The Work Breakdown Structure and Project Estimation
Learning Objectives

- Develop a work breakdown structure.
- Describe the difference between a deliverable and a milestone.
- Describe and apply several project estimation methods. These include the Delphi technique, time boxing, top-down estimation, and bottom-up estimation.
- Describe and apply several software engineering estimation approaches. These include lines of code (LOC), function point analysis, COCOMO, and heuristics.
Project Time Management

**PMBOK®**

- **Define Activities**
  - Identifying what activities must be completed to produce the project scope deliverables

- **Sequence Activities**
  - Determining whether activities can be completed sequentially or in parallel and any dependencies that may exist among them

- **Estimate Activity Resources**
  - Identifying the type of resources (people, technology, facilities, etc.) and the quantity of resources needed to carry out project activities

- **Estimate Activity Durations**
  - Estimating the time to complete each activity

- **Develop Schedule**
  - Based on the availability of resources, the activities, their sequence, and time estimates, a schedule for the entire budget can be developed

- **Control Schedule**
  - Ensuring that proper processes and procedures are in place in order to control changes to the project schedule
Work Breakdown Structure (WBS)

- The WBS represents a logical decomposition of the work to be performed and focuses on how the product, service, or result is naturally subdivided. It is an outline of what work is to be performed

  - PMBOK Guide® (17).
Work Package

- Project
  - Phase
    - Deliverable
      - Activity/Task
        - Milestone—Deliverable completion
        - Milestone—Phase completion
Deliverables versus Milestones

- **Deliverables**
  - Tangible, verifiable work products
    - Reports, presentations, prototypes, etc.

- **Milestones**
  - Significant events or achievements
  - Acceptance of deliverables or phase completion
  - Cruxes (proof of concepts)
  - Quality control
  - Keeps team focused
Developing the WBS

- A work package is developed for each of the phases and deliverables defined in the Deliverable Structure Chart (DSC)
Deliverable: Test Results Report

- **Logical Activities:**
  1. Review the test plan with the client so that key stakeholders are clear as to what will be tested, how the tests will be conducted, and when the tests will be carried out.
  2. Carry out the tests as outlined in the plan.
  3. Once the test results are collected, we need to analyze them.
  4. The results should be summarized in the form of a report and presentation to the client.
  5. If all goes well, the client will sign-off or approve the test results and then we can move on to the implementation phase of the project. If not, then we need to address and fix any problems.

What are the deliverables? Milestones?
Example Work Breakdown Schedule

-0.0 EC Bank Project
  +1.0 Conceptualize & initialize project
  +2.0 Develop charter & plan
  +3.0 Analysis
  +4.0 Design
  +5.0 Construction

-6.0 Testing
  +6.1 Test plan

-6.2 Test results report
  6.2.1 Review test plan with client
  6.2.2 Carry out test plan
  6.2.3 Analyze results
  6.2.4 Prepare test results report and presentation
  6.2.5 Present test results to client
  6.2.6 Address any software issues or problems
  6.2.7 **Milestone:** client signs off on test results

+6.3 **Milestone:** testing completed

+7.0 Implementation
+8.0 Close project
+9.0 Evaluate project success
The WBS Should Follow the Work Package Concept

-0.0 EC Bank Project
  +1.0 Conceptualize & initialize project
  +2.0 Develop charter & plan
  +3.0 Analysis
  +4.0 Design
  +5.0 Construction
-6.0 Testing
  +6.1 Test plan
  -6.2 Test results report
    6.2.1 Review test plan with client
    6.2.2 Carry out test plan
    6.2.3 Analyze results
    6.2.4 Prepare test results report and presentation
    6.2.5 Present test results to client
    6.2.6 Address any software issues or problems
    6.2.7 Milestone: client signs off on test results
  +6.3 Milestone: testing completed
+7.0 Implementation
+8.0 Close project
+9.0 Evaluate project success
The WBS...

- Should be “deliverable-oriented”
- Should support the project’s MOV
- Have enough detail to support planning and control
- Should involve those who will be doing the work
- Should include learning cycles and past lessons learned
Estimation Questions

What are you going to estimate?

Where do you start?

