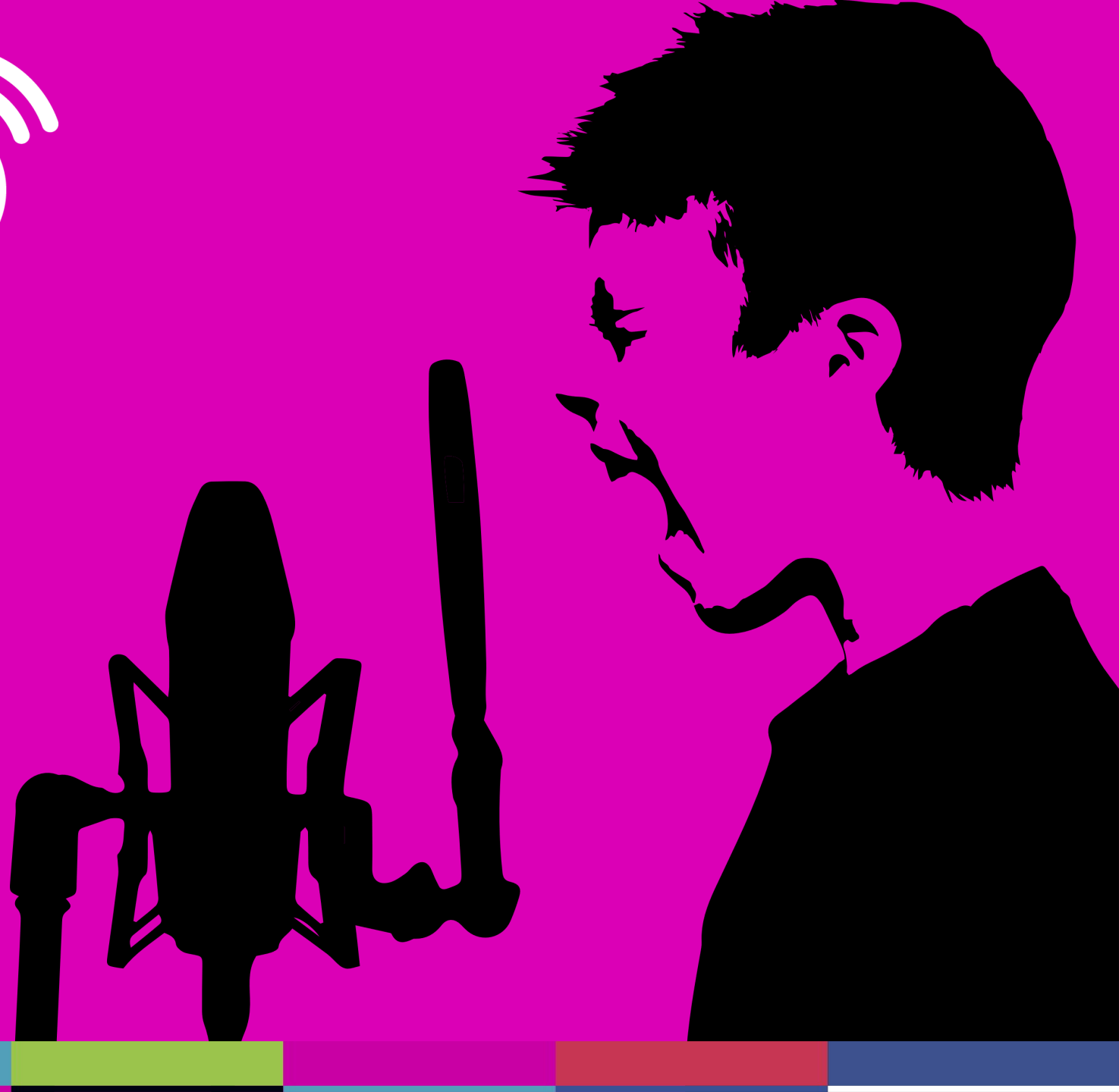
ALL DAY DEVOPS

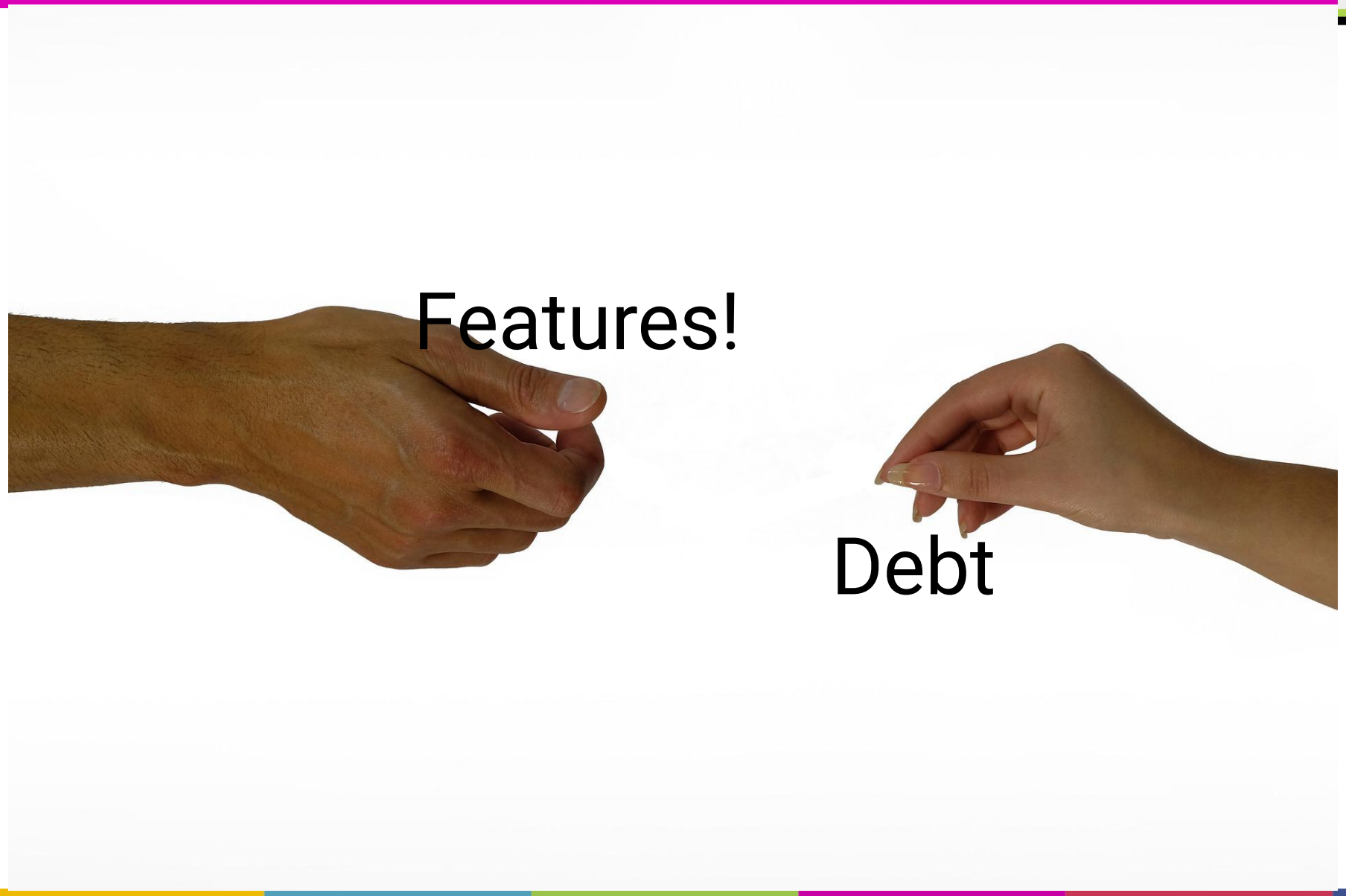
NOVEMBER 12, 2020

Hyrum Wright

Technical Debt as Pollution




50
Pesos
Veinte Pesos
UNITED STATES OF AMERICA
ONE DOLLAR
XP04573414


Features!
Debt

Technical Debt Axes

- Oversight
- Hindsight
- Context
- Lifestyle

Oversight

Early mistakes not being caught.

