outsystems

TECH TALKS

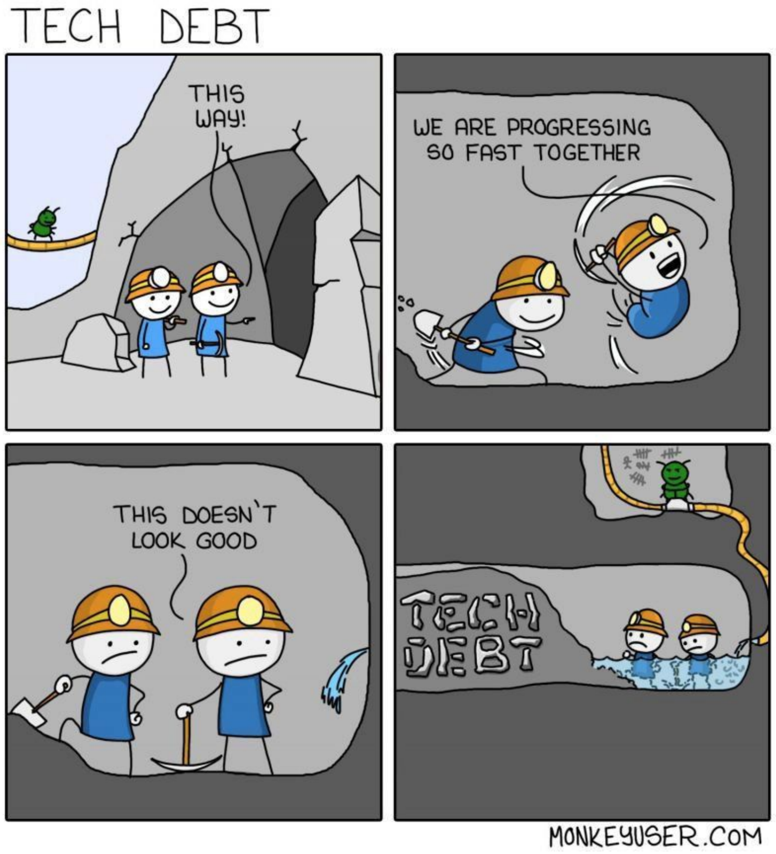
IN-DEPTH SERIES

Tech Debt

How to Diagnose and Manage Technical Debt

Joao Melo

MVP & Principal Tech Lead @ PhoenixDX


TECH DEBT
THIS WAY!
WE ARE PROGRESSING SO FAST TOGETHER
THIS DOESN'T LOOK GOOD
TECH DEBT
MONKEYUSER.COM

What you'll learn here

1. What it is...
2. Impact in Businesses
3. Root Causes
4. Model to Master Tech Debt
5. Q&A

Audience

Tech Leads

Developers

Architects

Engagement Managers

Product Owners / Managers

Project Managers

Simply put

Technical Debt
Is the coding you have to do now
Because of the shortcuts you
took yesterday

Tech Debt is...

Carnegie Mellon University:
“the tradeoff between the short-term benefit of rapid delivery and long-term value.”

Gartner:
“the deviation of an application from any nonfunctional requirements.”

But, am I dealing with Tech Debt?

Changes that follow the remediation of Tech Debts are not supposed to affect the behaviour of the App, from a functional perspective

Tech Debt is NOT...

A mess!

A mess is not a technical debt.
A mess is just a mess.

"Messy code, produced by people who are ignorant of good design practices, shouldn't be a debt."

Uncle Bob
Best-selling author on software design principles
Co-author of the Agile Manifesto

Lehman's laws of software evolution

Continuing Change

Continually adapt or it becomes progressively less satisfactory *(1974)*

Increasing Complexity

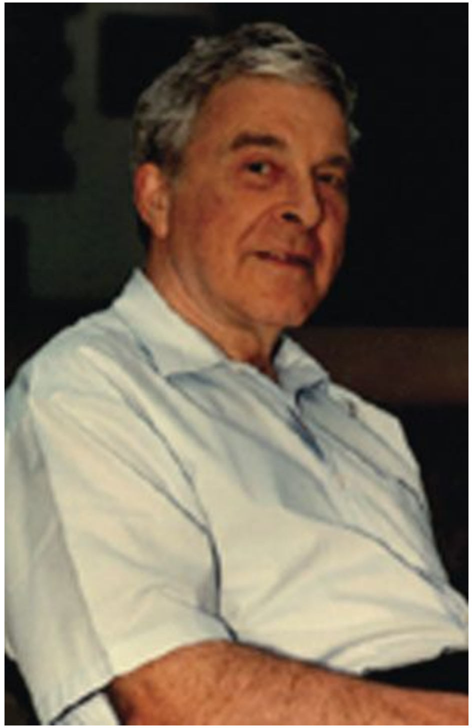
Complexity increases unless work is done to maintain or reduce it *(1974)*

Continuing Growth

Functional content must continually increase to maintain user satisfaction *(1991)*

