specification
- Search for adequate case to serve as original analogy
- Transfer solution from the source to the target case
- Adjust the initial solutions based on the differences found

ANGEL has been based on the generalisation of the approach [15]. According to this approach projects are presented by means of function point components. Analogue projects are neighbours of the new project and they are reached by calculating vector distance from the new project. Effort concerning the new project is estimated on the basis of the mean effort value in respect of the neighbouringprojects.

As one can see, ESTO and ANGEL have many common features in both cases, projects are represented by means of easily obtainable metrics and analogue project in both cases are identified by calculating vector distance. ESTOR uses only on analogue to determine the estimate, while ANGEL estimate can be based on several analogues.The advantage of analogue estimation models over the empirical parametric models is in their successful application in the cases where valid statistic data dependence cannot be determined. Schofield and Kitchenham[15] give an example of a set of eight projects for which ANGEL gives estimate with mean relative 60% error while regression linear model gives 22.6% error.

E. Downward Estimates
Estimation of total effort is made on the basis of the software product global characteristic [17]. This estimate is usually based on previous projects and takes into account effort in respect of all function projects. Total effort is then distributed as per components.

F. Upward Estimates
In this, case estimation is mad in respect of every project component individually and total effort is calculated as addition of individual efforts [17]. Quite often such approach leaves many global effort components overlooks such as those linked with integration, system testing and project management.

III. EFFORT ESTIMATION IN TRADITIONAL USING METRICS

Software effort can be estimated from size-oriented metrics, function oriented metrics, object point, test point and Use Case Point(UCP).[5][6][7]

A. Size oriented metrics
Source line of Code (SLOC) is software metric used to measure the size of software program by counting the number of lines in the text of the programs source code. This metric doesnot count blank lines, comment lines and library. SLOC measures are programming language dependent. They cannot easily accommodate non procedural languages. SLOC also can be used to measure other such as errors/KLOC, defects/KLOC, pages of documentation/KLOC, cost/KLOC.

B. Function oriented metrics:
Function Point (FP): FP defined by Allan Albrecht at IBM in 1979, is a unit of measurement to express the amount software functionality [5]. Function point analysis (FPA)is the method of measuring the size of software. The advantage is that it can avoid source code error when selecting different programming languages. FP is programming language independent, making ideal for applications using conventional and nonprocedural languages. It is based on data that are more likely to be known early in the evolution of project.
FP = UFP * VAF

The UFP is computed using predefined weights of each function type.Value Adjustment Factor (VAF) indicates the general functionality provided to the user for the application.

C. O*bjectpoint*
Object points are an alternative function related measure the function points when 4 GLS or similar languages are used for development. Object points are not the same as object classes.The number of object points in a program is considered a weighted estimate of 3 elements.Object points are easy to estimate. It is simply concerned with screen,reports and 3GL Modules.

D. T*est point*
Test point used for test point analysis, to estimate test effort for system and acceptance test. It covers black-box testing. Test point can be in two categories: dynamic test points and static test point.
a) Dynamic test points – Dynamic test points are calculated as the sum of the TP assigned to all functions. TP are calculated for each individual function using the amount of FP, function dependent factors (user-importance, user-intensity, complexity, uniformity and interfacing) as well as quality requirements.
b) Static test points – are a result of determining the number of test points required to test static measurable characteristics.
c) TTP = sum of dynamic + static TP(Total amount of TP)
d) Primary Test Hours: represent the volume of work required for the primary testing activities like preparation, specification, execution and completion test phases. Primary test hours can be calculated using the following formula.

$P_{\text{(TP)*environmental factors*productivity factor}}$

E. *Use case point*
Use case point is estimated from use cases. Use case is a system behavior under various conditions, based on request from a stakeholder. Use case point is used to map use cases to test cases. Use case serves as input for a specific test case. Required test effort for a project is calculated from use case point. Use case point is determined from UAW,UUCW,UUCP,TCF and AUCP

IV. EFFORT ESTIMATION IN AGILE USING METRICS

Agile methodology takes a considerably different approach to determining a team member’s capacity. First of all, it assigns work to an entire team, not an individual. Second, it refuses to quantify work in terms of time because this would undermine the self-organization central to the success of methodology. It does not prescribe a single way for teams to estimate their work. Teamuses a more abstracted metric to quantify effort. Normally effort estimation takes place at the beginning of new iteration during release planning. In agile effort can be estimated from following metrics:[1][3][4]

- story size metrics
- story complexity metrics
- friction factor metric
- variable factor metric
- Completion time (T) is calculated as

$$T=\sum_{i=1}^{n}((ES)/(Vi)^{D})*\left(\frac{1}{WD}\right) Months$$

Where WD is No of Work Days in a Month and ES is the User Story Effort, Calculated as
ES=Complexity x Size
Vi is the Initial or Raw Velocity, calculated as
Vi = Units of Effort Completed / Sprint Time.
Sprint Time is the No of Days in sprint.

Schedule and effort can be estimated using story points. Story points are evaluated from user stories, determining iteration length.

