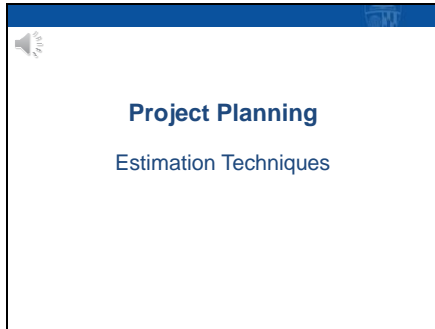
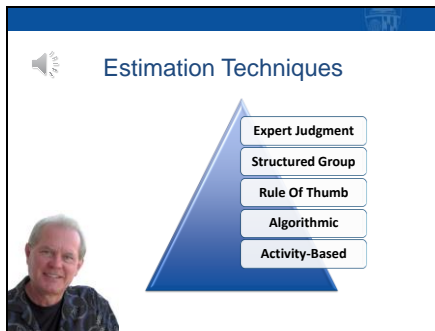


1



In this lecture we'll discuss some techniques that can be used in developing project estimates.

2

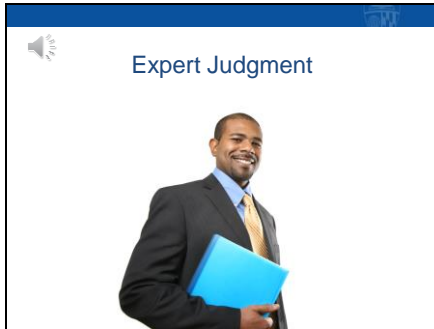


In an earlier lecture we discussed the kind of things that get estimated during the project planning process. But...how do estimates actually get made?

Well...if you remember the results of one of the earlier-referenced surveys...a lot of estimates are just guesstimates. Guesstimates aren't repeatable and they're usually not defensible in practice, so we're not going to say anything more about them.

Instead, we'll discuss five types of techniques that are commonly used in practice: expert judgment, structured group techniques, rule of thumb estimating, algorithmic models, and activity-based techniques.

3

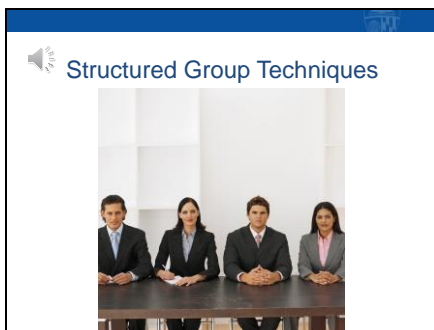


Expert judgment is a widely used method to come up with estimates. It relies on the judgment and experience of one or more individuals who will factor their knowledge into the specifics of the project at hand.

Expert judgment works best when there really are experts that can be called upon, when there is real historical information that can be accessed, when a structured process is used, and when the basis for the estimates is documented and communicated to project stakeholders.

The accuracy of estimates made this way can be good or not so good. They can be good if the project to be estimated is, in fact, similar to those the estimators have had experience with. If not, then results can vary substantially. For example, having experience in small-scale projects does not necessarily prepare one to deal with larger, more complex projects, where additional project activities and processes like quality assurance, configuration management, independent testing, and so forth may be required.

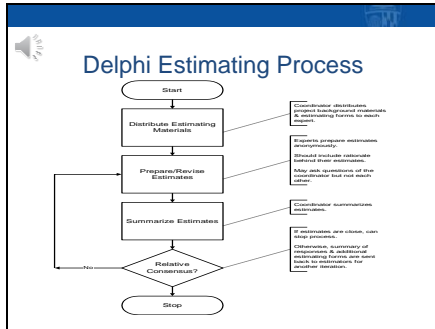
4



Another estimating technique, that is an improvement on the expert judgment technique, is a structured group technique. In this kind of technique multiple estimators, who are typically experts, form the estimating team...and they follow a structured process in arriving at their estimates. They may or may not rely on using expert judgment, but the structured nature of the process typically yields better results.

One such technique is the Delphi Method...and it has a number of variations. Let's take a quick look at it and see how it works.

5



The Delphi technique was developed by the Rand Corporation for use on “never done before” projects in order to get at least a reasonable ballpark estimate of project effort, cost, or schedule.

It uses a panel of expert estimators...usually at least three, a coordinator, and a set of rules.

This flowchart illustrates the process and some of the rules. The coordinator distributes project background information and estimating forms to each estimator. Each estimator prepares an estimate, anonymously, and documents their rationale. If they have questions, they may consult with the coordinator, but they can’t consult with each other.

