Chapter 4
Software Project Planning and Tracking

© Pearson Education Limited 2005

Topics
- Project plan development
- Project scheduling
- Project budget estimation
- Tracking project progress
  - Tracking the schedule
  - Tracking the budget

The theme
- Management aspect of the software engineering:
  - software project planning and tracking
  - software process management (Chapter 5)
- Software project planning and tracking is an ongoing activity of estimating how much time, money, effort and resources need to be expended for the project. It includes:
  - not only scheduling, budgeting and tracking,
  - but also multi-project issues of
    - risk analysis,
    - quality assurance,
    - people management and
    - configuration management.
- Software project planning continues from strategic planning and business modelling.

Planning documents
- **Budget/schedule** (cost/time) plan is but one of several planning documents
- Other documents may include
  - quality plan,
  - test plan,
  - people development plan,
  - configuration management plan, etc.
- These may be separate documents or parts of a comprehensive project plan document

Sections in a typical project plan
- **Introduction** (objectives, constraints)
- **Project organization**
- **Risk analysis**
- **Hardware and software resource requirements**
- **Work breakdown** (activities, milestones and deliverables)
- **Project schedule** (time and resources)
- **Monitoring and reporting mechanisms**
Steps in project planning

1. Define objectives, scope and constraints
2. Define deliverables - software products or services that meet project objectives
3. Define milestones, phases and tasks
   - milestone - a significant event in the project (can represent the completion of a deliverable or a phase)
   - task - activity with clear beginning and end
4. Estimate resource needs
   - Resources are people, equipment, materials, supplies, software, hardware, etc. required by project tasks
5. Establish the project team
6. Determine quality standards
   - quality product is a product that satisfies the user
7. Determine project risks (adverse events)
   - risk avoidance strategies and contingency plans
8. Establish team communications
   - consider technology solutions (workgroup, time management, etc.)
9. Define the schedule
   - includes also allocation of resources to tasks
10. Define the budget
    - estimation of costs required by resources needed for tasks
11. Write project plan document
12. Track project progress
    - and revise project plan

Project scheduling

- Building time schedules assumes that the work breakdown structure and the task list are known
  - Tasks are activities with clear start and end days, ideally of between one day and two weeks duration
- Project scheduling is a moving target
  - Critical path is any sequence of linked tasks in the project that must be done on time for the project to finish by the deadline
  - Slippage (delay) in any task on a critical path will result in the project missing the deadline
  - Tasks that are not on a critical path can afford delays (they have slack time)

Tasks, milestones and deliverables

- Tasks can be hierarchically organized in multiple levels of subtasks
  - Each task is assigned duration (in hours, days, or weeks)
  - Task at the top of hierarchy is known as summary task
  - Subtask is any task that is a part of the summary task or a part of any higher-level subtask
  - Recurring task is a task that repeats at regular time intervals (e.g. every day, once per week)
- Schedule can be constructed
  - forward (from the project's start date) or
  - backward (from the project's end date)
- Schedule reflects the project calendar
  - Milestone is a task that results in a significant event or outcome
  - it is customary to create milestones as separate tasks with zero duration
  - Milestone can represent a deliverable