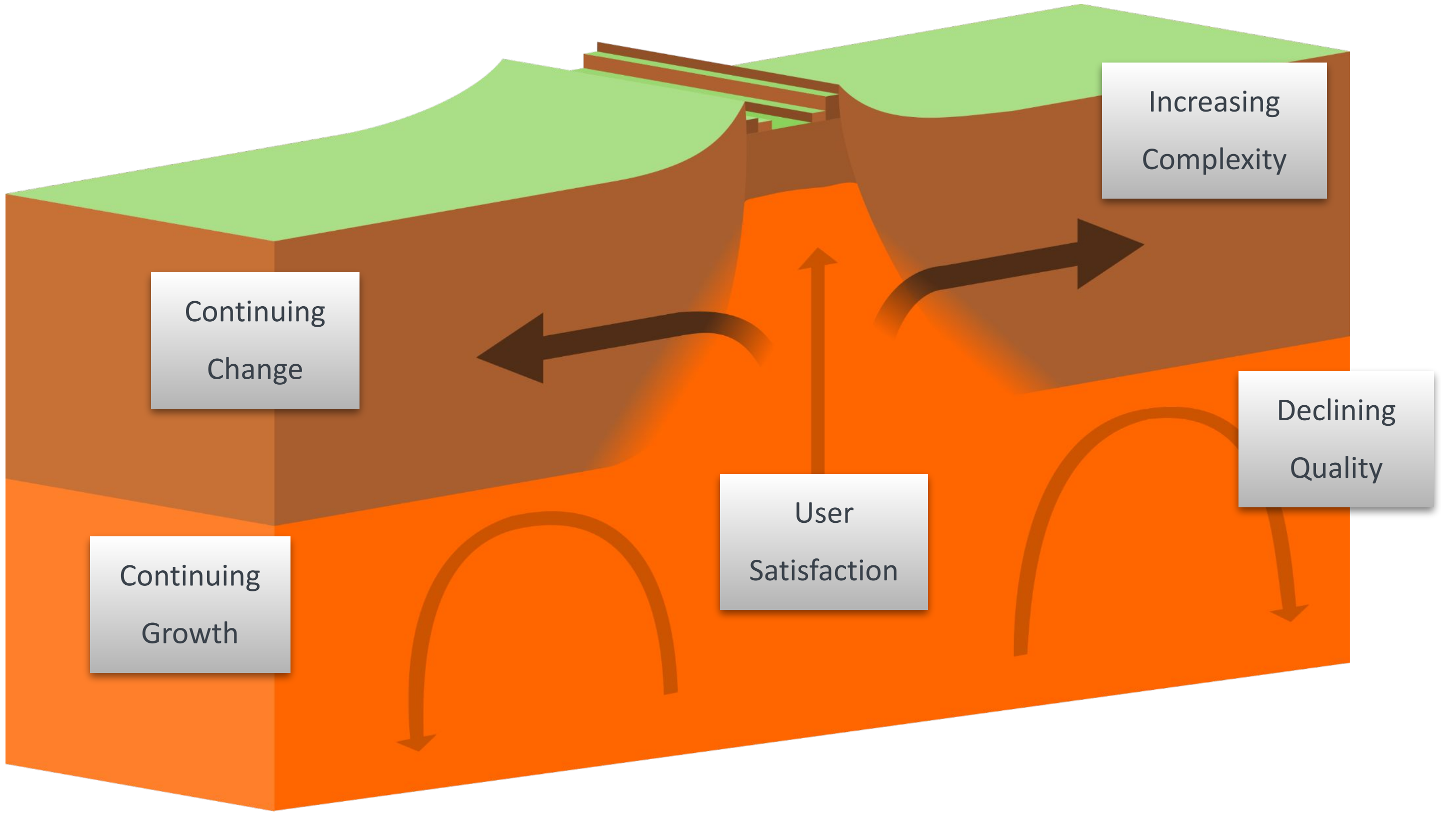
Declining Quality

Quality will appear to be declining unless it is rigorously maintained and adapted *(1996)*

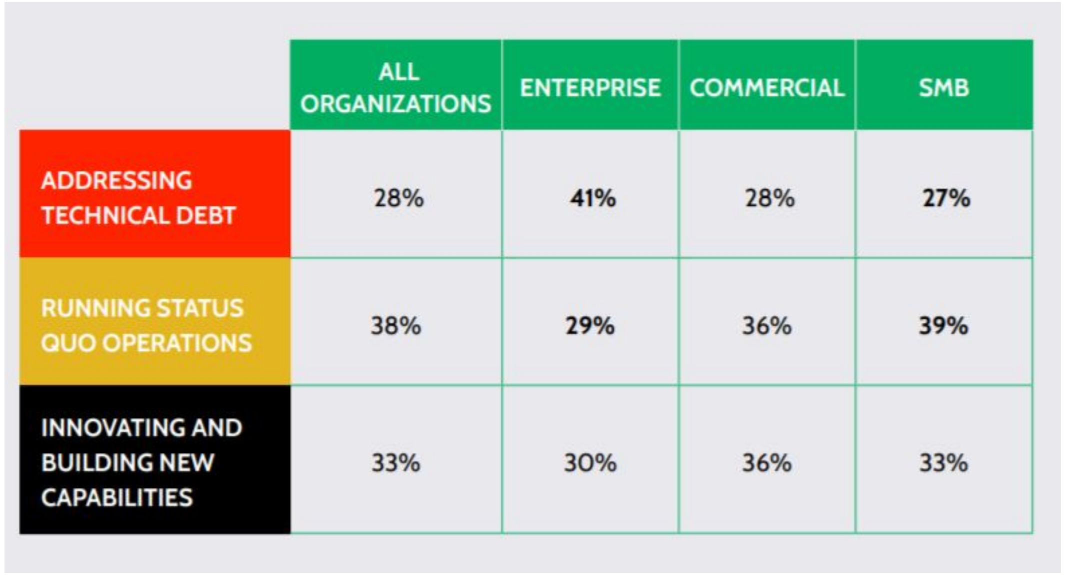


Manny Lehman

Father of Software Evolution

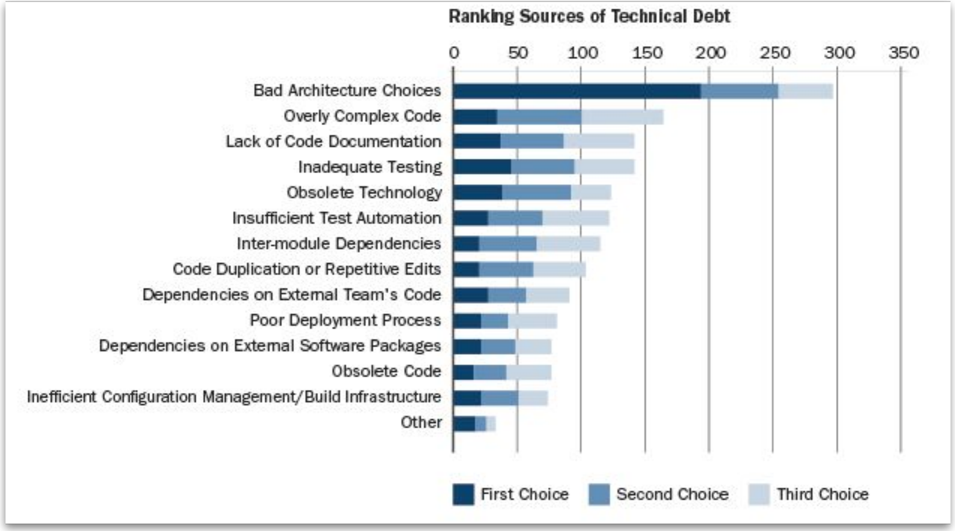

Continuing Change
Increasing Complexity
Declining Quality
User Satisfaction
Continuing Growth

Impact in Businesses

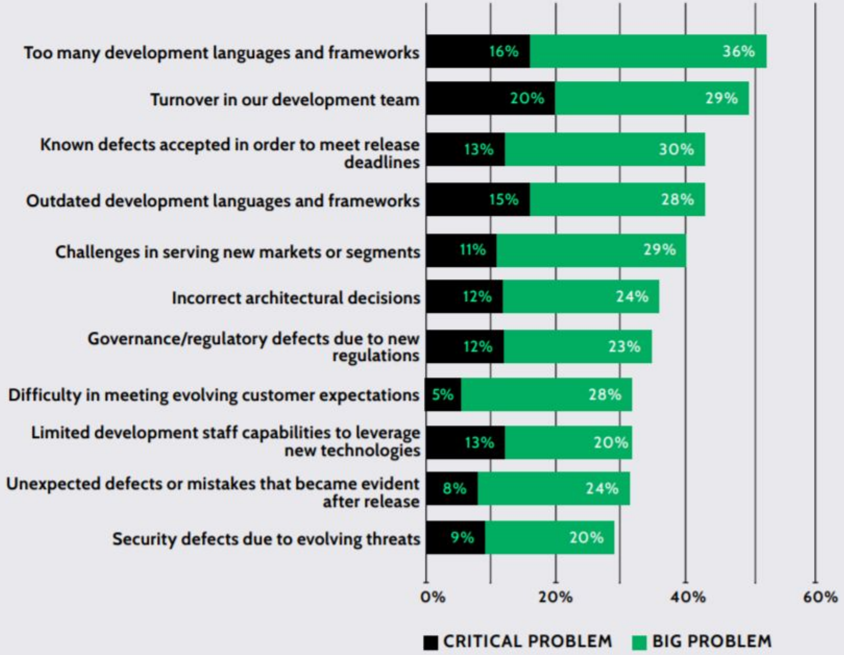


	ALL ORGANIZATIONS	ENTERPRISE	COMMERCIAL	SMB
ADDRESSING TECHNICAL DEBT	28%	**41%**	28%	**27%**
RUNNING STATUS QUO OPERATIONS	38%	**29%**	36%	**39%**
INNOVATING AND BUILDING NEW CAPABILITIES	33%	30%	36%	33%

Root Causes



Neil Ernst in 2015, at Carnegie Mellon University, A Field Study of Technical Debt



Financial Debt -> Technical Debt

A Metaphor

"We accumulate the learnings about the application over time by modifying the program to look as if we had known what we were doing all along." - Ward Cunningham

"I am never in the favor of writing code poorly, but I am in favor of writing code to reflect your current understanding of a problem." - Ward Cunningham

"A particular benefit of the debt metaphor is that it's very handy for communicating to non-technical people." - Martin Fowler

Ward Cunningham

First wiki

Co-author of the Agile Manifesto

The cost of avoidance

Like monetary debt, technical debt can accumulate “interest”.

