Institute of Computer Science
Vikram University, Ujjain (M.P.)

SOFTWARE ENGINEERING

M.Sc. Computer Science

BY

Mr. Shekhar Disawal

Planning a Software Project

Process Planning

Planning is the most important project management activity. It has two basic objectives—

1. Establish reasonable cost & schedule
2. Quality goals for the project & to draw out a plan to deliver the project goals.

Planning Guidelines

- ✓ A project succeeds if it meets its cost, schedule, and quality goals.
- ✓ Without the project goals being defined, it is not possible to even declare if a project has succeeded.
- ✓ Without detailed planning, no real monitoring or controlling of the project is possible.
- ✓ Projects are rushed toward implementation with not enough effort spent on planning. No amount of technical effort later can compensate for lack of careful planning.
- ✓ Lack of proper planning is a sure ticket to failure for a large software project.
- ✓ The inputs to the planning activity are the requirements specification and maybe the architecture description.
- ✓ A very detailed requirements document is not essential for planning, but for a good plan all the important requirements must be known, and it is highly desirable that key architecture decisions have been taken.
- ✓ Have people familiar with the tasks make the estimate.
- ✓ Use several people to make estimates.
- ✓ Base estimates on normal conditions, efficient methods, and a normal level of resources.
- ✓ Use consistent time units in estimating task times.
- ✓ Treat each task as independent, don’t aggregate.
- ✓ Don’t make allowances for contingencies.
- ✓ Adding a risk assessment helps avoid surprises to stakeholders.

There are generally two main outputs of the planning activity:

1. the overall project management plan document that establishes
   - the project goals on the cost
   - schedule
   - quality fronts
   - defines the plans for managing risk
   - monitoring the project
2. the detailed plan
   - The detailed project schedules
   - Specifying the tasks that need to be performed to meet the goals
   - The resources who will perform them, and their schedule.

The overall plan guides the development of the detailed plan, which then becomes the main guiding document during project execution for project monitoring.

Effort Estimation

For a software development project, overall effort and schedule estimates are essential prerequisites for planning the project. These estimates are needed before development is initiated, as they establish the cost and schedule goals of the project. Without these, even simple questions like "is the project late?" "Are there cost overruns?" and "when is the project likely to complete?" cannot be answered. A more practical use of these estimates is in bidding for software projects, where cost and schedule estimates must be given to a potential client for the development contract. Effort and schedule estimates are also required for determining the staffing level for a project during different phases, for the detailed plan, and for project monitoring. The accuracy with which effort can be estimated clearly depends on the level of information available about the project. The more detailed the information, the more accurate the estimation can be. Of course, even with all the information available, the accuracy of the estimates will depend on the effectiveness and accuracy of the estimation procedures or models employed and the process. If from the requirements specifications, the estimation approach can produce estimates that are within 20% of the actual effort about two-thirds of the time, then the approach can be considered good. Here we discuss two commonly used approaches.

Top-Down Estimation Approach

The top-down method is also known as the analogous method. It is used to determine order of magnitude estimates in the initiation phase of the project. The method uses the actual durations, effort or costs from previous projects as a basis for estimating the effort or costs for the current project.

1. Identify a previous project or section of a previous project that is similar to the current project.
2. Assess the extent to which the current project is similar to the previous project – the comparison factor (e.g 1.5 if the current project is estimated to be 50% larger).
3. Compute the estimate for the current project based on the actual durations, effort or costs from the previous project and the comparison factor

A more general function for determining effort from size that is commonly used is of the form:

$$\mathbf{EFFORT = a*SIZE^b}$$

where a and b are constants, and project size is generally in KLOC (Thousand lines of code is treated as KLOC. This metric helps in knowing the size and complexity of the Software Application). Values for these constants for an organization are determined through regression analysis, which is applied to data about the projects that have been performed in the past.

To develop an estimation model, we have to follow these steps:

1. Determine a list of potential / most important effort cost drivers.
2. Determine a scaling model for each effort and cost driver.
3. Select initial estimation model.
4. Measure and estimate projects and compare.
5. Evaluate quality of estimation as part of project post-mortem.
6. Update and validate model at appropriate intervals.

The COCOMO Model

The Constructive Cost Model (COCOMO) is an algorithmic software cost estimation model developed by Barry W. Boehm. The model uses a basic regression formula with parameters that are derived from historical project data and current as well as future project characteristics. In the COCOMO model, after determining the initial estimate, some other factors are incorporated for obtaining the final estimate. To do this, COCOMO uses a set of 15 different attributes of a project called cost driver attributes. Examples of the attributes are required software reliability, product complexity, analyst capability, application experience, use of modern tools, and required development schedule. Each cost driver has a rating scale, and for each rating, a multiplying factor is provided.

Boehm proposed three levels of the model: basic, intermediate, detailed.

1. The basic COCOMO'81 model is a single-valued, static model that computes software development effort (and cost) as a function of program size expressed in estimated thousand delivered source instructions (KDSI).
2. The intermediate COCOMO'81 model computes software development effort as a function of program size and a set of fifteen "cost drivers" that include subjective assessments of product, hardware, personnel, and project attributes.
3. The advanced or detailed COCOMO'81 model incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step (analysis, design, etc.) of the software engineering process.

COCOMO models depends on the two main equations

1. *development effort : MM = a * KDSI b*

based on MM - man-month / person month / staff-month is one month of effort by one person. In COCOMO'81, there are 152 hours per Person month. According to organization this value may differ from the standard by 10% to 20%.

2. *effort and development time (TDEV) : TDEV = 2.5 * MM c*

The coefficients a, b and c depend on the mode of the development.

There are three modes of development:

Development Mode	Project Characteristics			
	Size	Innovation	Deadline/constraints	Dev. Environment
Organic	Small	Little	Not tight	Stable
Semi-detached	Medium	Medium	Medium	Medium
Embedded	Large	Greater	Tight	Complex hardware/ customer interfaces

Table 1: development modes

****