Measure it? Manage it? Ignore it? Software Practitioners and Technical Debt

Neil A. Ernst, Stephany Bellomo, Ipek Ozkaya, Robert Nord, Ian Gorton (FSE)

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.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

DM-0004233

Background-1

Ipek Ozkaya, SEI CMU, PhD in Computational Design from Carnegie Mellon University

Robert Nord, SEI CMU, PhD in Computer Science from Carnegie Mellon University

Neil Ernst, SEI CMU, PhD in Software Engineering from University of Toronto

Stephany Bellomo, SEI CMU, MS in Software Engineering from George Mason University

Philippe Kruchten is a Prof of Software Eng. at University of British Columbia, Canada, known for RUP

About the SEI

- SEI is a Federally Funded Research and Development Organization
- Affiliated with Carnegie Mellon University
- SEI has Research and Practical focus

About our team

- Started with Agile and Architecture
- Could see projects with legacy code struggling with technical debt

Background-2

Motivating Definitions

- Cunningham, 1992: “Shipping first time code is like going into debt. A little debt speeds development **so long as it is paid back promptly** with a rewrite... The danger occurs when the debt is not repaid”
- McConnell “the obligation that a software organization incurs when it chooses a design or construction approach that's **expedient in the short term** but that increases complexity and is **more costly in the long term**.

Survey Introduction

- **RQ1:** Is the technical debt **metaphor useful**?
- **RQ2:** What are **most significant sources** of technical debt?
- **RQ3:** What **practices and tools** are practitioners using for managing technical debt?

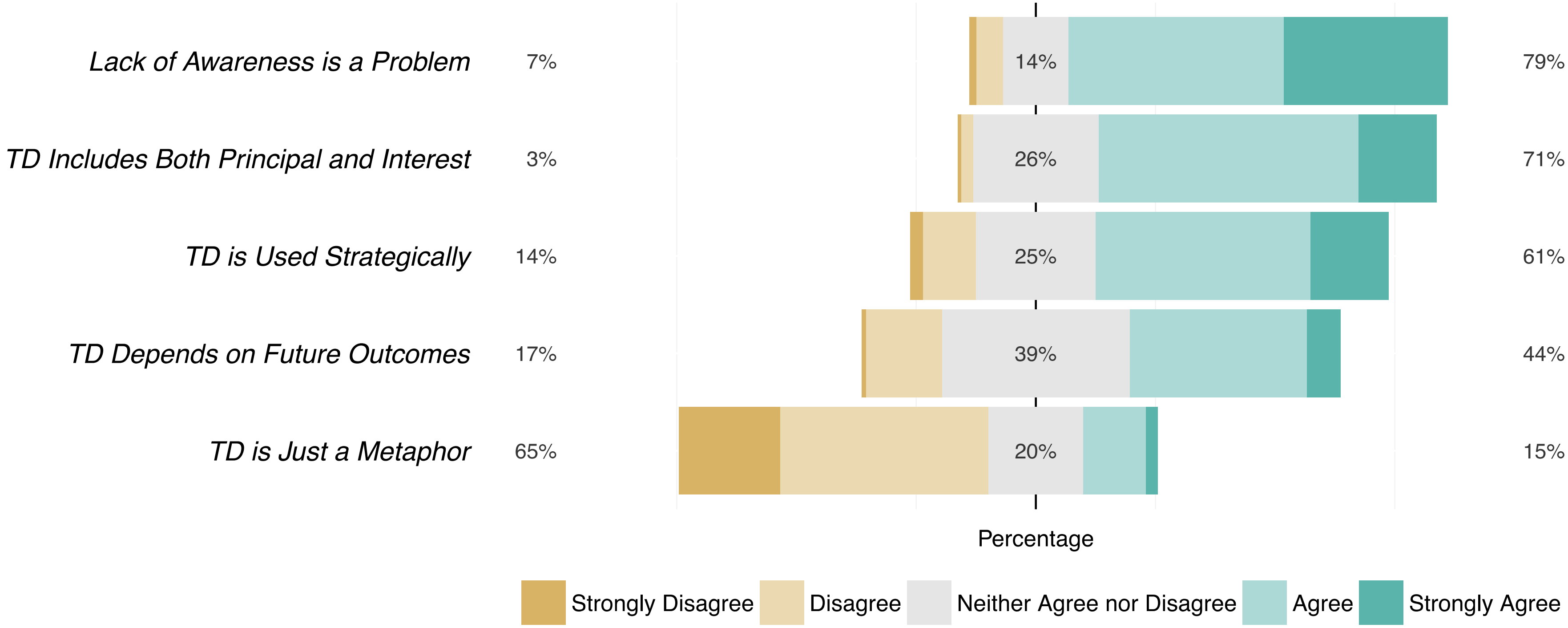
Org	Type	# Surveys out / received
A	Defense Contractor	3,500 / 248
B	Global automation, power robotics	15,000 / 1511
C	Government development/research lab	200 / 73
D	DoD sustainment	35 / 29
	Total	1861

