

Project planning

Unit – IV

8 Hours

Project Planning: Software pricing, Plan-driven Development: Project Plans, Planning process, Project scheduling: Schedule Representation, Agile Planning, Estimation techniques: Algorithmic Cost Modeling. The COCOMO II Model. Project Duration and Staffing

PROJECT PLANNING

- Software pricing
- Plan-driven development
- Project scheduling
- Agile planning
- Estimation techniques

Project planning

- Project planning involves **breaking down the work into parts** and **assign these to project team members**, **anticipate problems that might arise** and **prepare tentative solutions** to those problems.
- The **project plan**, which is **created at the start of a project**, is used to **communicate how the work will be done to the project team and customers**, and **to help assess progress on the project**.

Planning stages

- At the proposal stage, *when you are bidding for a contract to develop or provide a software system.*
- During the project startup phase, when you have to plan *who will work on the project, how the project will be broken down into increments, how resources will be allocated* across your company, etc.
- Periodically throughout the project, *when you modify your plan in the light of experience gained and information from monitoring the progress of the work.*

Proposal planning

- Planning may be necessary with only **outline software requirements**.
- The aim of planning at this stage is to provide information that will be used in **setting a price for the system** to customers.

Software pricing

- **Estimates** are made to discover the **cost**, to the **developer**, of **producing a software system**.
 - You take into account, *hardware, software, travel, training* and *effort costs*.
- **There is not a simple relationship** between the **development cost** and the **price charged** to the customer.
- **Broader organisational, economic, political and business considerations** *influence the price charged*.

Factors affecting software pricing

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 .

Factors affecting software pricing

Factor	Description
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.

Plan-driven development

- Plan-driven or plan-based development is an approach to software engineering **where the development process is planned in detail.**
 - Plan-driven development is based on **engineering project management techniques** and is the **‘traditional’ way of managing large software development projects.**
- A project plan is created that **records the *work to be done, who will do it, the development schedule* and the *work products.***
- **Managers** use the plan to ***support project decision making*** and as a ***way of measuring progress.***

Plan-driven development – pros and cons

- The arguments in favor of a plan-driven approach are that early planning allows ***organizational issues (availability of staff, other projects, etc.) to be closely taken into account***, and that ***potential problems and dependencies are discovered*** before the project starts, rather than once the project is underway.
- The principal **argument** against plan-driven development is that **many early decisions have to be revised** because of ***changes to the environment in which the software is to be developed and used.***

Project plans

- In a **plan-driven** development project, a **project plan** sets out the *resources available to the project*, the *work breakdown* and a *schedule* for carrying out the work.
- **Plan sections**
 - Introduction
 - Project organization
 - Risk analysis
 - Hardware and software resource requirements
 - Work breakdown
 - Project schedule
 - Monitoring and reporting mechanisms

Project plan supplements

Plan	Description
Quality plan	Describes the <i>quality procedures</i> and <i>standards</i> that will be used in a project.
Validation plan	Describes the <i>approach</i> , <i>resources</i> , and <i>schedule</i> used for system validation.
Configuration management plan	Describes the <i>configuration management procedures</i> and <i>structures</i> to be used.
Maintenance plan	Predicts the <i>maintenance requirements</i> , <i>costs</i> , and <i>effort</i> .
Staff development plan	Describes how the <i>skills</i> and <i>experience</i> of the <i>project team members</i> will be <i>developed</i> .

The planning process

- Project planning is an **iterative process** that ***starts*** when you create an ***initial project plan*** during the ***project startup phase***.
- 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***.
 - ***Changing business goals*** also leads to ***changes in project plans***. As business goals change, this could affect all projects, which may then have to be re-planned.

