

The background features a complex abstract graphic design. It includes several large, overlapping circles with black and white concentric patterns, some with yellow centers. There are also yellow splatters, black ink-like blotches, and thin black lines scattered across the light grey background. A thick black horizontal bar runs across the middle of the page, partially overlapping the text.

The Economics of Legacy Code

terry@odd-e.com



Who Am I

- I am Terry Yin
- Work for Odd-e
- Experienced in software development
- Love programming
- A father
- Live in Singapore

改善

```
1 using System;
2 using Microsoft.VisualStudio.TestTools.UnitTesting;
3
4 namespace GameOfLifeTests
5 {
6     [TestClass]
7     public class UnitTest1
8     {
9         //ctrl+R, A => run
10
11         [TestMethod]
12         public void SamePositionEachOther()
13         {
14             var world = new World();
15             Position startPos = world.StartPosition;
16             Assert.AreEqual(startPos, world.StartPosition);
17         }
18     }
19 }
```







What is Legacy Code?

The Odd-e CSD Classes



ONE WEEK. ONE SPRINT

THE WHOLE (10 PEOPLE) CLASS WORKS AS ONE TEAM.

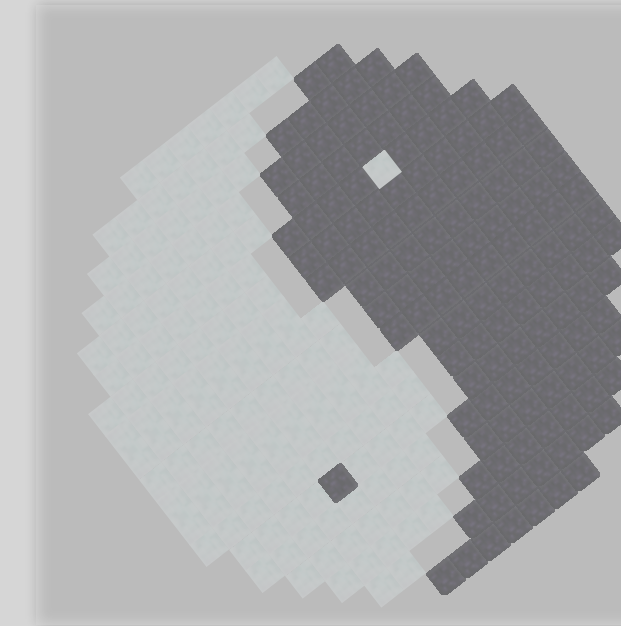
**INSTRUCTORS ACT AS PRODUCT OWNER,
SCRUMMASTER AND TECH COACH**



UNLIKE THIS GENTLEMAN WHO ITERATES ONLY FOR FUN...



**THE TEAM ITERATES TO DELIVER WORKING FEATURES
CONTINUOUSLY
ALONG WITH THE FUN.**



IN EACH CSD CLASS ...

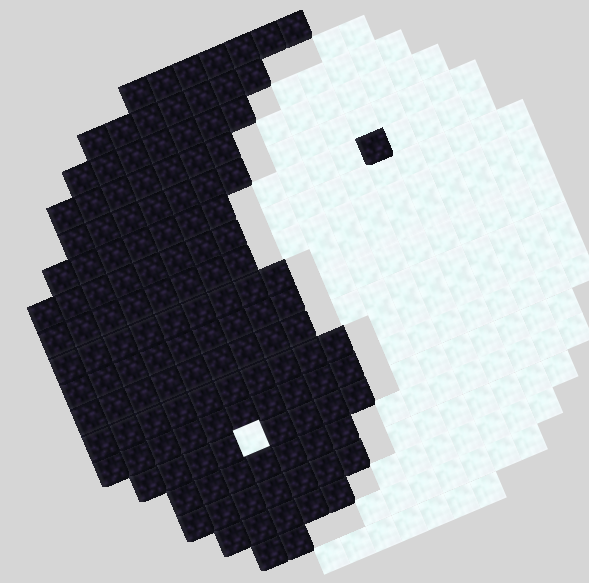


IN EACH CSD CLASS ...

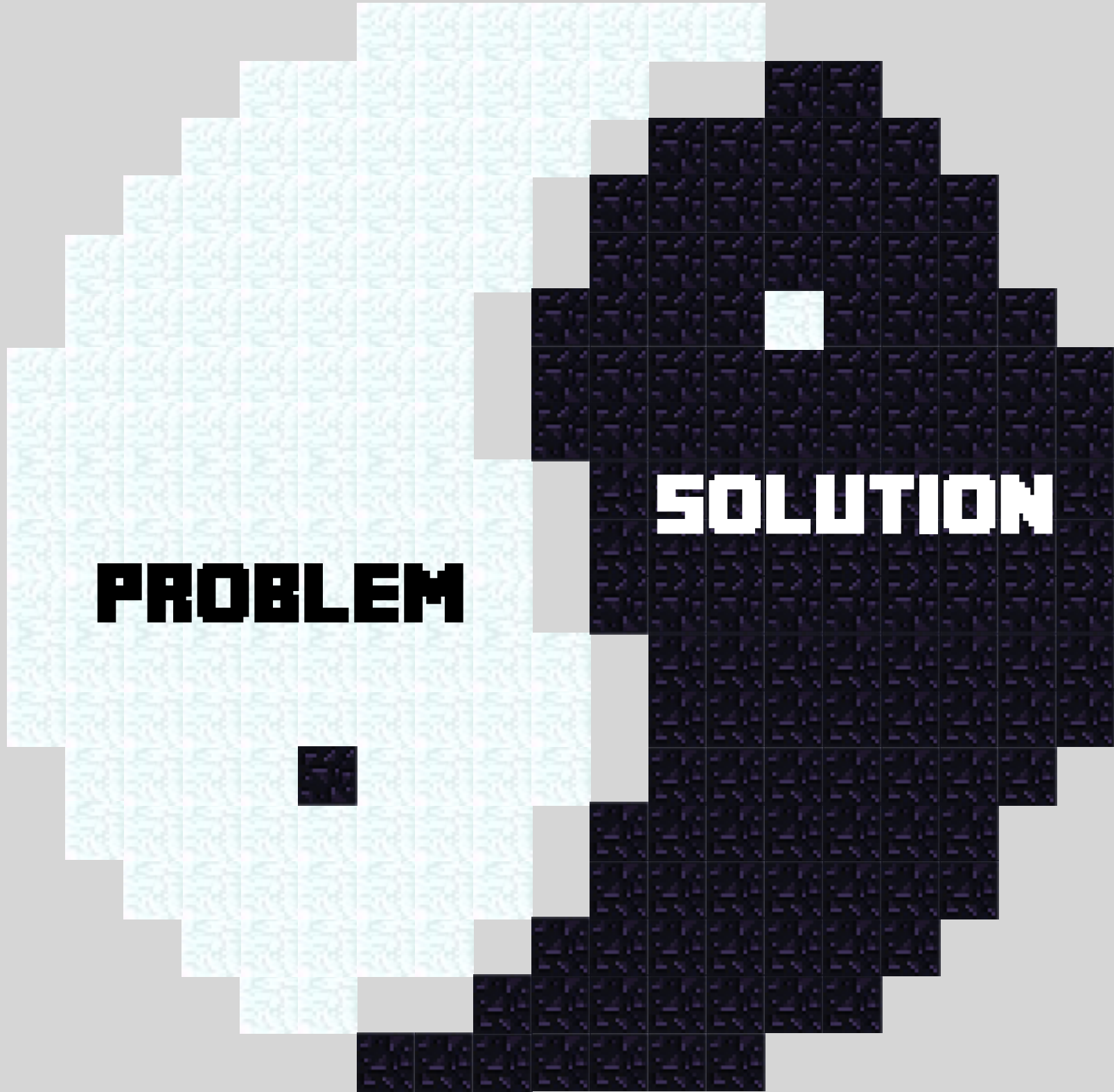
TEAM CONTINUES TO BUILD THE SAME PRODUCT ON THE SAME CODEBASE.



