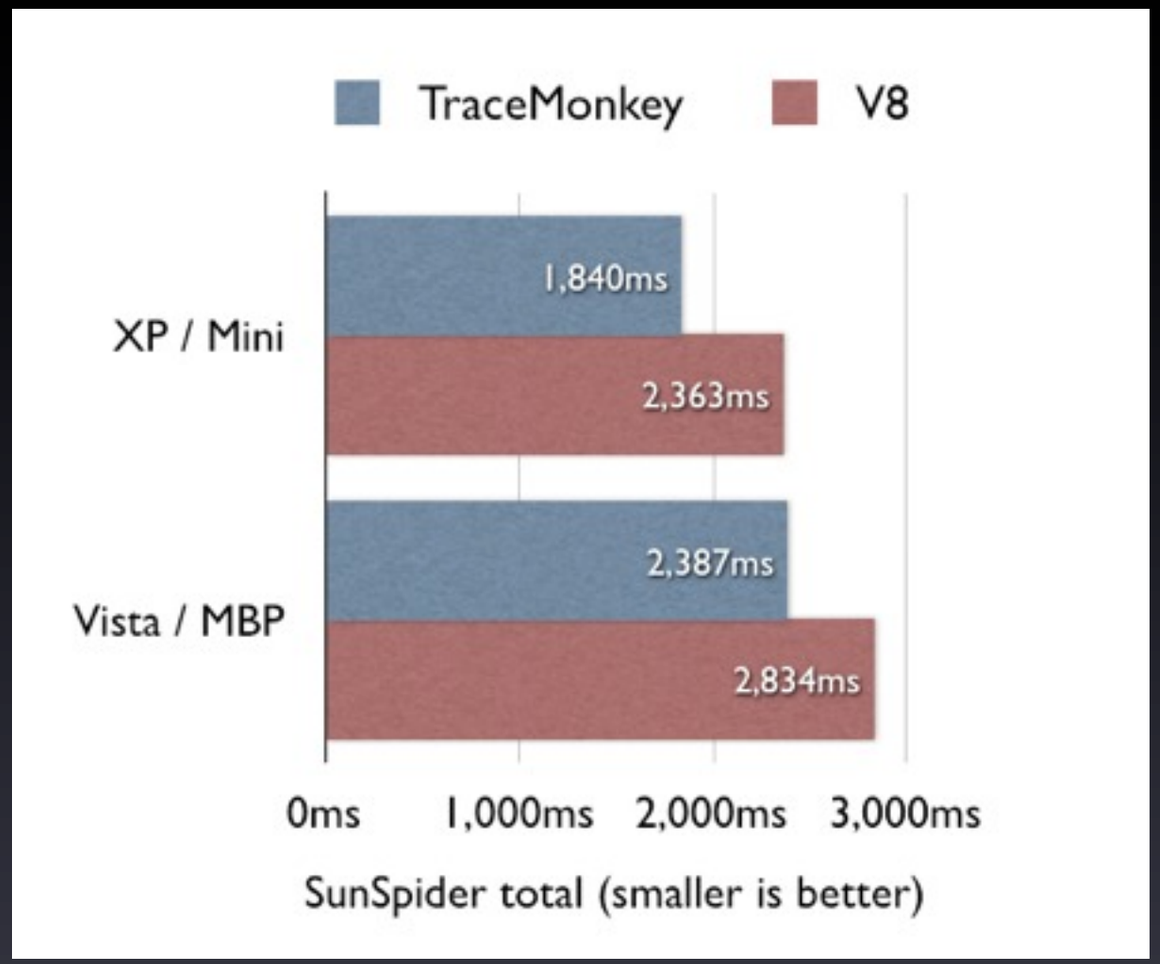
Technical Debt and Requirements

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

Overview

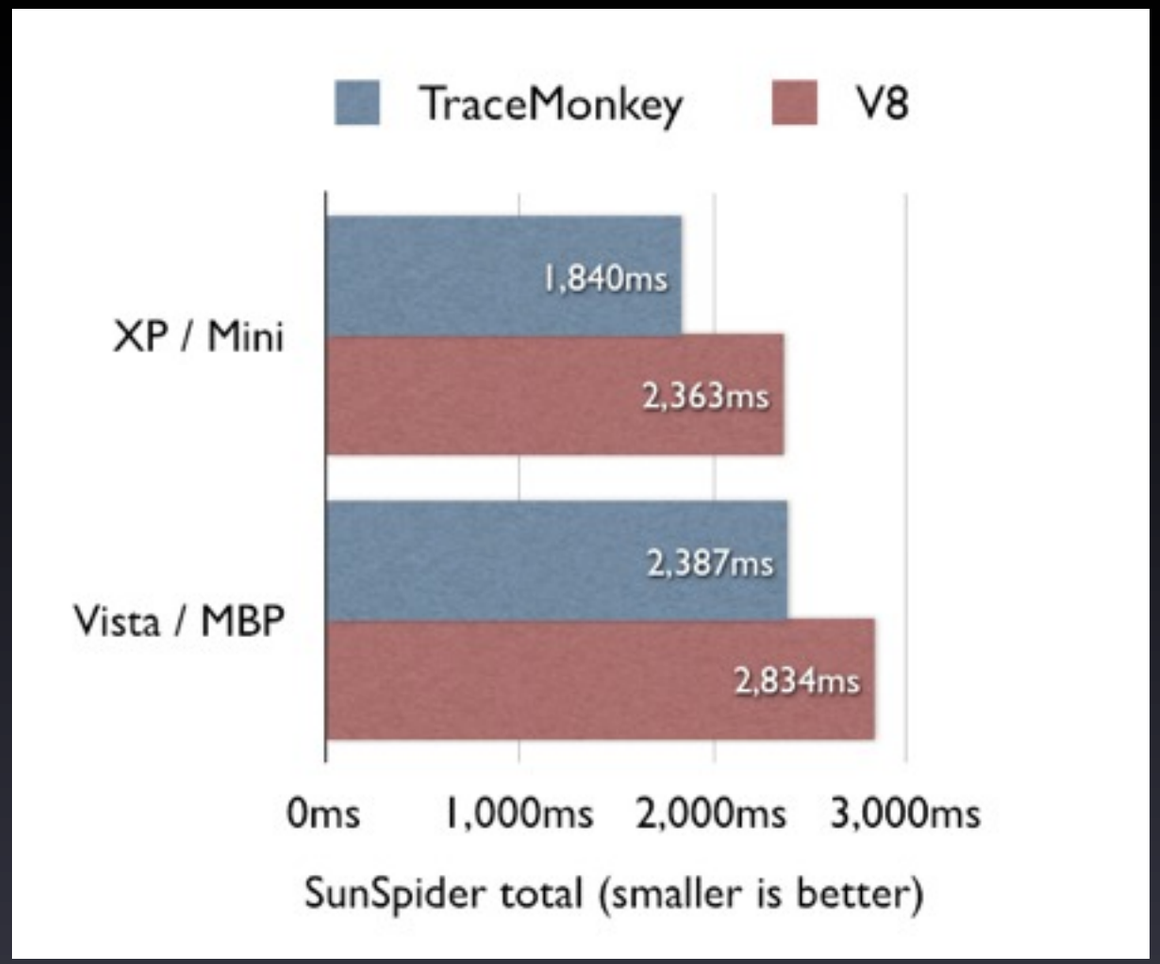
- Requirements represent product's business value and quality goals.
