Technical Debt and Requirements

Neil Ernst
University of British Columbia
@neilernst • neil@neilernst.net • neilernst.net
Overview

- Requirements represent product’s business value and quality goals.
- “Technical debt is acquired when engineers take shortcuts that fall short of best practices.” -- Eric Allman, CACM 55(5)

Short-cuts in requirements phase(s) a source of Technical Debt.
2008

Firefox
2008

Firefox

2012

SunSpider total (smaller is better)
Characteristics
Characteristics

• **Benefit** = Time saved *not* doing requirements work.
Characteristics

• **Benefit** = Time saved **not** doing requirements work.

• **Risk** = Possibility that we miss intent, leading to “rework”.

Wednesday, 6 June, 12
Characteristics

- **Benefit** = Time saved **not** doing requirements work.
- **Risk** = Possibility that we miss intent, leading to “rework”.
- **Interest** = Misunderstood req in V1 might lead to further misses in V2, if requirements build on top of one another.
Characteristics

- **Benefit** = Time saved *not* doing requirements work.
- **Risk** = Possibility that we miss intent, leading to “rework”.
- **Interest** = Misunderstood req in V1 might lead to further misses in V2, if requirements build on top of one another.
- **Repayment** = Reprioritize, re-analyse, process improvements.
Should have compiled ALL Javascript to begin with
Requirements Debt

Design Debt

Code Debt

Should have compiled ALL Javascript to begin with

Rearchitect to support base compilation
Requirements Debt

Design Debt

Code Debt

Should have compiled ALL Javascript to begin with

Rearchitect to support base compilation

Implement new code; test; deliver

Wednesday, 6 June, 12
Requirements Debt

Design Debt

Code Debt

Interest

Should have compiled ALL Javascript to begin with

Rearchitect to support base compilation

Implement new code; test; deliver

Wednesday, 6 June, 12
TD in Requirements

- Technical debt incurred when we do not conduct “sufficient” requirements analysis:
  - we gamble that more elicitation or analysis will not help,
  - because that issue may not even be relevant!
  - If it is relevant, than we go back and fix it.
- Key business decision: what is sufficient?
- Can tools help.
Other Examples

- “TBDs and maintenance” (MTD 10)
- Risk analysis and mitigation (JPL)
- Evolving user stories (SAP)
Optimizing decisions

• At start of iteration question is “what is the best trajectory to pick”?
• What is best set of ‘work items’ to prioritize?
• RE-KOMBINE automatically calculates the optimal strategy for satisfying the given set of requirements.
• Relations between requirements matter.
Surfacing requirements debt

• Mine repositories for requirements data
• Track usage data
<table>
<thead>
<tr>
<th>Command</th>
<th>Executions</th>
</tr>
</thead>
<tbody>
<tr>
<td>edit.Delete</td>
<td>5.4 M</td>
</tr>
<tr>
<td>file.Save</td>
<td>4.3 M</td>
</tr>
<tr>
<td>edit.Paste</td>
<td>3.8 M</td>
</tr>
<tr>
<td>edit.Copy</td>
<td>2.4 M</td>
</tr>
<tr>
<td>ContentAssist.proposals</td>
<td>1.4 M</td>
</tr>
<tr>
<td>Command</td>
<td>Executions</td>
</tr>
<tr>
<td>------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>edit.Delete</td>
<td>5.4 M</td>
</tr>
<tr>
<td>file.Save</td>
<td>4.3 M</td>
</tr>
<tr>
<td>edit.Paste</td>
<td>3.8 M</td>
</tr>
<tr>
<td>edit.Copy</td>
<td>2.4 M</td>
</tr>
<tr>
<td>ContentAssist.proposals</td>
<td>1.4 M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Executions</th>
</tr>
</thead>
<tbody>
<tr>
<td>window.previousView</td>
<td>9</td>
</tr>
<tr>
<td>navigate.Back</td>
<td>69</td>
</tr>
<tr>
<td>window.showViewMenu</td>
<td>89</td>
</tr>
<tr>
<td>window.previousPerspective</td>
<td>155</td>
</tr>
<tr>
<td>window.previousEditor</td>
<td>166</td>
</tr>
</tbody>
</table>

Data: Eclipse UPP, 200908, eclipse.ui, 3.5.0
Summary

- Debt is incurred when we do not do sufficient requirements work.
- Requirements capture value, and should be first-class citizens in software development.
- Support dev in understanding how software is meeting business and quality goals.
- Tracking historical tendency, we can improve our understanding of the problem space(s).
Research Directions

1. What is the relationship between process debt and requirements debt?
2. Analysis-paralysis vs. wearing blinders
3. Transitioning from ‘agile’ requirements to up-front design.
4. How do we track requirements debt?

Neil Ernst: @neilernst • neilernst.net