Creating task list

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>9th Apr 2014</th>
<th>12th Apr 2014</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task1</td>
<td>1 day</td>
<td>Mon 05:04:04</td>
<td>05-04 01</td>
<td>05-04 04</td>
</tr>
<tr>
<td>Task2</td>
<td>2 days</td>
<td>Mon 05:04:04</td>
<td>05-04 01</td>
<td>05-04 04</td>
</tr>
<tr>
<td>Task3</td>
<td>2 days</td>
<td>Mon 05:04:04</td>
<td>05-04 01</td>
<td>05-04 04</td>
</tr>
<tr>
<td>Milestone1</td>
<td>0 days</td>
<td>Mon 05:04:04</td>
<td>05-04 01</td>
<td>05-04 04</td>
</tr>
<tr>
<td>RecurringTask1</td>
<td>2 hrs</td>
<td>Fri 05:04:04</td>
<td>05-04 01</td>
<td>05-04 04</td>
</tr>
<tr>
<td>RecurringTask2</td>
<td>2 hrs</td>
<td>Fri 05:04:04</td>
<td>05-04 01</td>
<td>05-04 04</td>
</tr>
<tr>
<td>RecurringTask3</td>
<td>2 hrs</td>
<td>Fri 05:04:04</td>
<td>05-04 01</td>
<td>05-04 04</td>
</tr>
<tr>
<td>SummaryTask</td>
<td>2 days</td>
<td>Mon 05:04:04</td>
<td>05-04 01</td>
<td>05-04 04</td>
</tr>
<tr>
<td>Subtask1</td>
<td>1 day</td>
<td>Mon 05:04:04</td>
<td>05-04 01</td>
<td>05-04 04</td>
</tr>
<tr>
<td>Subtask2</td>
<td>2 days</td>
<td>Mon 05:04:04</td>
<td>05-04 01</td>
<td>05-04 04</td>
</tr>
<tr>
<td>Subtask3</td>
<td>1 day</td>
<td>Mon 05:04:04</td>
<td>05-04 01</td>
<td>05-04 04</td>
</tr>
<tr>
<td>Milestone2</td>
<td>0 days</td>
<td>Mon 05:04:04</td>
<td>05-04 01</td>
<td>05-04 04</td>
</tr>
</tbody>
</table>

Task scheduling in bar chart

- Task dependencies occur between two or more linked tasks
  - Unlinked tasks are considered to have no dependencies so that they can be conducted in parallel
  - There are two main kinds of task dependencies:
    - task dependencies based on the work organization
    - task dependencies due to competing for the same resources (discussed a bit later)
  - Categories of predecessor-to-successor dependencies:
    - finish-to-start (FS) - successor task cannot start until predecessor task is finished
    - start-to-start (SS) - successor task cannot start until predecessor task is started
    - finish-to-finish (FF) - successor task cannot finish until predecessor task is finished
    - start-to-finish (SF) - successor task cannot finish until predecessor task is started
Other predecessor-to-successor dependencies

- Delay dependencies in which a lag time is introduced between the predecessor and successor tasks
  - For example, a lag time of three days is specified if the successor task can start only after three days of completion of the predecessor task
- Overlap dependencies in which a lead time exists between the predecessor and successor tasks
  - This allows the successor task to start before the predecessor task is finished
- Lag and lead times can be specified
  - in absolute values (e.g. +3 days for lag time or -2 days for lead time)
  - as percentages of the predecessor task duration

Task scheduling in Gantt chart

- Delay dependencies
- Overlap dependencies

Scheduling constraints

- As Soon As Possible (ASAP)
  - task will be scheduled as soon as possible subject to dependencies on predecessor tasks and subject to availability of resources
- As Late As Possible (ALAP)
  - allows the task to obtain maximum input from earlier tasks before it starts
  - allows establishing how much a task can be delayed without delaying the project finish date (i.e. without extending the project's critical path)
- Inflexible constraints
  - Must Finish On (MFO)
  - Must Start On (MSO)
  - Finish No Earlier Than (FNET) – inflexible for projects scheduled from the finish date
  - Finish No Later Than (FNLT) – inflexible for projects scheduled from the start date
  - Start No Earlier Than (SNET) – inflexible for projects scheduled from the finish date
  - Start No Later Than (SNLT) – inflexible for projects scheduled from the start date

Resources and resource calendars

- Resources can be broadly classified into
  - work resources – people and equipment, including hardware and software
  - material resources – consumables and supplies
- Resource definition includes
  - giving the name to a resource,
  - specifying its type, and
  - indicating the number of resource units available for the work resource
- Tracking function
  - track the amount of work performed by people and equipment
  - monitor the use of materials

Effort-driven scheduling in bar chart

- Assignment of resources to tasks is based on the maximum number of working time that a resource can offer
  - time is determined from maximum available number of resource units and any constraints in the resource calendar
  - if the work assigned to the resource exceeds the resource daily availability, then such resource is over allocated
- The approach to scheduling where the duration of tasks can change (up or down) as resources are added or removed from tasks is known as effort-driven scheduling
  - In this approach, the effort required to complete a task remains unchanged, but the duration of the task can change.
Effort-driven scheduling in Gantt chart