E.g., “we didn’t know there was a better way"

Hindsight

Only learned after having built the first system.

Context

The choice was reasonable at the time, but circumstances changed.

We don’t exist in a vacuum.

Example: Betamax

Lifecycle

Debt resulting from questionable accounting or short-term priorities.

Occurs when we don't take into account total cost of system ownership.

"Eating your seed corn."

Technical Debt

The difference between the code and technical systems we have today

vs

what we wish we had.

Where the Financial Metaphor Breaks Down

- Different kinds of technical debt are qualitatively different from each other.
- Some kinds of debt do disappear on their own.
- Technical debt is not inherently fungible.

Technical Debt as Pollution

- There are ~∞ number of root causes for technical debt
- Far cheaper to prevent than clean up or mitigate
- Monitoring is required to ensure compliance
- Accurate accounting is essential
- Technical debt often imposes externalities on third-parties

Tradeoffs

Preventing Technical Debt

- Over large time horizons, prevention is far cheaper than cleanup or mitigation.
- People are poor at valuing prevention activities
  - Outages that didn't happen
  - Bugs that didn't manifest
  - Developer productivity which went down
- No finite list of specific *dos* and *don'ts*.

Prevention Recommendations

- Everyone:
  - Update design doc templates to identify the systemic technical debts being created.
  - Remind everyone that we are doing Software Engineering, not Software Development
  - Account for technical debt in project schedules
- Code Reviewers:
  - Take the long-term view
  - Don’t relax expectations

Prevention Recommendations

- Tech Leads:
  - Set time aside for addressing technical debt.
  - Set a good example for your own team.
  - Push back on designs which introduce overlapping technology.
- Managers:
  - Do not approve new projects which overlap existing systems.
  - Dedicate resources to addressing specific technical debt problems in your organization.

Monitoring

- An intervention without progress monitoring is just an activity.
- Build technical debt monitoring into your systems as early as possible
- Ask probing questions during 1:1s and Eng reviews
  - “What are your biggest sources of technical debt?”
  - “What are your biggest hindrances to engineering productivity?”

Monitoring

Possible metrics include:

- Outage frequency
- Canary deployment success rate
- Internal survey result

Consider appropriate technical debt early warning signals for your own project.

Enforcement

Establishing strong norms around acceptable behavior and the consequences for choosing to deviate from that behavior.

Many existing parts of an organization's social contract are implied.

What is locally optimal for your team, may not be globally optimal for your company.

Enforcement

Technical and social barriers surrounding technical debt:

- Explicitly document the SLA for users of in the ecosystem
  - “Don’t foul the water.”
- Improved tooling to detect attempts to introduce technical debt
- Better social norms around technical debt at all levels


EPA
CROWN VANTAGE
LANDFILL
SUPERFUND SITE
WARNING:
Hazardous materials
present at this site.
No Trespassing.
For further information call the
U.S. Environmental Protection Agency
(800) 346-5009