**SOFTWARE DEVELOPMENT IS MOSTLY ABOUT
KNOWLEDGE ACQUISITION AND PRESERVATION**

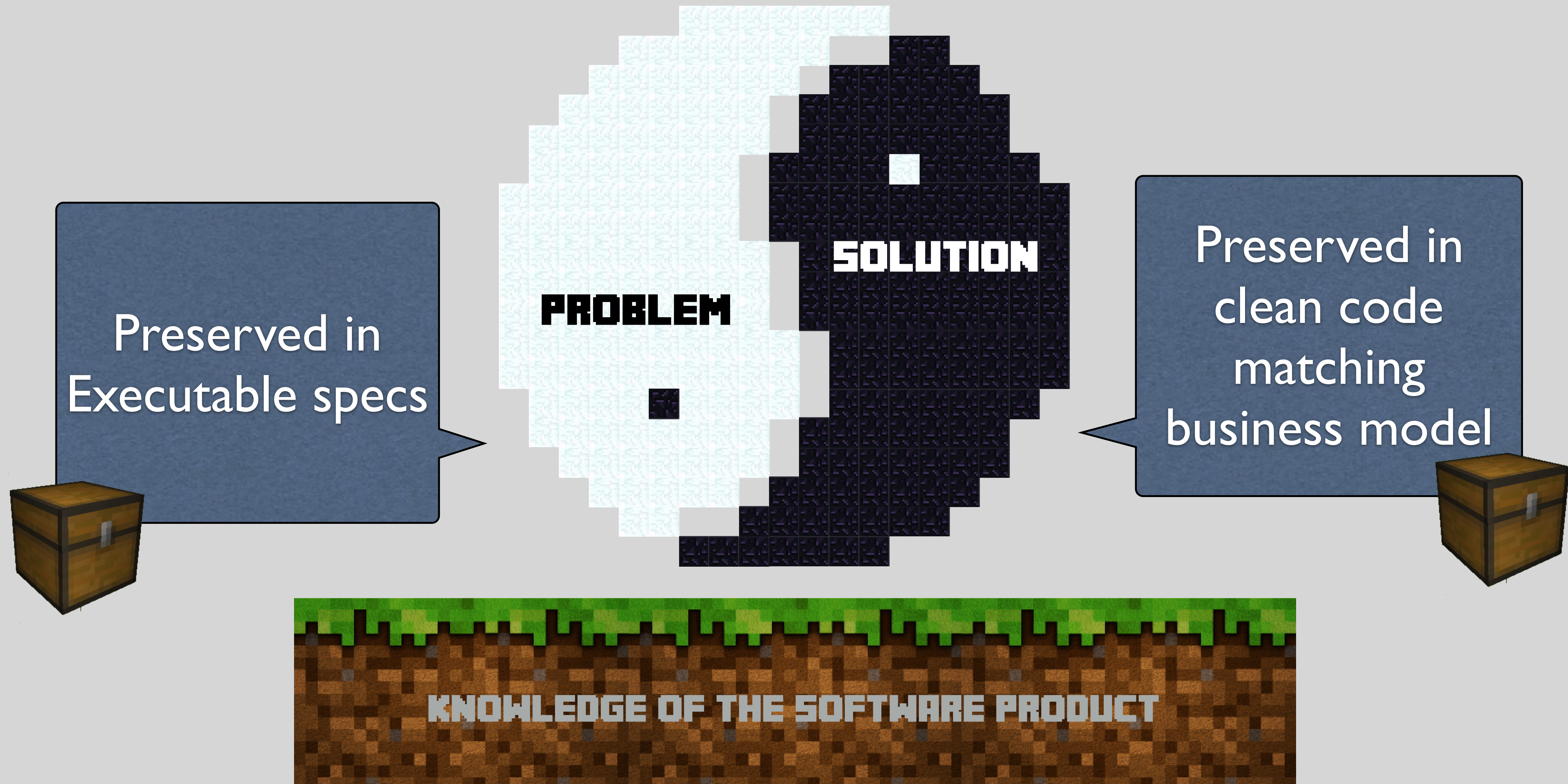


Knowledge acquired by understanding



Knowledge acquired by discovering





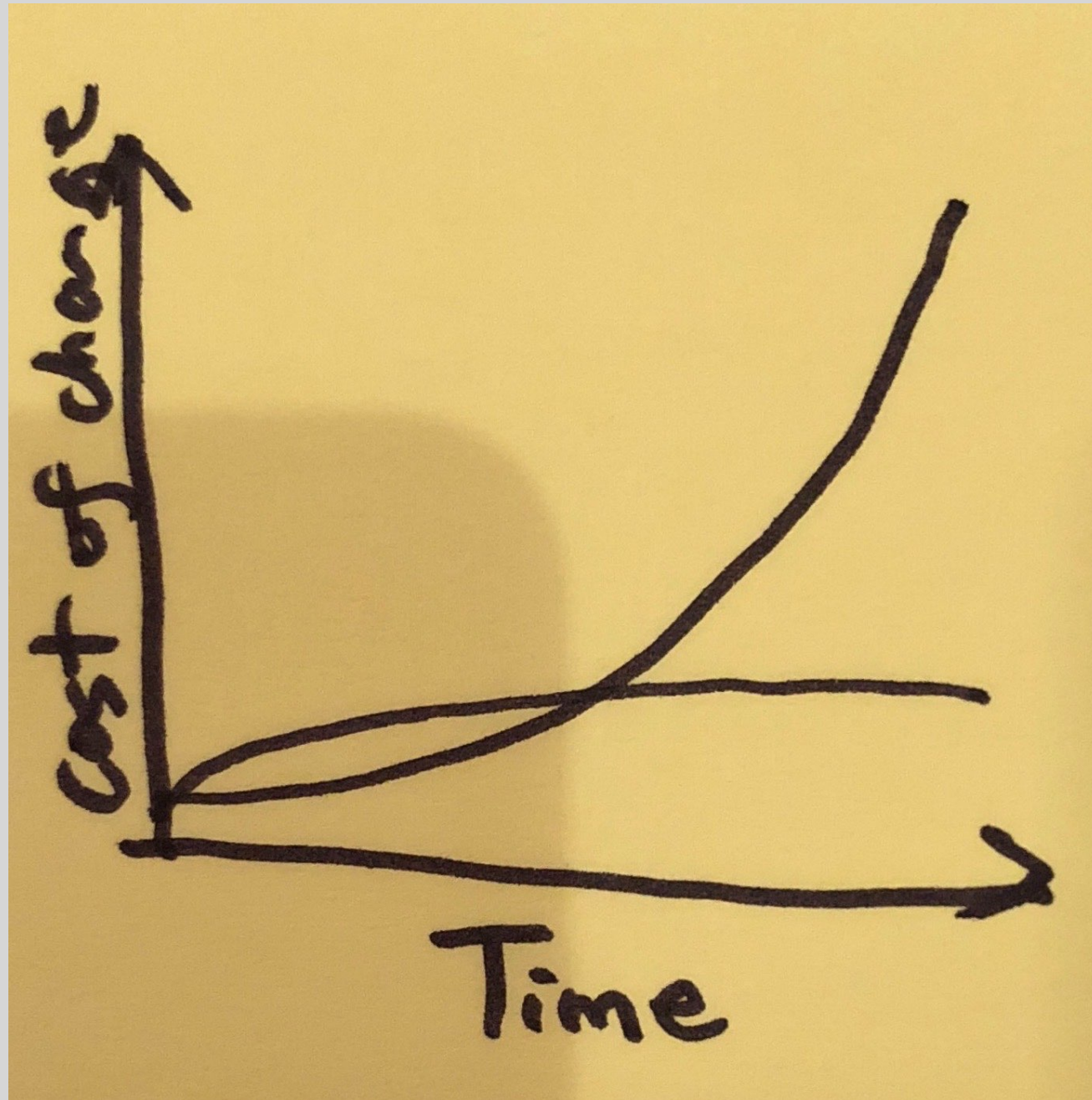
LEGACY CODE



**SOFTWARE DEVELOPMENT IS MOSTLY ABOUT
KNOWLEDGE ACQUISITION AND**

~~**PRESERVATION**~~

