Technical Debt: Why Should You Care?

Ipek Ozkaya and Robert L. Nord

Software Solutions Conference 2015

November 16–18, 2015

Software Engineering Institute | Carnegie Mellon University

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM-0003050

Is Technical Debt Real?

Popular media is recognizing major software failures as technical debt

- United Airlines failure *(July 8, 2015 "network connectivity")*
- New York Stock Exchange glitch *(July 8, 2015 "configuration issue)*
- Healthcare.gov *(February 2015,"users cannot access functionality")*

Researchers conservatively estimate US$361,000 of technical debt/100 KLOC, the cost to eliminate the structural-quality problems that seriously threatened the application's business viability.

Are we being fooled by scare tactics?

How do we understand the real problem and why should we care?

Agenda

What is technical debt anyways?

What is the technical debt timeline?

Common misconceptions about managing technical debt

What is Technical Debt?

We define technical debt as a software design issue that:

- Exists in a **system artifact**, such as code, build scripts, automated test suites, data;
- Is traced to **several locations** in the system, implying ripple effects of impact of change;
- Has a **quantifiable** effect on system attributes of interest to developers, such as increasing number of defects, negative change in maintainability and code quality indicators.

Types of Debt

	Reckless	Prudent
Deliberate	*"We don't have time for design"*	*"We must ship now and deal with consequences"*
Inadvertent	*"What's Layering?"*	*"Now we know how we should have done it"*

Fowler, M. 2009. Technical debt quadrant, Blog post at: http://martinfowler.com/bliki/TechnicalDebtQuadrant.html.

What is Technical Debt: Examples

"We have a model-view controller framework. Over time we violated the simple rules of this framework and had to retrofit later many functionality"

Modifiability violation, pattern conformance

"There were two modules highly coupled that should have been designed for from the beginning"

Modifiability violation, pattern conformance

"A simple API call turned into a nightmare <due to not following guidelines>"

Framework, pattern conformance

Timeline Perspective of Technical Debt

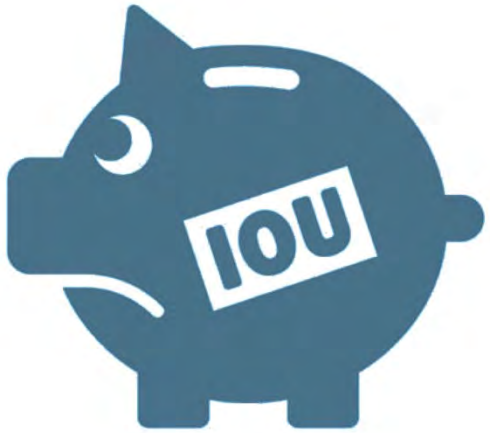
By the time the government owns the system the accumulation of debt

"Contractor developed our software tool and delivered the code to the government for maintenance. The code was poorly designed and documented therefore there was a very long learning curve to make quality changes. We continue to band aide over 1 million lines of code under the maintenance contract. As time goes by, the tool becomes more bloated and harder to repair."

Contr
intent
or uni
incurs

1. debt is incurred
2. debt is recognized
3. plan and re-architect
4. debt is paid-off
5. continuous monitoring