The project planning process

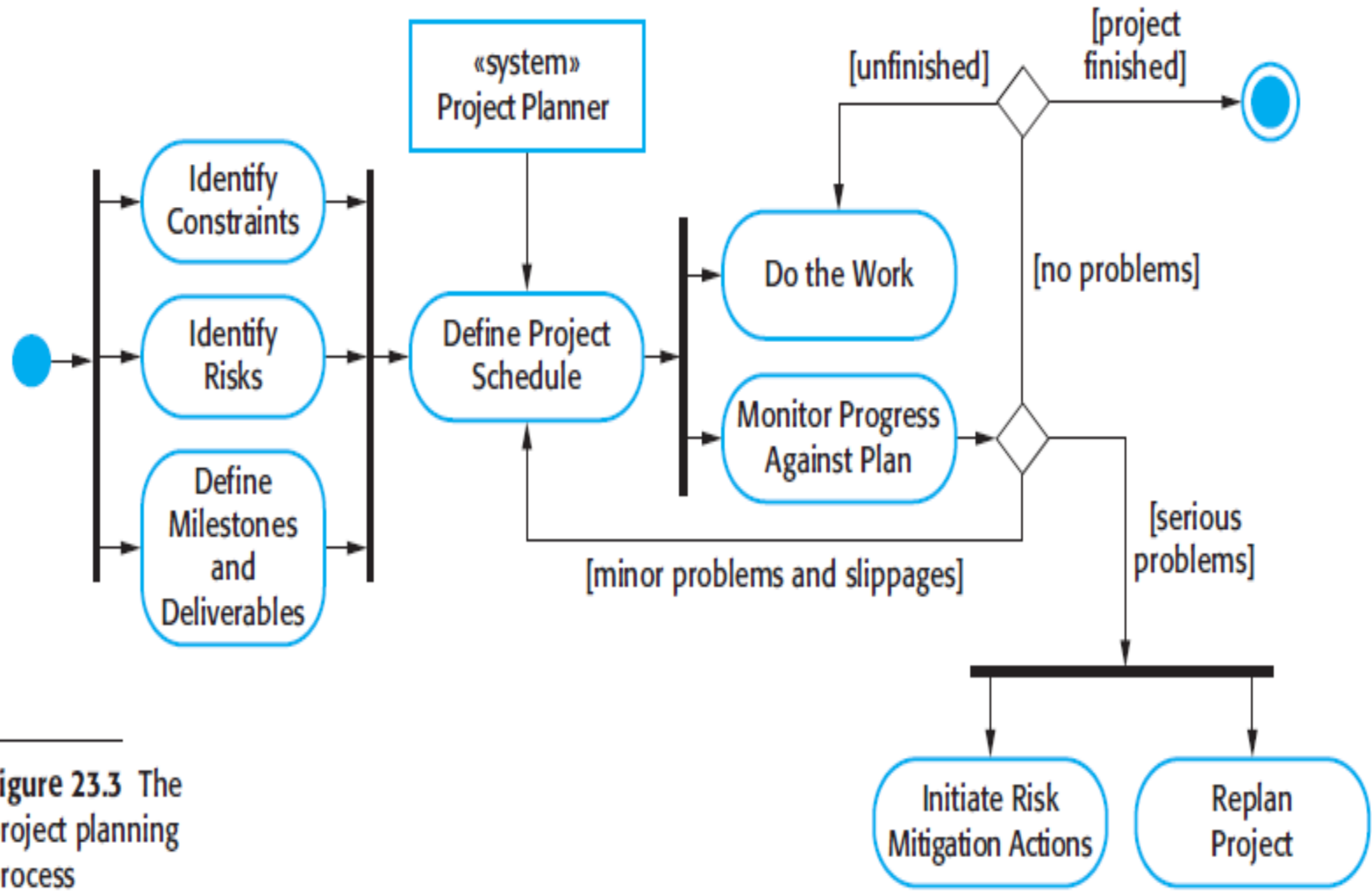


Figure 23.3 The project planning process

Project scheduling

- Project scheduling is the process of deciding **how the work in a project will be organized as separate tasks, and when and how these tasks will be executed.**
- You estimate the **calendar time** needed to complete **each task**, the **effort required** and **who will work on the tasks that have been identified.**
- You also have to estimate the **resources** needed to complete each task, such as the **disk space** required on **a server**, the **time required on specialized hardware**, such as a **simulator**, and **what the travel budget will be.**

