Quality Cost Tradeoff Model
Calibrated with TSP Data

William Nichols
Software Engineering Institute

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213
Quality Cost Tradeoff Model Calibrated with TSP Data

The Problem

Managing Software Project Cost, Schedule, Scope, and Quality Outcomes
TSP Defect Model

Is similar to COQUALMO in some key ways, but also has differences

1) Defect fix effort is explicitly quantified,

2) We calibrate with local TSP measurement (using TSP parameters)

As with COQUALMO, the assumption is that there is an underlying causal mechanism in software development. This leads us to the SCOPE project.
The Iron Triangle, (Quality and Beyond)

Fix two factors to determine the third

The Role of quality is misunderstood

The Iron Triangle represents the relationship among three interrelated variables, cost, schedule, and scope.

Quality is not included because it is not a simple trade-off. But,

- How does quality affect cost, schedule, and scope? (COQUALMO?)
- What factors determine quality? (SCOPE)
Defect Reduction Top 10 List

Expert option suggests mechanisms, some of which are causal.

Finding and fixing a software problem after delivery is often 100 times more expensive than finding and fixing it during the requirements and design phase. $(\text{find defects sooner may save money})$

About 80 percent of avoidable rework comes from 20 percent of the defects. $(\text{the vital few, but which ones are they?})$

Peer reviews catch 60 percent of the defects. $(\text{do we do enough?})$

Disciplined personal practices can reduce defect introduction rates by up to 75 percent. $(\text{prevention pays, but what does it cost?})$

All other things being equal, it costs 50 percent more per source instruction to develop high-dependability software products than to develop low-dependability software products. However, the investment is more than worth it if the project involves significant operations and maintenance costs. $(\text{where is the payoff tipping point?})$
A Mechanistic Model relates Effort, Product, and Defects

Product = ProdRate * Time
Defects = DefRate * Time

Defects_Removed = Defects_Present * RemovalYield

Similar to Jones “Tank and Filter”,
Simplifies assumptions found in Boehm/Chulani COQUALMO
Integrates defect and cost projections

Quality Cost Tradeoff Model Calibrated with TSP Data

Calibrate the Model Locally

With TSP Data
Close the “Cost Loop” with Rework Effort

Model how defects get into the system, And how they are removed

This is what we want to avoid. What are the control points?

Average Defect Removal Time by Activity

<table>
<thead>
<tr>
<th>Activity</th>
<th>Time in Minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design Review</td>
<td>5</td>
</tr>
<tr>
<td>Design Inspect.</td>
<td>22</td>
</tr>
<tr>
<td>Code Review</td>
<td>2</td>
</tr>
<tr>
<td>Code Inspect.</td>
<td>25</td>
</tr>
<tr>
<td>Unit Test</td>
<td>32</td>
</tr>
<tr>
<td>System Test</td>
<td>1405</td>
</tr>
</tbody>
</table>

Source: Xerox

Defect-removal Phase
Collect Project Level Data for Effort and Defects

Plan/Actual

Phase Effort

Defects Injected is predictable

Defects Removed is optional and variable
Calibrate with local data

Below are scatterplots of predicted and actual hours of effort found in individual components for a single project. Similar data is available for defects injected and removed. The data is variable, but predictable. Repeatable is one indication of an underlying causal structure, We will take advantage of this in SCOPE.
Where does the data come from?

>900 projects planning cycles of data for benchmarking

(Bring your own data!) Projects bootstrap by collecting local data to update the models.

<table>
<thead>
<tr>
<th>Defect injection rates and removal yields</th>
<th>Code Rate [LOC/Hr]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defect density (defects and size)</td>
<td>Phase Removal Yield [% removed]</td>
</tr>
<tr>
<td>Review rates</td>
<td>Zero Defect Test time [Hr]</td>
</tr>
<tr>
<td>Phase Injection Rate [defects/Hr]</td>
<td>Phase “Find and Fix” time [Hr/defect]</td>
</tr>
<tr>
<td>Phase Effort Distribution [%] total time</td>
<td>Review/Inspection Rate</td>
</tr>
<tr>
<td>Size [LOC]</td>
<td></td>
</tr>
</tbody>
</table>
Quality Cost Tradeoff Model Calibrated with TSP Data

Model Demonstration

Using the Model
Using the Model

Will you achieve your goals?
• How much do you want to deliver?
• What is your desired schedule?
• How many defects do you expect the user to find?

What do you know? Do you have relevant historical data?

What decisions can you make?
• Do you have enough staff?
• Do the staff have the right skills?
• How will you allocate the effort?
Compare our performance to a baseline
### Control Panel

<table>
<thead>
<tr>
<th></th>
<th>Rate [LOC/hr]</th>
<th>Yield (per insp)</th>
<th># Insps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design Review</td>
<td>200</td>
<td>50.0%</td>
<td>1</td>
</tr>
<tr>
<td>Design Inspection</td>
<td>200</td>
<td>50.0%</td>
<td>0</td>
</tr>
<tr>
<td>Code Review</td>
<td>200</td>
<td>50.0%</td>
<td>0</td>
</tr>
<tr>
<td>Code Inspection</td>
<td>200</td>
<td>50.0%</td>
<td>0</td>
</tr>
</tbody>
</table>

### Total Development and Test Time

- **Baseline**: Dev = 100, UT = 20, IT = 30, ST = 40
- **Revised**: Dev = 100, UT = 20, IT = 30, ST = 40