Technical Debt Analogy



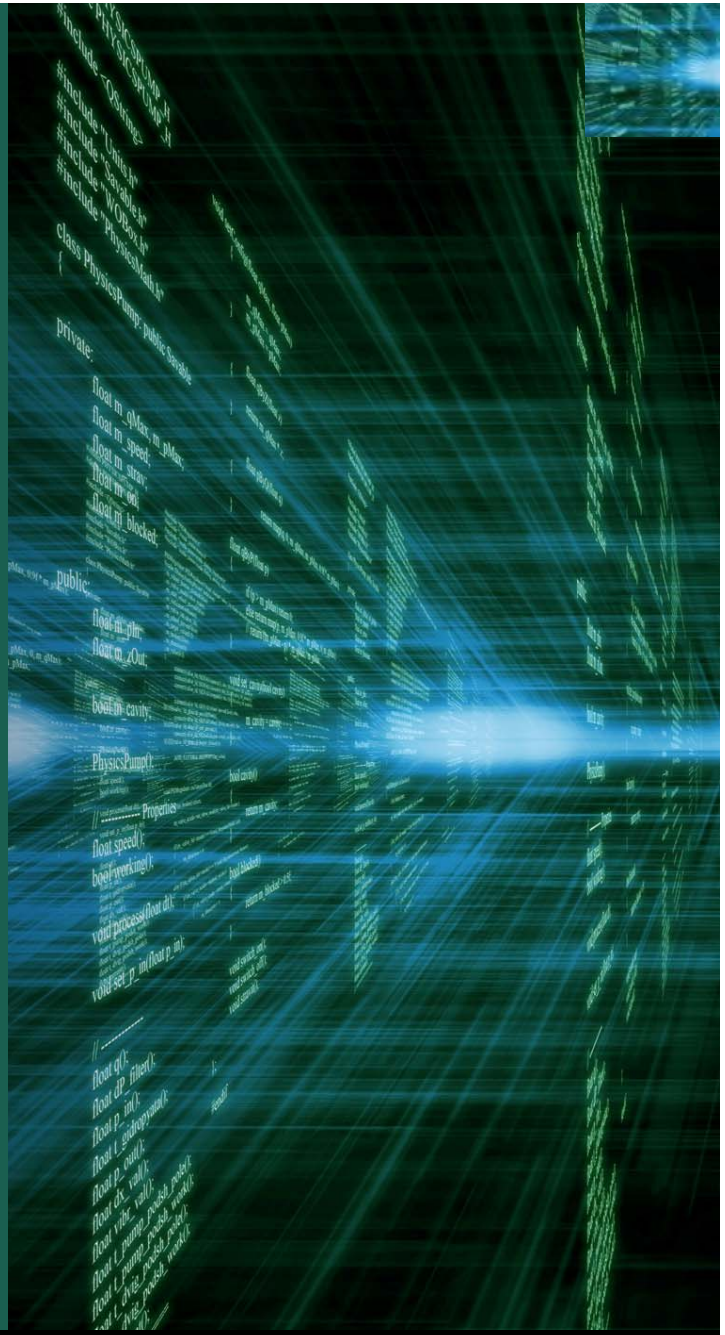
- What is the debt?
- How does debt accumulate?
- When should you pay back debt?

Is this a symptom of your debt?

- Teams spend almost all of their time fixing defects, and new capability development is continuously slipping.
- Integration of products built by different teams reveals that incompatibilities cause many failure conditions and lead to significant out-of-cycle rework.
- Progress toward meeting milestones is unsatisfactory because unexpected rework causes cost overruns and project-completion delays.

Technical Debt: Why You Should Care?

Avoid these common misconceptions



Misconception: Software quality management practices will help avoid technical debt

This view suffers from the following short-comings:

- Overlooks the impact of system complexity of long-lived, large-scale systems on technical debt management
- Underestimates the impact of technology change and evolution on the system
- Assumes technical debt can be and should be avoided, and potentially will fail to take advantage of it when needed

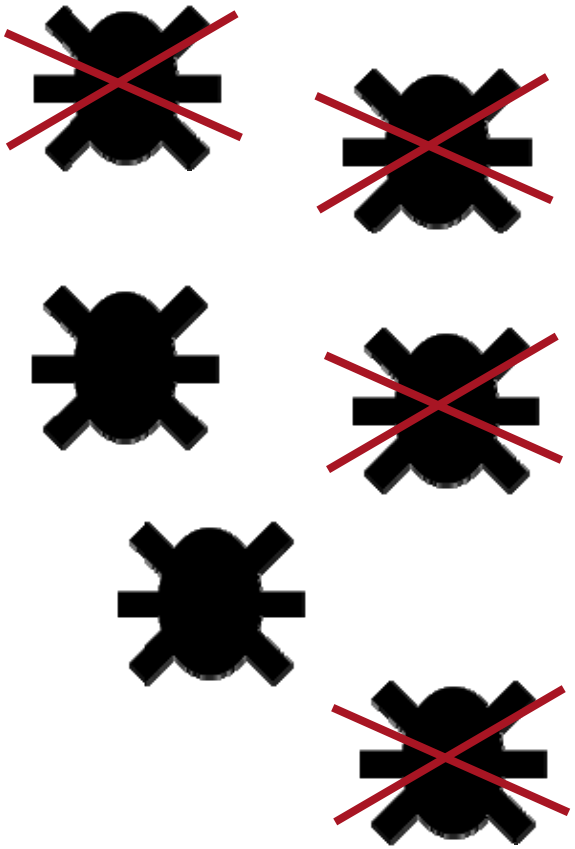
Correction: Software quality management practices will help manage technical debt

