Chapter 7

Measuring cost and effort

Project Costs

- Software project managers are responsible for controlling project budgets so, they must be able to make estimates of how much a software development is going to cost. The principal components of project costs are:
  - Hardware costs.
  - Travel and training costs.
  - Effort costs (the costs of paying software engineers).
- The dominant cost is the effort cost. This is the most difficult to estimate and control, and has the most significant effect on overall costs.
- Software costing should be carried out objectively with the aim of accurately predicting the cost to the contractor of developing the software.
- Software cost estimation is a continuing activity which starts at the proposal stage and continues throughout the lifetime of a project. Projects normally have a budget, and continual cost estimation is necessary to ensure that spending is in line with the budget.
- Effort can be measured in staff-hours or staff-months (Used to be known as man-hours or man-months).

COCOMO

- Most widely used model for effort and cost estimation.
- Projects fall into three categories: organic, semidetached, and embedded, characterized by their size.
- There is also an **intermediate** model which, as well as size, uses 15 other **cost drivers**.

- For details about COCOMO please check cocomo pdf

Cost Drivers for the COCOMO Model

- Software reliability
- Size of application database
- Complexity
- Analyst capability
- Software engineering capability
- Applications experience
- Virtual machine experience
- Programming language expertise
- Performance requirements
- Memory constraints
- Volatility of virtual machine
- Environment
- Turnaround time
- Use of software tools
- Application of software engineering methods
- Required development schedule
- Values are assigned by the manager.

Constraint model

- Constraint is something that limits or restricts something else
- A constraint is not a bias or a preference. Implementing a rule such as “games should be scheduled on Friday nights if possible” is not a constraint, it’s a preference. A rule such as “customer ABC’s orders should be processed out of DC 123” is also not a binding constraint if viable alternatives are allowed to be considered.
- Some industry pundits believe that using preferences and biases in the form of heuristic rules, in lieu of hard constraints, is perfectly acceptable — but is it? What it comes down to is how important profitability, predictability, agility, and accountability are to the company’s management and their investors.

Constraint model (continued)

- Constraint-based modeling is a scientifically-proven mathematical approach, in which the outcome of each decision is constrained by a minimum and maximum range of limits (+/- infinity is allowed). Decision variables sharing a common constraint must also have their solution values fall within that constraint's bounds. A constraint-based modeling approach is most commonly — and effectively — used with optimization techniques, such as the use of linear and mixed-integer programming to maximize an objective function.
- Conversely, constraint-based modeling is not hypothesis-driven (a.k.a. deterministic) modeling. No analyst using a constraint-based model should be able to predict the behavior of the entire model beforehand. Any "solution" that allows for this is not true constraint-based modeling.

Software Lifecycle Management (SLIM)

- The Putnam model is an empirical software effort estimation model. The original paper by Lawrence H. Putnam published in 1978 is seen as pioneering work in the field of software process modelling. As a group, empirical models work by collecting software project data (for example, effort and size) and fitting a curve to the data. Future effort estimates are made by providing size and calculating the associated effort using the equation which fit the original data (usually with some error).
- Created by Lawrence Putnam, Sr. the Putnam model describes the time and effort required to finish a software project of specified size. SLIM (Software Lifecycle Management) is the name given by Putnam to the proprietary suite of tools his company QSM, Inc. has developed based on his model. It is one of the earliest of these types of models developed, and is among the most widely used. Closely related software parametric models are Constructive Cost Model (COCOMO), Parametric Review of Information for Costing and Evaluation – Software (PRICE-S), and Software Evaluation and Estimation of Resources – Software Estimating Model (SEER-SEM).

Cont….d

Putnam used his observations about productivity levels to derive the software equation:

$$\frac{B^{1/3} \cdot \text{Size}}{\text{Productivity}} = \text{Effort}^{1/3} \cdot \text{Time}^{4/3}$$

where:

- Size is the product size (whatever size estimate is used by your organization is appropriate). Putnam uses ESLOC (Effective Source Lines of Code) throughout his books.
- B is a scaling factor and is a function of the project size.[4][5]
- Productivity is the **Process Productivity**, the ability of a particular software organization to produce software of a given size at a particular defect rate.
- Effort is the total effort applied to the project in person-years.
- Time is the total schedule of the project in years.

Cost models: advantages and drawbacks

- **Advantage**
- Flexible Areas of Calculation: Cost estimation models interpret costs. Some use an array of algorithmic models to assign values to certain factors in order to compute costs. Other models include an expert judgment model and an analogy estimation.
- Efficiency and Cost Control: Efficiency refers to the ability to do a task quickly and accurately, saving the business both time and money. When the right type of model is chosen, the business can realize gains from efficiency by using cost estimation to quickly calculate expenses and make choices on funding projects, choosing suppliers and other activities.
- **Drawbacks**
- Subjectivity: On the downside, cost estimation is somewhat subjective. Even with algorithmic models, it is usually up to the business to weight certain values over others and assign the correct values to factors. The other model options are even more subjective. This means that sometimes a manager can make mistakes just as easily when using a cost estimation model than when working without one.
- Variable Factors: In a perfect world, factors remain steady and cost estimation models always produce accurate results. Unfortunately, markets are in constant flux, prices change and technology is always moving onward. This leads to constant changes in price, which means that costs have to be frequently updated and values changed to match. This can be a drain on time, especially in highly complex models.