The longer technical debt is ignored or unaddressed, the more software entropy can occur.

A model to Master Tech Debt

#1: Don’t let it build up

#1: Don’t let it build up

#2: Make informed decisions driven by value to the business

#1: Don’t let it build up

#2: Make informed decisions driven by value to the business

#3: Bring the client / owner to the party

It’s about finding the balance

Preventing

Leverage good practices

It's Ok to replicate (some) data and need to search

a in OutSystems?

Product and Services Mkt

Customer

Yes

ld Cache Summary

ary fields frequently?

No

Ye

ch Sync

l field

ng a

ostly

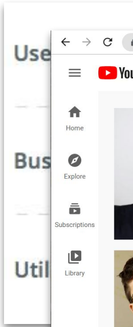
Ye

zy lo

More than one source for the same concept?

Yes

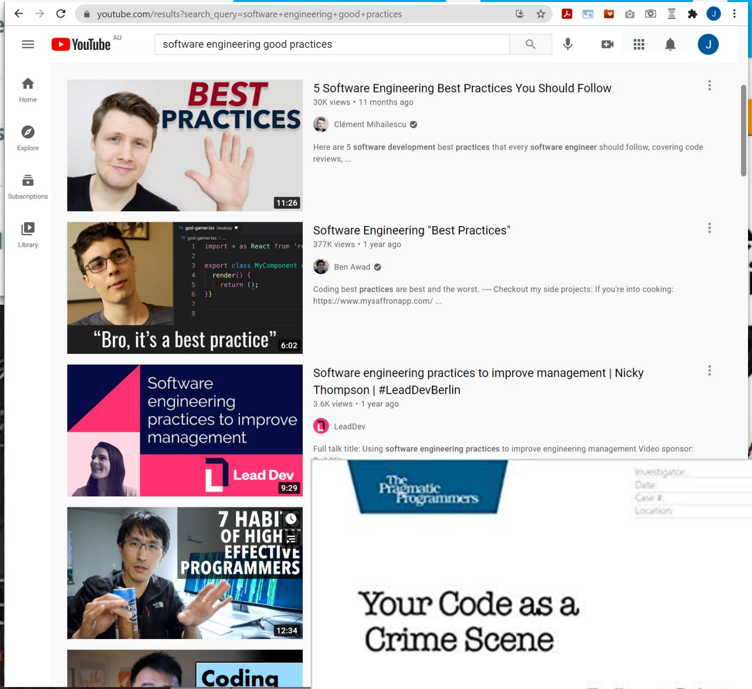
Tansparency service



Use

Bus

Util



youtube.com/results?search_query=software+engineering+good+practices

YouTube AU

software engineering good practices

Home

Explore

Subscriptions

Library

5 Software Engineering Best Practices You Should Follow

30K views • 11 months ago

Clément Mihailescu

Here are 5 **software development** best **practices** that every **software engineer** should follow, covering code reviews, ...

Software Engineering "Best Practices"

377K views • 1 year ago

Ben Awad

Coding best **practices** are best and the worst. ---- Checkout my side projects: If you're into cooking: https://www.mysaffronapp.com/ ...

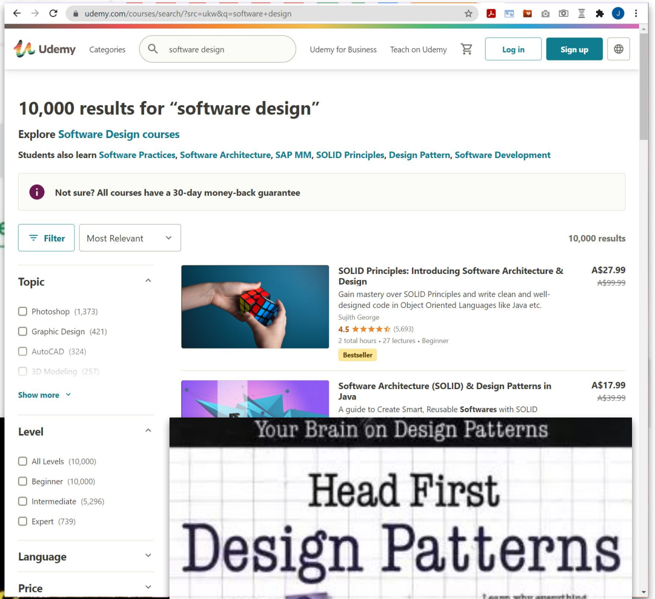
Software engineering practices to improve management | Nicky Thompson | #LeadDevBerlin

3.6K views • 1 year ago

LeadDev

Full talk title: Using **software engineering practices** to improve engineering management Video sponsor:

BEST PRACTICES
11:26
"Bro, it's a best practice"
6:02
Software engineering practices to improve management
Lead Dev
9:29
7 HABITS OF HIGHLY EFFECTIVE PROGRAMMERS
12:34
Coding



udemy.com/courses/search/?src=ukw&q=software+design

Udemy Categories software design Udemy for Business Teach on Udemy Log in Sign up

10,000 results for "software design"

Explore Software Design courses

Students also learn Software Practices, Software Architecture, SAP MM, SOLID Principles, Design Pattern, Software Development

Not sure? All courses have a 30-day money-back guarantee

Filter Most Relevant 10,000 results

Topic

- Photoshop (1,373)
- Graphic Design (421)
- AutoCAD (324)
- 3D Modeling (257)

Show more

Level

- All Levels (10,000)
- Beginner (10,000)
- Intermediate (5,296)
- Expert (739)

Language

Price

SOLID Principles: Introducing Software Architecture & Design — A$27.99 ~~A$99.99~~

Gain mastery over SOLID Principles and write clean and well-designed code in Object Oriented Languages like Java etc.