Cost and Interest





3 Perspectives

Automated (Unit) Test

Architecture

Organization

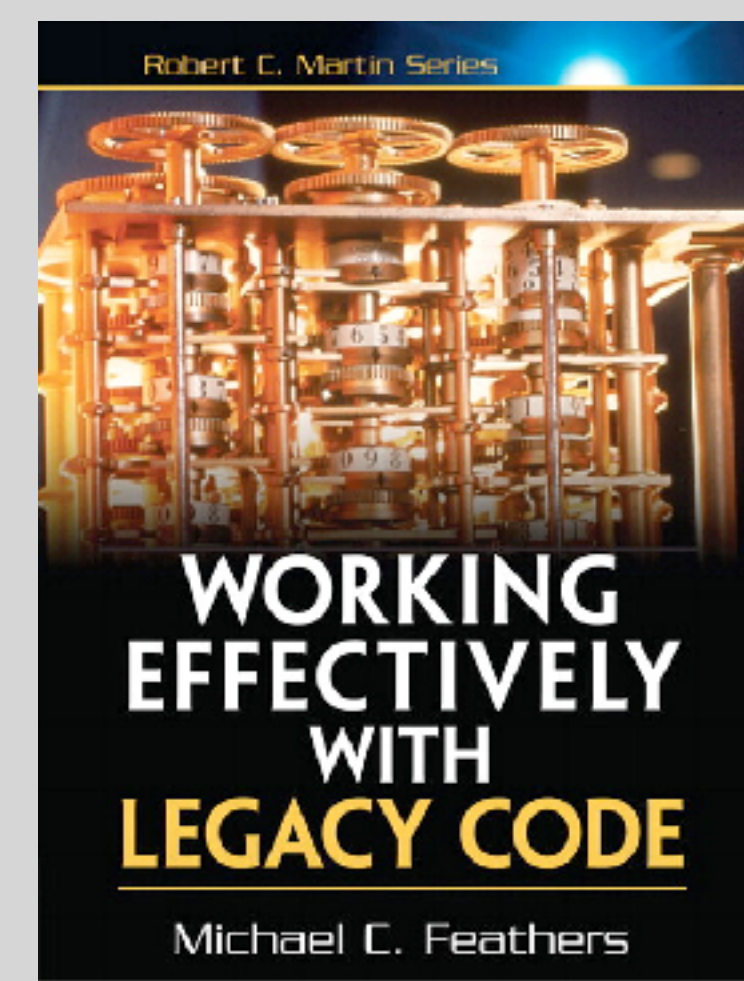
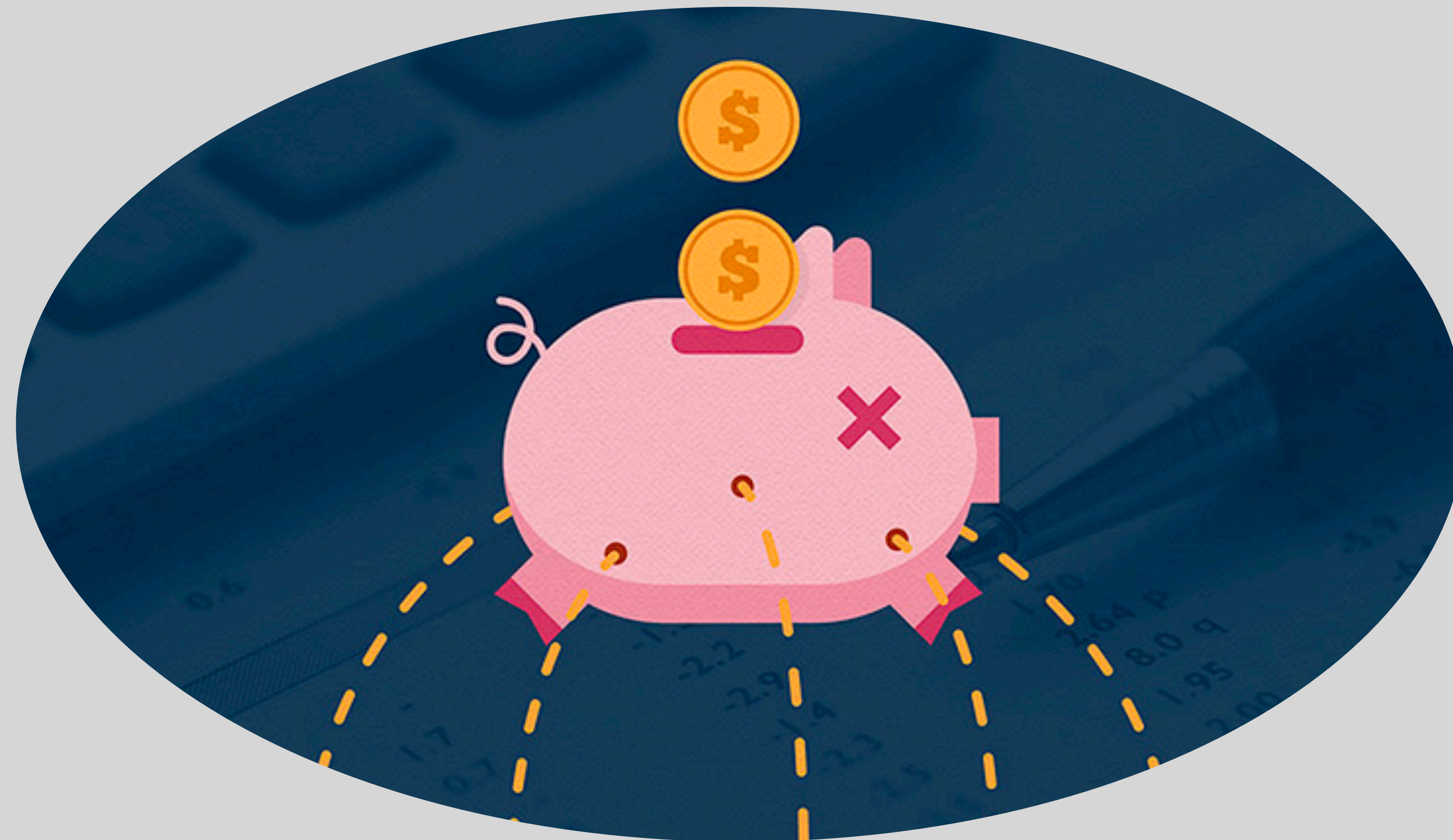


#1

Legacy Code
and

Automated (Unit) Test

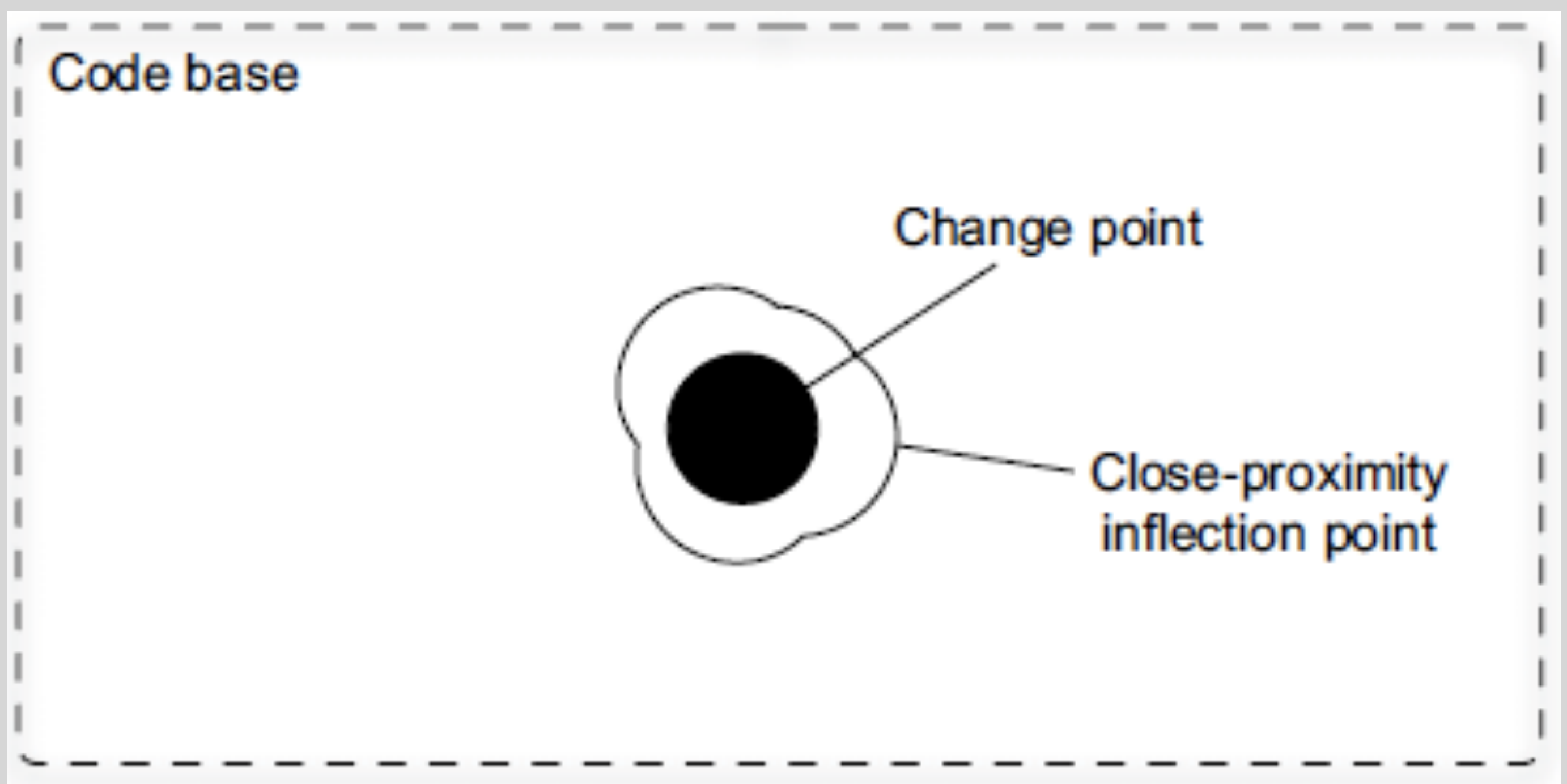
Code With No Test





What is the first thing to do,
when you find yourself in a hole?

