Estimating the Principal of Technical Debt

Bill Curtis, Jay Sappidi, & Alexandra Szynkarski
CAST Research Labs
June 5, 2012

The Technical Debt Metaphor

**Technical Debt** — the future cost of defects remaining in code at release, a component of the cost of ownership

- **Business Risk**
  - Opportunity cost
  - Liability from debt

- **Technical Debt**
  - Interest on the debt
  - Principal borrowed

- **Structural quality problems in production code**

- **Opportunity cost** — benefits that could have been achieved had resources been put on new capability rather than retiring technical debt

- **Liability** — business costs related to outages, breaches, corrupted data, etc.

- **Interest** — continuing IT costs attributable to the violations causing technical debt, i.e., higher maintenance costs, greater resource usage, etc.

- **Principal** — cost of fixing problems remaining in the code after release that must be remediated

*Today’s talk focuses on the principal*
Inputs for Estimating the Principal of Technical Debt

Data source

- Static analysis of applications
- Historical data on maintenance
- IT or contractor finance records

Inputs

- Structural quality problems
- Hours to correct problems
- Developer’s burdened hourly rate

Governance Dashboard

Project Trends

Drill Down Portal

CAST Application Intelligence Platform

ANALYZERS

- Oracle PL/SQL
- Sybase T-SQL
- SQL Server T-SQL
- IBM SQL/PSM
- C, C++, C#
- Pro C
- Cobol
- CICS
- Visual Basic
- VB.Net
- ASP.Net
- Java, J2EE
- JSP
- XML, HTML
- Javascript
- VBScript
- PHP
- PowerBuilder
- Oracle Forms
- PeopleSoft
- SAP ABAP
- Netweaver
- Tibco
- Business Objects
- Universal Analyzer

APP KNOWLEDGE BASE

APPLICATION HEALTH

- Risk Factors
  - Robustness
  - Performance
  - Security

- Cost factors
  - Transferability
  - Changeability

APPLICATION SIZE

- LOC
- Function Points

Application Metadata

Analysis of all system artifacts

CAST Confidential
Appmarq — CAST’s Structural Quality Repository

- Industry-leading repository on structural quality
  - 745 Applications
  - 160 Companies, 14 Countries
  - 321,259,160 Lines of Code; 59,511,706 Violations

<table>
<thead>
<tr>
<th>Kilo (thousands) of Lines of Code (KLOC)</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-20</td>
<td>60</td>
</tr>
<tr>
<td>20-50</td>
<td>110</td>
</tr>
<tr>
<td>50-100</td>
<td>121</td>
</tr>
<tr>
<td>100-200</td>
<td>122</td>
</tr>
<tr>
<td>200-500</td>
<td>149</td>
</tr>
<tr>
<td>500-1K</td>
<td>82</td>
</tr>
<tr>
<td>1K-5K</td>
<td>86</td>
</tr>
<tr>
<td>5K+</td>
<td>7</td>
</tr>
</tbody>
</table>

Formulas for Estimating Technical Debt Principal

<table>
<thead>
<tr>
<th></th>
<th>% Violations to be fixed</th>
<th>Hours to Fix</th>
<th>Cost /Hour</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Old</td>
<td>New</td>
<td>Old</td>
</tr>
<tr>
<td>High Severity</td>
<td>50%</td>
<td>100%</td>
<td>1</td>
</tr>
<tr>
<td>Medium Severity</td>
<td>25%</td>
<td>50%</td>
<td>1</td>
</tr>
<tr>
<td>Low Severity</td>
<td>10%</td>
<td>0%</td>
<td>1</td>
</tr>
</tbody>
</table>

Estimated Technical Debt Principal =
(Σ high severity violations) X (% to be fixed) X (average hours to fix) X ($s per hour) +
(Σ medium severity violations) X (% to be fixed) X (average hours to fix) X ($s per hour) +
(Σ low severity violations) X (% to be fixed) X (average hours to fix) X ($s per hour)

- This is an estimate of Technical Debt Principal
- Customers can get more accurate estimates by adjusting the parameters in the equation
### Technical Debt Principal Estimates for Both Formulas

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Median</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Std. Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Old</td>
<td>New</td>
<td>Old</td>
<td>New</td>
<td>Old</td>
</tr>
<tr>
<td>Sample (n=744)</td>
<td>3.61</td>
<td>10.26</td>
<td>2.79</td>
<td>7.94</td>
<td>0.02 0.01</td>
</tr>
<tr>
<td>.NET (n=63)</td>
<td>3.09</td>
<td>12.29</td>
<td>2.37</td>
<td>10.20</td>
<td>0.96 0.49</td>
</tr>
<tr>
<td>ABAP (n=72)</td>
<td>0.43</td>
<td>1.90</td>
<td>0.41</td>
<td>1.73</td>
<td>0.05 2.00</td>
</tr>
<tr>
<td>C (n=44)</td>
<td>2.62</td>
<td>7.65</td>
<td>2.18</td>
<td>6.46</td>
<td>0.02 0.01</td>
</tr>
<tr>
<td>C++ (n=30)</td>
<td>4.33</td>
<td>12.95</td>
<td>2.41</td>
<td>7.83</td>
<td>0.02 0.01</td>
</tr>
<tr>
<td>JavaEE (n=474)</td>
<td>5.42</td>
<td>14.68</td>
<td>5.13</td>
<td>13.66</td>
<td>0.07 0.23</td>
</tr>
<tr>
<td>Or-Forms (n=45)</td>
<td>4.57</td>
<td>21.16</td>
<td>1.12</td>
<td>3.87</td>
<td>0.49 1.13</td>
</tr>
<tr>
<td>V. Basic (n=16)</td>
<td>2.93</td>
<td>9.83</td>
<td>2.58</td>
<td>8.37</td>
<td>0.68 2.77</td>
</tr>
</tbody>
</table>

### Estimates of Technical Debt Principal by Health Factor

- **70% of Technical Debt is in IT Cost**
  - (Transferability, Changeability)
- **30% of Technical Debt is in Business Risk**
  - (Robustness, Performance, Security)
- **Health Factor proportions are mostly consistent across technologies**
Relating Technical Debt to Business Value

- **Health Factor**
  - Robustness
  - Performance
  - Security
  - Transferability
  - Changeability

- **Operational problems**
  - Outages, slow recovery
  - Degraded response
  - Breaches, Theft
  - Lengthy comprehension
  - Excessive effort

- **Output Measure**
  - Availability
  - Work efficiency
  - Data protection
  - IT productivity
  - Delivery speed

Technical Debt Management Cycle

1. **IT Executives**
   - Set policy and quality priorities

2. **Application Managers**
   - Set reduction targets & plans

3. **Developers**
   - Measure Technical Debt

4. **Build/Release/QA/Al Center**
   - Plan actions for remediation

5. **Step 5**
   - Remediate violations

6. **Step 6**
   - Track results

7. **Step 7**
   - Report to the business