A Fresh Look at Cost Estimation, Process Models and Risk Analysis

Frank Padberg*
Fakultät für Informatik
Universität Karlsruhe, Germany
padberg@ira.uka.de

Abstract

Reliable cost estimation is indispensable for industrial software development. A detailed analysis shows why the existing cost models are unreliable. Cost estimation should integrate software process modelling and risk analysis. A novel approach based on probability theory is proposed. A probabilistic cost model could provide a solid basis for cost-benefit analyses.

1 Introduction

Economics play a central role today in software development. This position paper examines some consequences that three important issues linking software development with economics have for software engineering research.

First, companies must make profit. It is not enough to have the technical ability to develop software. There always is some deadline and some budget to meet. To get a basis for realistic financial planning, we have to estimate the development time and the cost of our software projects, and our estimates must be reliable.

Second, technical decisions have an economical impact. This is true especially for software design decisions. For example, when comparing two different designs for the same software product, the technically better design might lead to a much more involved project and thus increase development cost. To make technical choices which are economically reasonable, we have to estimate the impact of our technical decisions on the cost of our projects.

Third, introducing new programming tools or a new software development technique is an investment which must pay off in the future. It must increase our profit in future software projects through a gain in productivity. For example, how much do we gain through the use of object-oriented programming? We have to reliably estimate how much our productivity would have to increase in future projects for such an investment to make sense economically.

This workshop's call for papers noted that cost estimation still is unreliable and did not pay off as expected. As a result, the existing cost models are hardly ever used in practice, see [2]. From the preceding discussion it is clear, however, that reliable cost estimation is indispensable for industrial software development – *there is no way out of cost estimation* ! As a starting point for improvement, let's analyze why the existing models are unreliable.

* supported by a postdoctoral fellowship of the Deutsche Forschungsgemeinschaft DFG at the Graduiertenkolleg Informatik, Karlsruhe

2 Problems

Known Problems There is a number of studies comparing and assessing the existing cost models, see [4] for an overview. The known problems can be summarized as follows.

- Current models rely on "global" project data such as total project size or application domain. However, projects that are similar at this global level can have totally different costs.
- It is almost impossible to obtain a valid size estimate for a project, but this is the basic input variable for most models.
- Important project data are not well represented in the models. Usually, there are simple scale factors which are multiplied, although they are not independent. Most of these "cost drivers" can't be properly determined in practice.
- The data sets which are used to develop and to adjust the models usually are rather small and inhomogeneous.

Much of this is still true for recent models using analogy, machine learning or neural networks, see [5][6][7]. For an overview and references see [1]. Neural networks currently seem to provide the most accurate results among these models, yet there are serious problems such as properly training a net without overtraining it. It is unsatisfactory that neural nets don't explain how the estimates are reached.

I have identified additional problems which I believe to be central reasons for the difficulties in current cost estimation.

No Process Modelling There is an obvious and close connection between the particular course that a project takes and the time and money needed to carry out the project. From a manager's point of view, the course of a project might be described as "what happens at what time". For example, two possible courses of a project may differ in the time it takes to reach a particular milestone or in the amount of rework which occurs on a particular component. We can't expect to get reliable estimates without looking at the variety of different courses that a project might take, but this is not done in current cost models. In other words, explicit development process modelling is missing.

Looking at the different possible courses of a project immediately raises a question: which course is the one that the project will take? I think *this is the key problem* that we have to address. I'll come back to this problem in the next section.

Unused Design Data The design of the software being developed is essential for the project structure and thus for the cost. For example, how the teams get assembled much depends on the design. Design data is hardly used in existing models. Scale factors such as the "product complexity" cost driver are too simple, and the function points method emphasizes items such as the number of screens, not the project structure. Lack of design data in the model makes it impossible to compute the economical impact of design decisions.

No Risk Estimates There naturally is some uncertainty about the future progress of a project. This uncertainty implies that there always is some risk that the time and money actually needed to carry out a project will exceed the estimated values. A cost estimate is of little use if no bounds are provided for how much deviation may occur. The tighter these bounds are the more reliable the estimate is. Though the risk of exceeding the cost (or time) estimate already has been identified in risk analysis, no bounds or estimates for that risk are provided in current cost models.

3 A Novel Approach

Requirements From the preceding analysis, we get a list of general requirements for developing better cost estimation models.

- We must integrate cost estimation with software process modelling.
- To see how reliable our cost and time estimates are, we must compute the risk of exceeding the estimates.
- Relevant project data must enter our models in some reasonable way as parameters.
- We must make sure that the kind of data that we use for building our models actually can be measured and that our databases will be sufficiently large and homogeneous.

When integrating cost estimation with software process modelling, we must find the right level of abstraction at which to describe the software process. It seems to me that current software process models are either too informal or much too detailed for the purpose of cost estimation, but there are a number of modelling techniques in the field that we can use.

The Idea To improve our estimates, we'd like to take into account that a project may take one of several different courses, but we face the problem that we are uncertain about which one it will take. From our experience with past projects we know that some courses are more likely to occur than others. This observation points a way to a solution of the problem: *the idea is to compute for each of the possible courses the probability that the project will take that course.* To compute the probabilities of the courses we shall need a formal definition of what a course of a project is. We shall also need statistical data about the courses of past projects.

Suppose for a moment that the probabilities of a project's courses already have been computed. How do we get cost and risk estimates from that? A cost estimate can be computed as the (probabilistic) expected value of a function which assigns to each course its cost. This expected value is a "weighted average" of the costs of all the possible courses, the weights being the probabilities of the courses. The chances that the project will succeed with a given budget can be computed by summing up the probabilities of those courses whose cost would be within budget. It is also straightforward to compute as an estimate for the risk the probability that the cost estimate will be exceeded by some given amount of money.