Sujith George

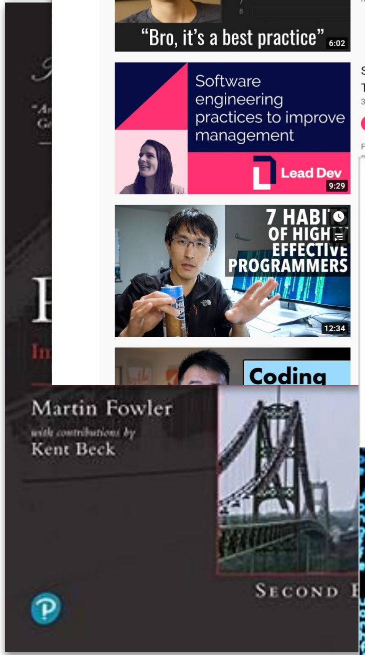
4.5 ★★★★★ (5,693)

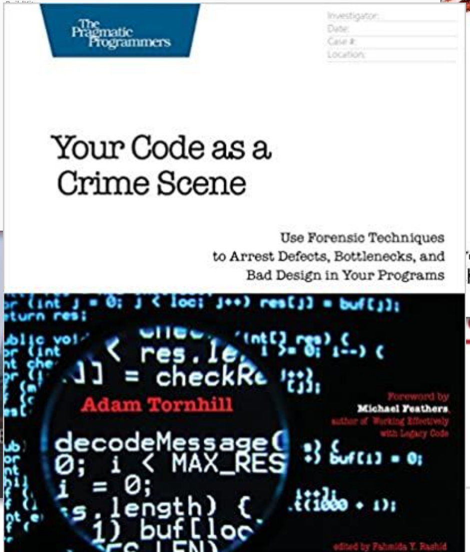
2 total hours • 27 lectures • Beginner

Bestseller

Software Architecture (SOLID) & Design Patterns in Java — A$17.99 ~~A$39.99~~

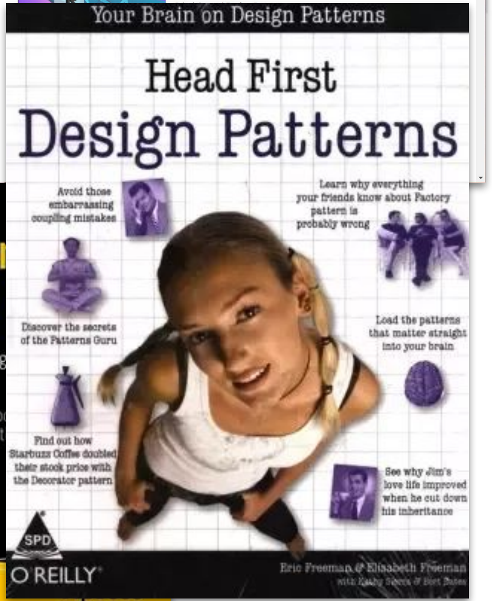
A guide to Create Smart, Reusable **Softwares** with SOLID





The Pragmatic Programmers
Your Code as a Crime Scene
Use Forensic Techniques to Arrest Defects, Bottlenecks, and Bad Design in Your Programs
Adam Tornhill
Foreword by Michael Feathers

Software Developmen
Design and
With Patterns, Debugging and Refactoring
Learn the principles of go design, and how to turn t into great code
Second Edition
John F. Dooley



Your Brain on Design Patterns
Head First
Design Patterns
Avoid those embarrassing coupling mistakes
Learn why everything your friends know about Factory pattern is probably wrong
Discover the secrets of the Patterns Guru
Load the patterns that matter straight into your brain
Find out how Starbuzz Coffee doubled their stock price with the Decorator pattern
See why Jim's love life improved when he cut down his inheritance
O'REILLY
Eric Freeman & Elisabeth Freeman

Challenge assumptions

But help find the balance

Avoid over engineered features

The more code the more Tech Debt

Bulletproof NFRs

Everchanging integrations tend to demand refactoring in several layers

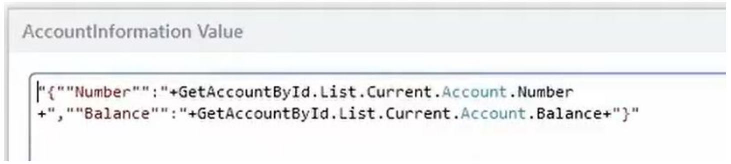
Identifying

Code Smell

CODE SMELLS ARE SYMPTOMS OF POOR DESIGN OR IMPLEMENTATION CHOISES

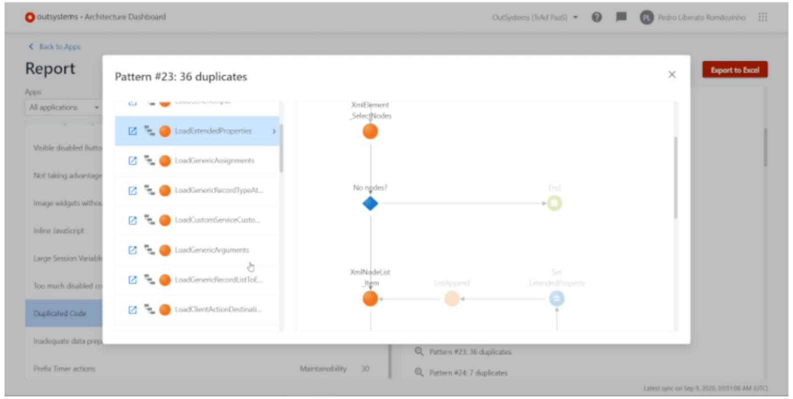
[Martin Fowler]

Code smells in OutSystems



AccountInformation Value

```
"{""Number"":"+GetAccountById.List.Current.Account.Number
+",""Balance"":"+GetAccountById.List.Current.Account.Balance+"}"
```



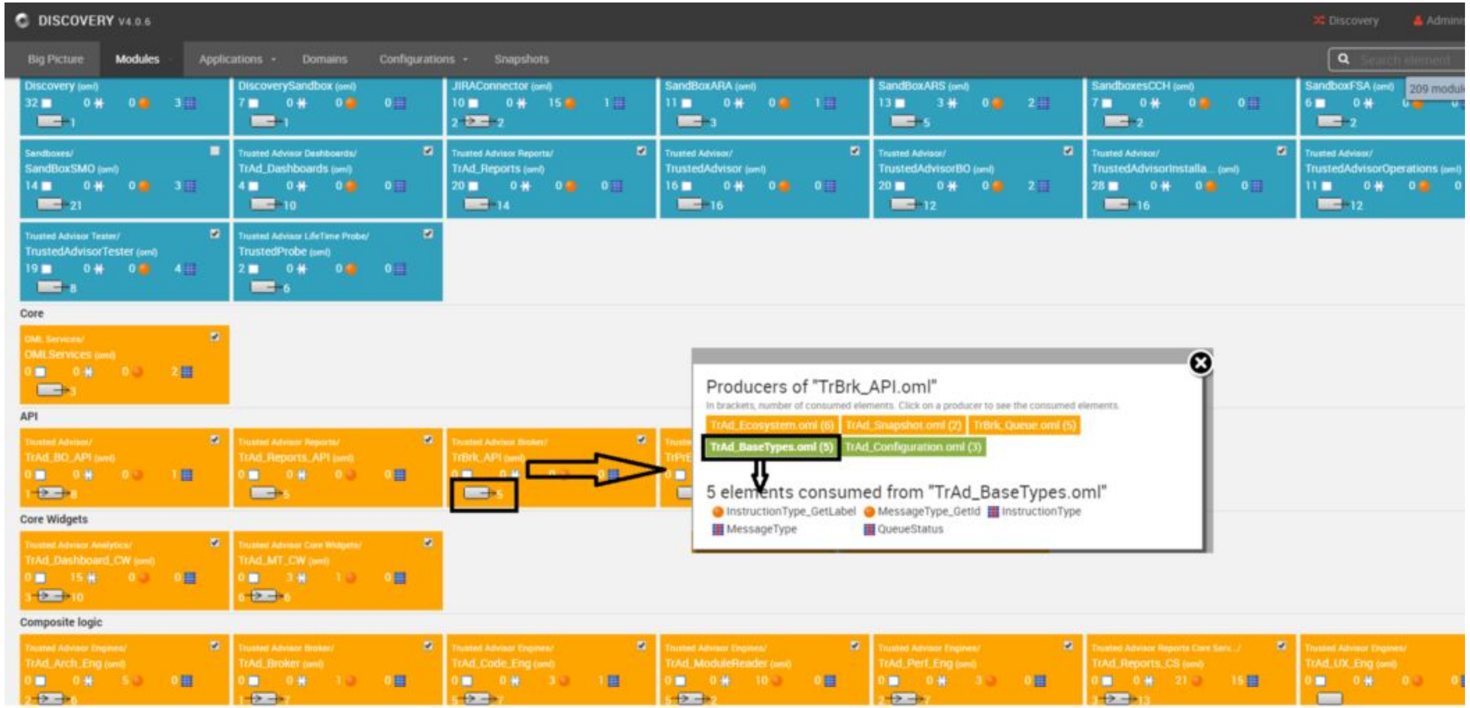
Easy to spot:

- Hardcoding
- Monolithic logic
- Duplicated code
- Cyclic references
- Incorrect dependencies
- Client x Server logic mixed up
- Undocumented components

In-Depth:

- Violation to domain rules
- Inconsistent data modeling
- Change frequency
- Cyclomatic complexity

OutSystems Discovery



Discovery (oml)
DiscoverySandbox (oml)
JIRAConnector (oml)
SandBoxARA (oml)
SandBoxARS (oml)
SandboxesCCH (oml)
SandboxFSA (oml)

Sandboxes/ SandBoxSMO (oml)
Trusted Advisor Dashboards/ TrAd_Dashboards (oml)
Trusted Advisor Reports/ TrAd_Reports (oml)
Trusted Advisor/ TrustedAdvisor (oml)
Trusted Advisor/ TrustedAdvisorBO (oml)
Trusted Advisor/ TrustedAdvisorInstalla... (oml)
Trusted Advisor/ TrustedAdvisorOperations (oml)

Trusted Advisor Tester/ TrustedAdvisorTester (oml)
Trusted Advisor LifeTime Probe/ TrustedProbe (oml)

Core

OML Services/ OMLServices (oml)

API

Trusted Advisor/ TrAd_BO_API (oml)
Trusted Advisor Reports/ TrAd_Reports_API (oml)
Trusted Advisor Broker/ TrBrk_API (oml)

Core Widgets

Trusted Advisor Analytics/ TrAd_Dashboard_CW (oml)
Trusted Advisor Core Widgets/ TrAd_MT_CW (oml)

Composite logic

Trusted Advisor Engines/ TrAd_Arch_Eng (oml)
Trusted Advisor Broker/ TrAd_Broker (oml)
Trusted Advisor Engines/ TrAd_Code_Eng (oml)
Trusted Advisor Engines/ TrAd_ModuleReader (oml)
Trusted Advisor Engines/ TrAd_Perf_Eng (oml)
Trusted Advisor Reports Core Serv.../ TrAd_Reports_CS (oml)
Trusted Advisor Engines/ TrAd_UX_Eng (oml)

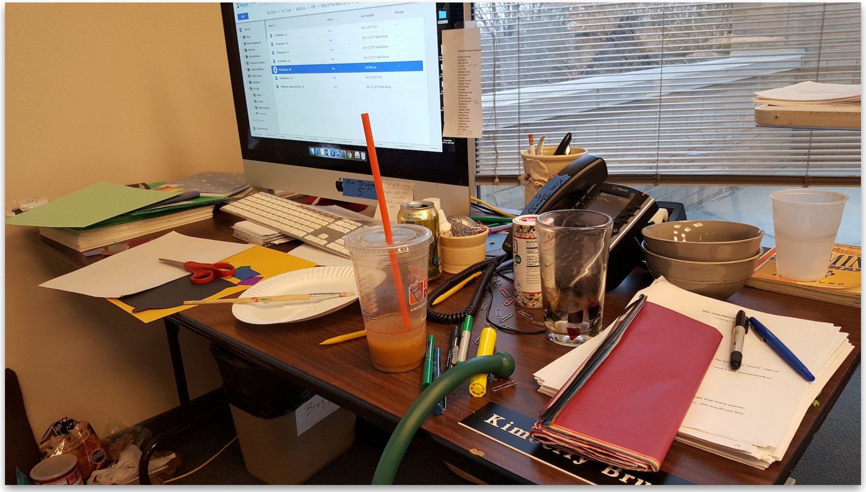
Producers of "TrBrk_API.oml"

In brackets, number of consumed elements. Click on a producer to see the consumed elements.

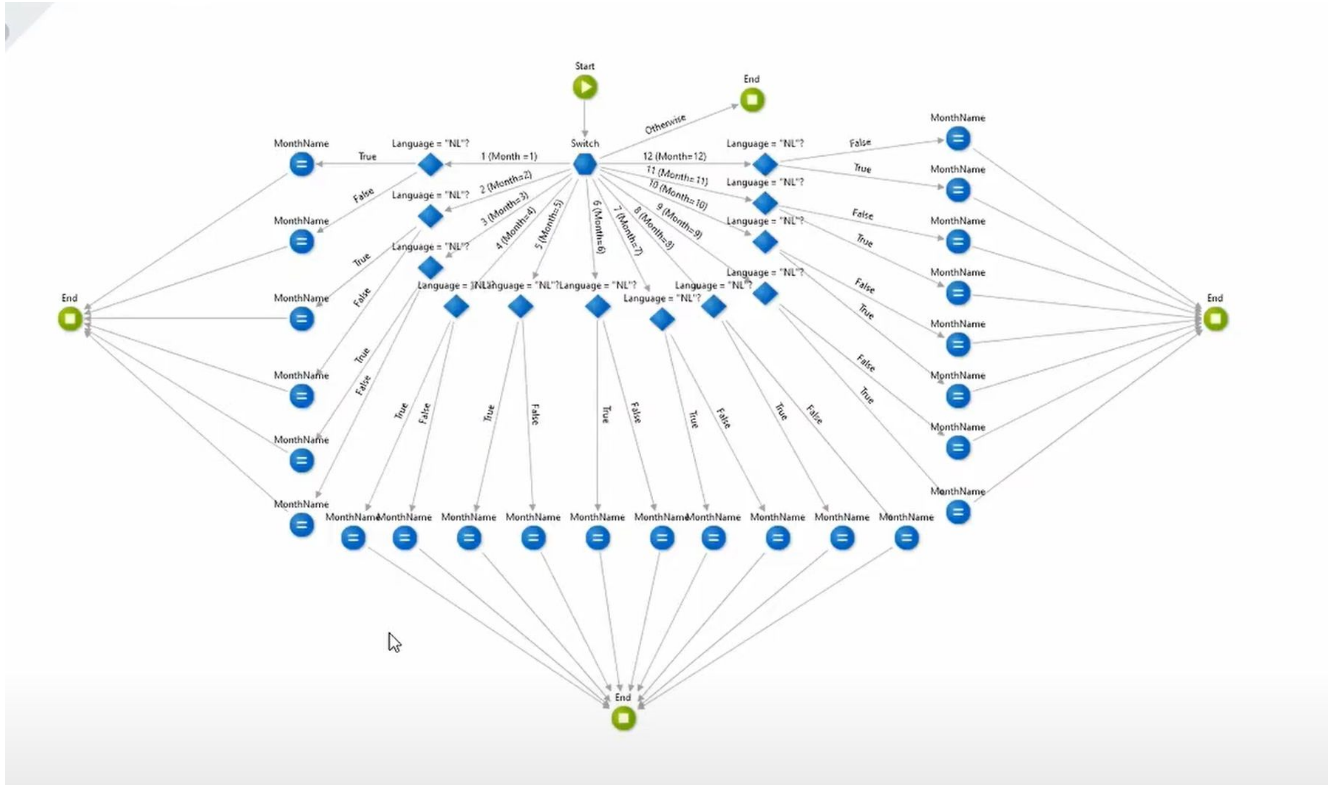
TrAd_Ecosystem.oml (6) TrAd_Snapshot.oml (2) TrBrk_Queue.oml (5)
TrAd_BaseTypes.oml (5) TrAd_Configuration.oml (3)