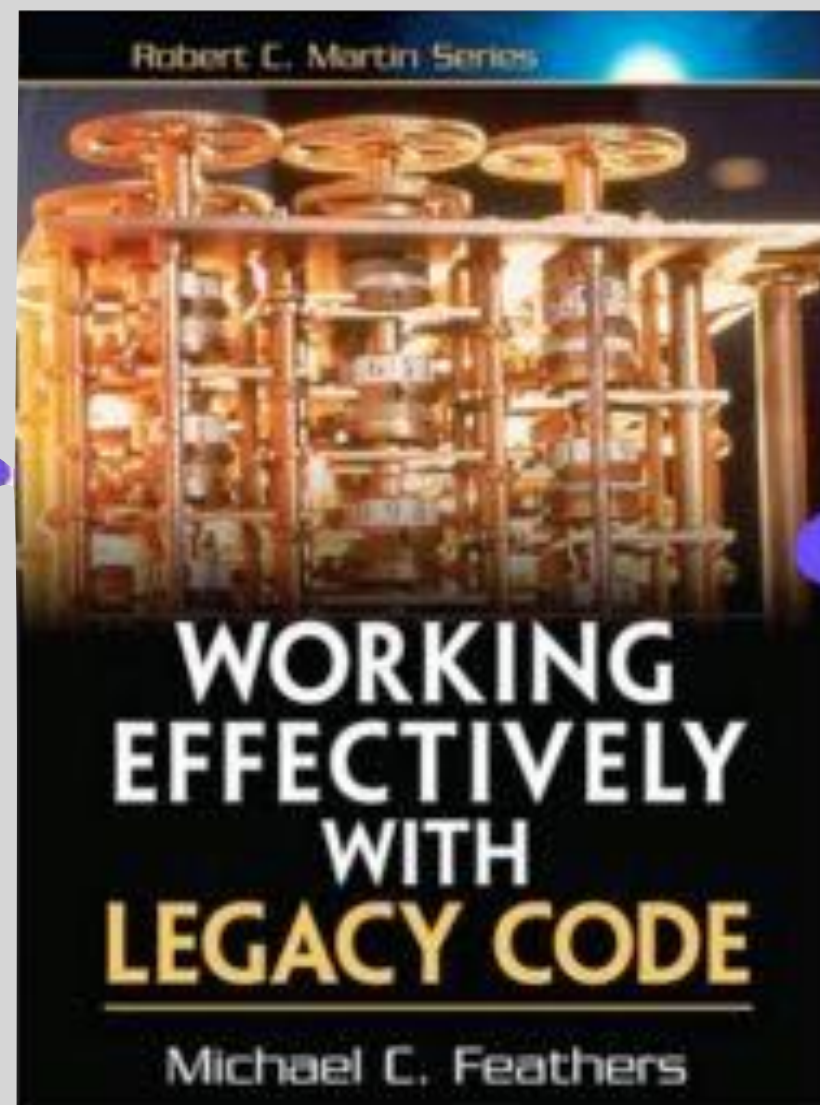
Legacy code algorithm



1. Identify change point
2. Find test points
3. Break dependencies
4. Write tests
5. Make changes and refactor

Challenge from working with legacy code: Breaking Dependencies

Sensing

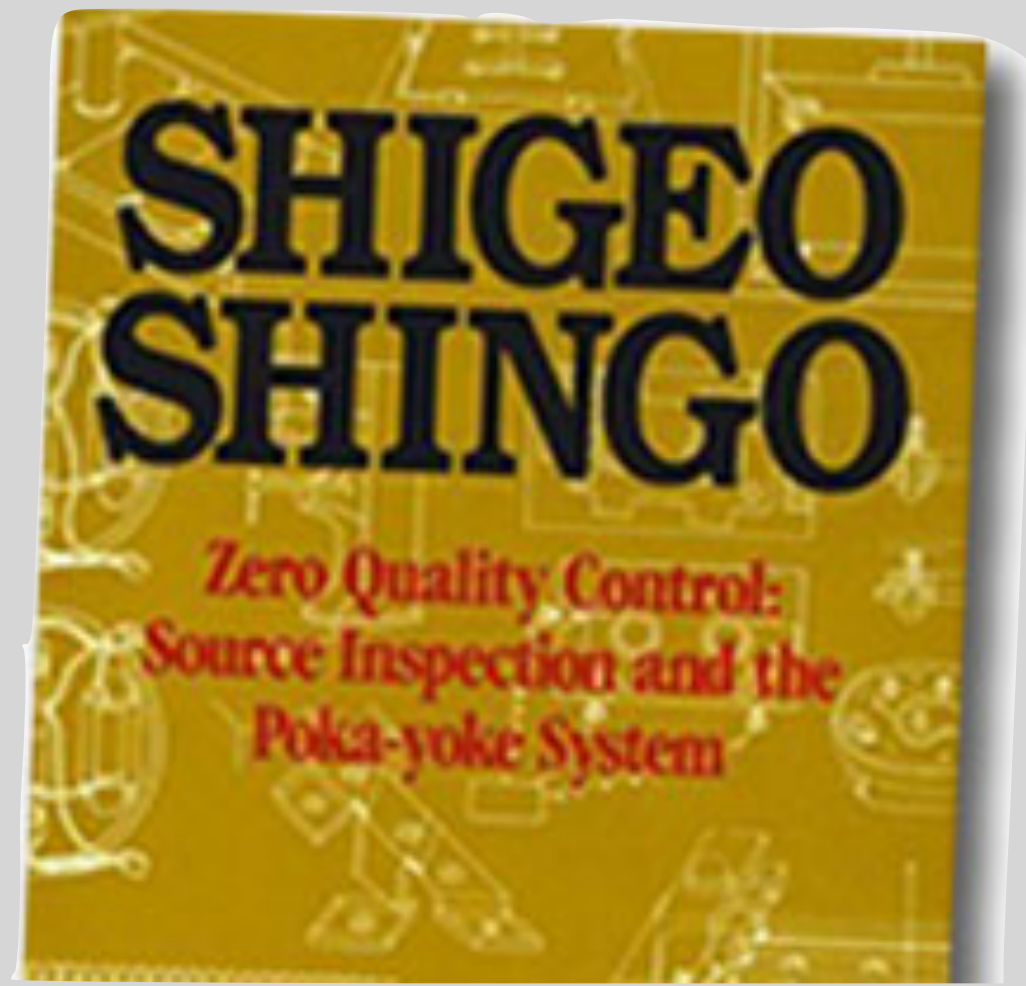


Separation



Autonomation vs Automation

自働化 ≠ 自動化





#2

Legacy Code
and

Architecture

High Cohesion, Low Coupling

**But high cohesion has
higher priority**



Craig Larman

High Cohesion



Cost of Not Having High Cohesion

Compound Interest Formula (including Principal)