Resource underallocation and overallocation

Change causing overallocation

Resource graph for overallocated resource

Project budget estimation

Methods of budget estimation

- Expert judgment
- Estimation by analogy
- Schedule-driven budget estimation
  - bottom-up methods, which consider tasks and resources allocated in the project schedule prepared earlier
- Algorithmic budget estimation
  - some estimated software size metric is used to derive the cost
Schedule-driven budget estimation

Schedule-driven budget estimation assumes that there exists a prior schedule with resources allocated to tasks:
- this is called the baseline schedule or plan
- project management tool can maintain multiple baselines and separate budgets for each baseline.

- The devil is in the detail:
  - resources can accrue costs at various times during the task duration
  - rates and fixed costs can differ in different time periods
  - there may be some legal rules affecting pay rates and costs
  - accounting principles can affect calculation, etc.

Assigning rates and other costs to resources

- standard rates
- overtime rates
- per-use rates (set cost for the use of a resource)

Cost accrual methods

- Accrual can be:
  - at the start of a task,
  - at the end of a task, or
  - prorated (as worked).

Task costs

- Effort can be expressed in:
  - lines of code (LOC)
  - function points (FP)
  - object points (OP)

Algorithmic budget estimation

- Algorithmic models use empirically derived formulas to estimate cost of human resources (effort) as a function of the project size.
  - The size can be expressed in:
    - the number of thousands (K) of non-comment source statements in the program (KLOC)
    - function points (FP)
    - object points (OP)
  - calculation is adjusted to remove reused object points.
Nominal form of algorithmic estimation

\[ \text{effort} = c \times \text{size}^k \]

- The **exponent** \( k \) reflects the complexity of the problem and the non-proportional increase of effort in larger projects (typically in the range from 1 to 1.5).
- The **multiplier** \( c \) is another scaling factor of project difficulty based on assessments of attributes such as required product reliability, skills of the team, available software tools, etc.
- COCOMO (Constructive Cost Model)
  - COCOMO 81
  - COCOMO II

COCOMO 81

- **Basic (or organic) COCOMO:**
  \[ \text{effort} = 3.2 \times \text{size}^{1.05} \]
- **Intermediate (or semidetached) COCOMO** – this mode includes a set of cost drivers to scale the nominal development effort based on assessments of product, hardware, personnel, and project attributes:
  \[ \text{effort} = 3.0 \times \text{size}^{1.12} \]
- **Advanced (or embedded) COCOMO** – this mode builds up on the intermediate version but uses cost drivers to assess their impact on each phase of the development lifecycle:
  \[ \text{effort} = 2.8 \times \text{size}^{1.20} \]

For example, using the basic COCOMO and assuming that the system will consist of 10,000 lines of code (10 KLOCs), the estimated effort expressed in person-months is:

\[ \text{effort} = 3.2 \times 100^{1.05} = 3.2 \times 125.9 \approx 403 \text{ person-months} \]

Cost drivers

- "product complexity" is a factor of 2.36 (1.65/0.70)
- "analyst capability" is a factor of 2.06 (1.46/0.71)
- "programmer capability" is a factor of 2.03 (1.42/0.70)
- "required software reliability" is a factor of 1.87 (1.40/0.75)
- "execution time constraints" is a factor of 1.66 (1.68/1.00)
- "application experience" is a factor of 1.57 (1.29/0.82)
- "use of modern programming practices" is a factor of 1.51 (1.24/0.82)
- "programming language experience" is a factor of 1.20 (1.14/0.95)

COCOMO II

- COCOMO II consists of three different models corresponding to three successive phases of the project:
  - **application composition model** – used in early analysis phase or during early prototyping
  - **early design model** – used after requirements analysis is completed
  - **post-architecture model** – used after the software architecture design is known

