COTS Integration

- Potential Benefits
  - Fast
  - Cheap
  - Known functionality

- Potential Risks
  - Architectural mismatch
  - Excess functionality ("bloatware")
  - Vendor defines features and interfaces
Origins of the Volatility Problem

- Vendor releases upgraded component
- Developers rework other components to restore compatibility of interfaces and functions
- Vendors usually don’t coordinate their upgrades

Quantifying the Costs of COTS Volatility

POTENTIAL COUPLING OF 8 NODES

Total two-way paths = 8 * 7 = 56
Changing one component may affect 7 interfaces.
Focus of the Study

- Objectives
  - Provide a simple model to estimate COTS volatility costs.
  - Support quantitative risk assessment
  - Include volatility costs in total project costs

- Approach
  - Build model based on a few parameters
  - Consider how to calibrate the model

Factors to Consider

- Number of components
- "Coupling" of the components (scope, difficulty)
- Costs of analysis, modification, retesting
- Expected release frequency
- Sequencing of releases relative to the development cycle
Notation

\( N = \) total modules in system
\( i = \) updated module (new release)
\( L_i = \) number of modules linked to module \( i \)
\( l = \) module linked to module \( i \)
\( M_i = \) number of modules modified to accommodate the new version of module \( i \)
\( m = \) module modified
\( E = \) effort

Cost Components for a Single Release

\[ E = E_{\text{analysis}} + E_{\text{modify}} + E_{\text{test}} \]

where, assuming the same effort per module,

\[ E_A = L \ast E_a \]
\[ E_M = M \ast E_m \]
\[ E_T = E_t \ast [M \ast (M-1)/2 + M \ast (N-M)] \]
The Costs Depend on Difficulty and Scope

- Analysis and Design Effort
  - None (changes to error message, prompt, value of a constant)
  - Easy (modifications straightforward to define and implement)
  - Hard (involves complex functions and/or interfaces to other modules or systems, performance constraints or tradeoffs).
  (Relates to COCOMO's SU parameter.)

- Modification Effort (parameters or lines added, changed, deleted)
  - Few items (<= 5)
  - Many items
  - All items (includes a total rewrite or adding new module)
  (Relates to COCOMO's AA and AAF* parameters.)

* Probably irrelevant since have no source code for most COTS components

Quantifying the Costs of COTS Volatility
Combine Analysis and Modification Costs

<table>
<thead>
<tr>
<th>Analysis</th>
<th>Modify</th>
<th>Few</th>
<th>Many</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>VL</td>
<td>LO</td>
<td>ME</td>
<td></td>
</tr>
<tr>
<td>Easy</td>
<td>LO</td>
<td>ME</td>
<td>HI</td>
<td></td>
</tr>
<tr>
<td>Hard</td>
<td>ME</td>
<td>HI</td>
<td>VH</td>
<td></td>
</tr>
</tbody>
</table>

This gives 5 "difficulty" values to define via calibration*

* Mapping VL to LO, and VH to HI gives only 3 values.

Quantifying the Costs of COTS Volatility

Initial Model: Assumptions

- All releases of a particular component have a single characteristic difficulty, $D_i$, and number of links, $L_i$

- The efforts are the same for all releases of a component

- $E_i$ scales linearly with $L_i$ (i.e., no quadratic terms)
Initial Model: Formulation

\[ E_{Total} = \sum_{i=1}^{N_{comp}} N_{R}(i) E_i(D_i) L_i \]

where \( E_i(D_i) = E_a(D_i) + E_m(D_i) + E_t(D_i) \)

This model has 3 parameters *per component.*

Simplified Model

- Further assumptions
  - All components require the same effort \([E_i(D_i) \rightarrow E(D_i)]\)
  - Replace \( L_i \) by \( f(S_i) \)

- Formulation

\[ E_{Total} = \sum_{D=VL}^{VH} E(D) \sum_{S=One}^{All} N_{REL}(D, S) f(S) \]

