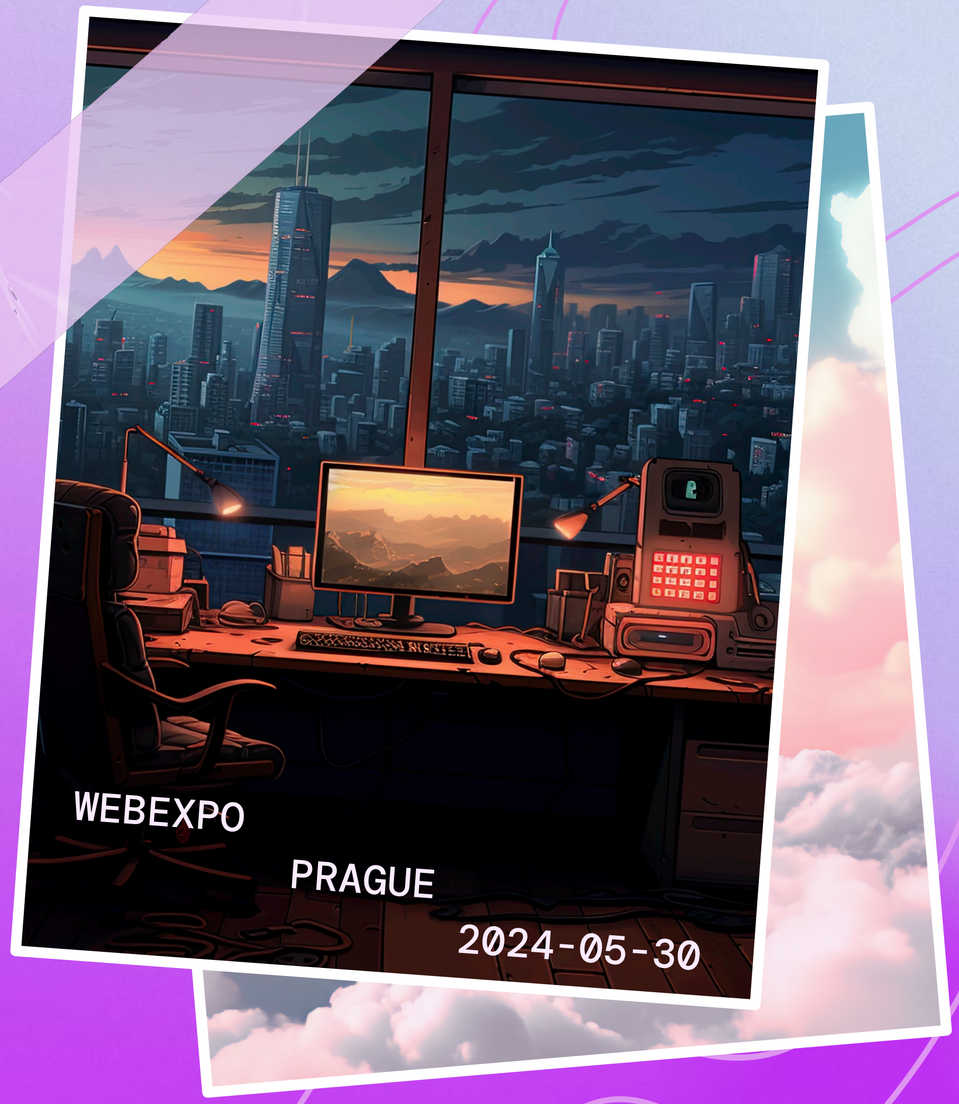
Robin
Pokorny

Evolutionary architecture

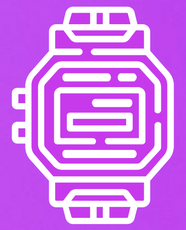
Agile meets software design



agile

(adjective)