Technical debt management is about managing the short-term and long-term design trade offs.

Technical debt can be used as a value-added design strategy.

Practices such as explicitly denoting technical debt and creating buffer-time to pay-off and analyze for technical debt will help manage it in the long term.

Misconception: Eliminating defects will eliminate technical debt



This view suffers from the following short-comings:

- Focuses only on the customer visible, functional aspects of the system problems
- Results in overlooking underlying contributors of defects as design issues
- Fails to recognize accumulating interest of potential technical debt that the defects might be signaling

Correction: Defects are key symptoms of technical debt

Defects, especially recurring defects, defects that have been open for a long time, and defects accumulating around particular aspects of the system point to technical debt to tackle.

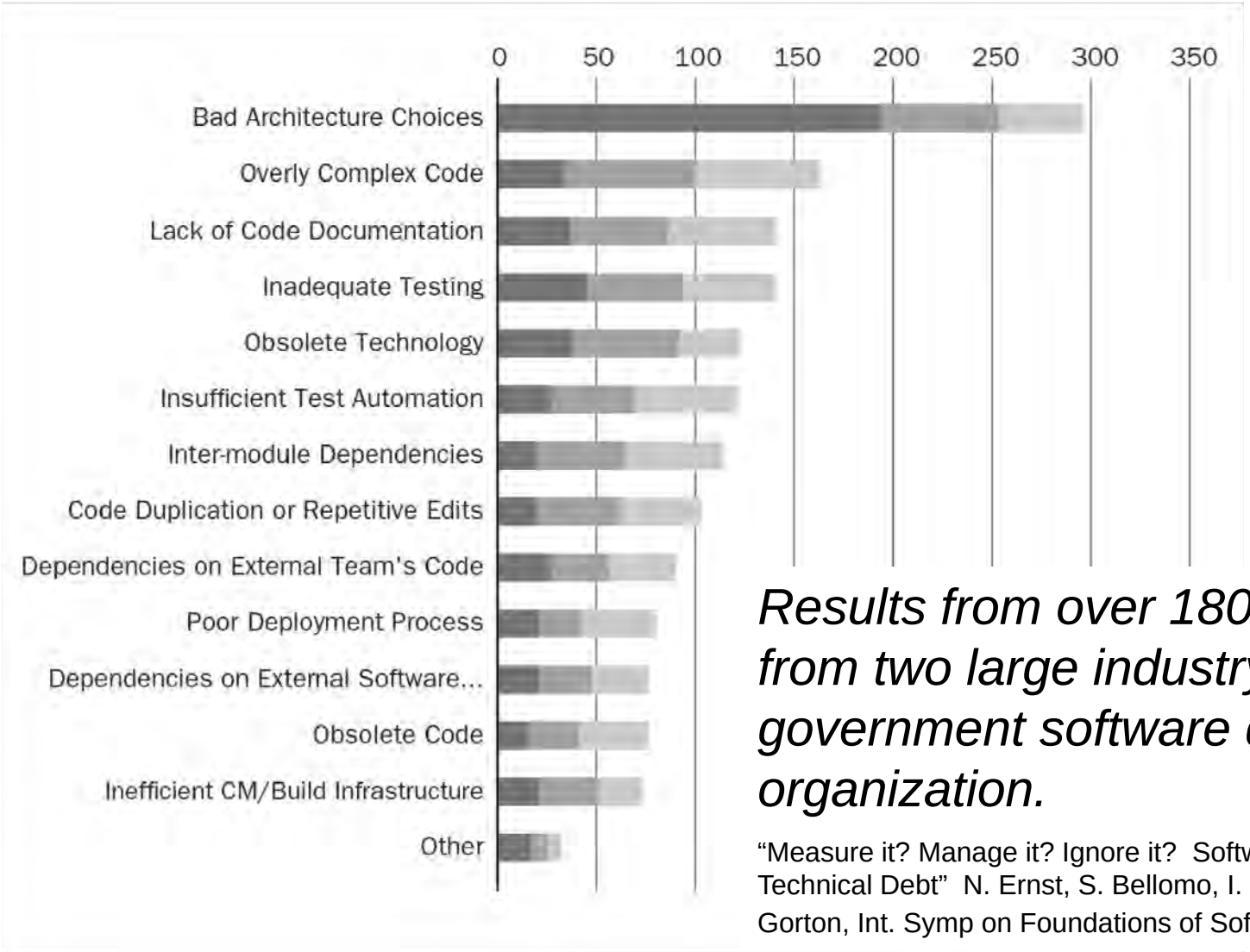
Understand the amount of resources and processes that go into defect management to better understand accumulating side effects of technical debt.

