Lecture 17: Chapter 26

- Estimation for Software Projects

*Slide Set to accompany*

*Software Engineering: A Practitioner’s Approach, 7/e*

*by Roger S. Pressman*


*For non-profit educational use only*

May be reproduced ONLY for student use at the university level when used in conjunction with *Software Engineering: A Practitioner's Approach, 7/e*. Any other reproduction or use is prohibited without the express written permission of the author.

All copyright information MUST appear if these slides are posted on a website for student use.
Software Project Planning

The overall goal of project planning is to establish a pragmatic strategy for controlling, tracking, and monitoring a complex technical project.

Why?

So the end result gets done on time, with quality!
Project Planning Task Set-I

- Establish project scope
- Determine feasibility
- Analyze risks
  - Risk analysis is considered in detail in Chapter 25.
- Define required resources
  - Determine require human resources
  - Define reusable software resources
  - Identify environmental resources
Project Planning Task Set-II

- **Estimate cost and effort**
  - Decompose the problem
  - Develop two or more estimates using size, function points, process tasks or use-cases
  - Reconcile the estimates

- **Develop a project schedule**
  - Scheduling is considered in detail in Chapter 27.
    - Establish a meaningful task set
    - Define a task network
    - Use scheduling tools to develop a timeline chart
    - Define schedule tracking mechanisms
Estimation

- Estimation of resources, cost, and schedule for a software engineering effort requires
  - experience
  - access to good historical information (metrics)
  - the courage to commit to quantitative predictions when qualitative information is all that exists

- Estimation carries inherent risk and this risk leads to uncertainty
Write it Down!

Project Scope
Estimates
Risks
Schedule
Control strategy

Software Project Plan
To Understand Scope ... 

- Understand the customers needs
- understand the business context
- understand the project boundaries
- understand the customer’s motivation
- understand the likely paths for change
- understand that ...

Even when you understand, nothing is guaranteed!
What is Scope?

- **Software scope** describes
  - the functions and features that are to be delivered to end-users
  - the data that are input and output
  - the “content” that is presented to users as a consequence of using the software
  - the performance, constraints, interfaces, and reliability that *bound* the system.

- Scope is defined using one of two techniques:
  - A narrative description of software scope is developed after communication with all stakeholders.
  - A set of use-cases is developed by end-users.
Resources

- People
  - Number
  - Skills
  - Location

- Environment
  - Software tools
  - Hardware
  - Network resources

- Project

- Reusable software
  - OTS components
  - New components
  - Full-experience components
  - Part-experience components
Project Estimation

- Project scope must be understood
- Elaboration (decomposition) is necessary
- Historical metrics are very helpful
- At least two different techniques should be used
- Uncertainty is inherent in the process
Estimation Techniques

- Wait till the end!!
- Past (similar) project experience
- Conventional estimation techniques
  - task breakdown and effort estimates
  - size (e.g., Function Points) estimates
- Empirical models
- Automated tools
Estimation Accuracy

- Predicated on …
  - the degree to which the planner has properly estimated the size of the product to be built
  - the ability to translate the size estimate into human effort, calendar time, and dollars (a function of the availability of reliable software metrics from past projects)
  - the degree to which the project plan reflects the abilities of the software team
  - the stability of product requirements and the environment that supports the software engineering effort.
Conventional Methods: LOC/FP Approach

- compute lines of code (LOC) using estimates of information domain values
- Computes function points (FP), information domain values
- use historical data to build estimates for the project
- Example: Computer-Aided Design application for mechanical components
  - Apply LOC and FP to this software
Example: LOC Approach

Average productivity for systems of this type = 620 LOC/pm.
Burdened labor rate =$8000 per month, the cost per line of code is approximately $13.

Based on the LOC estimate and the historical productivity data, the total estimated project cost is 33,200*620 = $431,000 and the estimated effort is 54 person-months.
Example: FP Approach

The estimated number of FP is derived:

$$FP_{\text{estimated}} = \text{count-total} \times \left[0.65 + 0.01 \times S(F_i)\right]$$

$$FP_{\text{estimated}} = 375$$

organizational average productivity = 6.5 FP/pm.

burdened labor rate = $8000 per month, approximately $1230/FP.

Based on the FP estimate and the historical productivity data, total estimated project cost is $461,000 and estimated effort is 58 person-months.
Process-Based Estimation

Obtained from “process framework”

Application functions

Framework activities

Effort required to accomplish each framework activity for each application function
Process-Based Estimation Example