Includes closed and open questions (follow-up interviews)

Demographics

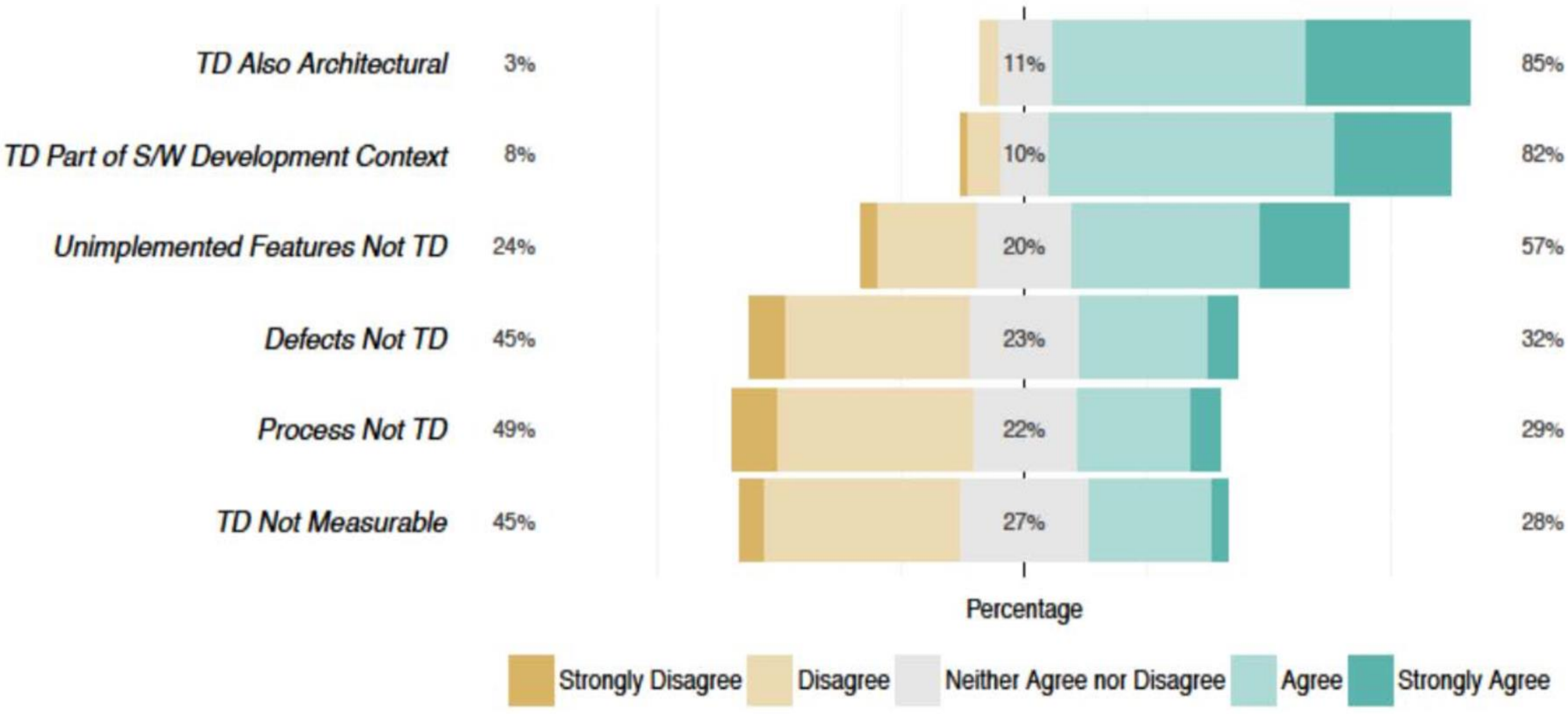
- **1831** surveys were started (across all three collaborators) and **536** surveys fully completed (all questions answered), an overall response rate of 29%
- Roles included developers (42%) and project managers (32%)
- Mixed web systems (24%) or embedded (31%).
- Mostly medium sized: 10-20 people
- The systems averaged 3-5 years old, but a significant number (29%) were over 10 years old.
- The systems between 100K LOC and 1M LOC in size.