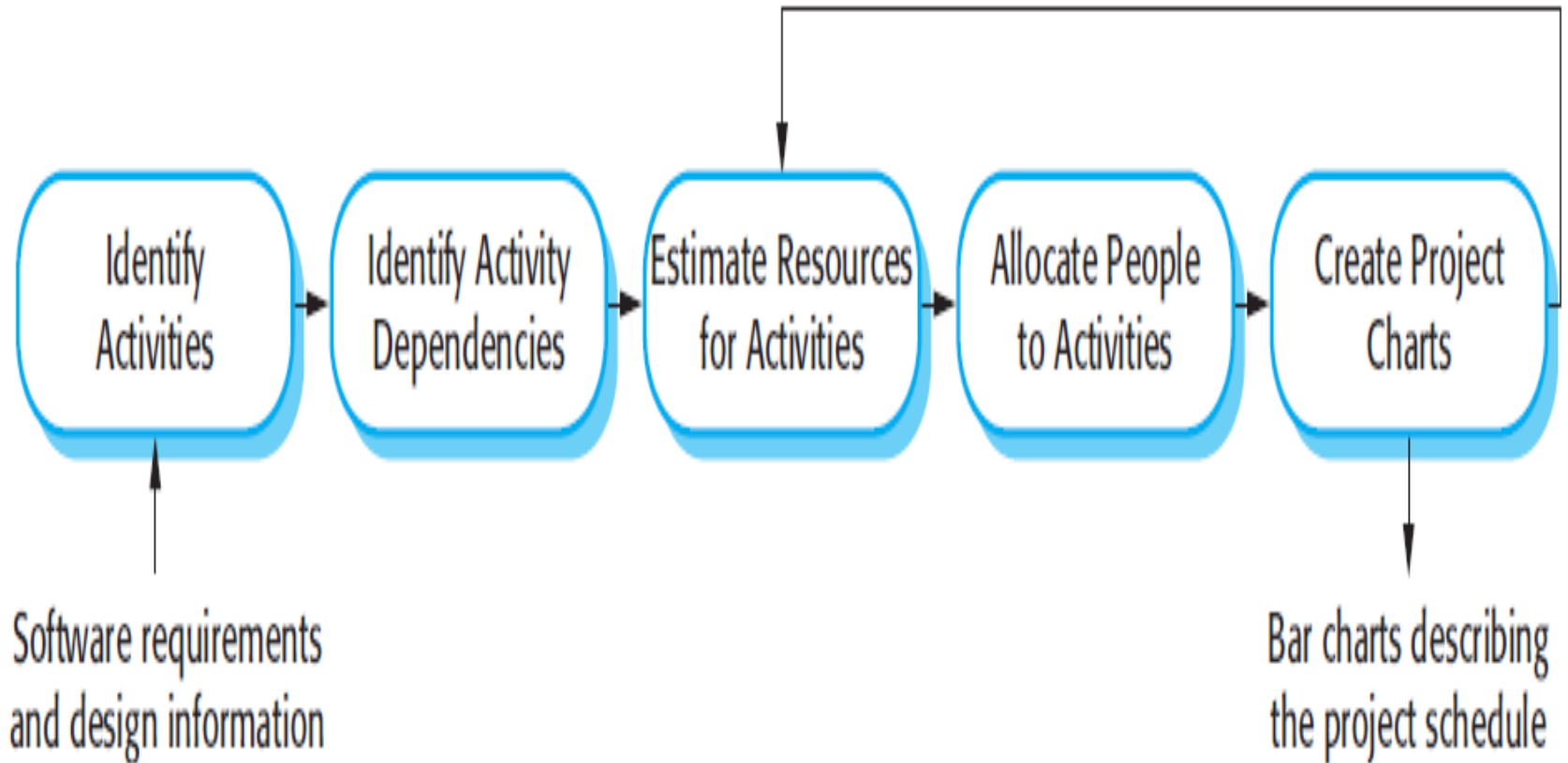
Project scheduling activities

- **Split** project into **tasks** and **estimate time** and **resources** required to complete each task.
- **Organize tasks concurrently** to make **optimal use of workforce**.
- **Minimize task dependencies** to avoid **delays** caused by **one task waiting for another to complete**.
- Dependent on **project managers intuition** and **experience**.

Milestones and deliverables

- **Milestones** are **points in the schedule** against which you can assess progress, for example, the *handover of the system* for *testing*.
- **Deliverables** are **work products** that are delivered to the customer, e.g. *a requirements document for the system*.

The project scheduling process



Scheduling problems

- **Estimating the difficulty of problems and hence the cost of developing a solution is hard.**
- **Productivity is not proportional to the number of people working on a task.**
- **Adding people to a late project makes it later because of communication overheads.**
- **The unexpected always happens. Always allow contingency in planning.**

Schedule representation

- **Graphical notations** are normally used to illustrate the **project schedule**.
- These show the **project breakdown into tasks**. **Tasks** should **not be too small**. They should **take about a week or two**.
- **Bar charts** are the most **commonly used** representation for **project schedules**. They show the **schedule as activities or resources against time**.