How do you estimate?
Estimation Techniques - Traditional Project Management Approaches

- Guesstimating
- Delphi Technique
- Time Boxing
- Top-Down
- Bottom-Up
- Analogous Estimates (Past experiences)
- Parametric Modeling (Statistical)
Estimation by guessing or just picking numbers out of the air is not the best way to derive a project’s schedule and budget. Unfortunately, many inexperienced project managers tend to guesstimate, or guess at the estimates, because it is quick and easy.
Delphi Technique

- Involves multiple, anonymous experts
- Each expert makes an estimate
- Estimates compared
  - If close, can be averaged
  - If not, do another iteration until consensus is reached
Time Boxing

- A “box” of time is allocated for a specific activity, task, or deliverable
- Can focus a team if used effectively
- Can demoralize a team if not used effectively
Top-Down

- Top & middle managers determine overall project schedule &/or cost
- Lower level managers are expected to breakdown schedule/budget estimates into specific activities (WBS)
Bottom-Up

- Schedules & budgets are constructed from WBS
- Starts with people who will be doing the work
- Schedules & budgets are the aggregate of detailed activities & costs
Analogous Estimates

- Similar to Top-Down approach
- Use information from previous, similar projects as a basis for estimation
Parametric Modeling

- Use project characteristics (parameters) in a mathematical model to estimate

- Example: $50/LOC based on:
  - Programming language
  - Level of expertise
  - Size & complexity
6.2 Test Results Report

<table>
<thead>
<tr>
<th>Activity</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2.1 Review test plan with client</td>
<td>1 day</td>
</tr>
<tr>
<td>6.2.2 Carry out test plan</td>
<td>5 days</td>
</tr>
<tr>
<td>6.2.3 Analyze results</td>
<td>2 days</td>
</tr>
<tr>
<td>6.2.4 Prepare test results report and presentation</td>
<td>3 days</td>
</tr>
<tr>
<td>6.2.5 Present test results to client</td>
<td>1 day</td>
</tr>
<tr>
<td>6.2.6 Address any software issues or problems</td>
<td>5 days</td>
</tr>
</tbody>
</table>

Estimates are made for each activity in the WBS

How did we come up with these estimates? Using a technique, or combination of techniques, with the exception of guestimating!
Estimating Techniques - Software Engineering Approaches

- Lines of Code (LOC)
- Function Points
- COCOMO
- Heuristics

Software engineering techniques focus on estimating the size of the system to be developed
Determinants of Estimating the Largest Deliverable of the Project – The Application System

Size

Application estimate

Complexity

Constraints & influencers
Function Point Analysis

- Allan Albrecht, IBM – 1979
- Synthetic metric
- Independent of the Technology
- IFPUG standards (www.ifpug.org)
- 5 Primary Elements
  - Inputs
  - Outputs
  - Inquiries
  - Logical Files
  - Interfaces
The Application Boundary for Function Point Analysis

Application boundary

External inputs

External outputs

External inquiries

Internal logical files (ILF)

External application

External inputs

External outputs

External inquiries

External interface files (ELF)
<table>
<thead>
<tr>
<th>Complexity</th>
<th>Low</th>
<th>Average</th>
<th>High</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Internal Logical Files (ILF)</strong></td>
<td>[3 \times 7 = 21]</td>
<td>[2 \times 10 = 20]</td>
<td>[1 \times 15 = 15]</td>
<td>56</td>
</tr>
<tr>
<td><strong>External Interface Files (EIF)</strong></td>
<td>[\text{<em><strong>} \times 5 = \text{</strong></em>}]</td>
<td>[2 \times 7 = 14]</td>
<td>[\text{<em><strong>} \times 10 = \text{</strong></em>}]</td>
<td>14</td>
</tr>
<tr>
<td><strong>External Input (EI)</strong></td>
<td>[3 \times 3 = 9]</td>
<td>[5 \times 4 = 20]</td>
<td>[4 \times 6 = 24]</td>
<td>53</td>
</tr>
<tr>
<td><strong>External Output (EO)</strong></td>
<td>[4 \times 4 = 16]</td>
<td>[2 \times 5 = 10]</td>
<td>[1 \times 7 = 7]</td>
<td>33</td>
</tr>
<tr>
<td><strong>External Inquiry (EQ)</strong></td>
<td>[2 \times 3 = 6]</td>
<td>[5 \times 4 = 20]</td>
<td>[3 \times 6 = 18]</td>
<td>44</td>
</tr>
<tr>
<td><strong>Total Unadjusted Function Points (UAF)</strong></td>
<td></td>
<td></td>
<td></td>
<td>200</td>
</tr>
<tr>
<td>General System Characteristic</td>
<td>Degree of Influence</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------------------</td>
<td>---------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Communications</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distributed Data Processing</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heavily Used Configuration</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transaction Rate</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>On-line Data Entry</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>End User Efficiency</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Online Update</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complex Processing</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reusability</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Installation Ease</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operational Ease</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiple Sites</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Facilitate Change</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total Degrees of Influence</strong></td>
<td>40</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Value Adjustment Factor $VAF = (TDI \times 0.01) + 0.65$

