Planning III-A: Estimating Software Size - Estimating Methods, Proxies

Outline
- Review of PSP Levels
- Background
- Criteria for a size estimating method
- Some popular estimating methods
- Proxy-based estimating
- Homework #3
Review of PSP Levels (Humphrey, 1995, p. 11)

- PSP0: Current process
  - Time recording
  - Defect recording
  - Defect type standard

- PSP1: Size estimating
  - Test report

- PSP2: Code reviews
  - Design reviews

- PSP2.1: Design templates

- PSP3: Cyclic development

Background (cf. Humphrey, 1995, p. 97-98)

- Poor planning is a major reason why projects have trouble, and many fail.
- Size-estimating is the generally-accepted practice in engineering, manufacturing, and construction.
  - Start with general estimate or demonstration of similar-feature product.
  - (Iteratively) refine requirements and estimates.
Review of the Project Planning Framework
(cf. Humphrey, 1995, p. 99)

- Define Requirements
- Produce Conceptual Design
- Estimate Product Size
- Estimate Resources
- Produce Schedule
- Develop Product
- Analyze the Process
- Tracking Reports

NOTE: Real life is not as linear as this framework suggests.

Background (cf. Humphrey, 1995, p. 98-100)

- “The tricky part of software size estimating is in characterizing the product elements and relating them to your historical experience.”
- The accuracy of any size-resource-cost model is “limited by the accuracy of the size estimates. So, even when you use an estimating model, you need an accurate size estimate.”
- Models must be calibrated to the organizations which use them.
- Estimation errors can be very large, even 100% or more.
- Very few professionals (22% in JPL study) use size estimation to make cost estimates.
- Early project uncertainty makes it hard to accurately estimate SW size.
  - PC, PI, and Objects may reduce this problem.
Criteria for a Size-Estimating Method
(cf. Humphrey, 1995, p. 100-101)

- Uses structured and trainable methods.
- Can use in all phases of development & maintenance.
- Usable for all SW product elements: code, files, reports, screens, & documentation.
- Suitable for statistical analysis.
- Applicable to future types of work.
- Provides a means to judge the accuracy of your work.

Popular Estimating Methods: Delphi
(cf. Humphrey, 1995, p. 102-103)

- Uses several estimators
- Coordinator calculates average and returns summary forms
- Estimators discuss results
- Iterate until consensus

  Ex:
  - Initial SLOC: A=100, B=500, C=350, avg=317
  - 2nd estimate: A=275, B=400, C=325, avg=333
  - 3rd estimate: A=300, B=375, C=300, avg=325

- Can be very accurate, but
- Relies on a few experts,
- Is time consuming, and
- Can be biased.
Popular Estimating Methods:
Fuzzy Logic (cf. Humphrey, 1995, p. 103-105)

- Roughly judge how predicted size compares with historical data.
- Historical data is divided into topical categories and subgroups based on size. You need a large amount of historical data for this.
- Look at examples on p. 103-105. (Note: \( \log_{10}(\text{LOC}) \) is used to create equally-sized ranges.)

Fuzzy Logic
In-Class Practice Problem

- See “In-Class Practice Problems” handouts
Popular Estimating Methods: Standard Components
(c.f. Humphrey, 1995, p. 102-103)

- Make a list of standard components
  - Files, modules, subsystems, screens, ...
- Determine historical average SLOC for each standard component
- Estimate min, most-likely, and max number of each required standard component
- Calculate estimated number of each required component
  - \( \text{Number} = (\text{min} + 4\times \text{most-likely} + \text{max})/6 \)
- Multiply each component’s estimated number by its historical average SLOC
- Sum these SLOC’s to obtain a total estimate for the project.
- Look at the example on p. 106.

- Easy to use, but
- Probably lack good historical base for large components, which must be estimated early in project.

Standard Components
In-Class Practice Problem

- See “In-Class Practice Problems” handouts

- Function-point = arbitrary unit.
- Most popular method for estimating the size of commercial SW app’s.
- Albrecht (1979) at IBM.
- Five basic functions that occur frequently in commercial SW:

<table>
<thead>
<tr>
<th>Function Types</th>
<th>Weights</th>
<th>Basic Counts</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>x4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Outputs</td>
<td>x5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inquiries</td>
<td>x4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logical Files</td>
<td>x10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interfaces</td>
<td>x7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Note adjustment factor calculation on p. 108.
- Not directly measurable in end-product, don’t reflect development language, skill not readily improved w/o large historical database.

Function Points
In-Class Practice Problem

- See “In-Class Practice Problems” handouts
Proxy-Based Estimating
(cf. Humphrey, 1995, lecture slides)

- **Basic issues:**
  - Good size measures are detailed and based on historical data.
  - Early estimators can rarely think in detail. (e.g. home construction and square feet vs. number, types, and sizes of rooms)

- **Alternatives**
  - Wait until you have the detail to generate an estimate
  - Make your best guess
  - Use a suitable proxy

Definition of a Proxy
(cf. Humphrey, 1995, p. 111)

- **Def:**
  - “A proxy is a substitute or stand-in”, something that is used in place of another.

- **A good proxy provides an easy early visualization of, and is related to, the size of the final product.**

- **Examples:**
  - Objects, screens, files, scripts, function points...
Criteria for a Good Proxy
(cf. Humphrey, 1995, p. 111-113)

- Related to Development Effort
- Automatically Countable
- Easily Visualizable at Project Start
- Customizable to Organization’s Needs
- Sensitive to Implementation Variations (e.g. language, design style, application categories, etc.)

Potential Proxies
(cf. Humphrey, 1995, p. 113)

- Objects, document chapters - seem to meet proxy criteria.
- Screens, reports, scripts - not enough data to draw conclusions.
- Collect data and assess each type’s applicability to your work.
Objects as Proxies
(cf. Humphrey, 1995, 113-116, and section slides)

- Object counts correlate well with development hours
- Object LOC correlates very closely - functions & procedures may work too (cf. graphs on p. 114-116, and below)


- Collect data on proxy
- Correlate proxy with total product LOC and development hours
- If good correlation then it is a potential proxy
- Divide into categories and size ranges (as in Fuzzy-Logic method)
- Estimate based on assessment of similar categories and sizes
- Best to normalize object LOC to average method LOC

Look at examples on p. 117.
Homework #3

- See “Homework Assignments” list and textbook instructions.