Cost Estimation

What are the costs of a Software Project?

- Why does it matter for us to know this?
- How do you measure productivity?
Why is project cost difficult to estimate?

Software cost estimation – Boehm (1981)

- Algorithmic cost modeling
  - Base estimate on project size (lines of code)
- Expert judgment
  - Ask others
- Estimation by analogy
  - Cost based on experience with similar projects
- Parkinson’s Law
  - Project time will expand to fill time available
- Pricing to win
  - Cost will be whatever customer is willing to pay
- Top-down estimation
  - Estimation based on function/object points
- Bottom-up estimation
  - Estimation based on components
Function points are computed by first calculating an unadjusted function point count (UFC). Counts are made for the following categories (Fenton, 1997):

- *External inputs* – those items provided by the user that describe distinct application-oriented data (such as file names and menu selections)
- *External outputs* – those items provided to the user that generate distinct application-oriented data (such as reports and messages, rather than the individual components of these)
- *External inquiries* – interactive inputs requiring a response
- *External files* – machine-readable interfaces to other systems
- *Internal files* – logical master files in the system

Each of these is then assessed for complexity and given a weighting from 3 (for simple external inputs) to 15 (for complex internal files).

### Function points -> Lines of code

<table>
<thead>
<tr>
<th>Language</th>
<th>Ratio: Source:Executable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembler</td>
<td>1:1</td>
</tr>
<tr>
<td>Micro-Assembler</td>
<td>1:1.5</td>
</tr>
<tr>
<td>C</td>
<td>1:2.5</td>
</tr>
<tr>
<td>ALGOL</td>
<td>1:3</td>
</tr>
<tr>
<td>COBOL</td>
<td>1:3</td>
</tr>
<tr>
<td>FORTRAN</td>
<td>1:3</td>
</tr>
<tr>
<td>Pascal</td>
<td>1:3.5</td>
</tr>
<tr>
<td>RPG</td>
<td>1:4.5</td>
</tr>
<tr>
<td>PL/I</td>
<td>1:4</td>
</tr>
<tr>
<td>MODULA-2</td>
<td>1:4.5</td>
</tr>
<tr>
<td>Ada</td>
<td>1:5</td>
</tr>
<tr>
<td>FORLOG</td>
<td>1:5</td>
</tr>
<tr>
<td>LISP</td>
<td>1:5</td>
</tr>
<tr>
<td>PERTH</td>
<td>1:5</td>
</tr>
<tr>
<td>BASIC</td>
<td>1:5</td>
</tr>
<tr>
<td>LOGO</td>
<td>1:5</td>
</tr>
<tr>
<td>48h-GL3</td>
<td>1:6</td>
</tr>
<tr>
<td>APL</td>
<td>1:6</td>
</tr>
<tr>
<td>OBJECTIVE-C</td>
<td>1:1.2</td>
</tr>
<tr>
<td>SMALLTALK</td>
<td>1:1.5</td>
</tr>
<tr>
<td>Query Languages</td>
<td>1:20</td>
</tr>
<tr>
<td>Spreadsheets</td>
<td>1:50</td>
</tr>
</tbody>
</table>

200-300 LOC/Function point in assembly

Multiplication factor by language

Programmer productivity? 30-900 LOC/month!
**Function Point Weights**

<table>
<thead>
<tr>
<th>Item</th>
<th>Weighting Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simple</td>
</tr>
<tr>
<td>External inputs</td>
<td>3</td>
</tr>
<tr>
<td>External outputs</td>
<td>4</td>
</tr>
<tr>
<td>External inquiries</td>
<td>3</td>
</tr>
<tr>
<td>External files</td>
<td>7</td>
</tr>
<tr>
<td>Internal files</td>
<td>5</td>
</tr>
</tbody>
</table>

Each count is multiplied by its corresponding complexity weight and the results are summed to provide the UFC Object points.

**Object Points**

Similar to function points (used to estimate projects based heavily on reuse, scripting and adaptation of existing tools)

- Number of screens (simple x1, complex x2, difficult x3)
- Number of reports (simple x2, complex x5, difficult x8)
- Number of custom modules written in languages like Java/C x10
- Productivity between 4 and 50 object points per month/developer
Factors affecting productivity

- Application domain expertise
- Quality requirements
- Project size
- Team size
- Technology support
- Working environment

Project & Risk Management

- Managing expectations & Trade-offs
- Planning for the unforeseen
Cost Breakdown

Affecting Delivery Dates

- More people != More faster
- Why?
Is That it?

- What does a man-hour really cost?
**Project scheduling**

- 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.

**Activity vocabulary**

- Activities in a project should be organized to produce tangible outputs for management to judge progress.

- *Milestones* are the end-point of a process activity.

- *Deliverables* are project results delivered to customers.
**Bar charts and activity networks**

- Show project breakdown into tasks. Tasks should not be too small. They should take about a week or two.

- Tasks should not be too big, or you loose your ability to track progress.

- Activity charts show task dependencies and the critical path.

- Bar charts show schedule against calendar time.

**Activity network: Critical Path**
Activity timeline: GANTT Charts

WBS 1 Summary Element 1
- WBS 1.1 Activity A: 75% complete
- WBS 1.2 Activity B: 67% complete
- WBS 1.3 Activity C: 50% complete
- WBS 1.4 Activity D: 0% complete

WBS 2 Summary Element 2
- WBS 2.1 Activity E: 0% complete
- WBS 2.2 Activity F: 0% complete
- WBS 2.3 Activity G: 0% complete

Estimations vs. Real-world

- Estimations will typically give you a best-case scenario.
- Even if you do everything perfectly, chances are the unforeseen will affect your productivity
Building a Buffer:
Planning for the Unexpected

• Buffer for unexpected features we thought up during development
• Buffer for unexpected competitive responses needed because our competitor did something
• Buffer to allow code written by different developers to be integrated so it works together (depending on the experience of your team this can be 25% - 100%)
• Buffer for non-development tasks that employees must perform, e.g. "1 day mandatory diversity training," "emergency company meeting," fire drills, birthday cake for the boss, etc. etc.
• Buffer because things take longer than estimated

How Much To Buffer?

• Probability distribution
  – Most optimistic date
  – Pessimistic date
  – Most likely date