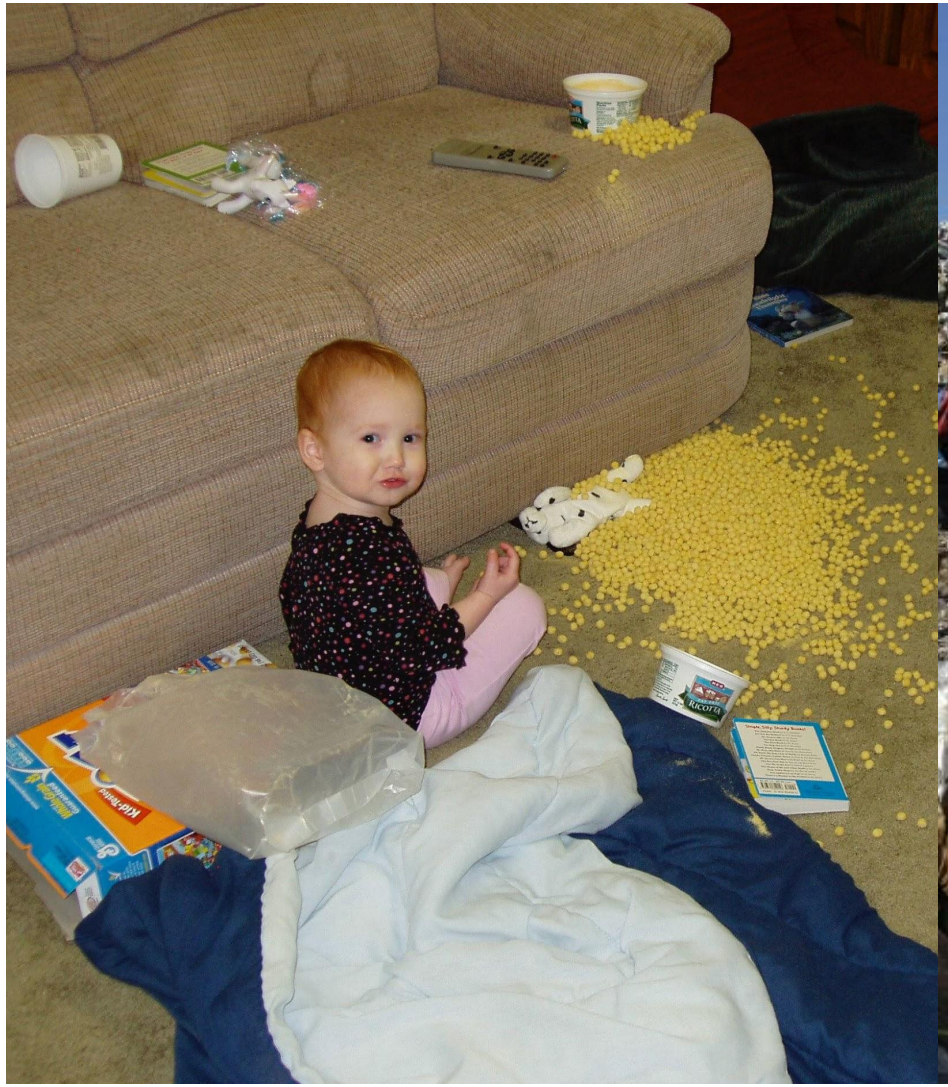
Cleanup

Cleanup efforts are most efficiently done when handled by a centralized team.

Consider a tanker spill at your local beach.

Cleanup

- Centralized teams have a concrete and quantifiable cost.
- These cleanups take a long time: prevention is better than cleanup.
- Leaders: Support the teams which own the cleanup.
  - Progress may be slow, if it were easy, the cleanup would already have been done.



Locality and Urgency

	Localized	Diffuse
Urgent	Fix or escalate immediately	Escalate; get centralized ownership
Non-urgent	Track or quickly fix	Track, investigate automation, or ignore

Suggestions for Engineers

- Focus on Software Engineering, not Development
- Learn to write good tests
- Learn to review code carefully
- Look carefully for new technical debts, and escalate them

Suggestions for Tech Leads

- Train less-experienced engineers to think about programming vs. software engineering
  - Lead by example
- Create discrete cleanup tasks to resolve technical debts
  - Put specific team members in charge of these tasks
- Recognize technical debt cleanup efforts
- Consider overlapping systems in design docs
- Have the courage to *not* build something

Suggestions for Infrastructure Teams

- Establish a “well-lit path” for using your infrastructure, and eliminate corner cases
- Be prepared to prioritize systematic technical debt issues your customers raise
- Be clear and explicit about best practices for use of your systems
  - Establish written contracts with the users of your systems
- *Resist the temptation to build new systems*

THANK YOU TO OUR SPONSORS

NORTHROP GRUMMAN

sonatype

StateFarm™

CloudBees®

Google Cloud

NowSecure™

D2 IQ

ConnectALL
An Orasi Company

moogsoft

NS1.

Carnegie Mellon University
Software Engineering Institute

CONTINO

ATARC

MACQUARIE

DZone
A DEVADA MEDIA PROPERTY

MediaOps

TechBeacon

SD Times

Forcepoint