5 elements consumed from "TrAd_BaseTypes.oml"

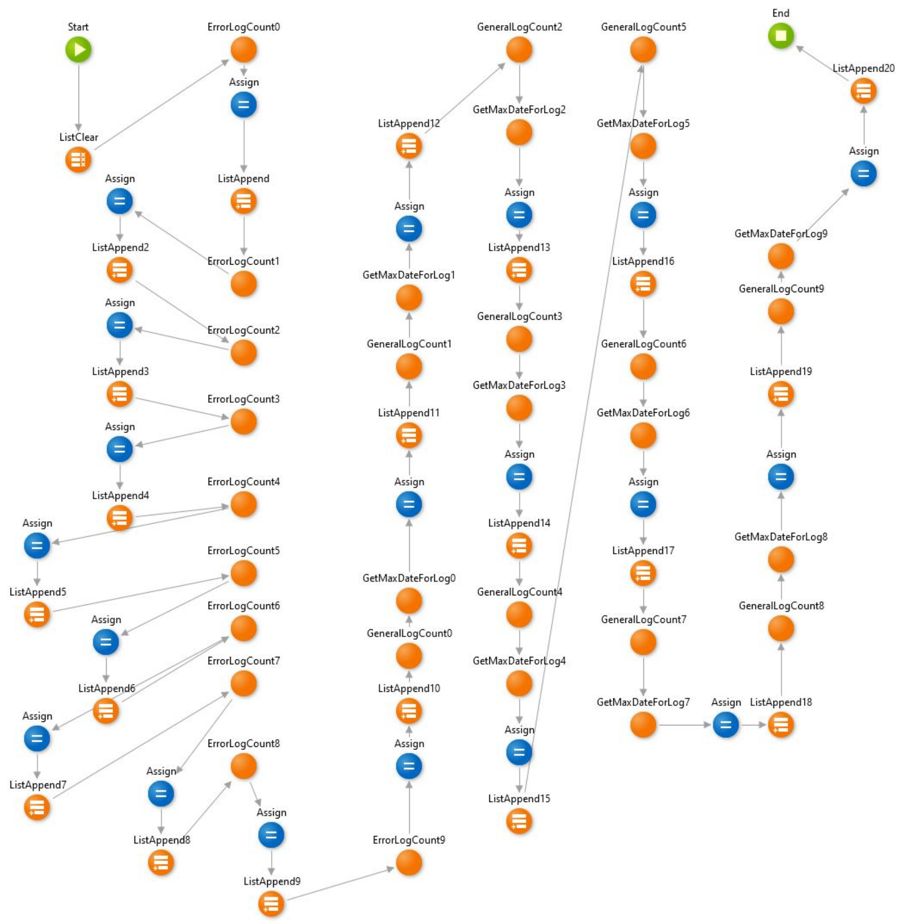
● InstructionType_GetLabel ● MessageType_GetId ▦ InstructionType
▦ MessageType ▦ QueueStatus



Coding Style



Coding Style



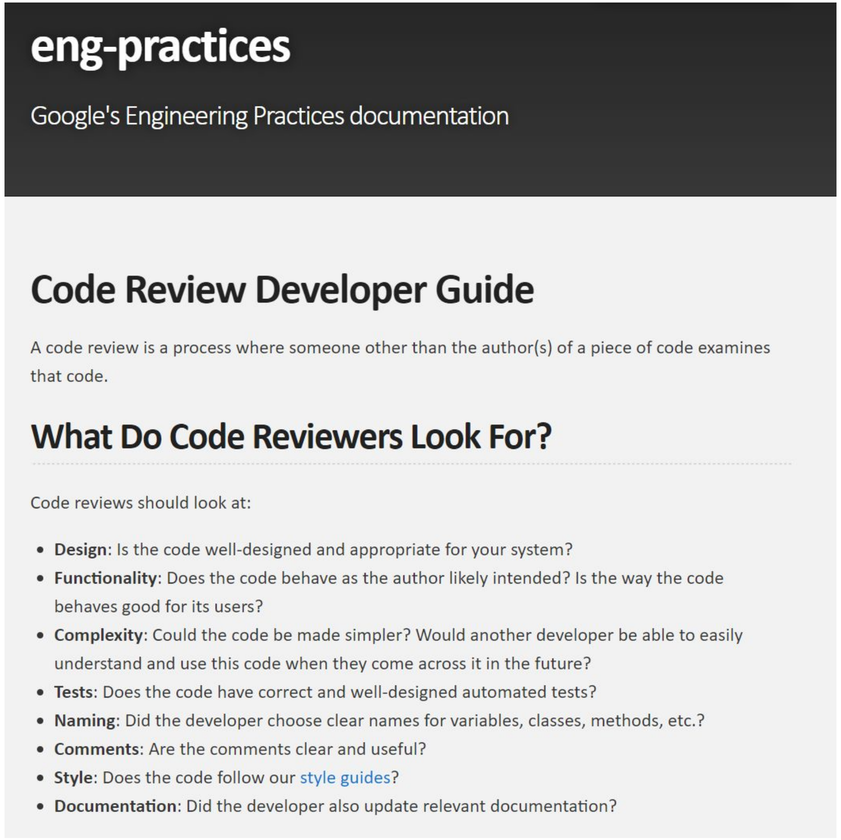
Code Reviews

Outer vision

Mentoring

Consensus

Shared ownership



eng-practices

Google's Engineering Practices documentation

Code Review Developer Guide

A code review is a process where someone other than the author(s) of a piece of code examines that code.