Amount **Interest Rate (decimal)**

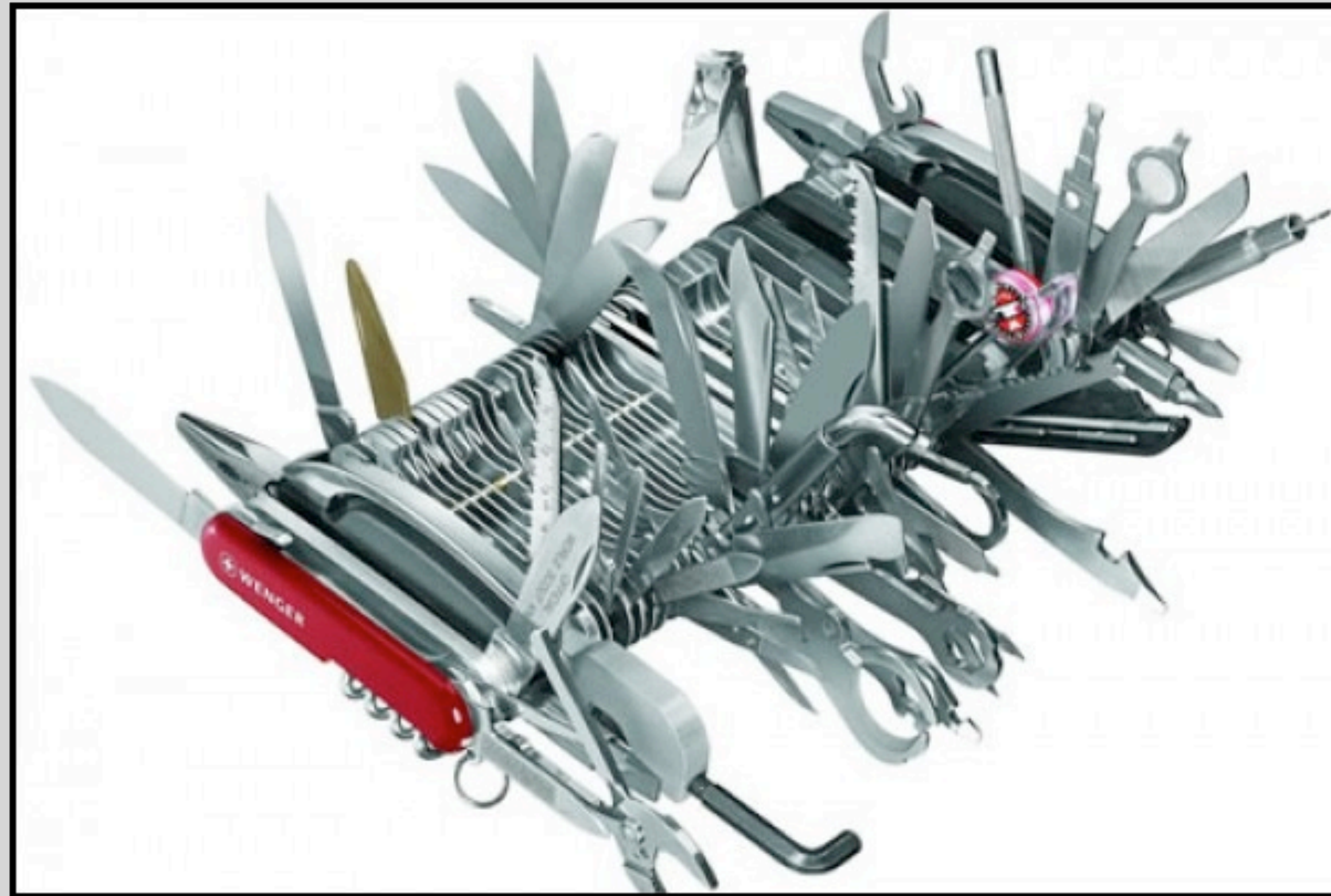
A = P (1 + $\frac{r}{n}$)^{nt} **Time (years)**

Principal **Number of times interest is compounded per year**

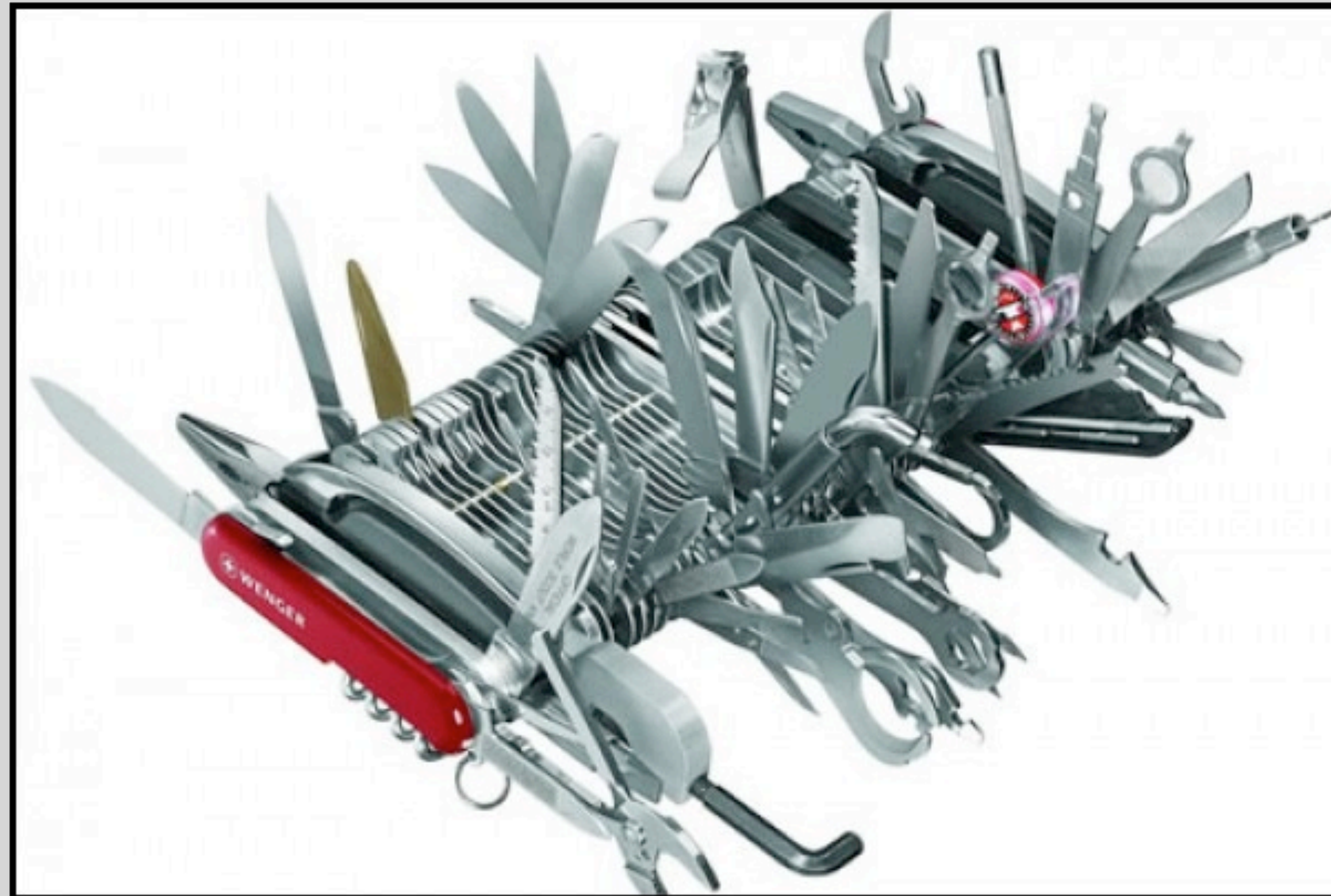
thecalculatorsite.com

Compound interest

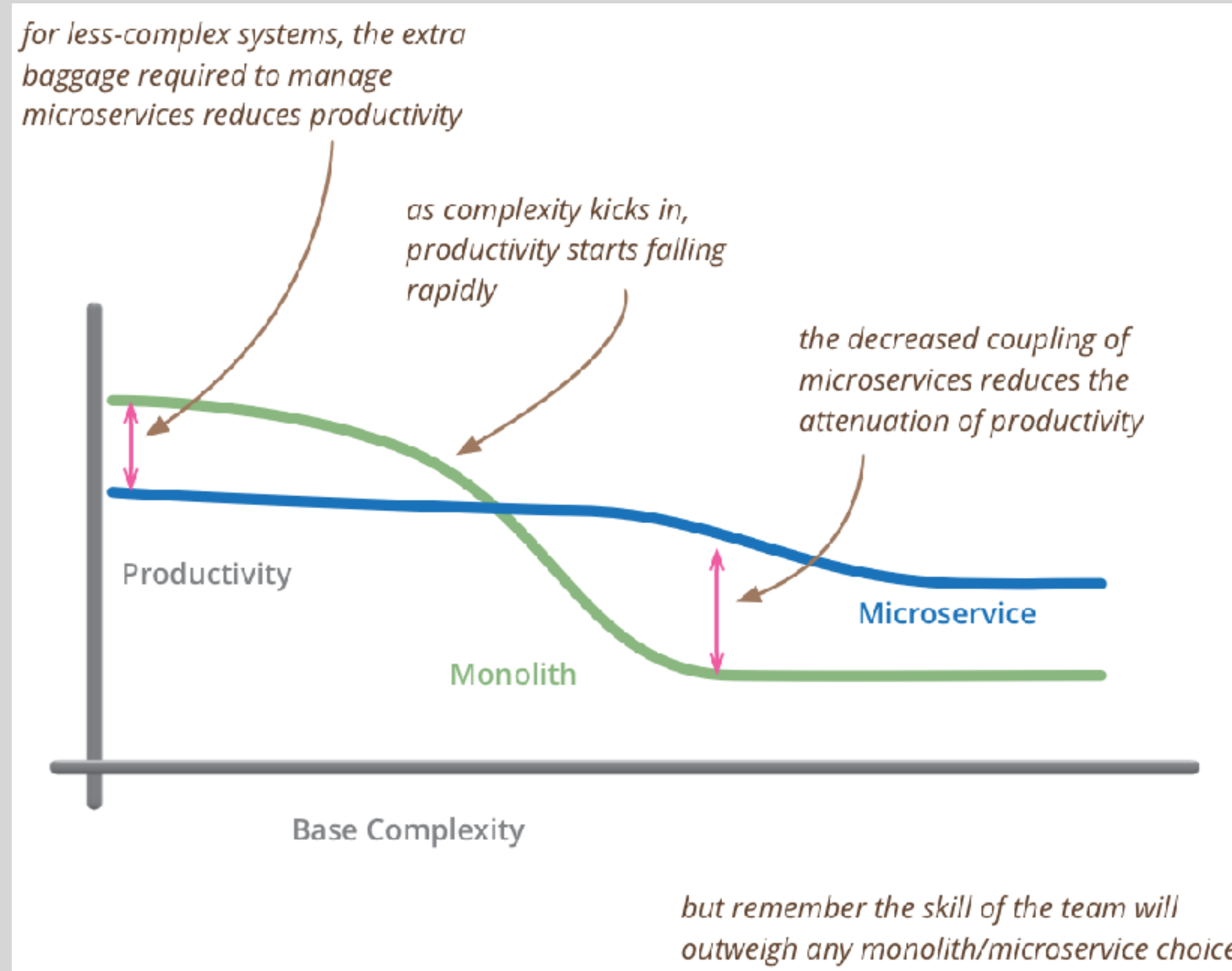
Low Coupling



Cost of Not Having Low Coupling?