The coordinator then summarizes the estimates. If the coordinator determines that the estimates are close enough so that a reasonable range estimate can be formed, the process stops. Otherwise, the estimates and rationale are summarized and given to the estimators, and another iteration takes place. Estimators know the other estimates, and the rationale that others used, but they don’t know which estimators are associated with them...so as to avoid bias.

The structured nature of the process, and the documentation of rationale, help to make this technique more repeatable than the basic expert judgment approach. And, even if the final result is a wide range of values, the documented rationale provides support for the variation.

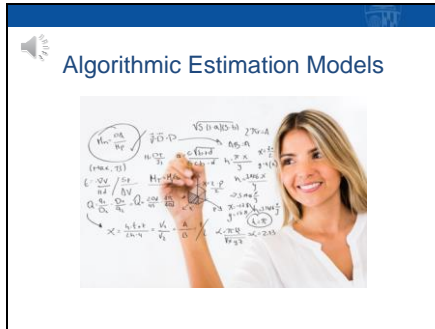


Another common approach to estimating is rule-of-thumb estimating. There's lots of specific techniques under this category.

For example, some techniques will involve making a size estimate of the software product, often in lines of code, then use a productivity factor like the average lines of code per week a programmer can produce to calculate effort and cost. Some techniques involve making an overall project estimate, say for total effort, and then use ratios to allocate how much effort is allocated to the various project phases.

The potential problem with rule-of-thumb techniques is that they may not scale up across different sizes and types of projects because the ratios and productivity factors can be different. As an example, the larger the project, in terms of people and ultimate product size...the more collaboration between project team members and other stakeholders is required...so productivity factors are different than for smaller projects. As another example, the percentage of time spent in various project phases will vary based on the size and type of project. For a small project, up to 30 percent of the overall effort may be spent coding, while in a large-scale government contracting project the percentage of effort spent in coding may be 15-20 percent due to all the additional processes and activities that need to take place.

7

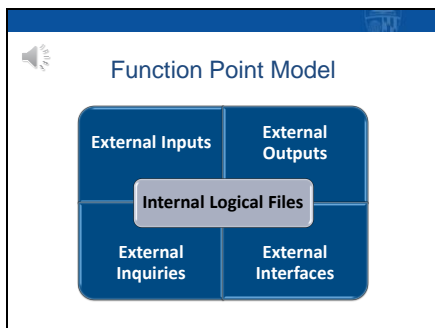


Algorithmic estimating models involve using calculations or formulas that are typically based on estimates of software product size to provide estimates of effort, cost, or project schedule. Sometimes these models use very complex formulas, and sometimes they're based on counts of software product attributes.

In practice, to use some algorithmic models effectively historical data must be used to calibrate the models to organizational specifics, and some models require estimating or choosing a dozen or more parameters based on project and organizational characteristics. If historical is not available, the resulting estimates can be quite inaccurate.

The advantage of using these models is that they are quick and cheap.

8



One example of an algorithmic estimation model is the function point analysis model. A function point is an abstract measurement that expresses the amount of functionality a software product provides to its stakeholders.

Function points are considered to be a measure of software size. The concept of function points was developed by Allan Albrecht at IBM. The International Function Point Users Group (IFPUG) is a non-profit organization that promotes the use of function point analysis and offers a certification program.

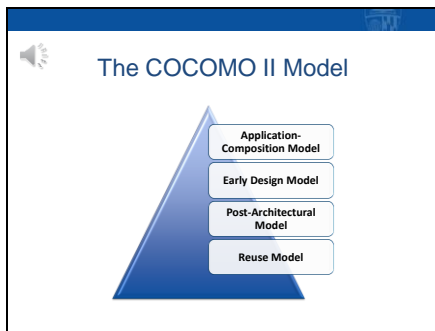
The basics behind the concept of function points is this. First, a product's functional requirements are developed. Then, each requirement is categorized into one of five types: external inputs, external outputs, external inquiries, external interfaces, and internal logical files. A requirement's complexity is then evaluated and assigned a number of function points. The function point counts for all the requirements are then summed to arrive at an initial function point count for the product. Adjustments to the initial function point count might then be made based upon specific

product characteristics.

The final function point count can then be correlated with historical data...like the average number of hours to implement a function point...to come up with estimates of effort, cost, and schedule. Or...the function point count might be plugged into another model, like the COCOMO model, to come up with the estimates.