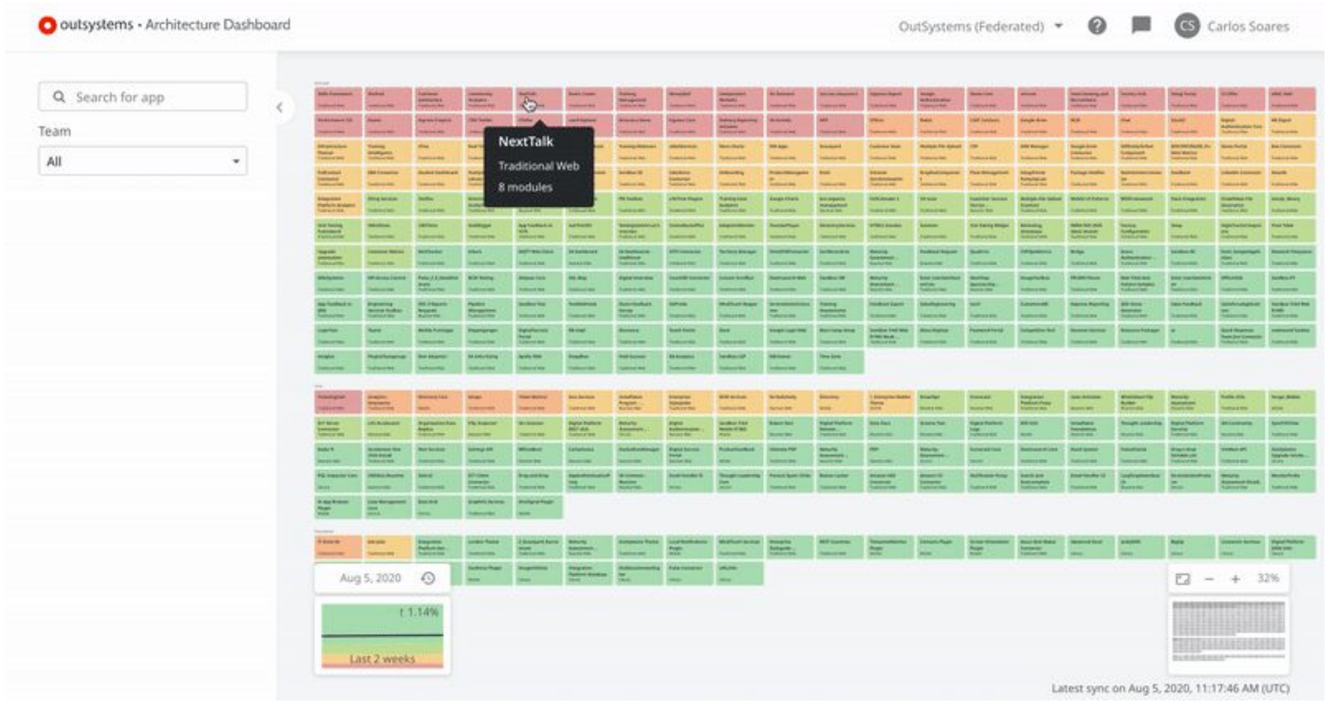
What Do Code Reviewers Look For?

Code reviews should look at:

- **Design**: Is the code well-designed and appropriate for your system?
- **Functionality**: Does the code behave as the author likely intended? Is the way the code behaves good for its users?
- **Complexity**: Could the code be made simpler? Would another developer be able to easily understand and use this code when they come across it in the future?
- **Tests**: Does the code have correct and well-designed automated tests?
- **Naming**: Did the developer choose clear names for variables, classes, methods, etc.?
- **Comments**: Are the comments clear and useful?
- **Style**: Does the code follow our style guides?
- **Documentation**: Did the developer also update relevant documentation?

https://google.github.io/eng-practices/review/reviewer/

OutSystems Architecture Dashboard



https://success.outsystems.com/Documentation/11/Managing_the_Applications_Lifecycle/Manage_technical_debt/Code_Analysis_Patterns

Managing

Bring the client to the party

Help them visualize it

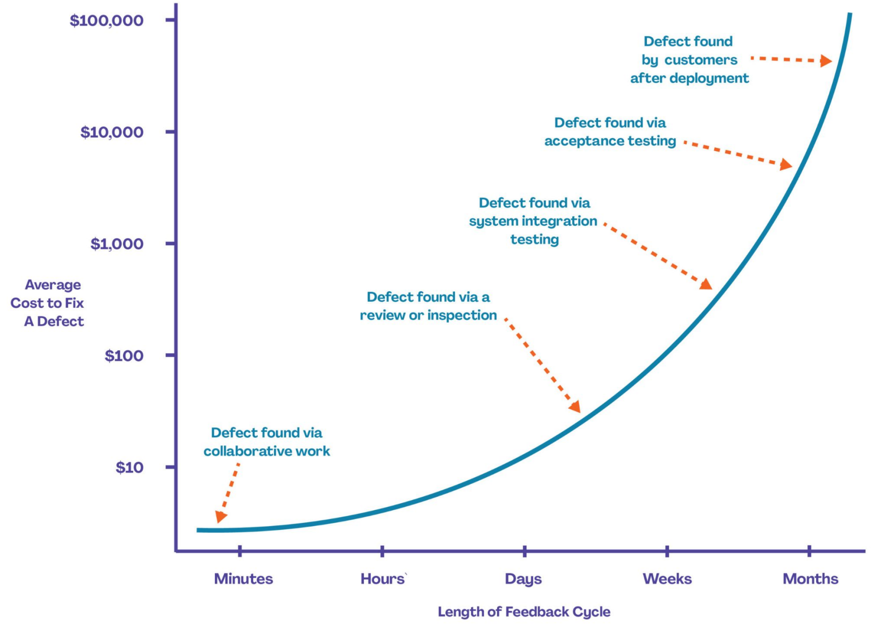
Communication is key!

Talk less techie

It's about business impact

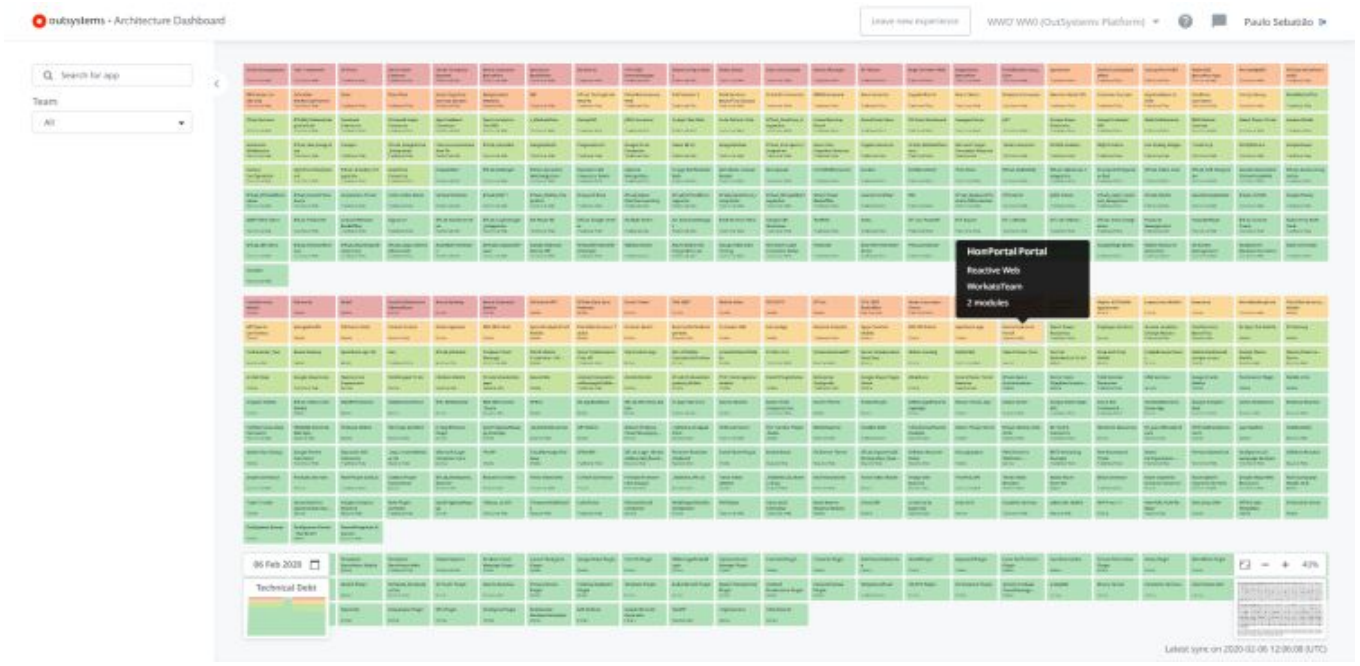
Visual storytelling

Be consistent



projectmanagement.com

Hotspot view



Combine tools to spice up the outcomes

Social aspects

How hard is it to keep evolving when a developer leaves the team?