Microservice



Martin Fowler

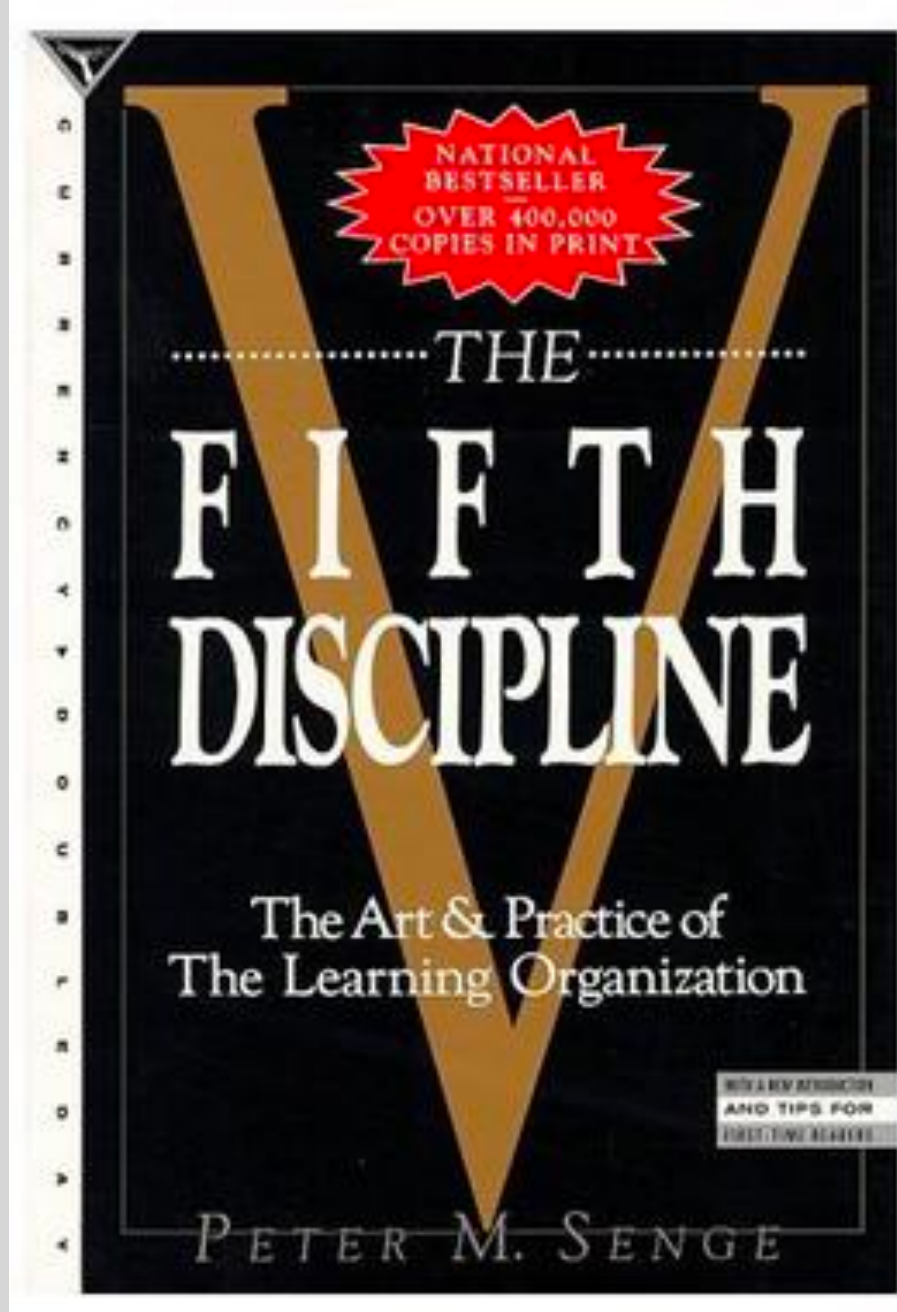
Picture from: <https://www.martinfowler.com/>



