Lecture 6: Estimation Techniques
Why estimations?

Based on estimations

- Project selection
- Price (bids)
- Change requests

Costs

- Schedule
- Resources

- Resource allocation
- Hiring

Duration

Milestones
Estimations in software projects

- Duration is essentially effort / resources

- Costs
  - Mostly personnel cost (effort)
  - Travel, training
  - Hardware, software

- Effort

- Schedule

- Resources
Overview

Estimation Techniques
  Empirical Estimations
  Algorithmical Estimations
Estimation Process
Estimation Exercise

- How many passenger planes does Lufthansa have?
  - Not counting regional subsidiaries

- How can we approach this problem systematically?
Overview

Estimation Techniques
  Empirical Estimations
  Algorithmic Estimations
Estimation Process
Empirical estimation: Expert judgment

- Estimate is based on experience and historical data
- Involve experts in
  - Development techniques
  - Application domain
- Most common technique in practice
Top-down estimation

- Estimation by **analogy**
  - Comparison with **similar projects**
  - Analysis of differences
  - Typical example: SAP introduction

**Pros**
- Quicker and less expensive than other methods
- Can be done early in the project

**Cons**
- Underestimation of difficult technical problems likely
- No detailed justification of estimate
- Be aware of scalability problems!
Top-down estimation: Delphi method

- **More accurate** than ordinary expert judgment
  - Eliminates outliers
- **More expensive** to produce

**Step 1:** Each expert submits
- Estimate
- Justification

**Step 2:** Each expert receives summary of all estimates

**Step 3:** Each expert submits
- New estimate
- Justification of deviation from average of previous estimates

**Step 4:** Iterate until consensus is achieved
Top-down estimation: Typical figures

- Typical figures for software development
  - Analysis: 20%
  - Design: 40%
  - Implementation: 15%
  - Test: 25%

- Very helpful to validate estimations
Bottom-up estimation

- Estimation by decomposition
  - Estimating the effort for individual work packages
  - Cost and accuracy depend on size of the work packages

**Pros**
- See “cons” of top-down estimation

**Cons**
- Underestimation because effort does not grow linearly (due to complexity, etc.)
- Underestimation of integration effort
- Requires initial system design
Program evaluation and review technique

- **Goal:** Manage probabilities with simple statistics
- **Approach:** Ask several experts for three estimates
  - Optimistic, Likely (mode), and Pessimistic
- **Important formulas**
  - Mean \( M = \frac{(O + 4 \times L + P)}{6} \)
  - Deviation \( V = \frac{(P - O)}{6} \)
- **Assumptions**
  - Project effort is normally distributed (more than 20 work packages)
  - Work package efforts are statistically independent (ignores single underlying cause of delay)
Overview

Estimation Techniques
   Empirical Estimations
      Algorithmical Estimations
Estimation Process

Algorithmic estimation of software

- Basic cost model

\[ \text{Effort} = A \times \text{Size}^B \times m(X) \]

Size: Some measurement of the software size
A: Constant factor that depends on
  - Organizational practices
  - Type of software
B: Usually lies between 1 and 1.5
X: Vector of cost factors
m: Adjustment multiplier
Cost models

- Define a way to determine the size
- Define cost factors $X$
- Provide defaults for parameters $A$, $B$, $m$
  (based on hundreds of projects)

- Important examples
  - Function point analysis
  - Constructive cost model (COCOMO)

$$\text{Effort} = A \times \text{Size}^B \times m(X)$$
Measuring size: Lines of code

- Software size can be measured in lines of source code
  - Most commonly used metric

- **Difficult in early phases** of the project (before design is known)
  - Reuse, make-or-buy decisions

- **Influenced** heavily by choice of **programming language**
- Should only be **used indirectly**
Function point analysis

- Size is estimated based on requirements

![Diagram of function point analysis with inputs, inquiries, outputs, internal files, and external files connected to a function circle.]
Functions

- **Inputs**
  - Forms, dialogs, messages, XML documents

- **Outputs**
  - Web pages, reports, graphs, messages, XML documents

- **Inquiries** (input/output combinations)
  - Simple web inputs, generally producing a single output

- **Logical internal files** (controlled by the program)
  - Tables, views or files in database

- **External files** (controlled by other programs)
  - Tables or files used from other systems or databases
Complexity of functions

<table>
<thead>
<tr>
<th>Factor</th>
<th>Simple</th>
<th>Average</th>
<th>Complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Outputs</td>
<td>4</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>Inquiries</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Ext. files</td>
<td>7</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>Int. files</td>
<td>5</td>
<td>7</td>
<td>10</td>
</tr>
</tbody>
</table>

- Determine complexity of each function
- **Weight** each function according to complexity

<table>
<thead>
<tr>
<th>Input</th>
<th>Simple</th>
<th>Average</th>
<th>Complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data elements</td>
<td>1-5</td>
<td>6-10</td>
<td>&gt;10</td>
</tr>
<tr>
<td>Checking</td>
<td>Formal</td>
<td>Formal, logical</td>
<td>Formal, logical, requires DB access</td>
</tr>
</tbody>
</table>
Cost factors

Rate each element from 0 - 5
- 0: no influence
- 1: insignificant influence
- 2: moderate influence
- 3: average influence
- 4: significant influence
- 5: strong influence

Technical complexity factor
- TCF = 0.65 + 0.01 \times \text{sum}
- Varies between 0.65 and 1.35

Data communications
Distributed processing
Performance
Heavy use
Transaction rate
Online data entry
Complex interface
Online data update
Complex processing
Reusability
Installation ease
Operational ease
Multiple sites
Facilitate change
### Function point computation

