Lessons Learned in Estimating the Software Cost of a Ground Station with COTS Integration

Kathy Bradford
22 February 2001
Short History of an Integrated COTS Procurement

• RFP requested a mostly COTS ground station
• Original proposal effort based on IR&D experience:
  – IR&D experience not well quantified
  – COTS integration not separately identified or estimated
  – Script, 4GL code was assumed to be 4x as productive, so estimated SLOC counts were divided by 4
• Project was a success - completion was on schedule
  – But actuals greater than bid
  – Adjusted actuals - enhancements still greater than bid
• How do we do better next time?
Standard TRW S&ITG Process for Auditing Software Productivity

- Detailed code count
- Identify COTS Products
- Identify Development Environment
- Map WBS to Common Model
- Identify Project Milestones
- Allocate Shared Support Functions
- Parametric Model of Software Size, Effort, Schedule
- Detailed Actual Cost, Effort by Subsystem, Phase, Function
Setting Up the Project Baseline

• Established 1 page outline of project
  – Major Milestones
    • ATP and DD250
    • Technical reviews
    • Incremental delivery schedule
    • Formal test events
  – IPT (product & functional) organization
    • Contractors, subcontractors
  – Major elements of cost (Product/Subsystems, Sites, Deliveries)
    • Major COTS elements by subsystem
Analysis of Actuals

• Map project work breakdown structure (WBS) to standard model
  – Standard WBS used to archive, compare completed projects
  – Has subsets by function for differentiating between products or subsystems

• Used map to resummarize effort, cost
  – Determine cost of each Product (subsystem)
  – Split development into H/W and S/W efforts
  – Provides framework for quantifying and analyzing effort by development phase

• Used time-phased actuals, program profile
  – Determine headcount by phase, product, function
Detailed Code Count: Will the real code stand up?

- Completed products captured in the project Configuration Management (CM) data base
- COTS Integration systems have LOTS of file types
- Differentiated contents by file extension:
  - Code include 3GL, 4GL source and scripts
  - Tables are info for COCOMO sizing (database size)

<table>
<thead>
<tr>
<th>File Extensions by Type</th>
<th>No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>3GL (C, C++ awk)</td>
<td>6</td>
</tr>
<tr>
<td>4GL (screen builders, code builders)</td>
<td>4</td>
</tr>
<tr>
<td>Scripts (make, shell, vendor specific)</td>
<td>13</td>
</tr>
<tr>
<td>Tables (vendor specific)</td>
<td>12</td>
</tr>
<tr>
<td>Intermediates (bin, dat, o, out, etc.)</td>
<td>23</td>
</tr>
<tr>
<td>Total file types:</td>
<td>58</td>
</tr>
</tbody>
</table>
Code Counting Terminology, Methods

- Source files include 3GL, 4GL language; scripts such as make files, sh/csh/ksh, COTS-specific scripting languages
- LOC = Lines of code, defined by line terminator, but not a comment or a blank line
- SLOC = Source LOC, executable statement as defined in context of the language

<table>
<thead>
<tr>
<th>Product</th>
<th>CM Files</th>
<th>Source Files</th>
<th>LOC</th>
<th>SLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2,770</td>
<td>333</td>
<td>118.6</td>
<td>46.8</td>
</tr>
<tr>
<td>B</td>
<td>264</td>
<td>196</td>
<td>24.9</td>
<td>22.2</td>
</tr>
<tr>
<td>C</td>
<td>1,118</td>
<td>172</td>
<td>44.0</td>
<td>18.4</td>
</tr>
<tr>
<td>D</td>
<td>810</td>
<td>172</td>
<td>28.8</td>
<td>18.9</td>
</tr>
<tr>
<td>E</td>
<td>2,616</td>
<td>643</td>
<td>176.5</td>
<td>78.4</td>
</tr>
<tr>
<td>F</td>
<td>866</td>
<td>128</td>
<td>55.0</td>
<td>16.7</td>
</tr>
<tr>
<td>G</td>
<td>49</td>
<td>44</td>
<td>12.7</td>
<td>12.4</td>
</tr>
<tr>
<td>H</td>
<td>3,602</td>
<td>293</td>
<td>60.7</td>
<td>38.8</td>
</tr>
<tr>
<td>Total</td>
<td>12,095</td>
<td>1,981</td>
<td>521.2</td>
<td>252.7</td>
</tr>
</tbody>
</table>
Further identification required - differentiating New from Improved

• More terminology:
  – Reuse - source code created else where, and used with some reengineering and retest effort.
  – Modified - reuse code with some changes or additions to the code, as well as reengineering and retest effort.
  – Derivative code - multiple source files derived from one original with some % modification
  – Non-product code - tests or stubs, not part of final delivery

• Summarized, counted code inspected by IPT leads:
  – Identified reuse, derivative code products and original source
  – Non-product code identified and eliminated from counts
  – Verified derivative parameters (%redesign, %recode, %retest)
Interviews result in better definition of the real work performed

