A cost-benefit framework for making architectural decisions in a business context

Jeromy Carriere  
Chief Architect, VistaPrint  
Rick Kazman and Ipek Ozkaya  
Software Engineering Institute

Background

• VistaPrint
• A belief in the promise of early-lifecycle cost-benefit analysis
  – Software is seldom (never?) developed from scratch
  – Ad-hoc or code-level cost-benefit is state-of-the-practice
Motivation

• Easy to place a value on revenue-generating projects
• Not easy to do the same for architectural transformations or refactoring
  – Risk reduction
  – Developer productivity
• We want architecture-based cost/benefit judgment
Approach

• Evaluate a completed project using
  – Tickets that document the work
  – Before and after codebase
  – Dependency structure matrix → coupling

• Train a model that can predict benefit based on a stream of classified changes

The Dependency Structure Matrix

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Dependency Structure Matrix

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Strength of A's dependency on B</td>
<td>Strength of B's dependency on A</td>
<td>Hidden slide – the previous builds to this</td>
</tr>
<tr>
<td>B</td>
<td>Strength of A's dependency on B</td>
<td>You get the idea...</td>
<td>Strength of C's dependency on B</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A (small) piece of VistaPrint’s DSM
Idealized process

Classification

- Price data
- Price test
- Tool
- Price logic
- Website UI
### Result ... the bad news

<table>
<thead>
<tr>
<th>Classification</th>
<th>Time</th>
<th>LOC</th>
<th>Coupling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price Test</td>
<td>-3%</td>
<td>11%</td>
<td>-19%</td>
</tr>
<tr>
<td>Price Data</td>
<td>61%</td>
<td>166%</td>
<td>-100%</td>
</tr>
<tr>
<td>Website UI</td>
<td>-54%</td>
<td>-73%</td>
<td>99%</td>
</tr>
<tr>
<td>Tool</td>
<td>-79%</td>
<td>-76%</td>
<td>271%</td>
</tr>
<tr>
<td>Price Logic</td>
<td>31%</td>
<td>5%</td>
<td>248%</td>
</tr>
</tbody>
</table>

In other words

<table>
<thead>
<tr>
<th>Classification</th>
<th>Time</th>
<th>LOC</th>
<th>Coupling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price Test</td>
<td>⬇</td>
<td>⬆</td>
<td>⬆</td>
</tr>
<tr>
<td>Price Data</td>
<td>⬆</td>
<td>⬆</td>
<td>⬇</td>
</tr>
<tr>
<td>Website UI</td>
<td>⬇</td>
<td>⬇</td>
<td>⬆</td>
</tr>
<tr>
<td>Tool</td>
<td>⬇</td>
<td>⬇</td>
<td>⬆</td>
</tr>
<tr>
<td>Price Logic</td>
<td>⬆</td>
<td>⬆</td>
<td>⬆</td>
</tr>
</tbody>
</table>
Why?

- The classification was wrong
- Ticket scope wasn’t normalized
Result ... the good news

<table>
<thead>
<tr>
<th></th>
<th>Number of tickets</th>
<th>Total LOC</th>
<th>LOC per ticket</th>
<th>Average time per ticket</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before</td>
<td>129</td>
<td>31035</td>
<td>240</td>
<td>8336</td>
</tr>
<tr>
<td>After</td>
<td>156</td>
<td>25162</td>
<td>161</td>
<td>5781</td>
</tr>
<tr>
<td>Improvement</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Improvement</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Okay, then

Anecdotal evidence

- Easier integration of new users of pricing (decreased coupling to concrete implementation)
  - Price a basket
  - Third party pricing
- Easier integration of new pricing logic (consolidation)
  - Shipping pricing
- Correctness
  - Actually pricing accurately
- New capabilities
  - Managing discounts
  - Predictive modeling for cross-sell discount
  - Dynamic discounts
- Reduced error rate
- Improved testability
  - Unit testing
  - QA automation
So?

- Effort by LOC by category isn’t informative or architecturally significant
- Anecdotal evidence and coarse effort analysis is positive
- Consider *propagation cost* to look for correlation

Propagation cost

- Start with the DSM, call it $M$
- Compute $M_P$ – visibility based on pricing only
- Then,
  \[ V = M^0 + M + \sum_{i=2}^{4} M^{i-1} M^P \]
- Transform V to 1’s and 0’s
- Compute propagation cost as density of V
Propagation cost example

\[ M^0 \]
\[
\begin{array}{cccc}
A & B & P1 & C \\
A & 1 & & \\
B & & 1 & \\
P1 & 1 & & \\
C & & & 1 \\
P2 & & & 1 \\
\end{array}
\]

\[ M^1 \]

\[ M \]

\[ M_P \]

\[ M^2 \]

\[ M^3 \]

\[ M^4 \]

\[ V \]

\[
\begin{array}{cccc}
A & B & P1 & C \\
A & 1 & 1 & 1 \\
B & 1 & 1 & 1 \\
P1 & 1 & 1 & 1 \\
P2 & 1 & 1 & 1 \\
\end{array}
\]

Propagation cost = 21/25 = 84%
Before (/during)

- Propagation cost: 20%

Before

- Propagation cost: 15%
After

• Propagation cost: 6%
**Actions**

- Collect different data: actionable, timely
- **New ticket classification, in-the-moment**
  - Project-/area-specific
  - Universal: arrival frequency, projected effort
  - ISO/IEC 14764 maintenance classification probably not a fit
- **Actual effort**

**Workflow**

- New
- Accepted
- Investigating
- In Progress
- Awaiting
- Test
- ...
Workflow

Initial design and scoping
Opportunity: “force”
ticket size normalization,
classification

New
Accepted
Investigating
In Progress
Awaiting Test

... 

Future work

• Immediate
  – Add propagation cost to metrics tracking
  – Finish defining new measures
  – Collect new measures during the process
  – Apply the model to upcoming projects, refine

• Medium term
  – Option model

• Speculative
  – Concept clustering on ticket descriptions