Estimation of Software Size and Effort
Distributions
Using Paired Ratio Comparison Matrices

Jairus Hihn
Karen Lum

Software Quality Improvement Project
Jet Propulsion Laboratory/California Institute of Technology

18th International Forum on COCOMO and Software Cost Modeling
October 21-24, 2003
Software Quality Improvement Project: Measurement & Estimation Element

Charter – Provide the basis for a quantitative approach to software management and assessment

Products:
- Handbooks, Guidelines, and Reports:
- Data Repositories
- Software Engineering Models to support:
  - Cost Estimation & Planning
  - Quality Assessment & Planning
  - Process Improvement
- Operational Measurement Program
- Training
- Support/Consulting
- Assessments
JPL Software Cost Estimation Procedure

Scope the Job
- Engineering Estimate
  - Estimate Effort
  - Schedule Effort
  - Calculate Cost
- Determine the Impact of Risk
- Track & Report Estimates

Estimate Software Size
- Model-based Estimate

Validation and Reconciliation
- Review & Approve Estimates

SW Cost Inputs
- Requirements
- Architectural Design
- Mission/Project Sched.
- Implementation Appr.
- Mission/Project WBS
- SW Implementation and Design Approach

Constraints
- Applicable Processes & procedures
- Design principles
- Std WBS
- NASA & OMB Reqs

When budget is too low “Do not look for a silver bullet” - DESCOPE

Estimation Mantra
- Data
- Multiple Estimates
- Uncertainty & Risk

Save History
Follow Through

October 21, 2003
Hihn & Lum,
Story Part 1: The Problem

- A new software manager had to make his first serious estimate for a ‘real’ delivery, Delivery 2. The previous build had
  - developed highly modular components (tools & libraries) and had not gone through formal acceptance test
  - been run very informally as a level-of-effort task and no records had been kept except high level schedule and cost

- He wanted us to help him with his estimate but his team
  - Did not trust sizing based on lines of code as size or cost models
  - Intuitively they thought of capabilities when describing a function

- So our job was
  - partly to play the outsider who was forcing the kids to follow the rules
  - to derive a formalism that would engage them so they would work with us and we could derive useful expert judgment information for direct effort estimates
Matrices of paired comparisons provide a systematic way to extract, combine, and capture expert judgments [Saaty, 1977]
- Variation of this approach used SEER-SSM
- Sometimes called judgment matrices

**Pairwise Comparison Matrix example:**

<table>
<thead>
<tr>
<th></th>
<th>Module 1</th>
<th>Module 2</th>
<th>Module 3</th>
<th>Module 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module 1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Module 2</td>
<td>0.5</td>
<td>1</td>
<td>1.5</td>
<td>2</td>
</tr>
<tr>
<td>Module 3</td>
<td>0.25</td>
<td>0.67</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Module 4</td>
<td>0.33</td>
<td>0.5</td>
<td>0.5</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ a_{ij} = 1/a_{ji} \]

- Use multiple estimates of comparisons (relative differences) to average out estimation errors
Pairwise Comparison Matrices: What are they?

- Creating a judgment matrix involves creating an \( n \times n \) matrix (\( A^{nxn}=[a_{ij}] \)), where \( n \) is the number of entities being compared.
  - Each element, \( a_{ij} \), in the matrix is an estimate of the relative size of entity \( i \) with respect to entity \( j \), that is \( \frac{\text{Size}_i}{\text{Size}_j} \).
  - The human mind is better at identifying relative differences than at estimating absolute values [Miranda, 1999; Shepperd, 2001]

- The properties of a judgment matrix require that elements be:
  - reciprocal, \( a_{ij} = 1/a_{ji} \), which means that entity \( i \) is \( a_{ij} \) times bigger than entity \( j \), then entity \( j \) is \( 1/a_{ij} \) times smaller than entity \( i \)
  - the same size as itself, which means that all diagonal elements \( a_{ii} = 1 \)

- Inconsistent matrices produces \( n \) different ranking vectors of adjustment factors; we want one vector to generate a final estimate
Deriving SW Size from a Pairwise Comparison Matrix

<table>
<thead>
<tr>
<th>Estimate</th>
<th>Module 1</th>
<th>Module 2</th>
<th>Module 3</th>
<th>Module 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module 1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Module 2</td>
<td>0.5</td>
<td>1</td>
<td>1.5</td>
<td>2</td>
</tr>
<tr>
<td>Module 3</td>
<td>0.25</td>
<td>0.67</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Module 4</td>
<td>0.33</td>
<td>0.5</td>
<td>0.5</td>
<td>1</td>
</tr>
</tbody>
</table>

1) Derive the judgment matrix by estimating the relative size of all modules
2) Compute the geometric mean across each row in the matrix
   - yields slope of line
3) Derive size/effort estimate by normalizing values to the reference module
   - yields origin of line
   - e.g. if the reference module, Module 3 = 2000 SLOC then derive origin as 2000/.76 = 2632.