Social aspects

Team coordination

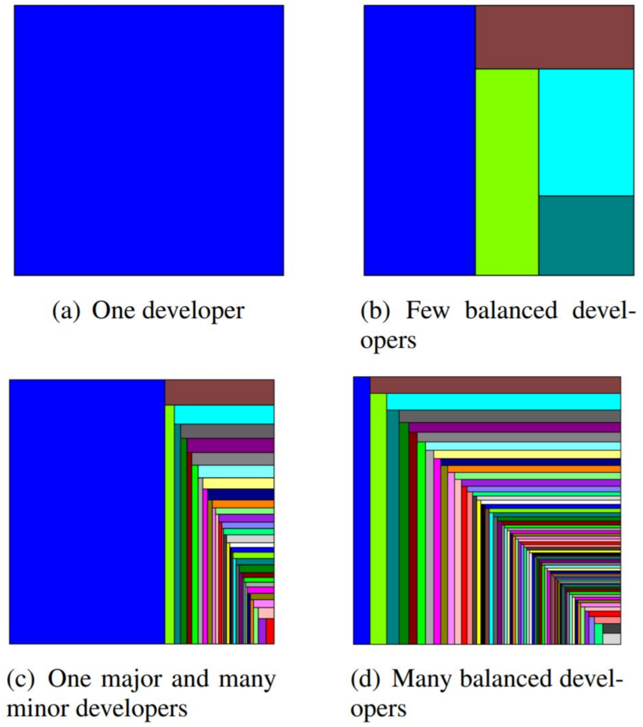
Diffusion of responsibility

Versioning system

Monitoring dependencies

Fractal Figures - Development patterns

The more balanced the better


(a) One developer

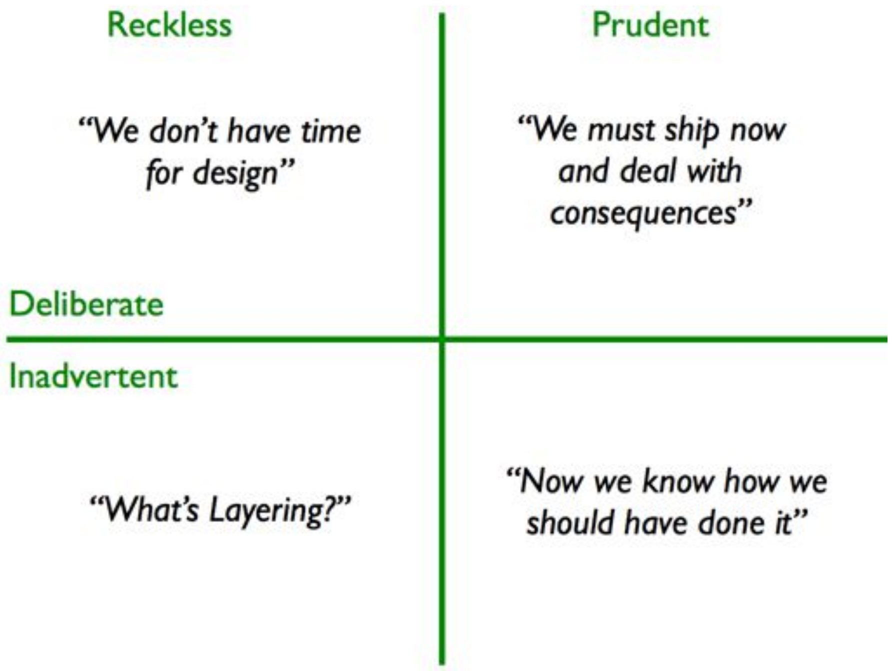
(b) Few balanced developers

(c) One major and many minor developers

(d) Many balanced developers

Fractal Figures: Visualizing Development Effort for CVS Entities
M. D'Ambros, M. Lanza, H. Gall

Check your temperature regularly


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

https://martinfowler.com/bliki/TechnicalDebtQuadrant.html

And how to prioritize?

It’s about:

- Finding a balance
- Aim at value
- Business continuity

Customers

Indicators:

- Interest rate
- Hotspot
- Change frequency
- Responsibility dispersion
- Early phases:
  - Architecture
  - Maintainability
- Close to Go Live
  - Security
  - Performance

Have constant checkpoints with the team

Learning opportunities

Be opportunistic

Technical Debt perception by OutSystems teams

References

- OutSystems’ Stop Tech Debt.com
- BMC’s Technical Debt explained - The complete guide to understanding and dealing with Technical Debt
- Gartner’s www.gartner.com/en/documents/3989188/manage-technology-debt-to-create-technology-wealth
- On Ward Cunningham:
    - en.wikipedia.org/wiki/Ward_Cunningham
    - www.youtube.com/watch?v=Jp5japiHAs4&t=8s
- Martin Fowler’s Technical Debt Quadrant
- Uncle Bob’s A Mess is not a Technical Debt
- Technical Debt - What to do?
- Does you OutSystems code smell?
- A code style guide for OutSystems
- refactoring.guru
- Google Engineering Practices
- Communicating with Management about Technical Debt
- Fractal Figures: Visualizing Development Effort for CVS Entities

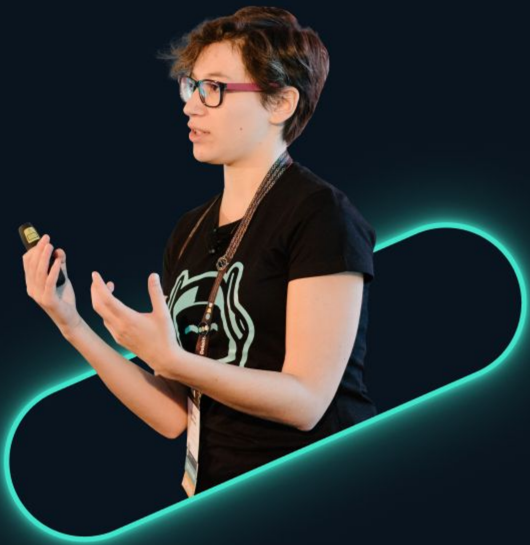
Q&A

OSDC

NOVEMBER 17-18

OSDC 2021

Build real solutions and make real connections, virtually.



50+ sessions

150+ countries

5+ tracks

outsystems.com/nextstep/OSDC

OSDC

Why Attend

Stay up to date

Find out about the latest developments in the OutSystems platform

Learn

Whether you're starting out or leveling up, benefit from hands-on expert led sessions

Connect

Join with other OutSystems developers around the world to collaborate, invent, and make new friends

Boost your career

See how to become an OutSystems expert and add new skills to your arsenal