#3

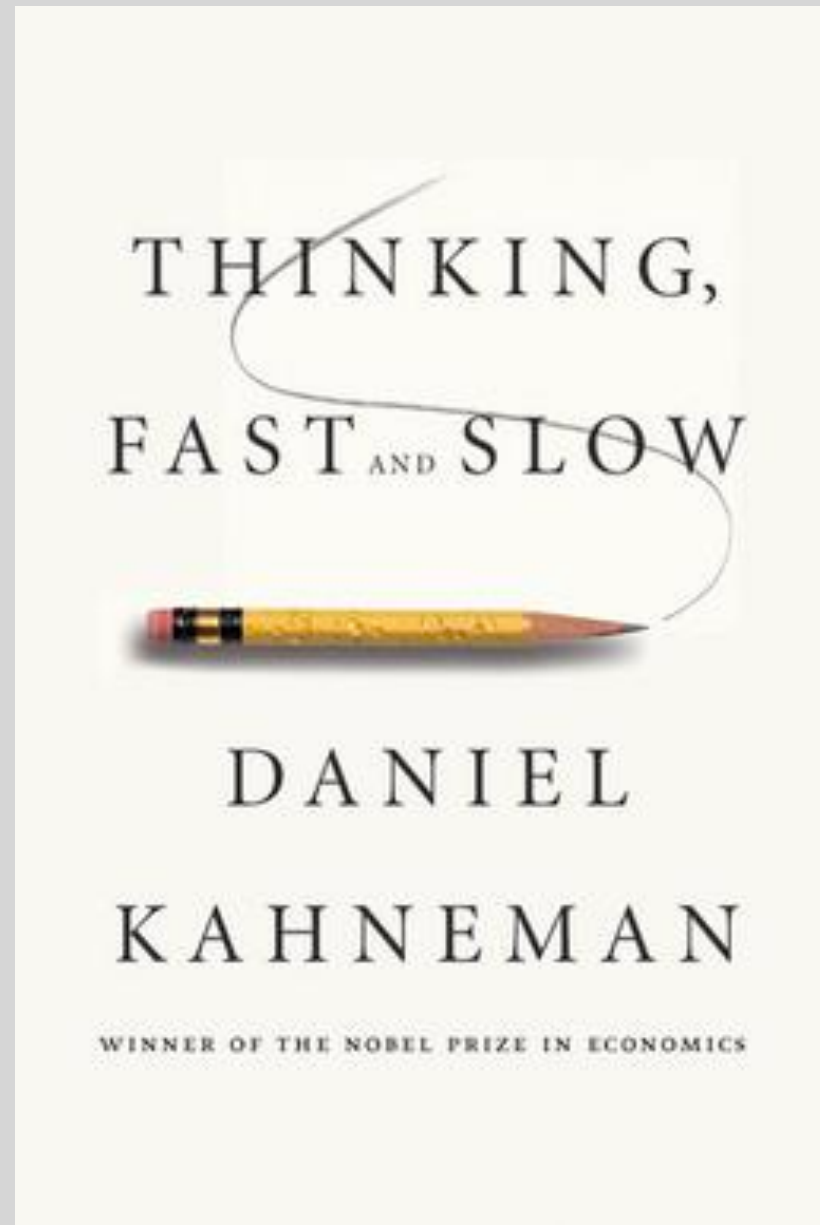
Legacy Code
and

Organization

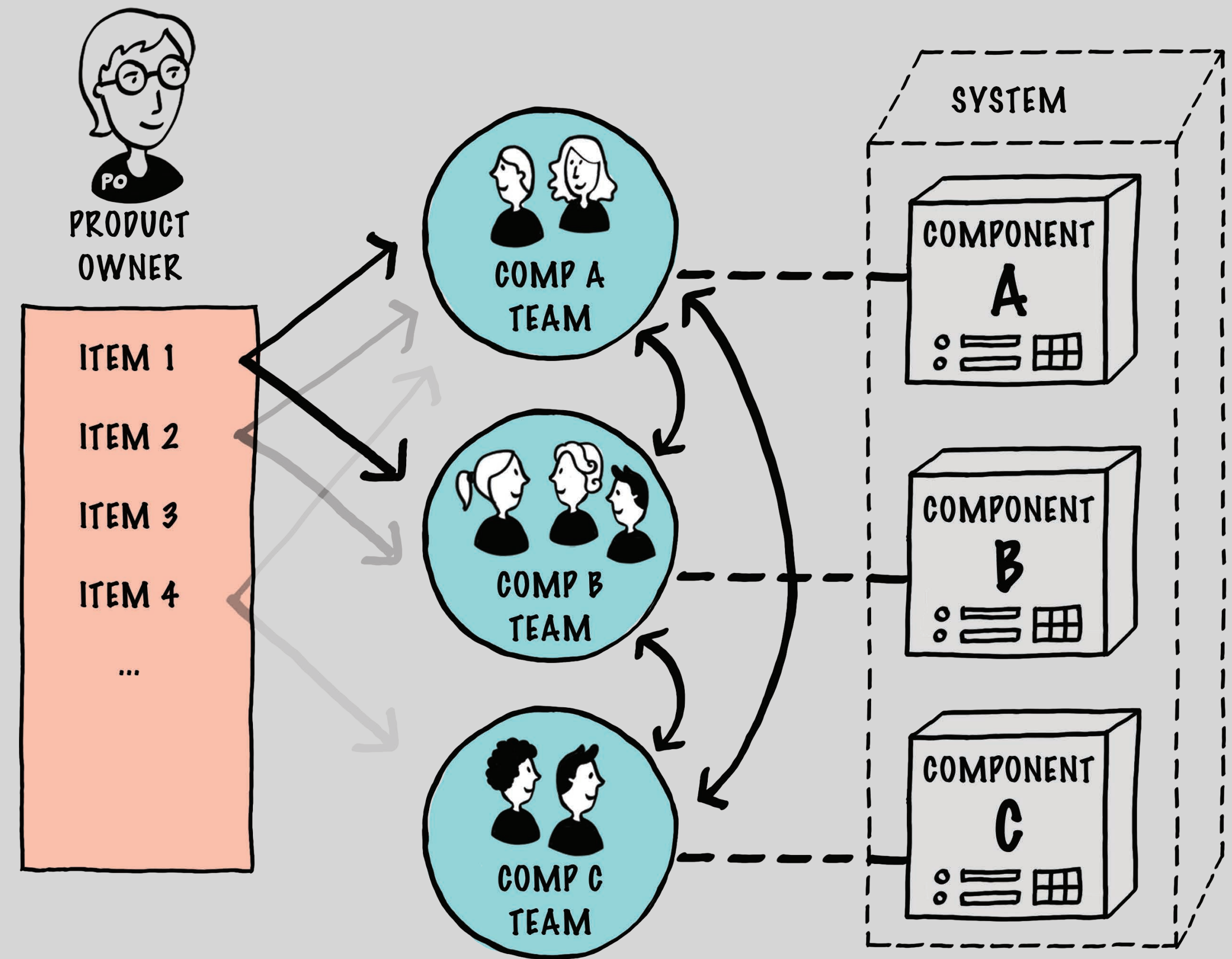


Global optimisation vs. Local optimisation

Human instinct are often
prone to local optimisation



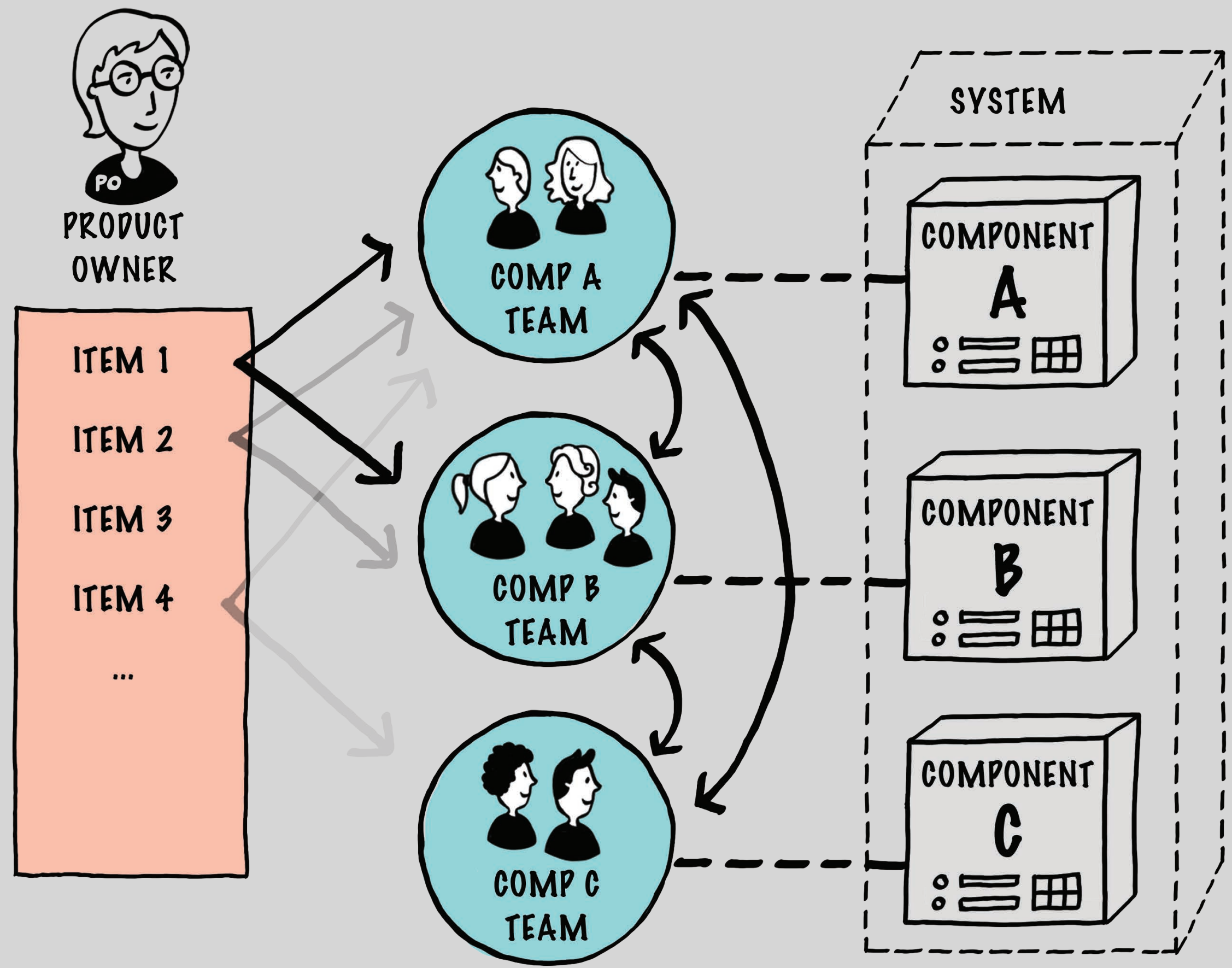
Conway's Law



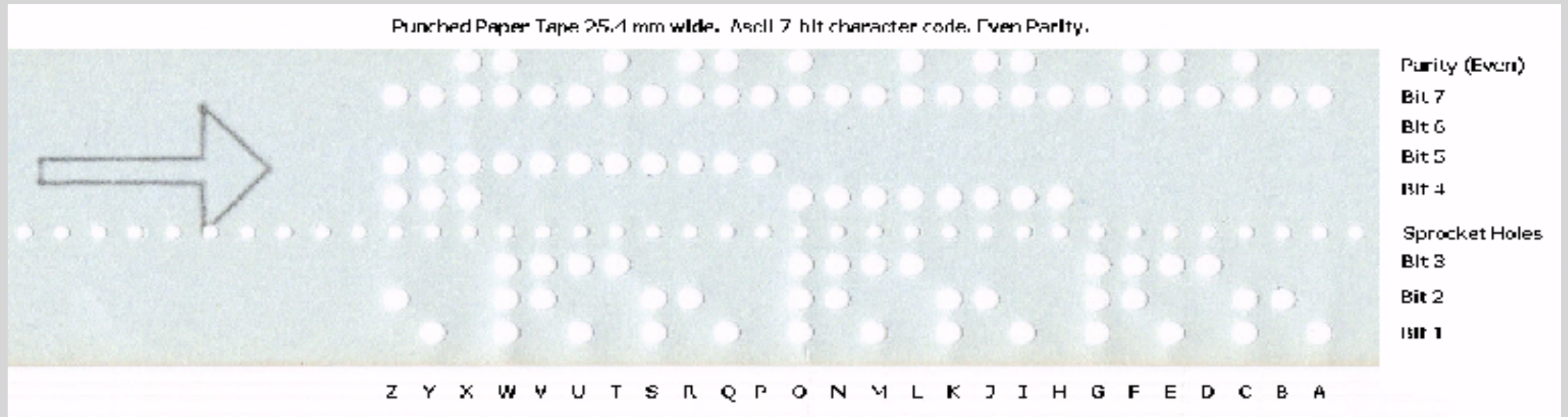
"organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations."