Misconception: Focusing on customer value-added functionality is sufficient to manage technical debt

This view suffers from the following short-comings:

- Assumes getting functionality into the hands of the customers as quickly as possible is what value is
- Follows the “if it works then there is nothing wrong with it, don’t touch it” culture
- Disregards the importance of architecturally significant requirements

Correction: Customer value-added functionality includes longevity and sustainability



Results from over 1800 developers from two large industry and one government software development organization.

"Measure it? Manage it? Ignore it? Software Practitioners and Technical Debt" N. Ernst, S. Bellomo, I. Ozkaya, R. Nord, I. Gorton, Int. Symp on Foundations of Software Engineering 2015.

Correction: Customer value-added functionality includes longevity and sustainability

Most challenging technical debt issues are architectural.

Differentiate strategic structural technical debt from technical debt that emerges from low code quality or poor engineering practices.

Invest time to understand architecturally significant requirements and their trade-offs.

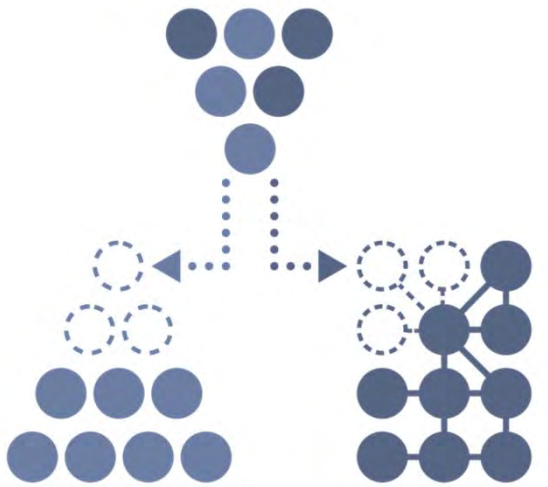
https://insights.sei.cmu.edu/sei_blog/2015/07/a-field-study-of-technical-debt.html

Misconception: Technical debt cannot be quantified

This view suffers from the following short-comings:

- Assumes measurement always happens with numbers
- Lumps project management and technical quality management together immaturely
- Expects one dollar figure for getting ahead of technical debt

Correction: Quantifying technical debt requires combination of qualitative and quantitative measures



Define clearly short-term and long-term structural goals of the system.

Invest in a sound development and testing infrastructure that incorporates automated quality measurement aspects.

Scout for project management and technical review practices that can easily be revised to include tracking technical debt.

Integrate technical debt reduction into iteration planning.

Ask the Hard Questions Early and Often

What are our dominant sources of debt?

- start with code, structure and process as larger areas of investigation

How can debt be visualized with effective tool support?

- decide on key business metrics and relate them to the product, that will drive what tool if any is right

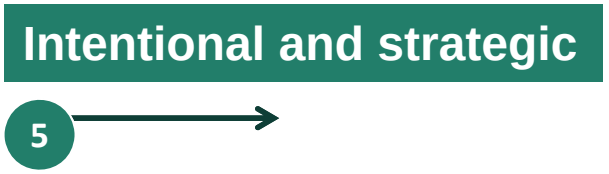
How to identify and manage strategic architectural debt?