Application composition model

- Uses weighted object points (OPs) for sizing information
  - OPs are counts of anticipated user interface screens, software reports, and software components
  - the OP count is reduced by the percentage of object points obtained by reuse, so that only new object points (NOPs) are considered for effort estimation.
- The formula for NOPs is:
  \[ \text{NOP} = \text{OP} \times \left[ \frac{100 - \%\text{reuse}}{100} \right] \]
- Uses only two **weighting factors** to determine the development productivity level (capability)
  - the developers’ experience/capability and
  - the organization’s maturity/capability
- The two factors are combined to determine five productivity levels (next slide)

Productivity metrics in application composition model

- Associated with the five levels are productivity "cost drivers" expressed in terms of NOPs that the project team is capable of producing per month

<table>
<thead>
<tr>
<th>Combined developers’ experience/capability and organization’s maturity/capability</th>
<th>Productivity in NOP/month</th>
</tr>
</thead>
<tbody>
<tr>
<td>very low</td>
<td>4</td>
</tr>
<tr>
<td>low</td>
<td>7</td>
</tr>
<tr>
<td>nominal</td>
<td>13</td>
</tr>
<tr>
<td>high</td>
<td>25</td>
</tr>
<tr>
<td>very high</td>
<td>50</td>
</tr>
</tbody>
</table>
Calculating effort in application composition model

- Once the productivity metric is chosen, the formula to calculate the effort in COCOMO II application composition model is:
  \[ \text{effort} = \frac{\text{NOP}}{\text{productivity}} \]
- For example, if
  - the count of new object points is 500 and
  - the productivity is 4,
  - then the effort (in person-months) will be estimated at 125.
- By contrast, if the productivity is 50, then the effort will be only 10 person-months.

Early design model

\[ \text{effort} = c \times \text{size}^k \times m + \text{autoeffort} \]
- constant coefficient \( c \) is set to 2.5
- exponent \( k \) varies between 1.01 and 1.26
- multiplier \( m \) expresses the influence of (seven) cost drivers on the estimated effort
  - each driver is assigned a value (the value of 1 means that the cost driver does not affect the estimation)
  - the value of \( m \) is the result of the multiplication of the seven numbers
- autoeffort – effort put by developers in the automatic code generation and the integration of this code with the manually created programs

\[ \text{effort} = 2.5 \times 100^{1.15} \times 1.3 + 35 \]
\[ \approx 678 \text{ person-months} \]

Cost drivers in post-architecture model

<table>
<thead>
<tr>
<th>Cost driver category</th>
<th>Cost driver code</th>
<th>Cost driver name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product attributes</td>
<td>RELY</td>
<td>Required system reliability</td>
</tr>
<tr>
<td></td>
<td>CPLX</td>
<td>Complexity of system modules</td>
</tr>
<tr>
<td></td>
<td>DOCU</td>
<td>Extent of documentation required</td>
</tr>
<tr>
<td></td>
<td>DATA</td>
<td>Size of database used</td>
</tr>
<tr>
<td></td>
<td>RUSR</td>
<td>Required percentage of reusable components</td>
</tr>
<tr>
<td>Computer attributes</td>
<td>TIME</td>
<td>Execution time constraints</td>
</tr>
<tr>
<td></td>
<td>PVOL</td>
<td>Volatility of development platform</td>
</tr>
<tr>
<td></td>
<td>STOR</td>
<td>Memory constraints</td>
</tr>
<tr>
<td>Personnel attributes</td>
<td>ACAP</td>
<td>Capability of project analysis</td>
</tr>
<tr>
<td></td>
<td>PCON</td>
<td>Personnel competency</td>
</tr>
<tr>
<td></td>
<td>PEXP</td>
<td>Programmer experience in project domain</td>
</tr>
<tr>
<td></td>
<td>PCAP</td>
<td>Programmer capability</td>
</tr>
<tr>
<td></td>
<td>AEXP</td>
<td>Analyst experience in project domain</td>
</tr>
<tr>
<td></td>
<td>LTEX</td>
<td>Language and tool experience</td>
</tr>
<tr>
<td>Project attributes</td>
<td>TOOL</td>
<td>Use of software tools</td>
</tr>
<tr>
<td></td>
<td>TEDS</td>
<td>Development schedule compression</td>
</tr>
<tr>
<td></td>
<td>SITE</td>
<td>Extent of multi-site working and quality of communications</td>
</tr>
</tbody>
</table>