$$VAF = (40 \times 0.01) + 0.65 = 1.05$$

**Total Adjusted Function Points** $= FP = UAF \times VAF$

$$FP = 200 \times 1.05 = 210$$
<table>
<thead>
<tr>
<th>Language</th>
<th>Average Source LOC per Function Point</th>
<th>Average Source LOC for a 210 FP Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>38</td>
<td>7,980</td>
</tr>
<tr>
<td>Basic</td>
<td>107</td>
<td>22,470</td>
</tr>
<tr>
<td>C</td>
<td>128</td>
<td>26,880</td>
</tr>
<tr>
<td>C++</td>
<td>53</td>
<td>11,130</td>
</tr>
<tr>
<td>COBOL</td>
<td>107</td>
<td>22,470</td>
</tr>
<tr>
<td>Delphi</td>
<td>29</td>
<td>6,090</td>
</tr>
<tr>
<td>Java</td>
<td>53</td>
<td>11,130</td>
</tr>
<tr>
<td>Machine Language</td>
<td>640</td>
<td>134,440</td>
</tr>
<tr>
<td>Visual Basic 5</td>
<td>29</td>
<td>6,090</td>
</tr>
</tbody>
</table>

Source: http://www.spr.com/library/0langtbl.htm
COCOMO

- COnstructive COSt MOdel
- Developed by Barry Boehm,
- Has been extended to COCOMO II

http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html
COCOMO Models (Effort)

- **Organic – Routine**
  - Person Months = $2.4 \times KDSI^{1.05}$

- **Embedded – Challenging**
  - Person Months = $3.6 \times KDSI^{1.20}$

- **Semi-Detached – Middle**
  - Person Months = $3.0 \times KDSI^{1.12}$
COCOMO – Effort Example

**Semi-Detached**

10,600 Java LOC = 200 FP * 53

Person Months = 3.0 * KDSI^{1.12}

= 3.0 * (10.6)^{1.12}

= 42.21
COCOMO Models (Duration)

- Organic
  - Duration = 2.5 * Effort^{0.38}

- Semi-Detached
  - Duration = 2.5 * Effort^{0.35}

- Embedded
  - Duration = 2.5 * Effort^{0.32}
COCOMO Duration Example

Duration = 2.5 * Effort^{0.35}
          = 2.5 * (42.21)^{0.35}
          = 9.26 months

People Required = Effort / Duration
                 = \frac{42.21}{9.26}
                 = 4.55
Heuristics (Rules of Thumb)

When scheduling a software task:

1/3 – Planning
1/6 – Coding
1/4 – Component test and early system test
1/4 – System test, all components in hand
The seeds of major software disasters are usually sown in the first three months of commencing the software project. Hasty scheduling, irrational commitments, unprofessional estimating techniques, and carelessness of the project management function are the factors that tend to introduce terminal problems. Once a project blindly lurches forward toward an impossible delivery date, the rest of the disaster will occur almost inevitably.

T. Capers Jones, 1988 Page 120
Brooks’ Law

Adding manpower to a late software project makes it later.
The Man Month

Time versus number of workers perfectly partitionable task – i.e., No communication among them e.g., reaping wheat.

When a task that cannot be partitioned because of sequential constraints, the application of more effort has no effect on the schedule.
Adding People

- Increases the total effort necessary
  - The work & disruption of repartitioning
  - Training new people
  - Added intercommunication
What can cause inaccurate estimates?

- Scope changes
- Overlooked tasks
- Poor developer-user communication
- Poor understanding of project goals
- Insufficient analysis
- No (or poor) methodology
- Changes in team
- Red tape
- Lack of project control
- Not identifying or understanding impact of risks
Other Factors to Consider When Estimating

- Rate at which requirements may change
- Experience & capabilities of project team
- Process or methods used in development
- Specific activities to be performed
- Programming languages or development tools to be used
- Probable number of bugs or defects & removal methods
- Environment or ergonomics of work space
- Geographic separation of team across locations
- Schedule pressure placed on the team
How can estimates be improved?

- Experience!
  - Lessons learned
  - Best Practices
- Revision
- Monitor
- Focus on deliverables
- Control