### Defect Density Phase Profile

- Baseline [Def/KLOC]
- Revised [Def/KLOC]
- Density Goal [Def/KLOC]

### Perform a personal design review

**Trial Parameter**

**Trial Outcomes**
Include a peer design review
### Control Panel

<table>
<thead>
<tr>
<th></th>
<th>Rate [LOC/hr]</th>
<th>Yield (per insp)</th>
<th># Insps</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design Review</strong></td>
<td>200</td>
<td>50.0%</td>
<td>1</td>
</tr>
<tr>
<td><strong>Design Inspection</strong></td>
<td>200</td>
<td>50.0%</td>
<td>1</td>
</tr>
<tr>
<td><strong>Code Review</strong></td>
<td>200</td>
<td>50.0%</td>
<td>1</td>
</tr>
<tr>
<td><strong>Code Inspection</strong></td>
<td>200</td>
<td>50.0%</td>
<td>0</td>
</tr>
</tbody>
</table>

### Total Development and Test Time

- **Baseline**
- **Revised**

### Defect Density Phase Profile

- **Baseline [Def/Kloc]**
- **Revised [Def/Kloc]**
- **Density Goal [Def/KLOC]**

---

**Have a peer inspect the code**
At some point we cross the “quality is free” boundary!
Crossing the “quality is free” point!
### Control Panel

<table>
<thead>
<tr>
<th>Rate [LOC/hr]</th>
<th>Yield (per insp)</th>
<th># Insps</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Design Review</strong></td>
<td>200</td>
<td>50.0%</td>
</tr>
<tr>
<td><strong>Design Inspection</strong></td>
<td>200</td>
<td>50.0%</td>
</tr>
<tr>
<td><strong>Code Review</strong></td>
<td>200</td>
<td>70.0%</td>
</tr>
<tr>
<td><strong>Code Inspection</strong></td>
<td>200</td>
<td>70.0%</td>
</tr>
</tbody>
</table>

### Total Development and Test Time

- **Baseline**
- **Revised**

### Defect Density Phase Profile

- Baseline [Def/Kloc]
- Revised [Def/Kloc]
- Density Goal [Def/KLOC]

Local parameters move the “quality is free” Boundary.
Conclusion

Implications and Future Work
Implications

Defect reduction improves project performance. *(quality is free?)*

By modeling the *causal* structural mechanisms

- Test the sensitivity and limits of *controllable factors*
- Examine “what-if” scenarios before work

What other factors for control should be included in cost models?

- Inspection effectiveness varies widely but is a teachable
- Training to affect developer performance ranges
- Design techniques, effort, documentation, quality
- Programming language or technology stack
- Effort allocation *(balance effectiveness with resource expenditure)*

Only matter if the relationship is *causal*.

**SCOPE** studies the causal nature and structural relationship of factors
Call for Data

SCOPE studies Causal Analysis with observational data from real projects. If you have data you can share, or If you want to learn more about Causal Analysis and structural modeling

Contact

Bill Nichols wrn@sei.cmu.edu
Bob Stoddard rws@sei.cmu.edu
Mike Konrad mdk@sei.cmu.edu
Quality Cost Tradeoff Model Calibrated with TSP Data

Section Title

Backup
Intuit Productivity Improvement

Faster, better, cheaper (reduced rework)

From data on over 40 TSP teams, Intuit has found that

- post code-complete effort is 8% instead of 33% of the project
- Standard test times were cut from 4 months to 1 month or less.

Organizations using TSP report productivity gains of 30% or more resulting in lower costs or more functionality in delivered software.

Source: Intuit
Duration/Effort to “project completion”

Duration and Effort by Method

Comparative Size Projects

Source: C. Jones http://www.infoq.com/articles/evaluating-agile-software-methodologies
Defect Levels for various development methods

Source: C. Jones http://www.infoq.com/articles/evaluating-agile-software-methodologies
Total Cost of Ownership

Comparable Size Projects

Cost by Method (comparable projects)

Source: C. Jones http://www.infoq.com/articles/evaluating-agile-software-methodologies
Application of Causal Analysis of PSP

DAG from PC search

DAG from FGES search

Common Direct Causal Edges

1. AsgAveMin → ConstMin
2. StuEffFactor → ConstMin
3. AsgAveMin → MinTot
4. StuEffFactor → MinTot
5. ConstMin → MinTot
6. MinTot → DefectTot
7. AsgAveMin → LOC
8. StuSizeFactor → LOC

<table>
<thead>
<tr>
<th>Expected Relation (log transformed for linear effects)</th>
<th>Edges found</th>
<th>PC</th>
<th>FGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \ln(LOC_{ij}) = \ln(ReqSize_i) + \ln(SSF_j) )</td>
<td>StuSizeFactor → LOC</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>AsgAveMin → LOC</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>( \ln(Min\text{Tot}_{ij}) = (ReqSize_i) + \ln(SEF_j) )</td>
<td>AsgAveMin → MinTot</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>StuEffFactor → MinTot</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>( \ln(Def\text{Tot}<em>{ij}) = \ln(ConstMin</em>{ij}) + \ln(StuDAR_j) )</td>
<td>StuDAR → DefectTot</td>
<td>Bi</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>Const → DefectTot</td>
<td>I</td>
<td>I</td>
</tr>
</tbody>
</table>