Note: We made the assumption that size scaled with capability or 
\[ \frac{\text{size}_i}{\text{size}_j} = w \cdot \frac{\text{capability}_i}{\text{capability}_j} \]

Total Size = 12,159
Multiple reference projects

- Multiple reference analogies with an inconsistent judgment matrix creates the dilemma that a different total size estimate is generated depending upon which reference module is used.

I cannot guess one number

I want to see the whole matrix

I have to understand how you got the answer
Rest of the Solution
Incorporate Uncertainty

<table>
<thead>
<tr>
<th>Estimate</th>
<th>Module 1</th>
<th>Module 2</th>
<th>Module 3</th>
<th>Module 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module 1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Module 2</td>
<td>0.5</td>
<td>1</td>
<td>1.5</td>
<td>2</td>
</tr>
<tr>
<td>Module 3</td>
<td>0.25</td>
<td>0.67</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Module 4</td>
<td>0.33</td>
<td>0.5</td>
<td>0.5</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\begin{bmatrix}
2.21 \\
1.11 \\
0.76 \\
0.54
\end{bmatrix} \times 2632
\]

Total Size
= 12,159

<table>
<thead>
<tr>
<th>Estimate</th>
<th>Module 1</th>
<th>Module 2</th>
<th>Module 3</th>
<th>Module 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module 1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>TriPDF (1, 3, 4)</td>
</tr>
<tr>
<td>Module 2</td>
<td>0.5</td>
<td>1</td>
<td>TriPDF (1, 1.5, 3)</td>
<td>TriPDF (1, 2, 3)</td>
</tr>
<tr>
<td>Module 3</td>
<td>0.25</td>
<td>1/TriPDF (1, 1.5, 3)</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Module 4</td>
<td>1/TriPDF (1, 3, 4)</td>
<td>1/TriPDF (1, 2, 3)</td>
<td>0.5</td>
<td>1</td>
</tr>
</tbody>
</table>

Multiplier CDF

Total Size CDF

October 21, 2003

Hihn & Lum,
Mission Critical ground software

Estimate cost of Delivery 2 based on cost of Delivery 1

Delivery 1 consisted of 1 function (A) with 5 modules

Delivery 2 consisted of 2 functions (B & C) with 5 modules each

Used Pairwise Comparisons Technique to estimate Delivery 2 effort and size, based on Function A

- Inputs provided by developers
- Core information based on analogical comparisons to Function A actuals
- Major inconsistencies between estimators were reduced by having them discuss the differences until a consensus was achieved
- Estimate 1: Effort-based estimate
- Estimate 2: Size-based estimate
**Constructed Distributional Pairwise Comparisons**

### Function B (Delivery 2)

<table>
<thead>
<tr>
<th>Function B</th>
<th>User Interface</th>
<th>Core</th>
<th>Mapping</th>
<th>Stochastic</th>
<th>Smoother</th>
<th>FM</th>
<th>FPI</th>
<th>PD</th>
<th>CI</th>
<th>UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>1 TriPDF(4,4,5)</td>
<td>1</td>
<td>1 TriPDF(5,75,1)</td>
<td>TriPDF(5,75,1)</td>
<td>0.2</td>
<td>2</td>
<td>1</td>
<td>1.5</td>
<td>TriPDF(1,5,75,2)</td>
<td></td>
</tr>
<tr>
<td>Mapping</td>
<td>1 TriPDF(25,375,5)</td>
<td>TriPDF(5,75,1)</td>
<td>0.2</td>
<td>1</td>
<td>1</td>
<td>0.75</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stochastic</td>
<td>1 TriPDF(1,5,2)</td>
<td>TriPDF(25,625,1)</td>
<td>TriPDF(1,2.3)</td>
<td>TriPDF(5,25,2)</td>
<td>TriPDF(2,3,4)</td>
<td>TriPDF(1,3,3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ a_{ij} = 1/a_{ji} \]

