Project Planning

**Factors affecting software pricing (10 Marks)

Factor	Description
Market opportunity	A development organization may quote a low price because it wishes to move into a new segment of the software market. Accepting a low profit on one project may give the organization the opportunity to make a greater profit later. The experience gained may also help it develop new products.
Cost estimate uncertainty	If an organization is unsure of its cost estimate, it may increase its price by a contingency over and above its normal profit.
Contractual terms	A customer may be willing to allow the developer to retain ownership of the source code and reuse it in other projects. The price charged may then be less than if the software source code is handed over to the customer.
Requirements volatility	If the requirements are likely to change, an organization may lower its price to win a contract. After the contract is awarded, high prices can be charged for changes to the requirements.
Financial health	Developers in financial difficulty may lower their price to gain a contract. It is better to make a smaller than normal profit or break even than to go out of business. Cash flow is more important than profit in difficult economic times.

**Project plans (10 Marks)

A project plan sets out the resources available to the project, the work breakdown, and a schedule for carrying out the work. The plan should identify risks to the project and the software under development, and the approach that is taken to risk management. Although the specific details of project plans vary depending on the type of project and organization, plans normally include the following sections:

1. Introduction This briefly describes the objectives of the project and sets out the constraints (e.g., budget, time, etc.) that affect the management of the project.

2. Project organization This describes the way in which the development team is organized, the people involved, and their roles in the team.

3. Risk analysis This describes possible project risks, the likelihood of these risks arising, and the risk reduction strategies that are proposed.

4. Hardware and software resource requirements This specifies the hardware and support software required to carry out the development. If hardware has to be bought, estimates of the prices and the delivery schedule may be included.

5. Work breakdown This sets out the breakdown of the project into activities and identifies the milestones and deliverables associated with each activity.

6. Project schedule This shows the dependencies between activities, the estimated time required to reach each milestone, and the allocation of people to activities.

7. Monitoring and reporting mechanisms This defines the management reports that should be produced, when these should be produced, and the project monitoring mechanisms to be used.

The project planning process (10 Marks)

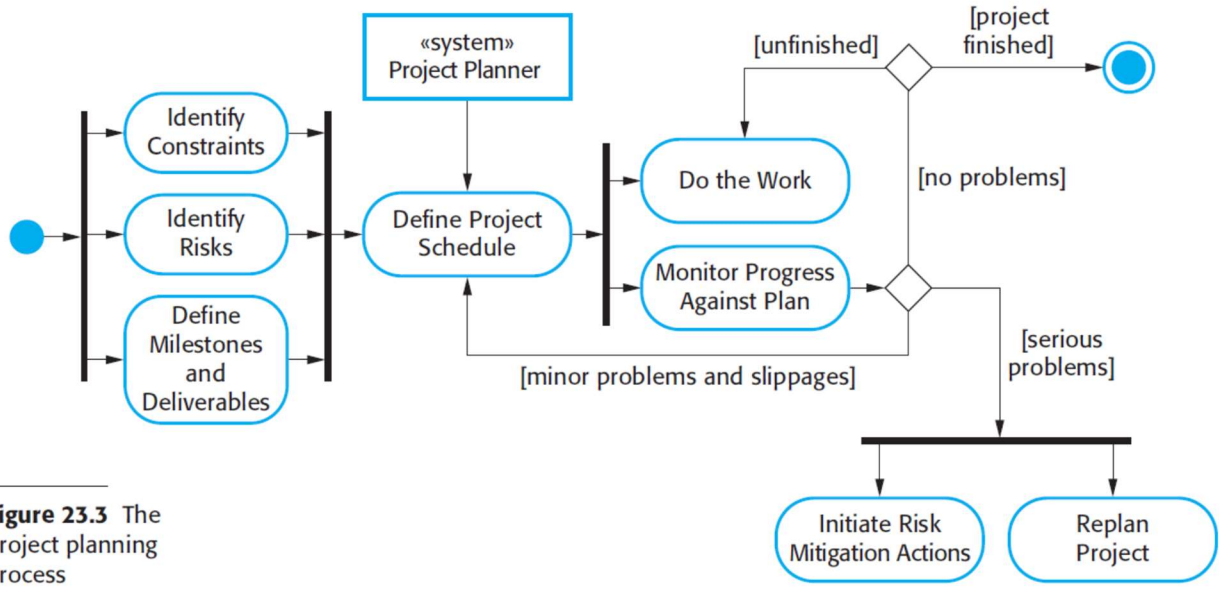


Figure 23.3 The project planning process

The diagram shows a typical workflow for a project planning process. Plan changes are inevitable. As more information about the system and the project team becomes available during the project, you should regularly revise the plan to reflect requirements, schedule, and risk changes.

At the beginning of a planning process, you should assess the constraints affecting the project. These constraints are the required delivery date, staff available, overall budget, available tools, and so on. In conjunction with this, you should also identify the project milestones and deliverables.

The process then enters a loop. You draw up an estimated schedule for the project and the activities defined in the schedule are initiated or given permission to continue. After some time (usually about two to three weeks), you should review progress and note discrepancies from the planned schedule. Because initial estimates of project parameters are inevitably approximate, minor slippages are normal and you will have to make modifications to the original plan.