RQ1: Is The Metaphor Useful?



- “I think the vocabulary of technical debt is useful for getting the interests aligned.”
- ‘helpful in convincing product managers and stakeholders on the value proposition of managing the debt.’

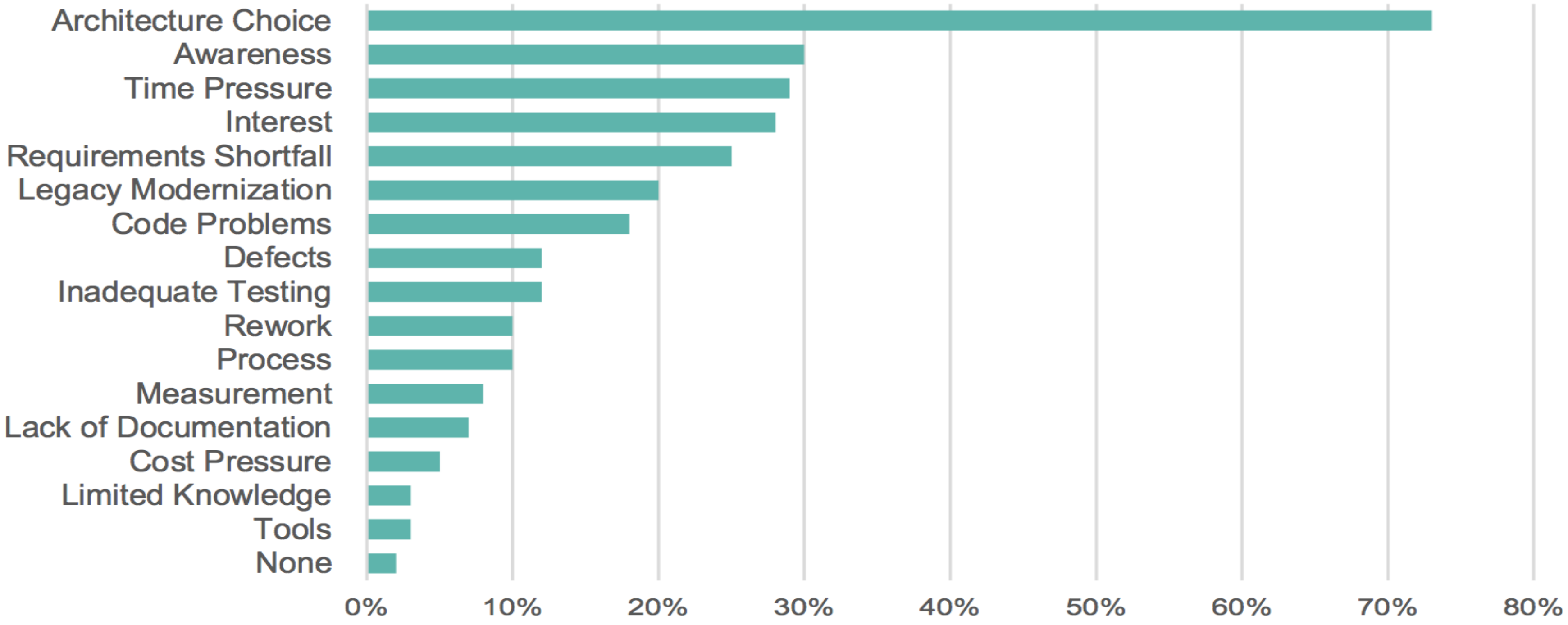
RQ2: Most significant source of technical debt?