- "Technical debt is acquired when engineers take shortcuts that fall short of best practices." -- Eric Allman, CACM 55(5)

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



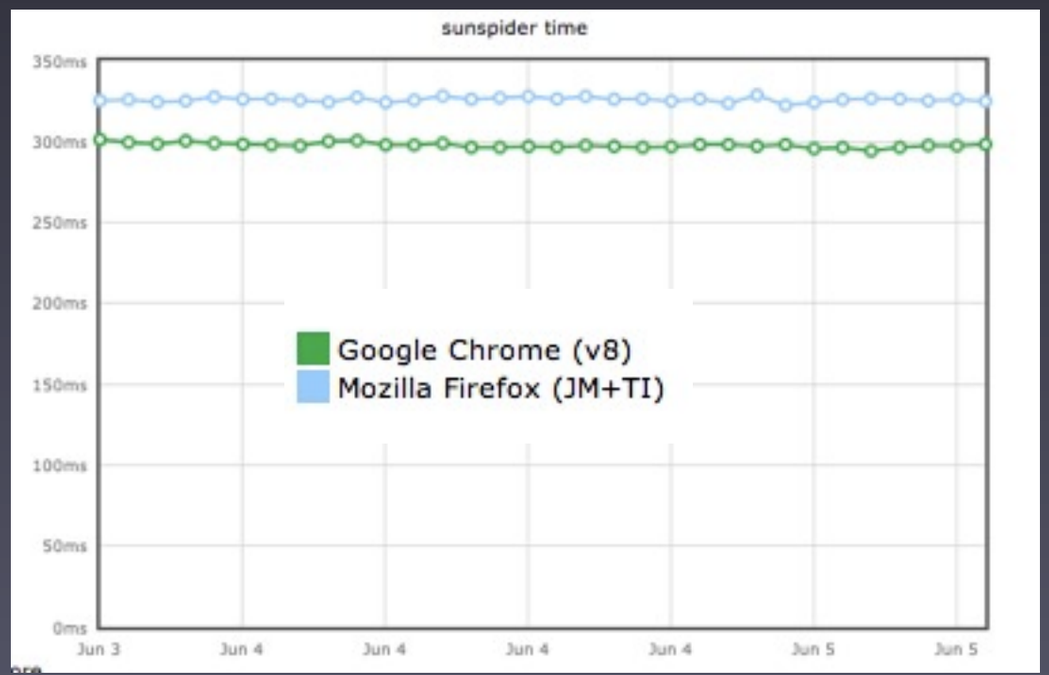
2008

Firefox



2008

Firefox



2012

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”.