A First Model To make these ideas explicit, I have developed a probabilistic model which is described in detail in [3]. The model is not tailored to a specific software development method.

PROJECTS. In the model, a project consists of several development teams and a project manager. Based on the high-level design of the software being developed, each team is assigned one component to work on. The teams work simultaneously, but not independently. Problems with the software's high-level design which are detected in one component may lead to rework in other components.

STATES. As a project advances, the software's high-level design will be revised from time to time because of problems. The time span between two consecutive designs is called a phase. The course of a project is modelled as a sequence

$$\zeta(1),\ \zeta(2),\ \ldots$$

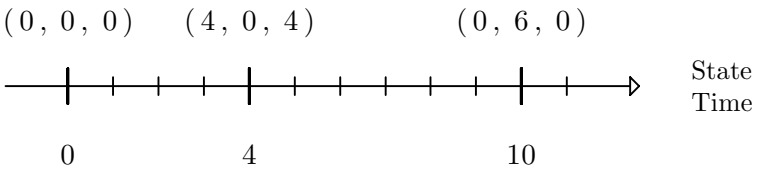
of states, where $\zeta(j)$ is the project's state at the end of phase j. The state of the project at the end of a phase is defined as a vector

$$\zeta = (\zeta_1,\ \zeta_2,\ \ldots\ \zeta_N)$$

where ζ_i is the state of team i at the end of the phase, and N is the number of teams. Time is discrete in the model. The state of a team is defined as the number of time slices that have passed since the team last reported a problem or was affected by changes in the software's high-level design. The definition is discussed in subsection "Remarks". The sequence of states is supplemented by the sequence of numbers

$$d_1\,,\quad d_2\,,\quad \ldots$$

giving the length of each of the project's phases. This might be the the first two phases of a sample project:



PROBABILITIES. The transitions between the project's states are controlled by transition probabilities. The transition probability

$$\mathrm{P}_{\zeta}\,(\,d,\,\eta\,)$$

is defined as the probability for ending the next phase after d time slices with state η given that the previous phase ended with state ζ. The formulas for the transition probabilities are involved, see [3]. The probability that the project will take a particular course can be computed by multiplying the transition probabilities which correspond to that course. For example,

$$\mathrm{P}_{(0,0,0)}(4,\ (4,\ 0,\ 4))\ \cdot\ \mathrm{P}_{(4,0,4)}(6,\ (0,\ 6,\ 0))$$

is the probability that the sample project will advance from state $(0,\ 0,\ 0)$ after 4 time slices into state $(4,\ 0,\ 4)$ and after 6 more time slices into state $(0,\ 6,\ 0)$. The model is a Markov process.

DATA. To compute the transition probabilities, statistical data and design data are required as input for the model. The statistical data are a measure of the pace at which the teams have made progress in past projects. For example, the model takes as input the probability that a team will finish its component after, say, five time slices if the high-level design doesn't get changed during that time. The raw data needed to compute these probabilities could be collected by the managers during running projects with little effort, see [3]. The design data required for the model are a measure of the degree of coupling between the software's components. The stronger the coupling is the more likely it is that problems detected in one component will propagate into other components.

Remarks The model should be regarded as sort of a prototype. I have made simplifying assumptions in [3] which currently limit its applicability. Despite the simplifications, the model already captures much of the dynamics of software projects, because it reflects that the progress of a team depends on the other teams' progress. Modelling of this dependence is possible because suitable high-level design data is used as input.

The statistical input data for the model are much more "local" than the input data for the existing models. I expect projects to be better comparable at this local level, which would give us larger databases.

Examples show that the model behaves as expected when the input data are changed, see [3].

The state of a team describes the progress the team has made so far. The state must be defined in such a way that it can be measured in practice. Thus a metric such as "x percent completed" would be hard to use. Instead of software size metrics in the past, let's explore *project progress metrics* in the future !

4 Benefit

A probabilistic model for cost estimation reflects the software development process much closer than the existing models. Instead of estimated software size data, such a model naturally inputs high-level design data and statistical productivity data. As a benefit, a probabilistic cost model ...

- promises to provide cost estimates that are more reliable;
- provides risk estimates in a straightforward way;
- gives guidelines for collecting and utilizing detailed project data;
- measures how the development cost depends on the software design;
- allows to examine the cost-benefit ratio of software development tools and techniques.

To help practitioners meet their economic constraints, let's select areas for research where progress would be most *cost-effective* ! A probabilistic cost model that includes relevant project data could provide a solid basis for the necessary cost-benefit analyses.

References

1. Gray, MacDonell: "A Comparison of Techniques for Developing Predictive Models of Software Metrics" Information and Software Technology 39 (1997) 425-437
2. Lederer, Prasad: "Nine Management Guidelines for Better Cost Estimating" Communications of the ACM 35-2 (1992) 51-59
3. Padberg: "A Probabilistic Model for Software Projects", submitted
4. Sallis, Tate, MacDonell: *Software Engineering*, Addison-Wesley 1995
5. Shepperd, Schofield, Kitchenham: "Effort Estimation Using Analogy" Proceedings of the 18th International Conference on Software Engineering (1996) 170-178
6. Srinivasan, Fisher: "Machine Learning Approaches to Estimating Software Development Effort" IEEE Transactions on Software Engineering 21-2 (1995) 126-137
7. Wittig, Finnie: "Using Artificial Neural Networks and Function Points to Estimate 4GL Software Development Effort", Australian Journal of Information Systems 1 (1994) 87-94

Author's Address

Dr.-Ing. Frank Padberg
Fakultät für Informatik
Universität Karlsruhe
Am Fasanengarten 5
76131 Karlsruhe
Germany