<table>
<thead>
<tr>
<th></th>
<th>Simple</th>
<th>Average</th>
<th>Complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>$6 \times 3 = 18$</td>
<td>$2 \times 4 = 8$</td>
<td>$3 \times 6 = 18$</td>
</tr>
<tr>
<td>Outputs</td>
<td>$7 \times 4 = 28$</td>
<td>$7 \times 5 = 35$</td>
<td>$0 \times 7 = 0$</td>
</tr>
<tr>
<td>Inquiries</td>
<td>$0 \times 3 = 0$</td>
<td>$2 \times 4 = 8$</td>
<td>$4 \times 6 = 24$</td>
</tr>
<tr>
<td>Ext. files</td>
<td>$9 \times 5 = 45$</td>
<td>$0 \times 7 = 0$</td>
<td>$2 \times 10 = 20$</td>
</tr>
<tr>
<td>Int. files</td>
<td>$5 \times 7 = 35$</td>
<td>$2 \times 10 = 20$</td>
<td>$3 \times 15 = 45$</td>
</tr>
<tr>
<td><strong>Unadjusted function points (UFP)</strong></td>
<td></td>
<td></td>
<td><strong>304</strong></td>
</tr>
<tr>
<td><strong>Technical complexity factor (TCF)</strong></td>
<td></td>
<td></td>
<td><strong>1.15</strong></td>
</tr>
<tr>
<td><strong>Adjusted function points</strong></td>
<td></td>
<td></td>
<td><strong>350</strong></td>
</tr>
</tbody>
</table>
Determining effort and size

- Empirical value for effort
  - Or use a table
- Empirical value for size
- Huge differences in productivity
  - Factor 10-20 between individual programmers
  - Factor 4 between companies

\[
\text{Effort} = \frac{FP^{1.4}}{150}
\]
Observation about software size

- Consider a project that requires 10 Web pages, 15 reports, and 20 database tables
  - 315 function points, if each item is medium complexity
- How many lines of C code would it have?
  - About 32,000 lines
- What if you used Excel?
  - About 2,000 lines
- Why do you think there are so many spreadsheets out there?
Function point analysis: Discussion

Pros
- Based on requirements (instead of code size)
- Can be applied in early project phases
- Can be calibrated (for company, project type)
- Counting standards by “International Function Points User Group”
- Technology-independent

Cons
- Estimation of overall effort (not per phase)
- Tailored towards functional decomposition (rather than OO)
- Tailored towards information systems
- Needs calibration to produce reliable results
Estimation techniques: Discussion

Empirical Estimation
- Accurate if experts are experienced
- Experts can be strongly biased (over-optimism)

Algorithmic Estimation
- Very accurate if model is calibrated
- Calibration is very difficult and expensive
- Estimation is expensive
Other estimation strategies

**Parkinson’s Law**
- Work expands to fill the time available
  - Gold plating
- Effort is determined by available resources
- Important for team management

**Pricing to win**
- Cost is estimated to whatever the customer is willing to spend
- Common strategy to win projects
- Features are negotiated later, constrained by agreed costs
- Costs are fixed, not requirements
Overview

Estimation Techniques
  Empirical Estimations
  Algorithmical Estimations

Estimation Process
### Estimate types

<table>
<thead>
<tr>
<th>Rough order of magnitude</th>
<th>Budgetary</th>
<th>Definitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial estimates</td>
<td>Decision making, response to proposals</td>
<td>Project plan, proposals</td>
</tr>
<tr>
<td>-25 / +75%</td>
<td>-10 / +25%</td>
<td>-5 / +10%</td>
</tr>
</tbody>
</table>

- **Refine** your estimates at each project stage
  - Requirements document, system design, detailed design, working code
Estimating process

1. Establish objectives
2. Determine project details
3. Select strategy and plan
4. Generate effort estimate
5. Determine team size and duration
6. Validate and finalize
7. Document

Why? Accuracy? Audience?

Duration = \sqrt{\text{Effort}}
Duration = 3.0 \times \text{Effort}^{1/3}
(Effort in person months, Duration in months)
Effort = \text{Duration} \times \text{Team Size}

Different method, review

Estimators, type of validation, historic data

Document assumptions
Estimation tips

• Avoid off-the-cuff estimates
• Allow time for the estimate, and plan it
• Use historic data
• Use developer-based estimates
• Estimate by walkthrough
• Estimate by categories
  ➢ e.g. easy, medium, hard
• Estimate at a low level of detail
• Don’t omit common tasks (management; use checklists)
• Use different techniques and compare the results
• Change estimation practices as the project progresses
From effort to costs

- **Direct costs**: Costs incurred for the benefit of a specific project
  - Salaries of project staff
  - Equipment bought specifically for the project
  - Travel expenses

- **Indirect costs**: Costs incurred for the joint benefit over multiple projects ("overhead")
  - Accounting, quality assurance department
  - Line management
  - Rooms, electricity, heating
Unit costs

- Projects have to budget for
  - Direct costs
  - A certain share of indirect costs
- Budgets are usually determined by using unit costs
  - Unit cost: Price per unit of a resource
  - **Loaded rate**: Including indirect costs
  - **Unloaded rate**: Without indirect costs
- Examples
  - Loaded day rate for senior IT consultant: CHF 3.500
  - Loaded day rate for internal developer: CHF 1.200
From costs to prices

- The price is often based on the costs and a margin
- \[ \text{Price} = \frac{\text{Costs}}{1 - \text{Margin}} \]
- Example
  - Costs = CHF 1,000,000
  - Margin = 5%
  - Price = CHF 1,052,632
- Price is influenced by
  - Market situation
  - Business strategy