Software Project Duration Estimation Using Metrix Model

Marius VETRICI
Economy Informatics Department, Academy of Economic Studies, Bucharest, România
mariusvetrici@softmentor.ro

This paper analyzes the existing types of duration estimation models for software projects and advances a new, user-friendly model for project duration estimation. The Metrix model is a stochastic model for software project duration estimation using Monte Carlo simulation over an activity graph. The first advantage of using the Metrix model is that it produces a probability distribution of the software project duration and not a single estimate for duration. Further to this approach, we diminish the project uncertainty by giving the manager better control over the project duration and the associated probability of a certain duration outcome. The second advantage of the Metrix model is that for Monte Carlo simulation it relies upon the historic duration estimation of the team members and not on probability distribution functions which are rather difficult to come with.
JEL classification: L86 computer software, O22 project analysis.
***Keywords**: Software project duration estimation, Monte Carlo simulation, Metrix model.*

1 Introduction

The grand majority of software development projects are known to be late and over the budget. Most of them hit schedule and budget overruns of 25% to 100% and sometimes even more [1], [2], [3], [4].
The prerequisite for defining an accurate project delivery date is a precise estimation of the project duration. Existing models are rather imprecise because the forecasted value is to a certain extent distant from the real one. The large discrepancies between the estimated duration and the actual schedule of an ongoing project ended it in order to prevent further damages and losses. The [2] research reveals that only one project in three is considered successful, while one project in five is a total disaster. Given this, it is imperative to look for new software project duration forecast models that will be able to outspring results that are more realistic.
The aim of this research is to bridge the gap between the forecasted software project duration and the actual project duration.
The necessity of this research comes hand in hand with the explosive evolution of the software development projects from the last decades. The private sector has generated a myriad of empirical techniques for project duration estimation, which unfortunately did not benefit from much rigorous academic support. The nowadays software development companies have developed advanced systems for project planning and progress tracking. Using these tools, companies are able to record high volumes of data sets, such as initial task duration estimation, the daily progress of a task, the number of bugs found and the daily rate of fixed bugs. With all this data at hand, the companies were faced with the situation of being overflown with data, while lacking pertinent information and precise duration estimations regarding ongoing projects.
This paper elaborates the Metrix model, a new user-friendly duration estimation model for software projects that is able to forecast the delivery date of a project function of the historical estimation errors of the same team.

2. The classification of duration estimation models

The range of duration estimation techniques and methods significantly broadened its coverage in the last years so that now we have sophisticated mathematical and statistical models and even expert system based estimation models.
Figure 1 depicts the classification of existing models [5].
Expertise-based methods are based on the subjective judgment of a human expert or a

group of experts and are the most widely used methods for project duration estimation [6]. Unlike rigorous estimation methods, these methods rely on the personal intuition and on the experience gained by the human expert in question [7]. For example, according to Delphi method, a panel of experts is required to make an estimation regarding a project. After the first step, the estimations are debated and then the experts go to a second stage of estimations. After each estimation stage, some elements and some details will be left out, while others will be greatly emphasized. The process is iteratively repeated until a common agreed duration is reached [8].

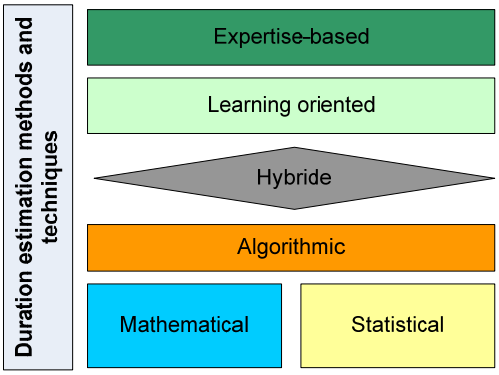

Fig.1. The classification of duration estimation models and techniques for software projects

Learning-oriented techniques instead try to identify a similar software development project and infer the duration estimation out of the past experiences and the differences between the old and the new project [9]. The advantage of this class of techniques over the expertise-based ones is that in this case estimations are grounded on real life facts and on palpable examples and not on the general experience of the experts. The disadvantage of these techniques is the fact that it is not very obvious how the two projects should be compared, what are the key variables that should be tracked and what are the issues that should be left out. The identification of the key variables is a tedious, time-consuming task because of the very particular nature of software projects.