RQ2 Source of TD: Open Coding

We triangulated answers with open coding of question data

Question: What is the biggest technical debt challenge your project faces?



Quotes from TD Examples (related to R2 most significant impact)

“the work that we’re doing now to **introduce a service layer** and also building some clients using other technology is an example of decisions that could have been **done earlier** if we had had more time and had the funding...”

“‘**platform**’ was **not designed with scalability** in mind”

“In retrospect we put **messaging/communication ... in the wrong place** in the model view controller architecture”

Architecture Choices and System Age

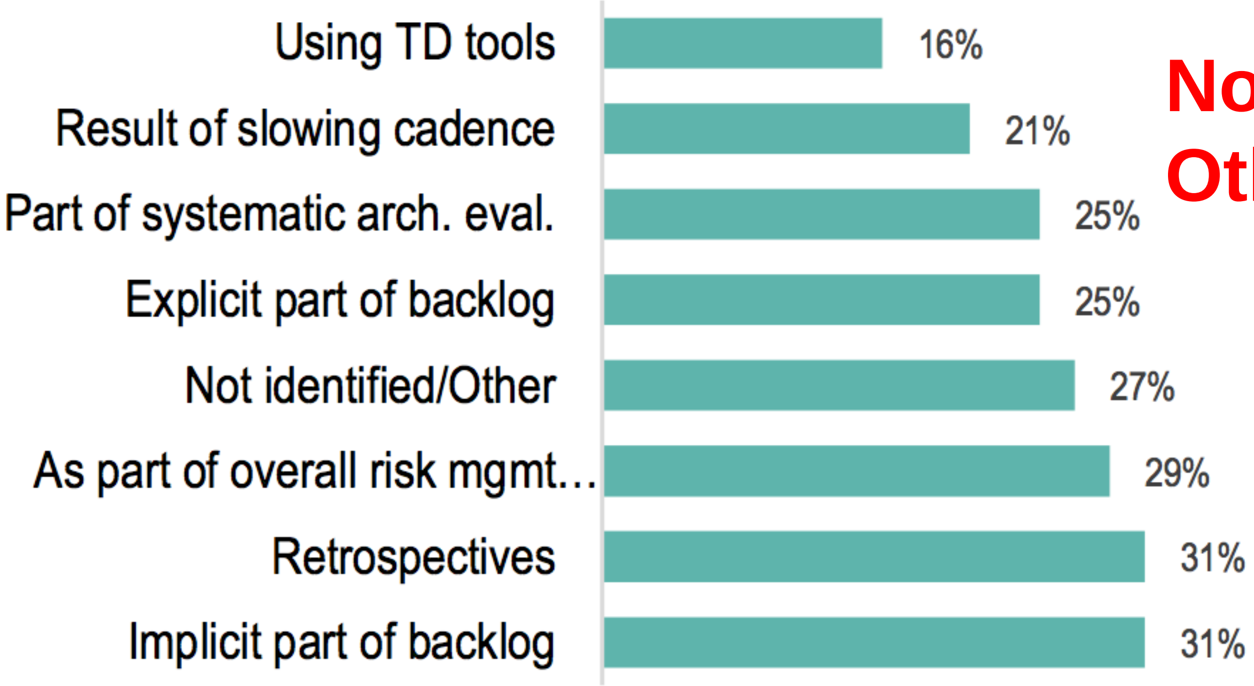
- Weak association between system age and the perceived importance of architectural issues
  - 89% of those with systems > 6 years old agreed that architectural issues are a significant source of debt
  - 80% of those with newer systems (<3 years old) agreed