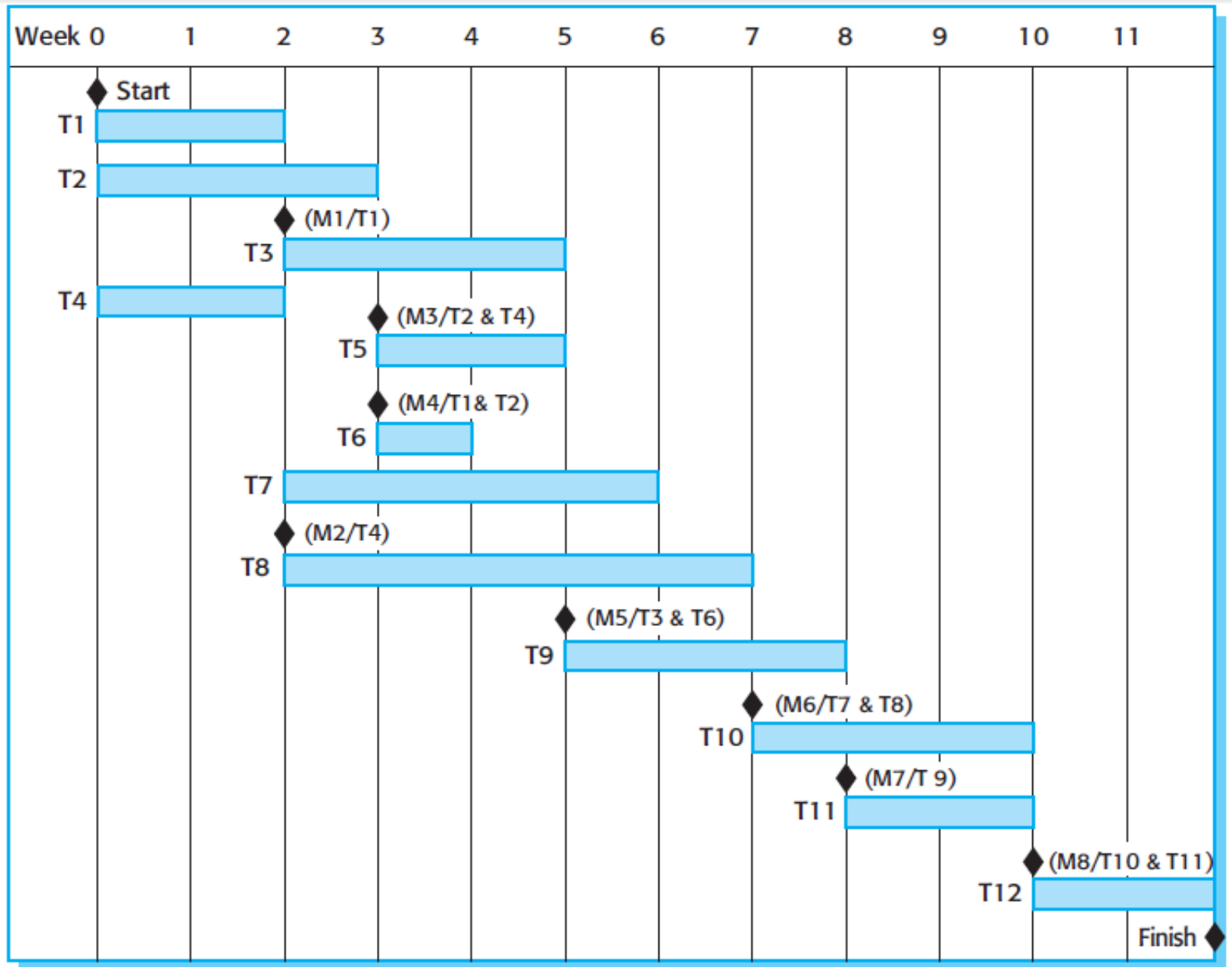
Tasks, durations, and dependencies

Task	Effort (person-days)	Duration (days)	Dependencies
T1	15	10	
T2	8	15	
T3	20	15	T1 (M1)
T4	5	10	
T5	5	10	T2, T4 (M3)
T6	10	5	T1, T2 (M4)
T7	25	20	T1 (M1)
T8	75	25	T4 (M2)
T9	10	15	T3, T6 (M5)
T10	20	15	T7, T8 (M6)
T11	10	10	T9 (M7)
T12	20	10	T10, T11 (M8)

- A **Man-day or Person-day** Simply means **8 hours of work for 1 person.**

- 1) Estimate how long it will take to perform each task
- 2) Add them up
- 3) divide by 8= Man-days or Person-days