The algorithmic methods use iterative approaches based on mathematical formulae. They take as input data the size of the software project (counted in function points or lines of source code) and parameters like hardware and software development platform, team experience, manager's experience and the development methodology employed. The data is inputted and the algorithm produces an estimation of the duration together with an index of the estimation accuracy. The algorithmic methods are iteratively run several times in order to refine the input parameters' values and to enhance the estimation accuracy. The limitation of this class of methods comes out when the algorithm is fed with uncalibrated or not validated data. Most of the algorithmic methods offer estimations for duration, for effort and even for the total cost of the projects. Among them are COCOMO and COCOMO 2.0 [10], SLIM, Neural Networks, Critical Path Method, Critical Chain Method, PERT.

Mathematical-statistical models are particularly useful when confronted with large sets of historical data available for analysis. Such models include the linear regression and the multiple regression. [11] advances a new method for project duration estimation that takes into account the time consumed with inter task communication.

Hybrid methods have been created in order to overcome the increasing uncertainty and complexity of software projects. This class of methods combines algorithmic, statistical, mathematical and expertise-based methods into a single unitary method.

Table 1 generalizes existing duration estimation methods for software projects together with their pros and cons.

3. The Metrix model for software project duration estimation

In order to overcome the disadvantages of the existing duration estimation models described in section 2 of this paper we advance a new, hybrid type model for estimating the duration of software projects. This is a stochastic model that addresses the project duration uncertainty by running Monte Carlo simulations over the activity graph. The advantage of this approach is that the model pro-

duces an interval for the possible project durations and a probability distribution. Thus, one is able to know the possible project durations together with the probability that certain duration will materialize.

Table 1. Duration estimation methods for software projects

Name	Pros	Cons
Expertise-based methods	These are the most flexible methods that can be easily adapted from project to project in order to enhance the quality of duration estimations.	Are too subjective. Depend on the experience of the experts in question.
Learning-oriented techniques	Are based on real life examples that have been previously executed.	The necessity to identify the key-variables is a daunting, time-consuming task because of the specifics of every project.
Algorithmic methods	These methods are able to refine their estimates on subsequent iterative algorithm execution. Can be easily adapted to the variations of the input values.	The estimations can have a very low quality when the input data has not been properly validated and calibrated.
Mathematical-statistical models	Are easy to develop and have a very good academic background.	Need a large set of historical data.
Hybrid methods	Are the most efficient by combining key aspects from all other methods.	Are immature, undeveloped and lack solid formalization.

The components of Metrix model are described in Figure 3:

a) an expertise-based component: task duration estimation is performed by the software developer himself who will be responsible with the task completion;

b) a learning oriented component: individual task duration estimations will be automatically adjusted with historical individual estimation errors, this way enhancing the accuracy of estimations;

c) a mathematical-statistical component: the Monte Carlo simulation is used in order to produce a distribution of probability for the possible project durations;

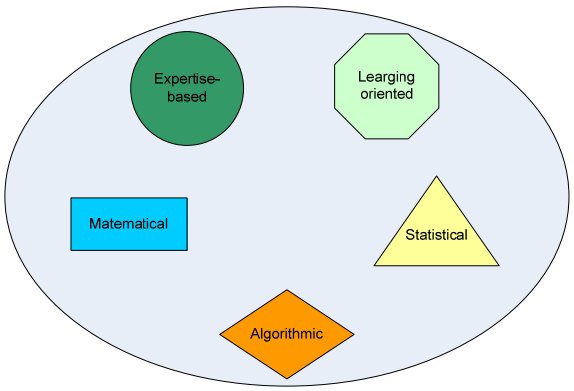

Fig.2. The components of the Metrix Model for software project duration estimation

d) an algorithmic component: the model has input data, it iteratively executes several steps and ramifications and in outputs clearly defined results.

From the approach used viewpoint, this is a bottom-up model that takes task duration estimations as input, it aggregates the task into project stages and then it combines them into the project as a whole (see Figure 4):

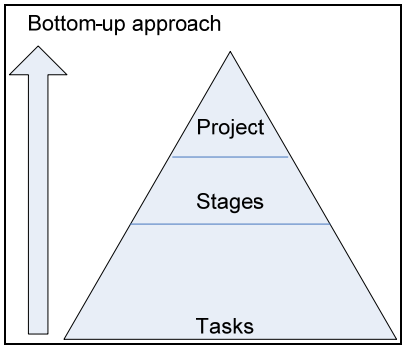

Fig.3. Hybrid model with bottom-up approach

As follows, the Metrix model structure and the steps it encompasses are presented in greater detail.