A complete function point example is provided in the course materials for this course module and can be downloaded from the course website.

9



Another example of an algorithmic estimation model is COCOMO II. COCOMO stands for Constructive Cost Model. The COCOMO model was developed by, Barry Boehm, a well-known mathematician and software engineer, now at the University of Southern California's Center for Software Engineering, and who is sometimes referred to as the father of software engineering economics. COCOMO has been around for decades, and is constantly being refined to improve accuracy and applicability.

COCOMO II consists of four sub-models that are based on the type of project and the point in the project life cycle that the estimating takes place.

The Application-Composition Model is used for estimating on prototyping projects.

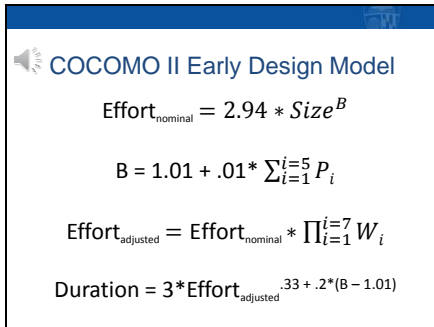
The Early Design Model is used to for estimating once an initial set of requirements for the project has been established.

The Post-Architectural Model is the most detailed model, and is used once the design architecture has been established.

The Reuse Model is used on projects that involve reusable software components.

I'm going to focus my discussion on the Early Design Model.

10



COCOMO II Early Design Model

$$\text{Effort}_{\text{nominal}} = 2.94 * \text{Size}^B$$
$$B = 1.01 + .01 * \sum_{i=1}^{i=5} P_i$$
$$\text{Effort}_{\text{adjusted}} = \text{Effort}_{\text{nominal}} * \prod_{i=1}^{i=7} W_i$$
$$\text{Duration} = 3 * \text{Effort}_{\text{adjusted}}^{.33 + .2 * (B - 1.01)}$$

Some of the formulas for the COCOMO II Early Design Model are shown here. A complete example is provided in the course materials for this course module and can be downloaded from the course website.

In the COCOMO II model, the nominal effort, in units of person-months, is estimated based upon some size estimate for the software product, in thousands of lines of source code, raised to some power, B.

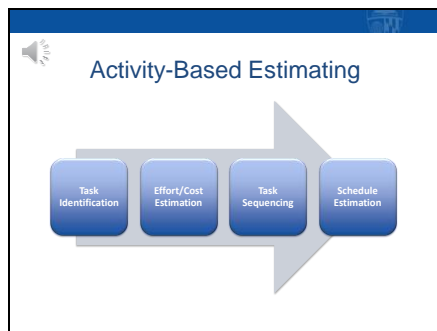
B is a parameter that is based on five product and process characteristics, and its value ranges from 1.1 to 1.24.

To refine the effort estimate, the nominal effort can be adjusted by a multiplier based on seven project and process attributes.

An estimate of project duration, in calendar-months, can then be made using the duration formula. And...not shown here...the duration estimate can be adjusted, if necessary, with a multiplier that reflects a project compression or expansion factor.

The advantage of this estimating approach is that it is fast and cheap. The challenges are that organizations must usually adjust the estimates for specific internal characteristics, and the estimates are all based on a size estimate...which can be difficult and may not be very accurate in practice.

11



Yet another estimation technique is called activity-based estimation. This technique starts with what's called a work-breakdown structure. A work-breakdown structure is a list of all the tasks that are needed to complete the project. They can be defined at a high level or at a very detailed level. Once the work tasks are identified, the effort associated with each task...usually measured in person-hours, person-days, person-weeks, or person-months...is estimated, along with the costs...which are primarily labor costs. Non-labor costs also need to be estimated in order to come up with a total project cost estimate. The next step is to analyze the dependencies between tasks. Some tasks can be done in parallel, and some must be done sequentially...so these dependencies help to determine the sequencing of all project tasks. Finally, the tasks, the dependencies, and the effort estimates, along with the staff resources that are expected to be committed to the project are all combined to estimate the project schedule. In practice, it is common to use project planning software tools to assist with these steps.

Activity-based estimating has been shown to be the most accurate way of estimating in practice. And...it may be required for projects that involve contracts. It's primary disadvantage is that for very large projects it can be quite time-consuming, even with assistance from project management tools. In practice, very large projects are often decomposed into smaller sub-projects to control complexity.

The remaining lectures in this course module will explore activity-based estimating in more depth.