Open-ended quote

- "**over the years**, other sites would begin using the system and would require changes to how the workflow operated"

Our data for this study does not support correlation between system age and perceived importance of architecture issues, however, we see indicators that may warrant further investigation

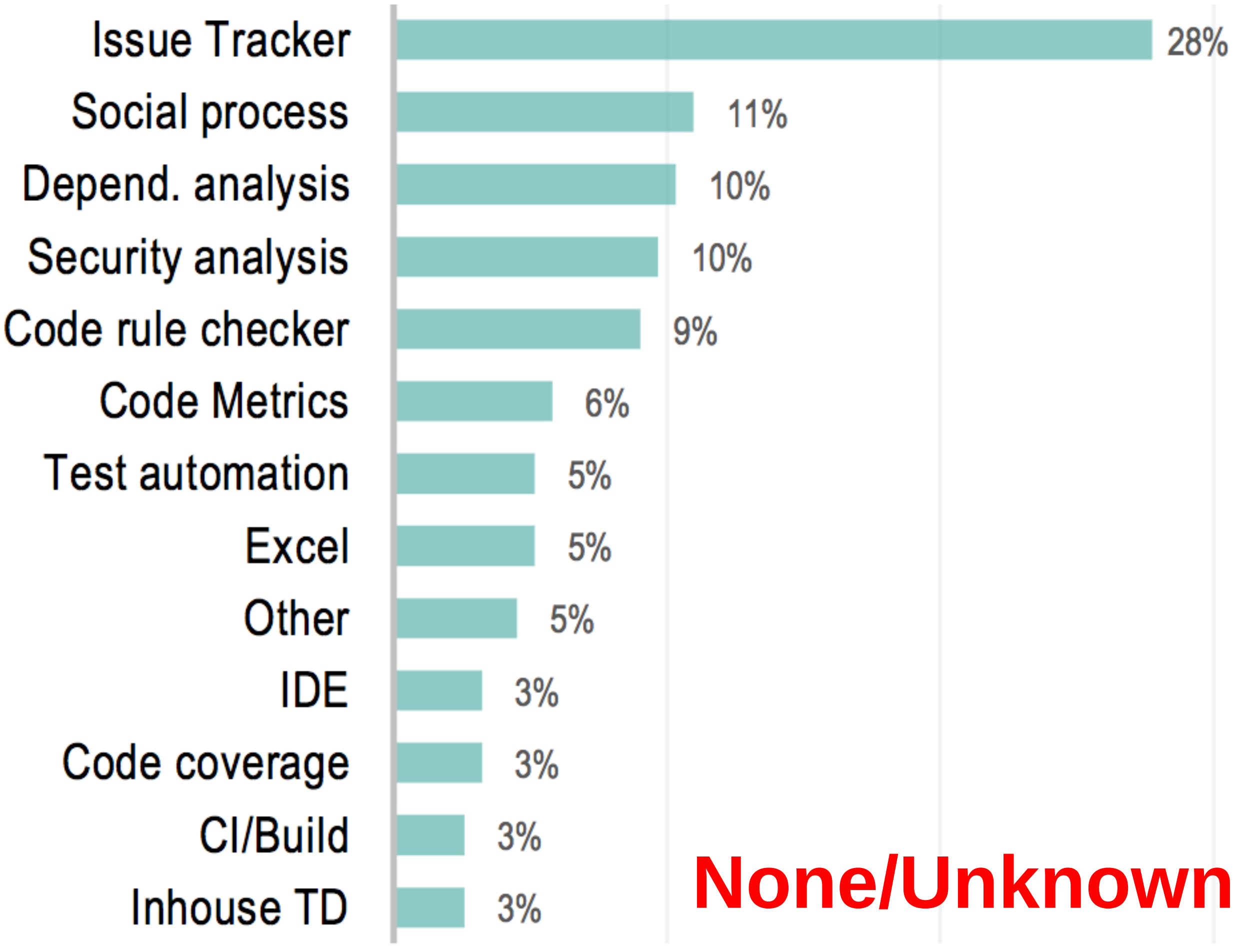
RQ3: What approaches are people using for managing TD?