Individual task duration estimations and task interdependency represent the input data of the model. The model will also get the histo-

ry of the duration estimations for the tasks that have already been finished.

The result of running the model is a probabilistic distribution of the project duration.

The steps performed are described here under:

Step 1. The historical task duration estimations are collected for every developer. Will be considered both current project finished tasks and the tasks finished in other projects during the last 6 months.

Step 2. For every historical task duration estimation from step 1 we calculate the Estimation Accuracy Index (EAI) using the following formula:

$$EAI = \frac{ED}{AD} \quad (1)$$

where: ED–estimated task duration (in hours); AD–actual, elapsed task duration.

Using the results above we calculate the discreet probability distribution for the EAI indexes for every developer part of the team.

Step 3. On build the activity graph using the task dependency and estimated task durations.

Step 4. On find the critical path through the graph and on calculate the deterministic duration of the software project.

Step 5. On run the Monte Carlo simulation. The following operations are performed at each stage:

a) for every task, on adjust the estimated duration with a randomly chosen EAI (using the probability distribution from step 2).

b) on recalculate the critical path method and the project duration.

On repeat the simulation between 1000 and 10000 times.

Step 5. We calculate the project duration frequencies obtained as a result of Monte Carlo simulation. We display the project duration probability distribution. See figure 5:

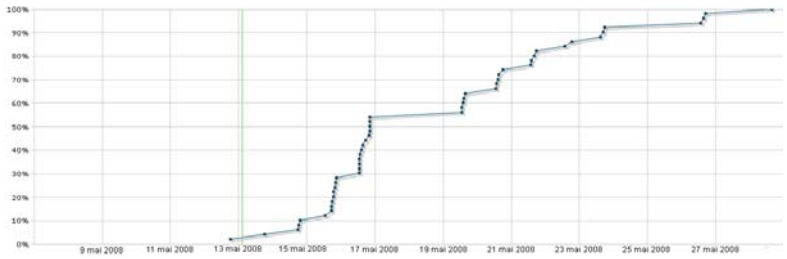

Fig.4. Probability distribution for project duration and project deadline

4. Conclusion

The Metrix model is a stochastic model for software project duration estimation using Monte Carlo simulation over an activity graph.

The first benefit of the Metrix model is that unlike classical deterministic models, which offer a single value for the estimated project duration, this model produces a probability distribution of the software project duration. By using this approach we reduce the project uncertainty by allowing the manager to gain better control over the project duration and the associated probability of a certain duration outcome.

The second benefit of the Metrix model is that it relies on the historic duration estimation of the team members. Similar models based on Monte Carlo simulations require a duration probability distribution function for every task. This requirement unfortunately set Monte Carlo simulations out of the practical domain into the academic universe. The innovation brought by the Metrix model is the elimination of the probability distribution functions requirement and the use of discreet probability distribution of the EAI (defined in this paper). The EAI probability distribution can be easily determined using the historical estimation errors which are at the disposal of most software companies..

References

[1] A. W. Chow, B. D. Goodman, J. W. Rooney, C. D. Wyble, *Engaging a corporate community to manage technology and embrace innovation*, IBM Systems Journal, Vol. 46, No. 4, 2007.

[2] ***, *The Chaos Report of IT Project Failure*, Standish Group, 2006
[3] ***, *ERP Software Implementation Success Rates*, Robbins-Gioia 2001
[4] Steve McConnell, *Rapid Development*, Microsoft Press, Washington, 1996
[5] Temnenco V., *Software Estimation, Enterprise-Wide*, IBM The Rational Edge, Vol. June 2007
[6] Jørgensen M., *A review of studies on expert estimation of software development effort*, The Journal of Systems and Software, Vol. 70, No. 1-2, February 2004.
[7] Callahan J., Moretton B., *Reducing software product development time*, International Journal of Project Management Vol. 19, No. 1, January 2001.
[8] Wiegers C., *Stop Promising Miracles*, Software Development Magazine, February 2000.
[9] Sungbin Cho, *An exploratory project expert system for eliciting correlation coefficient and sequential updating of duration estimation*, Expert Systems with Applications, Vol. 30, No. 4, May 2006.
[10] Boehm, Barry W. et al., *Software Cost Estimation with COCOMO II*, Prentice Hall, 2000.
[11] J. Uma Maheswari, Koshy Varghese, *Project Scheduling using Dependency Structure Matrix*, International Journal of Project Management, Vol. 23, No. 3, April 2005.