Activity bar chart



Staff allocation chart

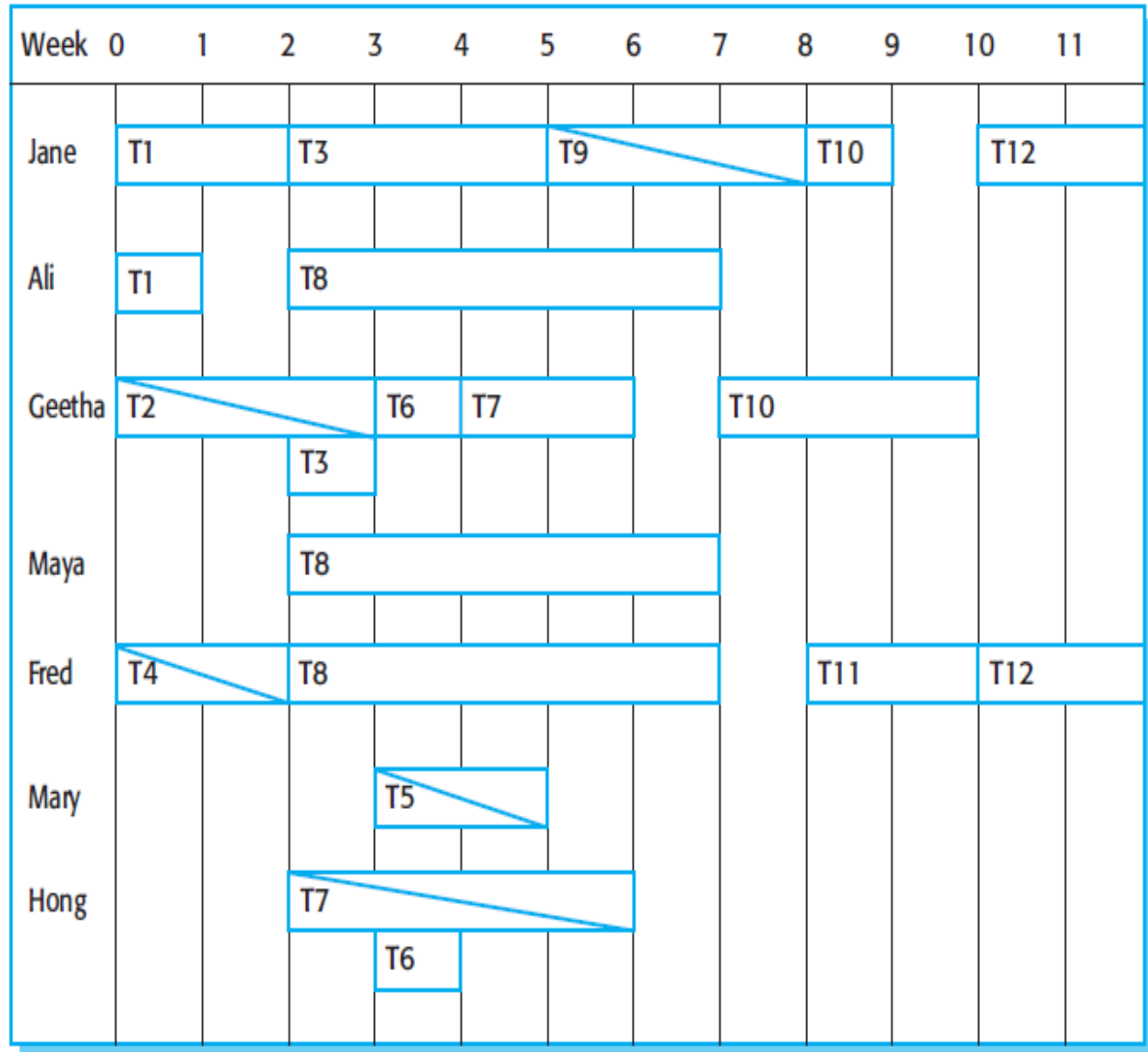


Figure 23.7 Staff allocation chart

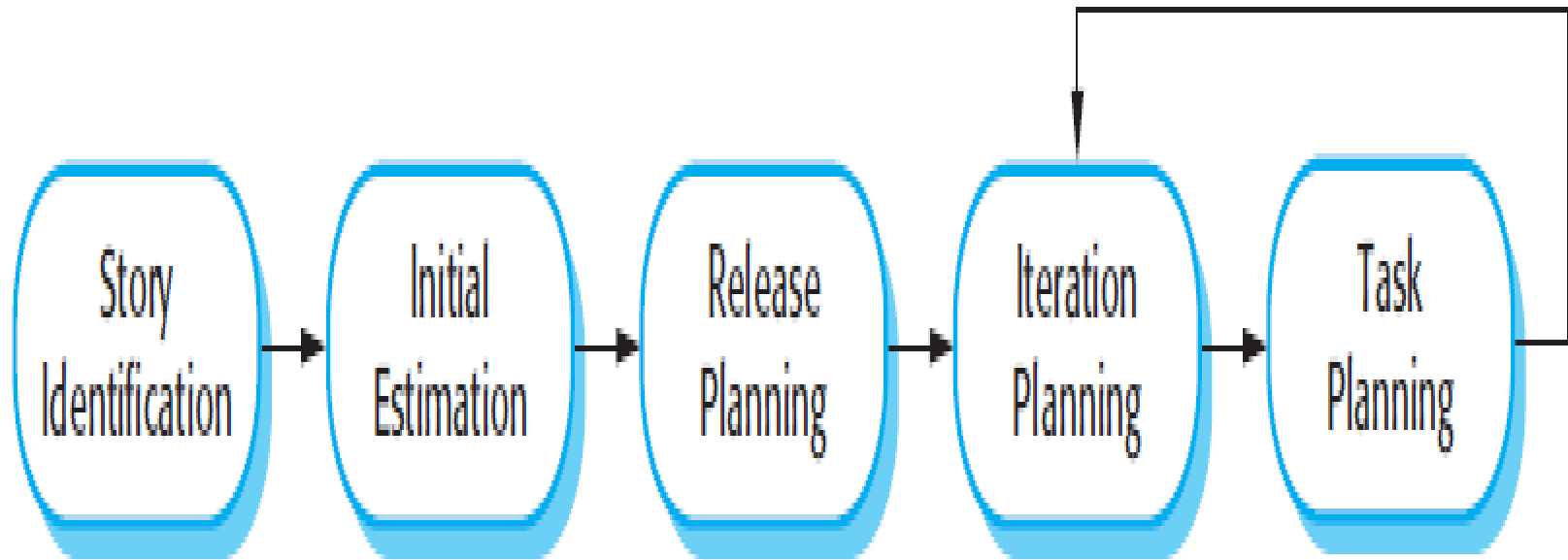
Agile planning

- Agile methods of software development are **iterative** approaches where the **software is developed and delivered to customers in increments.**
- Unlike plan-driven approaches, the functionality of these increments is **not planned in advance but is decided during the development.**
 - The decision on **what to include** in an increment depends ***on progress*** and on the ***customer's priorities.***
- The customer's priorities and requirements change so it makes sense to have a **flexible plan** that **can accommodate these changes.**

Agile planning stages

- **Release planning**, which looks ahead for *several months* and **decides on the features** that should be included in a **release of a system**.
- **Iteration planning**, which has a *shorter term outlook*, and focuses on planning the *next increment of a system*. This is typically **2-4 weeks of work for the team**.

Planning in XP



Story-based planning

- The **system specification** in XP is based on **user stories** that reflect the **features that should be included in the system**.
- The **project team** *read* and *discuss* the stories and *rank* them in order of the *amount of time* they think it will take to *implement the story*.
- **Release planning** involves *selecting* and *refining the stories* that will reflect the *features* to be implemented in a release of a system and the *order* in which the stories should be implemented.
- Stories to be implemented in *each iteration are chosen*, with the *number of stories* reflecting the *time to deliver an iteration (usually 2 or 3 weeks)*.

Estimation techniques

- 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 approaches

- 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*.

Algorithmic cost modelling

- Cost is estimated as a mathematical function of ***product***, ***project*** and ***process attributes*** whose values are estimated by ***project managers***:
 - **Effort = $A \times \text{Size}^B \times M$**
 - **A** is an ***organisation-dependent constant***, **B** reflects the ***disproportionate effort for large projects*** and **M** is a ***multiplier reflecting product, process and people attributes***.
 - **SIZE** may be ***code size*** or ***function points***
 - **B** usually lies between **1 to 1.5**
- 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.

Estimation accuracy

- The *size of a software system* can only be *known accurately when it is finished*.
- Several factors influence the final size
 - *Use of COTS and components;*
 - *Programming language;*
 - *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**.