Scale factors in post-architecture model

<table>
<thead>
<tr>
<th>Scale factor</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precedentedness</td>
<td>To capture the prior experience of the organization with similar projects.</td>
</tr>
<tr>
<td>Development flexibility</td>
<td>To capture the extent to which the project client intervenes in the development. A high flexibility means that the client is not involved or sets only generic goals.</td>
</tr>
<tr>
<td>Architecture/risk resolution</td>
<td>To capture the extent to which risk management is done on the project.</td>
</tr>
<tr>
<td>Team cohesion</td>
<td>To capture the extent to which the development team is integrated and effective.</td>
</tr>
<tr>
<td>Process maturity</td>
<td>To capture the software engineering maturity of the organization in conducting projects.</td>
</tr>
</tbody>
</table>

Tracking project progress

- Comparing the baseline and the actual:
  - the baseline schedule versus the actual schedule
  - the baseline budget versus the actual budget
- Interim plans may also be created as checkpoints of project progress
- Information (possibly conflicting) comes from two sources:
  - changes to the planned schedule
  - accounting system
- Also, time tracking and cost tracking can bring up conflicting information:
  - project can be behind schedule but on or under budget
  - it can also be on or ahead of schedule but over budget
Tracking the schedule

- The lower bar represents the baseline task
- The upper bar represents the actual task colored to indicate if they are completed, not started yet, or completed in some percentage
- Variance is the difference between the baseline and the actual

Tracking Gantt chart

Tracking the work on task by task basis

Tracking the work on resource by resource basis

Tracking the budget

- Actual costs from schedule = budget tracking as a direct continuation of schedule tracking
  - tracking actual costs on tasks completed, or in progress, by applying resource rates, accrual methods, and any fixed costs
- Actual costs from accounting
  - because actual costs reported by the accounting system may be different from the costs calculated from actual work performed
  - in some cases, the accounting system may be the only source of information of actual cost
- Earned value analysis – one of the most useful methods that succinctly informs if the project is on budget
**Actual costs from accounting**

- Calculating actual costs from the schedule rarely gives true and final account of costs committed:
  - schedules are unlikely to include all resources consumed by tasks or even to include all tasks
  - the sluggishness of the accounting system can distort the real story

- Some possible difficulties:
  - fixed cost for the task failed to be entered during scheduling
  - per-use cost charged for a work resource in addition to normal rate-based fees

- Frequently, budget changes reported by the accounting system relate to non-human resources

**Budget changes reported by accounting system**

**Earned value**

- **Earned value analysis** = performance management + management by objectives
  - quantitative technique of determining if the project is on budget
    - uses the baseline estimate of the schedule and the actual progress to date to determine the completeness status ("completeness health") of the project
    - conducted for individual tasks and for the whole project
    - can also be taken down to the level of resources within each task
    - can only be conducted if resources and their rate/costs have been assigned to tasks

- **Earned value analysis** is a quantitative technique of determining if the project is on budget

**Summary**

- **Software project planning and tracking** is an ongoing activity of estimating how much time, money, effort and resources need to be expended for the project
- **Scheduling** is about estimation of time needed for the development and about allocation of resources to tasks
- **Budgeting** is about estimation of costs required by resources needed for tasks
- The approach to scheduling where the duration of tasks can change (up or down) as resources are added or removed from tasks is known as effort-driven scheduling
- Schedule-driven budget estimation assumes that there exists a baseline schedule with resources already assigned to tasks
- Algorithms budget estimation takes some measure (metric) of the size of the problem and applies it to an empirically obtained algorithm to arrive at the budget
- Tracking project progress involves comparisons of: (1) the baseline schedule versus the actual schedule, and (2) the baseline budget versus the actual budget
- **Earned value analysis** is a quantitative technique of determining if the project is on budget