Not Identified/ Other: 27%

How tracked … **Where tracked …**

RQ3: What tools are practitioners using for managing TD?



None/Unknown: 58%

RQ3: Quotes on Tools and TD

“regarding **static analysis** we have the source code static analysis tools, but this is to assure proper **quality of source code.** But **h**ow architectural changes are impacting I don’t know. And, in fact, this is something we don’t do.”

“**there’s a billion little warnings [from static analyzers]**. And so it seems a little bit overwhelming.”

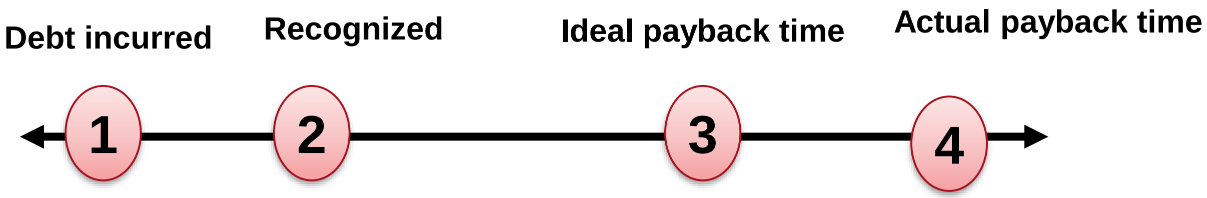
“[we track] occasionally by explicit tech debt items [in issue tracker]**, usually by pain**, or not at all...”

Summary

- Software practitioners agree on the usefulness of the technical debt metaphor
- Survey open and closed questions suggest architectural choices have biggest impact on accumulation
  - Most pain in terms of effort or funds
- Responses suggest standard practices and tools to manage technical debt do not currently exist
- Respondents said issue trackers are heavily used for managing technical debt on their projects

Future Work on Strategic Management of TD

Conceptual TD Timeline



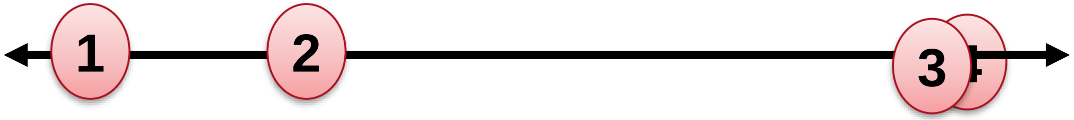
Debt incurred

Recognized

Ideal payback time

Actual payback time

What we observe in practice



TD payback is delayed causing significant accumulation

Our goal: Shorten the time between 2-3 by handling technical debt more strategically

Future Work Cont.

Three aspects inform our future work

- Better understand states of technical debt and evolve our conceptual model
- Help practitioners strategically and proactively manage technical debt (as close as possible to the ideal time to pay it back)
- Improve the state of the practice for detecting impactful technical debt
    - Preferably using artifacts that are a natural bi-product of the SDLC