Estimate uncertainty

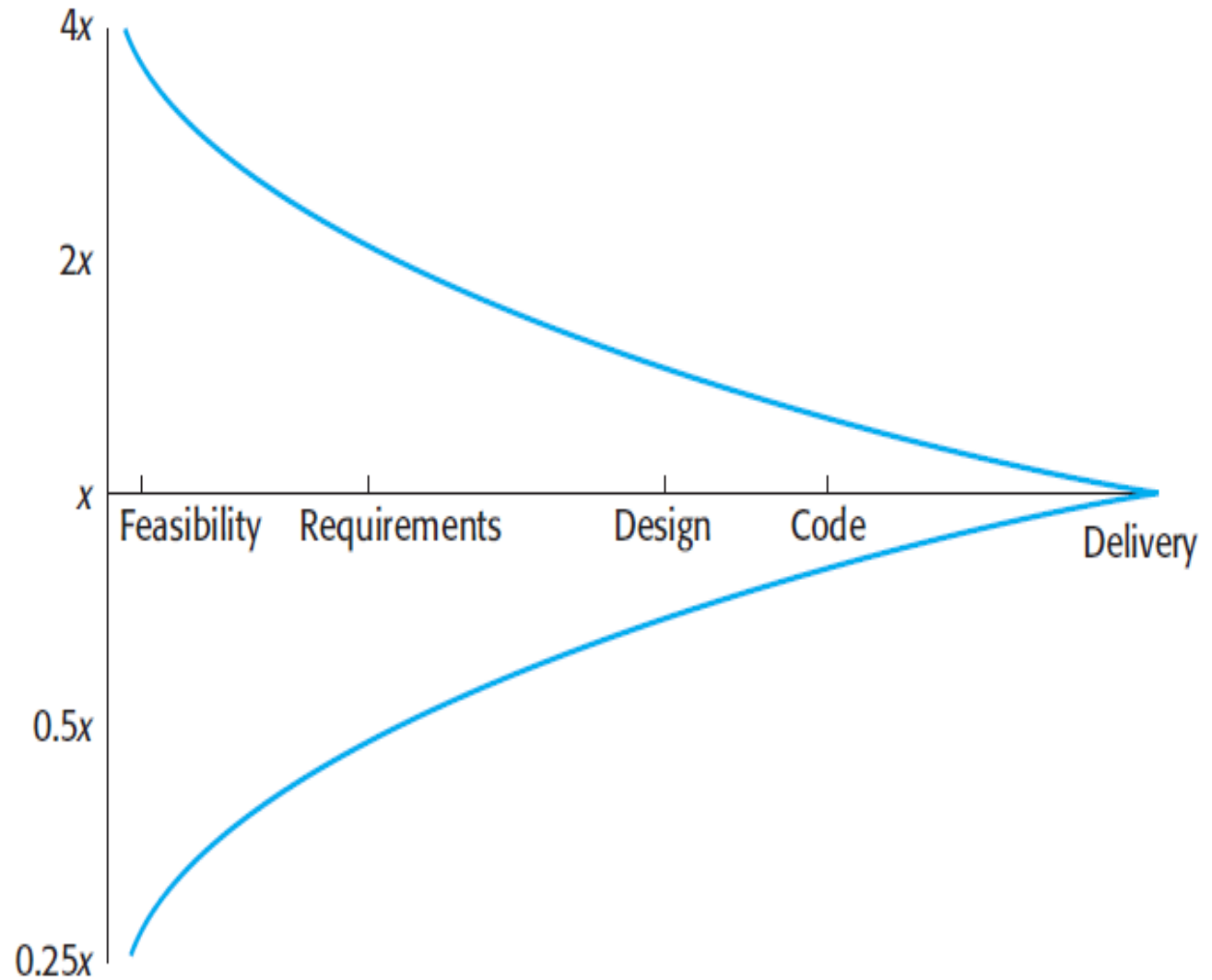


Figure 23.9
Estimate uncertainty

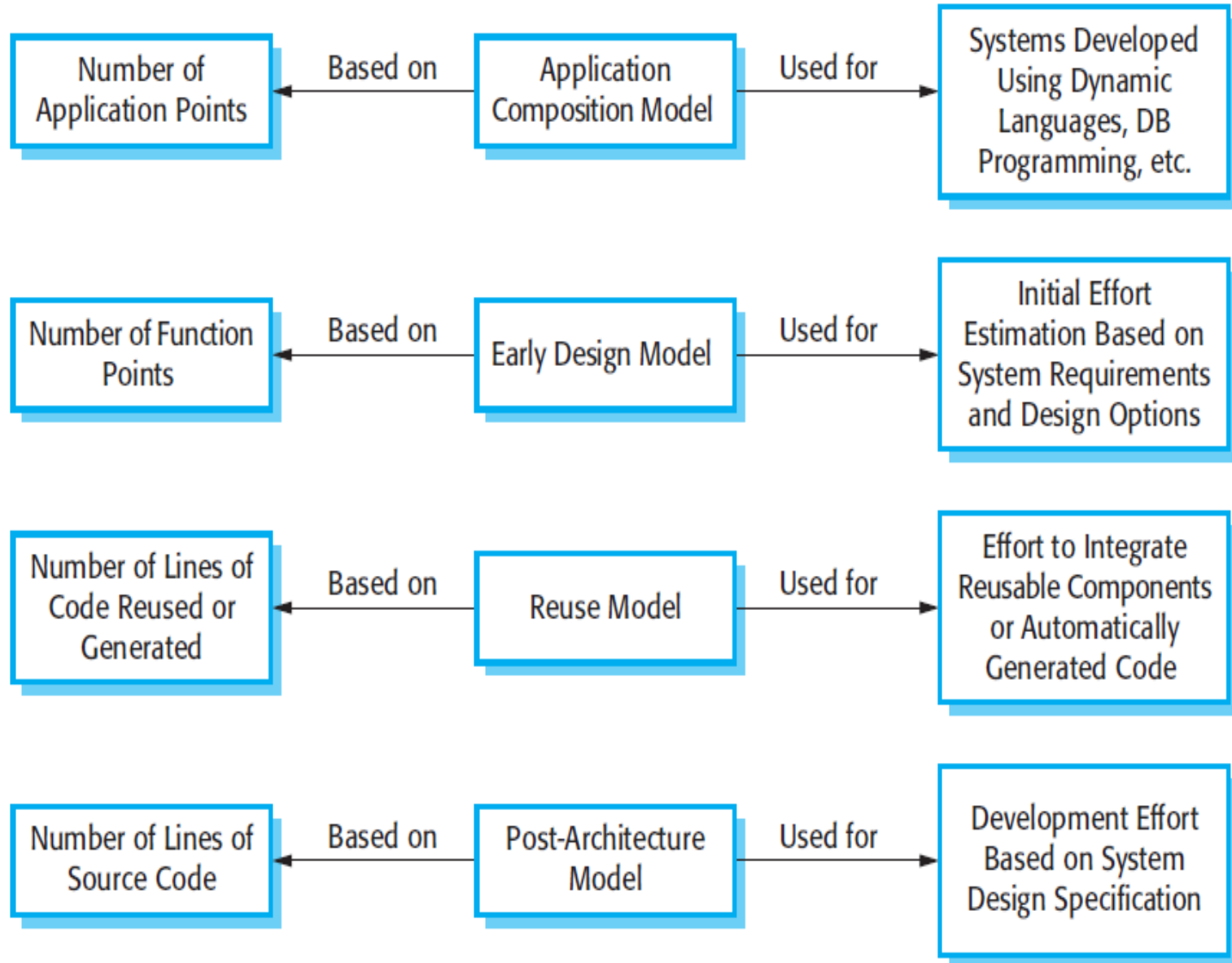
The COCOMO 2 model

- An *empirical model* based on *project experience*.
- **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 models

- 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.*

COCOMO estimation models



Project duration and staffing

- As well as **effort estimation**, managers *must estimate* the **calendar time required to complete a project** and **when staff will be required**.
- Calendar time can be estimated using a COCOMO 2 formula
 - $TDEV = 3 \times (PM)^{(0.33+0.2*(B-1.01))}$
 - **PM** is the *effort computation* and **B** is the *exponent computed* as discussed above (B is 1 for the early prototyping model). This computation predicts the nominal schedule for the project.
- The *time required* is independent of the *number of people working on the project*.

Staffing requirements

- Staff required **can't** be computed by dividing the **development time** by the required schedule.
- The **number of people working on a project** varies **depending on the phase of the project**.
- The **more people who work on the project**, the **more total effort** is usually required.
- A **very rapid build-up of people** often correlates with **schedule slippage**.

Review Questions

- What is significance of plan driven development? [3M]