***Algorithmic cost modeling (10 Marks)

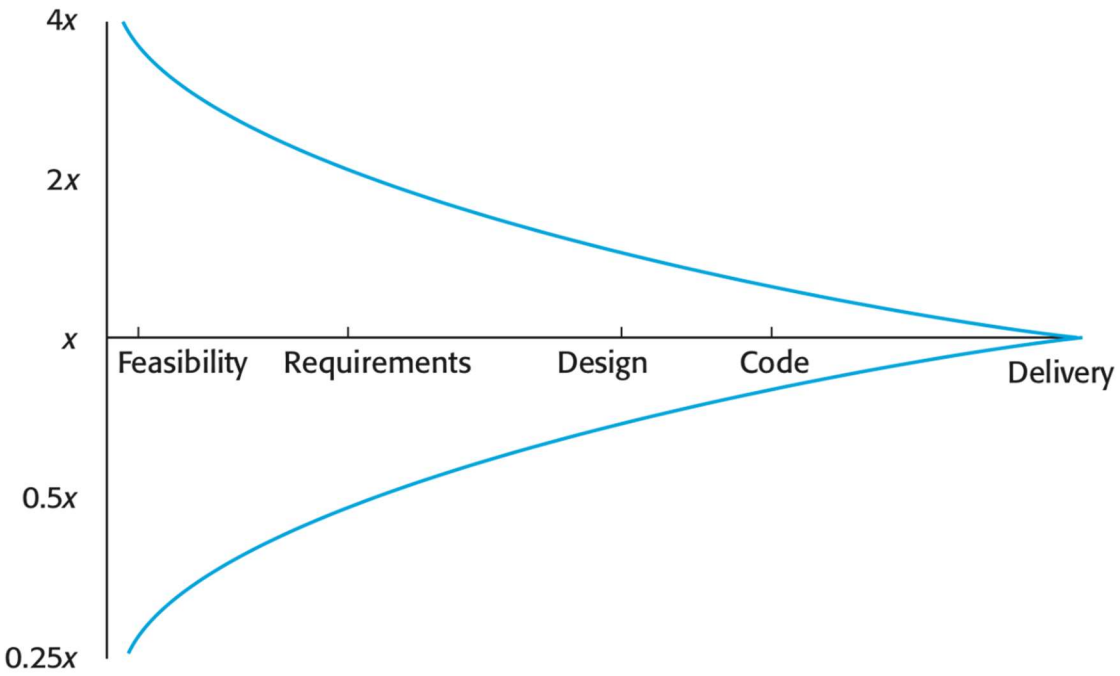
Organizations need to make software effort and cost estimates. There are two types of technique that can be used to do this:

Experience-based techniques

The estimate of future effort requirements is based on the manager's experience of past projects and the application domain. Essentially, the manager makes an informed judgment of what the effort requirements are likely to be.

Algorithmic cost modeling

In this approach, a formulaic approach is used to compute the project effort based on estimates of product attributes, such as size, and process characteristics, such as experience of staff involved.



Experience-based techniques rely on judgments based on experience of past projects and the effort expended in these projects on software development activities. Typically, you identify the deliverables to be produced in a project and the different software components or systems that are to be developed. You document these in a spreadsheet, estimate them individually and compute the total effort required. It usually helps to get a group of people involved in the effort estimation and to ask each member of the group to explain their estimate. The difficulty with experience-based techniques is that a new software project may not have much in common with previous projects. If you have not worked with these techniques, your previous experience may not help you to estimate the effort required, making it more difficult to produce accurate costs and schedule estimates.

With **algorithmic cost modeling**, Cost is estimated as a mathematical function of product, project and process attributes whose values are estimated by project managers:

$$\text{Effort} = A * \text{Size}^{B} * M$$

A is an organization-dependent constant, B reflects the disproportionate effort for large projects and M is a multiplier reflecting product, process and people attributes. The most commonly used product attribute for cost estimation is code size. Most models are similar but they use different values for A, B and M. The size of a software system can only be known accurately when it is finished. Several factors influence the final size: use of reused systems and components; programming language; and distribution of system. As the development process progresses then the size estimate becomes more accurate. The estimates of the factors contributing to B and M are subjective and vary according to the judgment of the estimator. Algorithmic cost models are a systematic way to estimate the effort required to develop a system. However, these models are complex and difficult to use. There are many attributes and considerable scope for uncertainty in estimating their values. This complexity means that the practical application of algorithmic cost modeling has been limited to a relatively small number of large companies, mostly working in defense and aerospace systems engineering.

COCOMO cost modeling (05Marks)

COCOMO (Constructive Cost Modeling) cost modeling is an empirical model based on project experience. It is a well-documented, 'independent' model which is not tied to a specific software vendor. Long history from initial version published in 1981 (COCOMO-81) through various instantiations to COCOMO 2. COCOMO 2 takes into account different approaches to software development, reuse, etc. COCOMO 2 incorporates a range of sub-models that produce increasingly detailed software estimates. The sub-models in COCOMO 2 are:

- **Application composition model** used when software is composed from existing parts.
- **Early design model** used when requirements are available but design has not yet started.
- **Reuse model** used to compute the effort of integrating reusable components.
- **Post-architecture model** used once the system architecture has been designed and more information about the system is available.