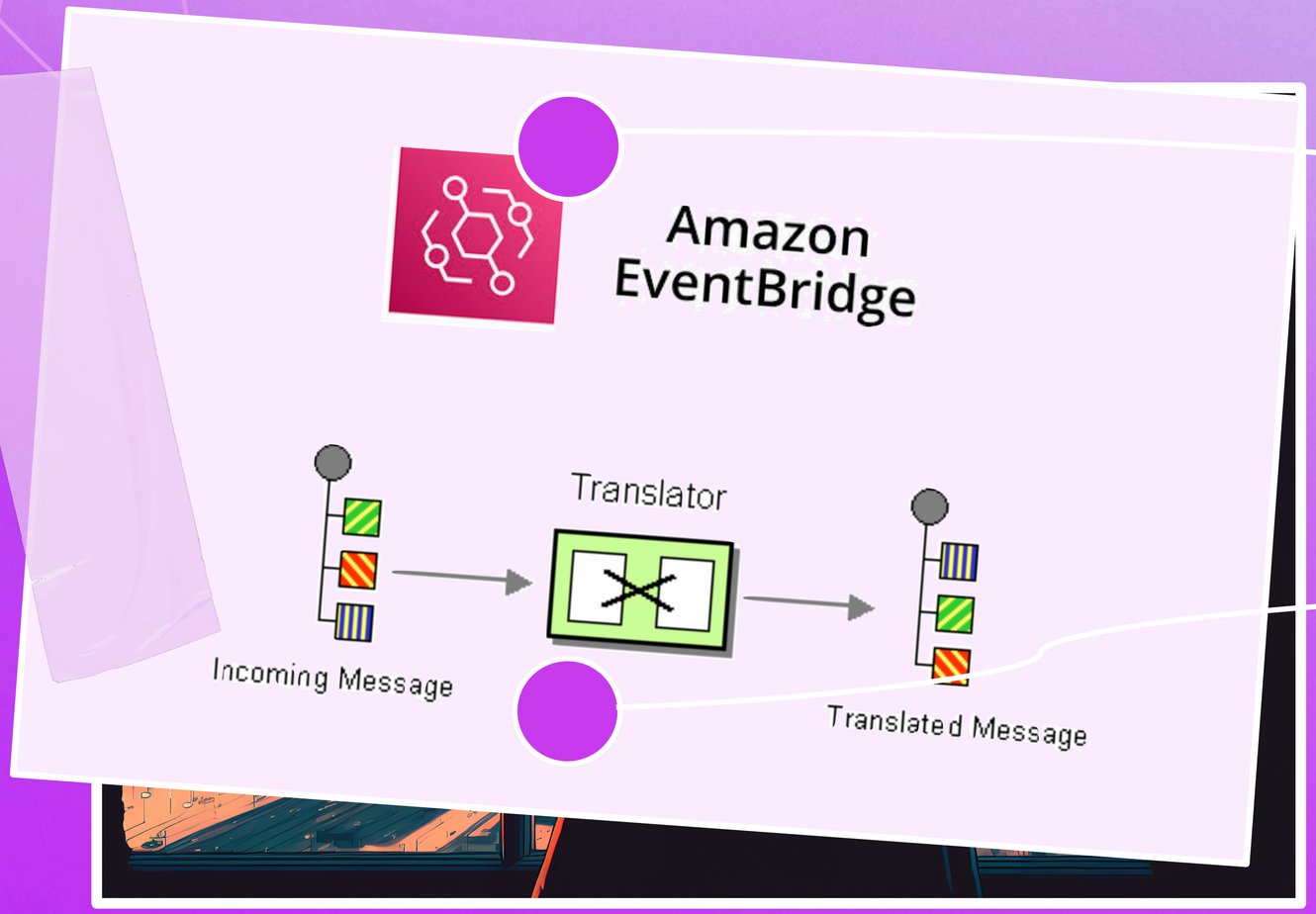
‘responding to change over following a plan’



architecture

(noun)

‘stuff that's hard to change’



Infrastructure

Cloud services,
tools, networking

Architecture

Intents, patterns,
and boundaries

Gregor Hohpe: enterpriseintegrationpatterns.com

Best architecture

Got it right the first time — Docker — You can’t predict technological shifts

Provides extensibility — Complexity — I’ll-need-it-later accumulates

Used everywhere — Insufficient — There is no one universal architecture

Potom lituje,
když už pozdě je
bycha honiti.

Potom běduje,
když už se nedá
čeho měniti.