- optimize for cost of rework, if you need to minimize it, more up-front analysis is needed

How and when to pay-back debt?

- make it an ongoing process

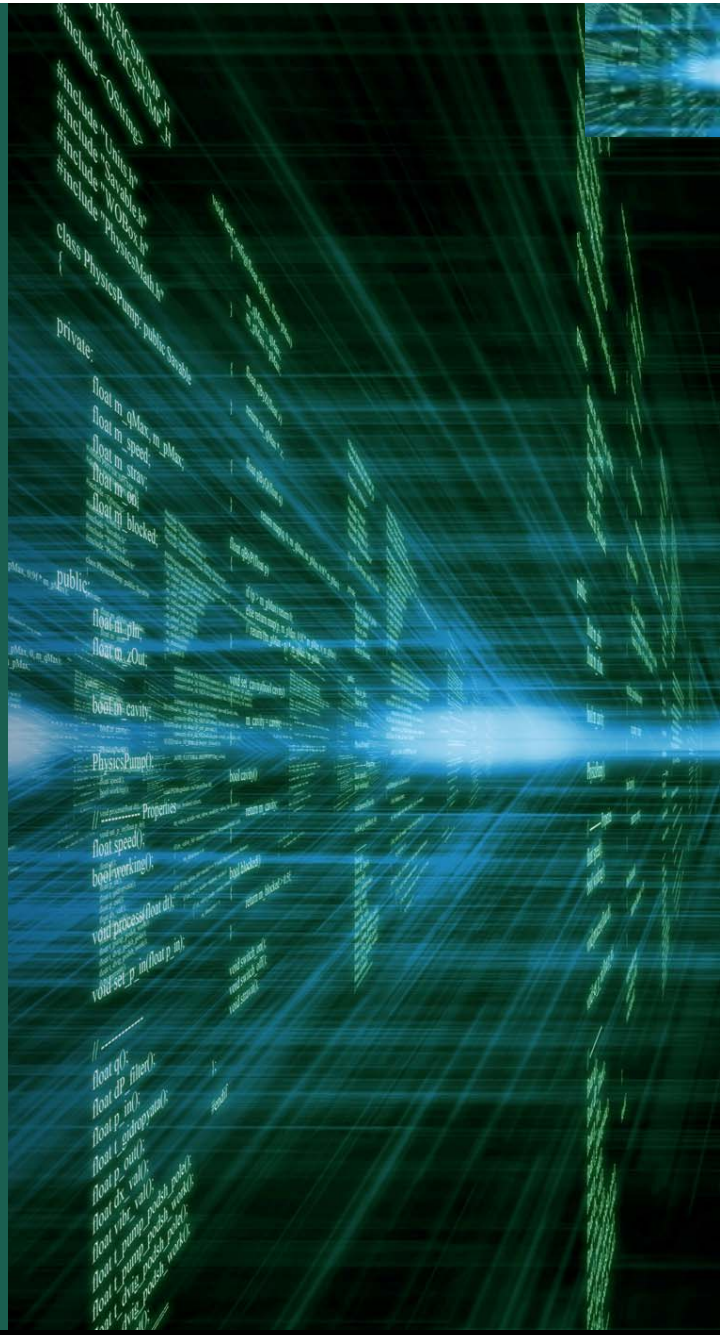
Manage the Technical Debt Timeline



Start today!

- Make architecture features and technical debt visible.
- Differentiate strategic structural technical debt from technical debt that emerges from low code quality.
- Invest in tools and technique that help elicit and track leading indicators.
- Engage business and technical persons in making tradeoffs.
- Integrate technical debt management into planning and standard operating procedures.
- Associate technical debt with risk management.

Questions?



Contact Information

Ipek Ozkaya

email: ozkaya@sei.cmu.edu

Robert Nord

email: rn@sei.cmu.edu

http://www.sei.cmu.edu/architecture/