### Function A (Delivery 1)

\[ a_{ij} = a_{ik} \cdot a_{kj} \]

### Function C (Delivery 2)

<table>
<thead>
<tr>
<th>Function C</th>
<th>Measurement Models</th>
<th>Delay Models</th>
<th>User Interface</th>
<th>File Formats</th>
<th>Stations</th>
</tr>
</thead>
<tbody>
<tr>
<td>UI</td>
<td>MM</td>
<td>DM</td>
<td>UI</td>
<td>FT</td>
<td>FM</td>
</tr>
<tr>
<td>Core</td>
<td>1 TriPDF(0.6,625,0.75)</td>
<td>TriPDF(1,25.1,375,1)</td>
<td>TriPDF(0.5,0.7,1)</td>
<td>TriPDF(1,25.1,375,1)</td>
<td>TriPDF(0.5,0.7,1)</td>
</tr>
</tbody>
</table>

\[ a_{ij} = 1/a_{ji} \]

### Function A (Delivery 1)

\[ a_{ij} = a_{ik} \cdot a_{kj} \]

### Function A (Delivery 1)

\[ a_{ij} = a_{ik} \cdot a_{kj} \]

### Function A (Delivery 1)

\[ a_{ij} = a_{ik} \cdot a_{kj} \]
Software Size estimates

<table>
<thead>
<tr>
<th></th>
<th>Core</th>
<th>Mapping</th>
<th>Smoother</th>
<th>Stochastics</th>
<th>Total SLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>5th Percentile</td>
<td>3621</td>
<td>2800</td>
<td>3316</td>
<td>5880</td>
<td>15898</td>
</tr>
<tr>
<td>Mode</td>
<td>5740</td>
<td>3831</td>
<td>4985</td>
<td>10374</td>
<td>22155</td>
</tr>
<tr>
<td>95th percentile</td>
<td>6796</td>
<td>5230</td>
<td>6444</td>
<td>11878</td>
<td>29950</td>
</tr>
</tbody>
</table>

Measurement

<table>
<thead>
<tr>
<th></th>
<th>Models</th>
<th>Delay Models</th>
<th>File Formats</th>
<th>Stations</th>
<th>Total SLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>5th Percentile</td>
<td>8209</td>
<td>3100</td>
<td>4525</td>
<td>3976</td>
<td>19866</td>
</tr>
<tr>
<td>Mode</td>
<td>10737</td>
<td>3789</td>
<td>5971</td>
<td>4833</td>
<td>24968</td>
</tr>
<tr>
<td>95th percentile</td>
<td>12837</td>
<td>4897</td>
<td>7053</td>
<td>6241</td>
<td>30899</td>
</tr>
</tbody>
</table>

JPL Probabilistic COCOMO II (SCAT) estimates

- Model estimated 129 to 137 work months for 50th to 70th percentile
- Budget for Build 3 was 135.6 work months
Performance

- It is now one year later.
- Delivery 2 has completed at 133.7 WM so the model estimate appears to have performed well.
- However, the lines of code is significantly less than expected which is partly due to not all the planned tasks being completed.
# Changes in Model Inputs

<table>
<thead>
<tr>
<th></th>
<th>COCOMO II Estimated Size and Driver Ratings</th>
<th>COCOMO II Actual Size and Driver Ratings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Function B</td>
<td>Function C</td>
</tr>
<tr>
<td></td>
<td>Low, Likely, High</td>
<td>Low, Likely, High</td>
</tr>
<tr>
<td>New SLOC</td>
<td>15898, 22155, 29950</td>
<td>19886, 24968, 30899</td>
</tr>
<tr>
<td>REVL</td>
<td>10%</td>
<td>15%</td>
</tr>
<tr>
<td>Adapted SLOC</td>
<td>7226</td>
<td></td>
</tr>
<tr>
<td>% Design Modified</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>% Code Modified</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>% Integration Modified</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