- List the sections included in project plan document with a brief note. [6M]
- Illustrate a project planning process using an UML activity diagram reflecting the workflow. [6M]
- List the supplementary plans with a brief note carried out along with the project plan.[4M]
- Define 'scheduling' in a plan driven project. What are the time bounds suggested for breaking down the total work involved into separate tasks? [5M]
- Describe the factors affecting 'Software Pricing'. [6M]
- Show the scheduling process diagram reflecting activities and dependencies with a brief note on individual component. [5M]

- What are the popular 'Agile planning' approaches? Identify the two stages commonly used in these approaches for planning? Show the same using an UML diagram. [6M]
- Using algorithmic cost modeling formula show the efforts put-in to predict project costs.[5M]
- Which software development model used as a base for COCOMO-II cost estimation? Narrate all sub-models of COCOMO-II that produce increasingly detailed estimates for a project cost estimation. [8M]
- Derive the 'Calendar Time' required for completing a project with reference to the COCOMO model time estimation.[assume suitable data wherever necessary] [7M]

- For the following project schedule draw the 'Activity Bar-chart'. [7M]

Task	Duration	Dependency
T1	5	
T2	7	
T3	10	T1(M1)
T4	5	
T5	10	T2,T4(M3)
T6	12	T4(M2)
T7	7	T5(M4)
T8	15	T6(M5)
T9	8	T7(M6)
T10	10	T7,T9(M7)