<table>
<thead>
<tr>
<th>Activity</th>
<th>CC</th>
<th>Planning</th>
<th>Risk Analysis</th>
<th>Engineering</th>
<th>Construction Release</th>
<th>CE</th>
<th>Totals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UICF</td>
<td>0.50</td>
<td>2.50</td>
<td>0.40</td>
<td>5.00</td>
<td>n/a</td>
<td>8.40</td>
<td></td>
</tr>
<tr>
<td>2DGA</td>
<td>0.75</td>
<td>4.00</td>
<td>0.60</td>
<td>2.00</td>
<td>n/a</td>
<td>7.35</td>
<td></td>
</tr>
<tr>
<td>3DGA</td>
<td>0.50</td>
<td>4.00</td>
<td>1.00</td>
<td>3.00</td>
<td>n/a</td>
<td>8.50</td>
<td></td>
</tr>
<tr>
<td>CGDF</td>
<td>0.50</td>
<td>3.00</td>
<td>1.00</td>
<td>1.50</td>
<td>n/a</td>
<td>6.00</td>
<td></td>
</tr>
<tr>
<td>DSM</td>
<td>0.50</td>
<td>3.00</td>
<td>0.75</td>
<td>1.50</td>
<td>n/a</td>
<td>5.75</td>
<td></td>
</tr>
<tr>
<td>PCF</td>
<td>0.25</td>
<td>2.00</td>
<td>0.50</td>
<td>1.50</td>
<td>n/a</td>
<td>4.25</td>
<td></td>
</tr>
<tr>
<td>DAM</td>
<td>0.50</td>
<td>2.00</td>
<td>0.50</td>
<td>2.00</td>
<td>n/a</td>
<td>5.00</td>
<td></td>
</tr>
<tr>
<td>Totals</td>
<td>0.25</td>
<td>0.25</td>
<td>0.25</td>
<td>3.50</td>
<td>20.50</td>
<td>4.50</td>
<td>16.50</td>
</tr>
<tr>
<td>% effort</td>
<td>1%</td>
<td>1%</td>
<td>1%</td>
<td>8%</td>
<td>45%</td>
<td>10%</td>
<td>36%</td>
</tr>
</tbody>
</table>

CC = customer communication    CE = customer evaluation

Based on an average burdened labor rate of $8,000 per month, the total estimated project cost is $368,000 and the estimated effort is 46 person-months.
Estimation with Use-Cases

<table>
<thead>
<tr>
<th></th>
<th>use cases</th>
<th>scenarios</th>
<th>pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>User interface subsystem</td>
<td>6</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>Engineering subsystem group</td>
<td>10</td>
<td>20</td>
<td>8</td>
</tr>
<tr>
<td>Infrastructure subsystem group</td>
<td>5</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td><strong>Total LOC estimate</strong></td>
<td><strong>42,568</strong></td>
<td><strong>3,366</strong></td>
<td></td>
</tr>
</tbody>
</table>

Using 620 LOC/pm as the average productivity for systems of this type and a burdened labor rate of $8000 per month, the cost per line of code is approximately $13. Based on the use-case estimate and the historical productivity data, the **total estimated project cost is $42,568*13 = $552,000 and the estimated effort is 68 person-months.**
Empirical Estimation Models

*General form:*

\[
\text{effort} = \text{tuning coefficient} \times \text{size}^{\text{exponent}}
\]

- Usually derived as person-months of effort required.
- Either a constant or a number derived based on complexity of project.
- Exponent usually derived.
- Usually LOC but may also be function point.
COCOMO-II

- COCOMO II is actually a hierarchy of estimation models that address the following areas:
  
  - *Application composition model.* Used during the early stages of software engineering, when prototyping of user interfaces, consideration of software and system interaction, assessment of performance, and evaluation of technology maturity are paramount.
  
  - *Early design stage model.* Used once requirements have been stabilized and basic software architecture has been established.
  
  - *Post-architecture-stage model.* Used during the construction of the software.

- Empirical model that relies on size information
The Software Equation

A dynamic multivariable model

\[ E = \left[ \text{LOC} \times B^{0.333}/P \right]^3 \times (1/t^4) \]

where

- \( E \) = effort in person-months or person-years
- \( t \) = project duration in months or years
- \( B \) = “special skills factor”
- \( P \) = “productivity parameter”
Estimation for OO Projects-I

- Develop estimates using effort decomposition, FP analysis, and any other method that is applicable for conventional applications.
- Using object-oriented requirements modeling (Chapter 6), develop use-cases and determine a count.
- From the analysis model, determine the number of key classes (called analysis classes in Chapter 6).
- Categorize the type of interface for the application and develop a multiplier for support classes:
  - **Interface type**
    - No GUI: 2.0
    - Text-based user interface: 2.25
    - GUI: 2.5
    - Complex GUI: 3.0
Multiply the number of key classes (step 3) by the multiplier to obtain an estimate for the number of support classes.

Multiply the total number of classes (key + support) by the average number of work-units per class. Lorenz and Kidd suggest 15 to 20 person-days per class.

Cross check the class-based estimate by multiplying the average number of work-units per use-case.
Estimation for Agile Projects

- Each user scenario (a mini-use-case) is considered separately for estimation purposes.
- The scenario is decomposed into the set of software engineering tasks that will be required to develop it.
- Each task is estimated separately. Note: estimation can be based on historical data, an empirical model, or “experience.”
  - Alternatively, the ‘volume’ of the scenario can be estimated in LOC, FP or some other volume-oriented measure (e.g., use-case count).
- Estimates for each task are summed to create an estimate for the scenario.
  - Alternatively, the volume estimate for the scenario is translated into effort using historical data.
- The effort estimates for all scenarios that are to be implemented for a given software increment are summed to develop the effort estimate for the increment.
The Make-Buy Decision
Computing Expected Cost

Expected cost = \sum_{i} (\text{path probability})_{i} \times (\text{estimated path cost})_{i}

For example, the expected cost to build is:

\text{expected cost}_{\text{build}} = 0.30 (\$380K) + 0.70 (\$450K) = \$429 \text{ K}

Similarly,

\text{expected cost}_{\text{reuse}} = \$382\text{K}
\text{expected cost}_{\text{buy}} = \$267\text{K}
\text{expected cost}_{\text{contr}} = \$410\text{K}