<table>
<thead>
<tr>
<th>Product</th>
<th>SLOC</th>
<th>New</th>
<th>Reuse</th>
<th>Derivative</th>
<th>ESLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>46.8</td>
<td>12.8</td>
<td>34.0</td>
<td></td>
<td>16.6</td>
</tr>
<tr>
<td>B</td>
<td>22.2</td>
<td>22.2</td>
<td></td>
<td></td>
<td>22.2</td>
</tr>
<tr>
<td>C</td>
<td>18.4</td>
<td>16.1</td>
<td>2.3</td>
<td></td>
<td>16.3</td>
</tr>
<tr>
<td>D</td>
<td>18.9</td>
<td>18.9</td>
<td></td>
<td></td>
<td>18.9</td>
</tr>
<tr>
<td>E</td>
<td>78.4</td>
<td>69.6</td>
<td>8.8</td>
<td></td>
<td>69.8</td>
</tr>
<tr>
<td>F</td>
<td>16.7</td>
<td>16.7</td>
<td></td>
<td></td>
<td>16.7</td>
</tr>
<tr>
<td>G</td>
<td>12.4</td>
<td>4.9</td>
<td>7.5</td>
<td></td>
<td>5.9</td>
</tr>
<tr>
<td>H</td>
<td>38.8</td>
<td>33.0</td>
<td>5.8</td>
<td></td>
<td>33.8</td>
</tr>
<tr>
<td>Total</td>
<td>252.7</td>
<td>194.3</td>
<td>42.8</td>
<td>15.6</td>
<td>200.2</td>
</tr>
</tbody>
</table>
COTS products count, too.

- Terminology again:
  - COTS - a product purchased or supplied, which is used as delivered. Most products supply some well-defined customization interfaces; but most of capability is in proprietary, execute only code.

- IPT Leads verify COTS products integrated into product:

- COTS size in FPs; sources include:
  - Vendor (rarely)
  - Comparison to well-known product lines, knowledge bases
  - Entries in reference manual index, or function list
  - Best to compare results of methods 2 and 3.

- Product Functionality required (in % of total functions)
  - Search on well-known function names
  - Inspection of index for known elements
  - Product quality, support, knowledge characteristics
    - Maturity vs. volatility
    - Vendor responsiveness
    - Ease of use, familiarity
IPT leads also define the characteristics of the development environment, staff

- Questionnaires derived from popular models:
  - SEER-SEM, COCOMO II, COCOTS, Price-S
  - Staff experience levels
  - Host/Target complexity, stability/volatility
  - Requirements completeness, volatility

- Parametric model of delivered software product created:
  - Use new, reuse, derivative code sizes
  - Development environment parameters as supplied by IPT leads
  - Constraints as identified in program plan, IPT interviews

- Compare modeled effort and schedule to actuals
Actual vs Model of Delivered Product

A calibrated model provides:

• Validation of provided data
• Basis of estimate for future enhancements

<table>
<thead>
<tr>
<th>Product</th>
<th>% Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>12%</td>
</tr>
<tr>
<td>B</td>
<td>-9%</td>
</tr>
<tr>
<td>C</td>
<td>31%</td>
</tr>
<tr>
<td>D</td>
<td>4%</td>
</tr>
<tr>
<td>E</td>
<td>20%</td>
</tr>
<tr>
<td>F</td>
<td>-43%</td>
</tr>
<tr>
<td>G</td>
<td>0%</td>
</tr>
<tr>
<td>H</td>
<td>6%</td>
</tr>
<tr>
<td>Total</td>
<td>7%</td>
</tr>
</tbody>
</table>

• Product F was forced to restart with alternate COTS product 28% into the schedule.

• Product C was severely constrained by staffing problems.

• Difference <2% with these corrections
Creating Modified Model of Original Proposal

• Needed to verify the “improved” process
• Created another parametric model using:
  – Original 3GL, 4GL and script code from the proposal
    • In proposal, script and 4GL assumed to be 25% as hard, so were counted 4:1 in models
    • Corrected script, 4GL lines counts back to original estimate
  – COTS function counts, parameters from IPT interviews
    • Used SEER-SEM “quicksize” of COTS products to estimate impact of COTS integration
• Compared the proposal to the Quicksize model, and to the actuals and model of the delivered product
Comparison of Proposal to Other Methods

- 7 of 8 products are far closer to reality using the modified modeling methods.
Quicksize using COTS FP arrives at Effort and Schedule within 2% of Actuals

Comparison of Models to Proposed and Actual Effort (not normalized - scaled to effort)
What Next?

• Using COTS FPs and experience parameters, created a much better model of this project
• More COTS integration programs are nearing delivery now; verify the process, productivity profiles on other efforts
• Use the results to help current program management:
  – Use results to help quantify effects of mid-term COTS fall-out
  – Use the history and models to better determine intermediate milestones
• Use the results to establish better cost-risk profiles for COTS integration programs