TABLE 1 Comparisionof Agile and Traditional Metrics

Agile	Traditional
Backlog	Backlog Is Not Desirable
Complexity Point	Function Point
Epics	Mission Threads
Planning Poker	Wide-Band Delphi
Sprint	Work Package
Velocity	Schedule Performance Index
Burn-Up/Down Chart	Barchart/Ganttchart)
Earn Value Managementn	Earned Business Value
Use Case Point Not Included	Use Case Point Included
Object Point Not Included	Object Point Included

Table 1 provides information about comparison of agile metrics with traditional metrics. Traditional Method have many metrics to predict effort at initial stage of the development. But, Agile having a few metrics in effort estimation. So, if Use Case Point and Object Point metrics are used, then prediction of effort estimation can be improved in agile methodologies. Traditional methodologies use object point and use case point for their initial estimates in software development. Reliable initial estimates are quite difficult to obtain due to lack of detailed information at an early stage of the development. To overcome this problem usecase point and object point methods are used by many software practitioners to estimate effort. In Agile predicting effort at an initial stage is a challenging one. To improve the accuracy of effort product backlog can be combined with UCP and object point. A different cases study shows that UCP can support Agile environment and fulfills object oriented development without major adjustments. Along with UCP, object point can be added to the product backlog to estimate no. of screen, reports and database for every iteration. Object point can be calculated by the following steps.

- Assess object count, number of screens, report and 3GL components.
- Classify object: Simple, medium and difficult depending on the values of characteristic dimensions.
- Weight the number in each cell using the following scheme. The weights reflect the relative effort required to implement an instance of the at complexity level.
- Determine object points: add all the weighted object instance to get one number the object point count.
- Estimate percentage of reuse you expect to be achieved in this project. Compute new object points to be developed as NOP = (object Point) * (100 - % reuse)/100

Where %reuse is the percentage of screens, reports and 3GL modules reused from previous applications.

V. CONCLUSION

Efforts are estimated by using metrics. Numerous effort models and effort estimation methods are developed for the traditional software development, whereas Agile has very limited number of effort estimation model. To improve the agile methodology, agile need some additional metrics. This paper focuses on different estimation model in traditional and agile. It also discusses about what are the different metrics used in effort estimation model. In future, this work is continued by investigating how UCP and Object Point performs on different type of projects, in particular regarding size and complexity of the software project implemented in Agile Environment.

REFERENCES

[1] Amrita Raj mukker,Dr.LatikaSingh(2014) Systematic Review of Metrics in Software Agile projects.COMPUSOFT, IJACT Vol. III, Issue-II
[2] Vajargah B. and Jahanbin A., Approximation theory of matrices based on its low ranking and stochastic computation, Advances in Computer Science and its Applications (ACSA) Vol. 2, No. 1, 2012; pp 270-280
[3] Evitacoelho,Anirban Basu,2012,Effort Estimation in Agile Software Development Using Story Points.IJAIS.
[4] Zlauddin, Shahid Kamal Tipu,Shahrukh Zia,2012 An Effort Estimation Model for Agile Software Development,ACSA.
[5] JairusHihn, Hamid Habib-agahi, “Cost Estimation of Software Intensive Projects: A Survey of Current Practices”, IEEE, 2011.
[6] Peter Hill, 2010 “Practical Software Project Estimation – A Toolkit for Estimating Software Development Effort and Duration “McGraw Hill Education
[7] KhaledHamdan, Hazem El KhatibShuaib,” Practical Software Project Total Cost Estiamtion Methods”, MCIT 10, IEEE, 2010
[8] Barbara Kitchenham and Pearl Brereton,2010. Problems Adopting Metrics from other Disciplines, Workshops on Emerging Trends in Software Metrics, May pp 1-7.
[9] IFPUG-FSM Method: ISO/TEC 20926:2009, Software and systems engineering – Software measurement – IFPUG functional size measurement method.
[10] SCHMIETENDORF A., KUNZ M., DUMKE R. (2008) Effort estimation for AgileSoftware Development projects, Proceedings 5th Software Measurement European Forum, Milan.
[11] Chetan Nagar, Anurag Dixit, “Software efforts and cost estimation with systematic approach”, IJETCIS, ISSN: 2079-8407. Vol. 2, No. 7, Jul 2011. [20]. Pressman, Roger S., “Software Engineering: A Practioners Approach”, 6thEdn., McGraw-Hill New York, USA., ISBN: 13: 9780073019338, 2005.
[12] Lindstrom, L & Jeffries, R., 2004 Extreme Programming and Agile Software Development Methodologies Information Systems Management, 21, 41-52.
[13] Putnam, Lawrence H. ,2003, Ware Myers Five Core Metrics: The Intelligence Behind Successful Software Management, Dorset House Publishing, p86-97.
[14] Suresh Nageswaran, “Test Effort Estimation Using Use Case Points”, Quality Week, San Francisco California, USA, June2001.
[15] ShepperdM.J., Schofield, C.,andKitchenham, B. 1996.,Effort Estimation using analogy, Proceedings of the 18th International Conference on Software Engineering,Berlin.

[16] Briand L. C., Emam K. El, Surmann D., Wiezczork I., and Maxwell K. D. 1999, An Assessment and comparison of common software cost estimation modeling techniques, in Proceedings of the International Conference on Software Engineering, pp. 313-323.
[17] Shooman M.L.,1996 Avionics Software Problem Occurrence rates, issre, The Seventh International Symbosiums on Software Reliability Engineering, pp. 55-68.
[18] Mukhopadhyay, T. Vincinanza, S., and Prietula, M.J. 1992. Estimating the Feasibility of a Case-Based Reasoning Model for Software Effort Estimation, MIS Quarterly, vol 16 no. 2, pp. 155-171.
[19] Albrecht, A.J., and Gaffney, J.E. 1983, Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation, IEEE Transaction of Soft. Engineering, vol 9 no 6, pp 639-648.
[20] Brooks, F.P. 1975, The Mythical Man Month, Addison-Wesley, Reading.