This model has only 15 inputs \([N_{REL}(D,S)]\)
The Function $E(D)$

Treat $E(D)$ similarly to adapted code:

$$E(D) = E_{Glue} \times EAF(D)$$

where

$$E_{Glue} = \text{average effort to analyze, modify and test one link for a particular release of a specific component.}$$

and we choose equally spaced values for $EAF(D)$:

<table>
<thead>
<tr>
<th>$D$</th>
<th>$EAF(D)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>VL</td>
<td>0.0</td>
</tr>
<tr>
<td>LO</td>
<td>0.4</td>
</tr>
<tr>
<td>ME</td>
<td>0.8</td>
</tr>
<tr>
<td>HI</td>
<td>1.2</td>
</tr>
<tr>
<td>VH</td>
<td>1.6</td>
</tr>
</tbody>
</table>

Quantifying the Costs of COTS Volatility

The Function $f(S)$

$S$ reflects the number of modules "touched", $L$.

Assume that $L_{max} \sim 20$.

Assume that effort scales as $L$.

<table>
<thead>
<tr>
<th>$S$</th>
<th>Range</th>
<th>$&lt;L&gt;$</th>
<th>Rounded</th>
</tr>
</thead>
<tbody>
<tr>
<td>One</td>
<td>0-1</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>Several</td>
<td>2-9</td>
<td>5.5</td>
<td>6</td>
</tr>
<tr>
<td>All</td>
<td>10-20</td>
<td>15.0</td>
<td>15</td>
</tr>
</tbody>
</table>

Quantifying the Costs of COTS Volatility
Usage of the Model

(1) For each COTS component, estimate:
   1. Values of A and M, and use table to get D
   2. Value of S (or $L_i$)
   3. The number of releases during the project

(2) For the project, estimate the effort to construct an average link, $E_{Glue}$

(3) Sum the values for the products to get $N_{REL}(D,S)$

(4) Compute $E_{TOTAL}$

Validation and Calibration

• We need to collect actual project data to:
  – Validate the many assumptions
  – Calibrate the coefficients ($EAF(D)$, $f(S)$ and possibly $E_{GLUE}$)

• Data needed for each modification:
  – Links analyzed and modified ($L_i$ and $M_i$)
  – Total effort to analyze, to modify, and to test ($E_a$, $E_m$, $E_t$)
  plus, for each component:
    – Difficulty ($D$)
    – Scope ($S$)
  plus
  – Total components and total modules ($N_{comp}$, $N$)
Modification Costs Increase with Time

- Reference Boehm's data [Boehm, 1989]
- This will force disaggregation of the model since each release occurs at a different time (unsynchronized)
- Reformulating using continuous distributions may be a way to obtain some level of parameterization

Simple Models for Time Dependence

- Assume cost doubles at each release:
  \[ E_i = E_i(0) \sum_{j=1}^{N_{REL}(i)} (2^{j-1}) = E_i(0) \left(2^{N_{REL}(i)} - 1\right) \]
- Assume cost tracks the number of completed modules:

  ![Graph showing cost over time]

  Early releases are cheaper to incorporate.
Implications for Project Management

- Choose COTS components that are stable
  - Complete (full featured)
  - Proven capability (mature)
  - Decreases N_R(i)

- Choose COTS components that are compatible
  - Architecturally matched
  - Decreases effort needed to analyze, modify, and test

- Design the product to reduce coupling
  - Choose a “clean” partitioning
  - Isolate volatile components via bindings
  - Decreases L_4

Quantifying the Costs of COTS Volatility

Implications for Project Management (continued)

- Shorten the development cycle or exposure time
  - Accept no upgrades after PDR or Code Complete
  - Decreases N_R(i)

- Commit to specific versions
  - Not “the most current release”
  - Decreases N_R(i)

- Handle modifications in batches
  - A “pathfinder team” evaluates and integrates a set of releases
  - The pathfinders prepare for the cutover (“advancemanship”)”
  - The main team makes the changes at an opportune time
  - Reduces the effort to implement each link