Melvin Conway, 1967

Following Conway's Law — Component Team



Trying to Preserve knowledge by organisational structure

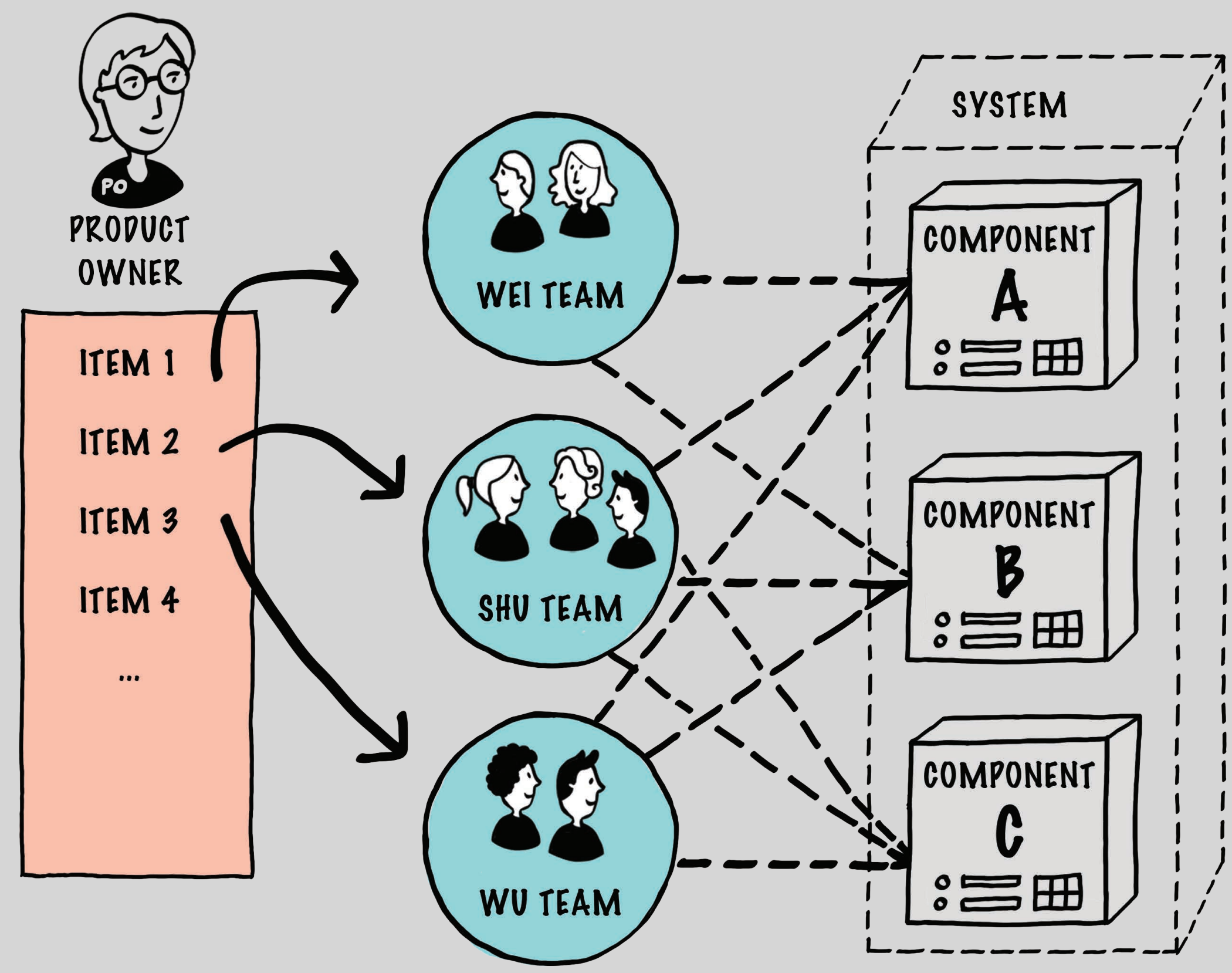


vs.

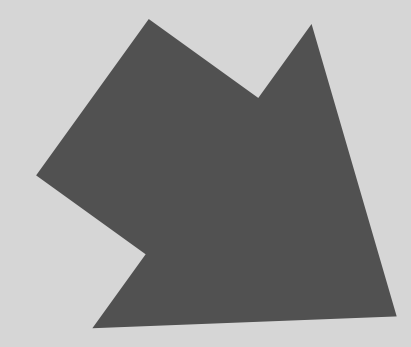


Representational Gap

Business Domain vs. Solution Domain

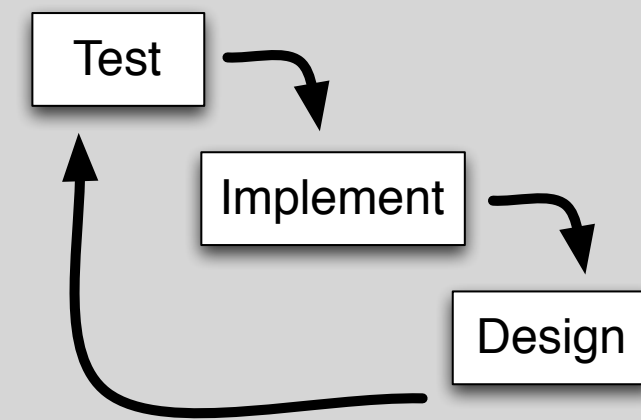


Lower representational gap



Less legacy code

Collective Code Ownership Needs Supporting Practices



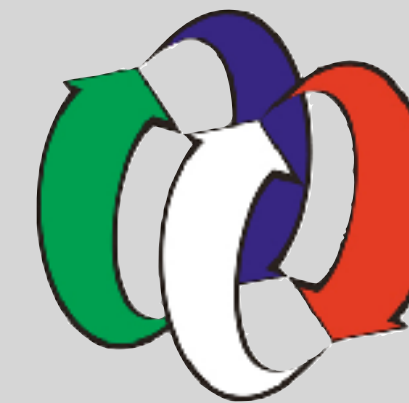
Test Driven Development



Pair Programming

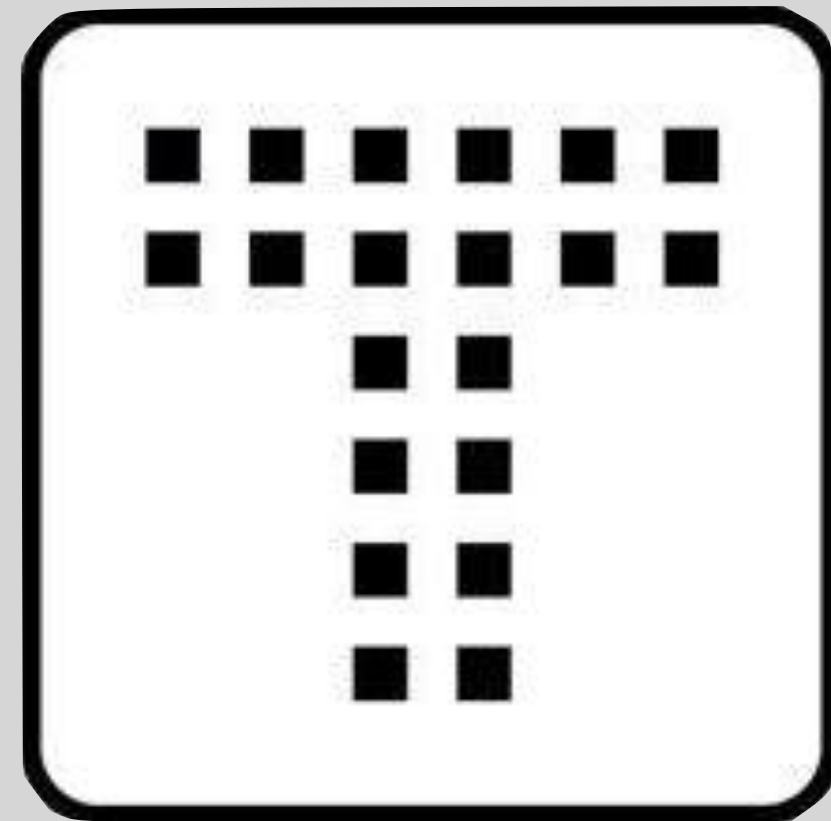


Refactoring



Continuous Integration

Acknowledgement



“91” Joey Chen