## EFFORT MULTIPLIERS

|                  | RELY | DATA | DOCU | CPLX | RUSE | TIME | STOR | PVOL | ACAP | APEX | PCAP | PLEX | LTEX | PCON | TOOL | SCED | SITE | PREC | FLEX | RESL | TEAM | PMAT |
|------------------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
|                  | H    | N    | N, N+25, H | VL, L, VH | N   | N    | N    | L    | VH   | L    | H    | VH   | L, H+50, VH | N   | N    | VL, N, H | L    | H    | VH   | L, L+50, N |
|                  | H    | N    | N    | L    | H    | L    | N    | VH   | VH   | VH   | N    | VH   | H+50 | N    | H+50 | L+50 | L+50 | L+50 | L+50 |
### Size Estimates Compared to Actuals

#### Function B

<table>
<thead>
<tr>
<th></th>
<th>Core</th>
<th>Mapping</th>
<th>Smoother</th>
<th>Stochastics</th>
<th>Total SLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>5th Percentile</td>
<td>3077.85</td>
<td>2380</td>
<td>1658</td>
<td>2940</td>
<td>10237</td>
</tr>
<tr>
<td>Mode</td>
<td>4879</td>
<td>3256.35</td>
<td>2492.5</td>
<td>5187</td>
<td>14054</td>
</tr>
<tr>
<td>95th percentile</td>
<td>5776.6</td>
<td>4445.5</td>
<td>3222</td>
<td>5939</td>
<td>19129</td>
</tr>
<tr>
<td>Delivered SLOC</td>
<td>5750</td>
<td>6728</td>
<td>244</td>
<td>748</td>
<td>13470</td>
</tr>
<tr>
<td>Eq SLOC</td>
<td>3928</td>
<td>2912</td>
<td>244</td>
<td>748</td>
<td>7832</td>
</tr>
</tbody>
</table>

#### Function C

<table>
<thead>
<tr>
<th></th>
<th>Measurement Models</th>
<th>Delay Models</th>
<th>File Formats</th>
<th>Stations</th>
<th>Total SLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>5th Percentile</td>
<td>4104.5</td>
<td>0</td>
<td>4072.5</td>
<td>3976</td>
<td>12372</td>
</tr>
<tr>
<td>Mode</td>
<td>5368.5</td>
<td>0</td>
<td>5373.9</td>
<td>4833</td>
<td>13842</td>
</tr>
<tr>
<td>95th percentile</td>
<td>6418.5</td>
<td>0</td>
<td>6347.7</td>
<td>6241</td>
<td>18758</td>
</tr>
<tr>
<td>Delivered SLOC</td>
<td>1959</td>
<td>689</td>
<td>7611</td>
<td>2105</td>
<td>12364</td>
</tr>
</tbody>
</table>

- Considering we had never applied this technique before at JPL to estimate SLOC and we were adapting the technique it performed fairly well based on:
  - JPL SW Engineers underestimate delivered SLOC by 70% (1990-1996)
  - In this case study Delivered SLOC was over estimated by 4-11% on average compared to mode. We were originally asked to assume no inheritance.
The main issue that arose was with numerical and algorithmically intensive code

Spent much more time in algorithm development for Fn B than originally planned
  - Very Complex to derive and implement algorithms
  - 6 months on algorithm development, even with excellent, highly capable, experienced development team
    • Existing software not well documented, so needed to reverse engineer

Also underestimated amount of unit testing for Fn C capabilities again because of numerical issues
  - Very advanced and time consuming numerical testing
  - Have to match unit testing to legacy software and Zuma in every case
  - Hundreds of hours spent trying to refine interaction
  - These aspects of the job were not anticipated
Adapting the Technique to Deal with Algorythmic Intensive Code

The Pairwise Comparison Matrices as described in this talk were introduced by Satty as part of the Analytical Hierarchy Process. Can decompose capability to address size and algorithms separately.

In this example one would need to construct three pairwise comparison matrices

- One to compare the algorithms vs size contribution to ‘Capabilities’. This is used to combine the algorithm and code vectors into a single capability vector. One could work in either an effort or ‘bigness’ dimension.
- One each for comparing functions by relative Algorithm size and code size
Conclusion

- **Technical conclusions:**
  - Matrices of pairwise comparisons are an effective way to capture expert judgment
  - Incorporating uncertainty is relatively straightforward
  - The reason for incorporating uncertainty is that it addresses major issues
    - Estimator uncertainty
    - Multiple reference projects

- **Organizational conclusions:**
  - Successful example of introducing quantitative techniques into software development teams
  - Working with the technical staff in the way they wanted allowed us to evolve their ‘consciousness’ eventually leading to a relatively rigorous size estimate and use of a cost model.