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.

Requirements Debt

Design Debt

Code Debt

Requirements Debt

Should have compiled ALL Javascript to begin with

Design Debt

Code Debt

Requirements Debt

Should have compiled ALL Javascript to begin with

Design Debt

Rearchitect to support base compilation

Code Debt

Requirements Debt

Should have compiled ALL Javascript to begin with

Design Debt

Rearchitect to support base compilation

Code Debt

Implement new code; test; deliver

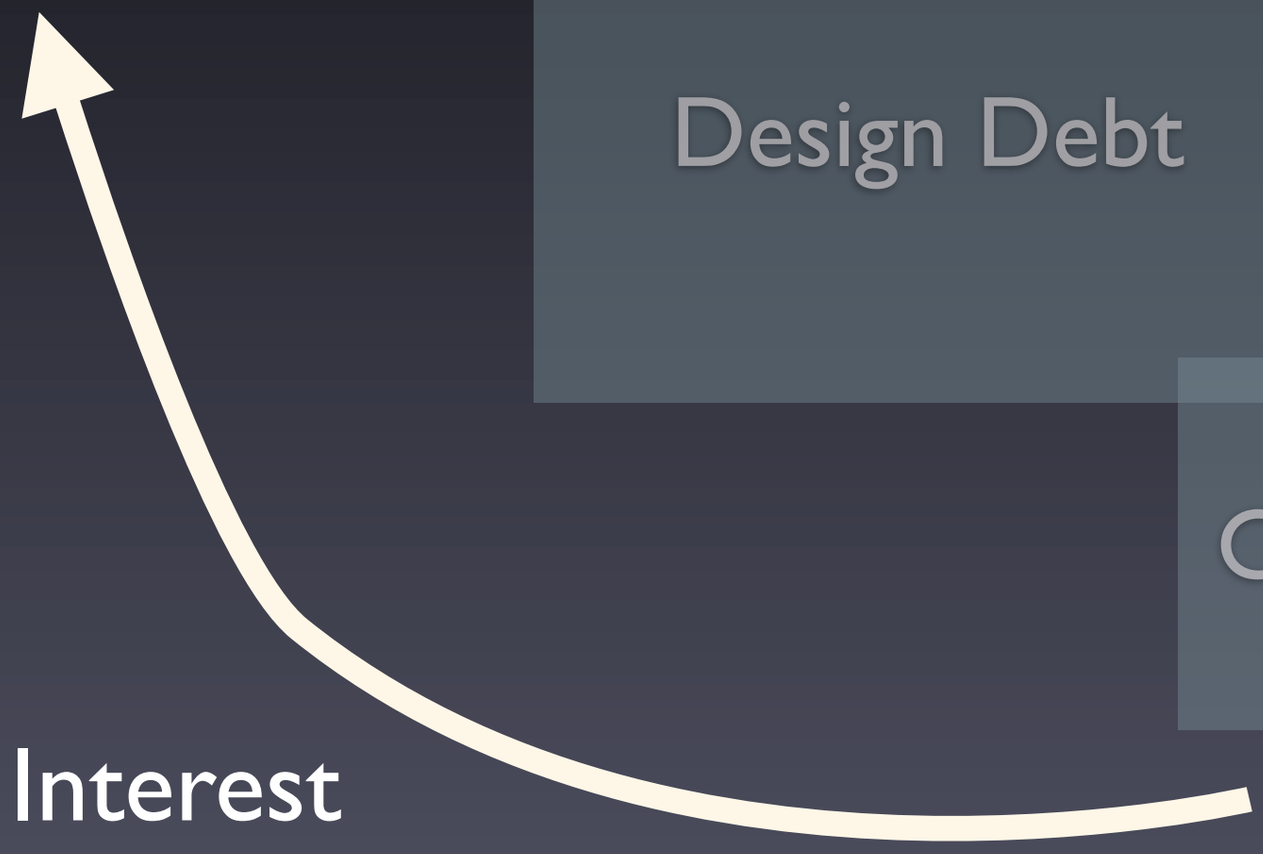
Requirements Debt

Should have compiled ALL Javascript to begin with

Design Debt

Rearchitect to support base compilation

Code Debt



Implement new code; test; deliver

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

Command	Executions
edit.Delete	5.4 M
file.Save	4.3 M
edit.Paste	3.8 M
edit.Copy	2.4 M
ContentAssist.proposals	1.4 M

Command	Executions
edit.Delete	5.4 M
file.Save	4.3 M
edit.Paste	3.8 M
edit.Copy	2.4 M
ContentAssist.proposals	1.4 M

Command	Executions
window.previousView	9
navigate.Back	69
window.showViewMenu	89
window.previousPerspective	155
window.previousEditor	166

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