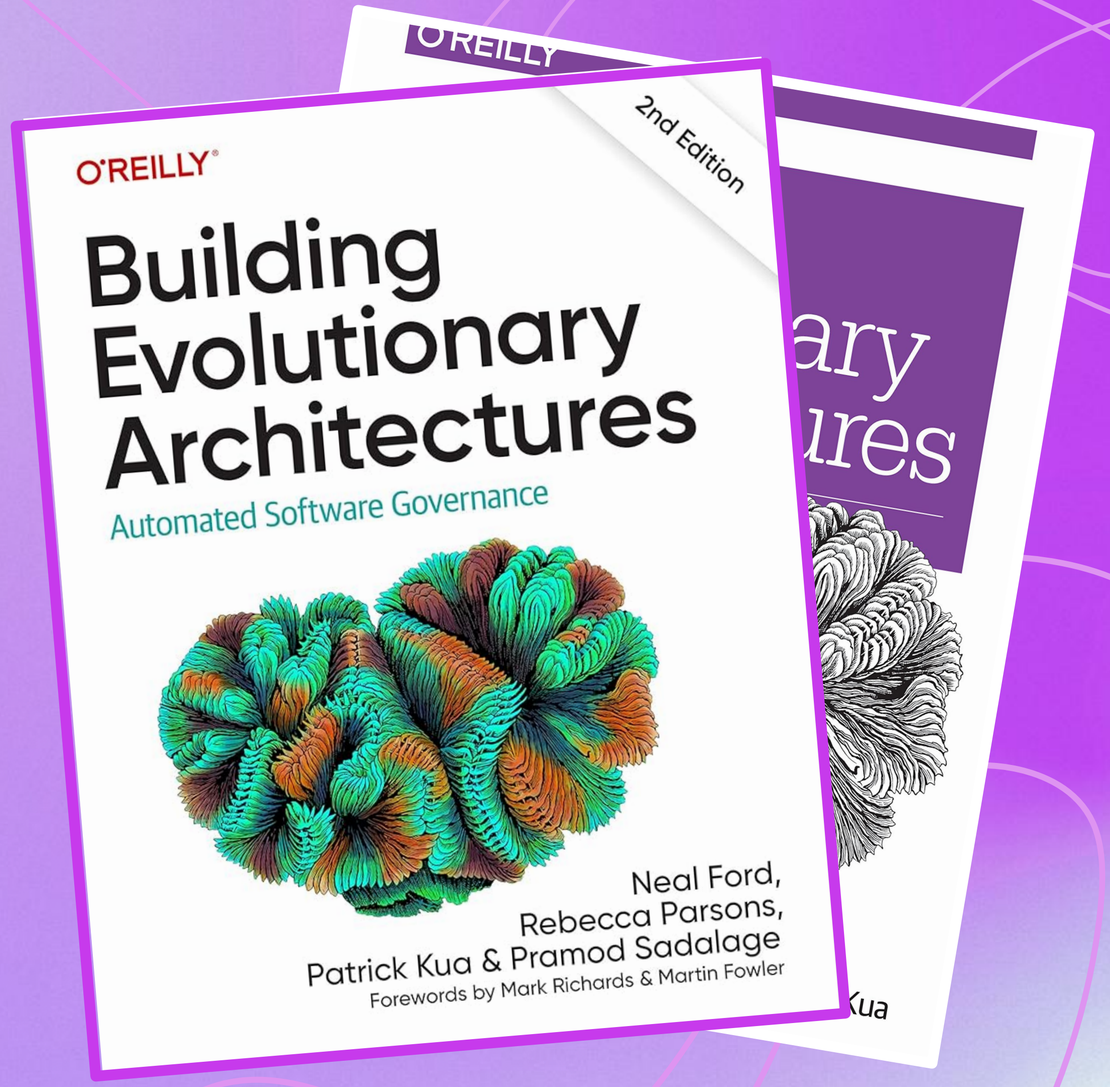
~ Kecal, dohazovač
Prodaná nevěsta

In the end,
he feels really sorry
but what’s done cannot be
undone again.

Then he starts lamenting
when it's much too late
to change anything.

~ Kecal, a marriage broker
The Bartered Bride

An evolutionary architecture supports **guided**, **incremental** change as a first principle across **multiple dimensions**



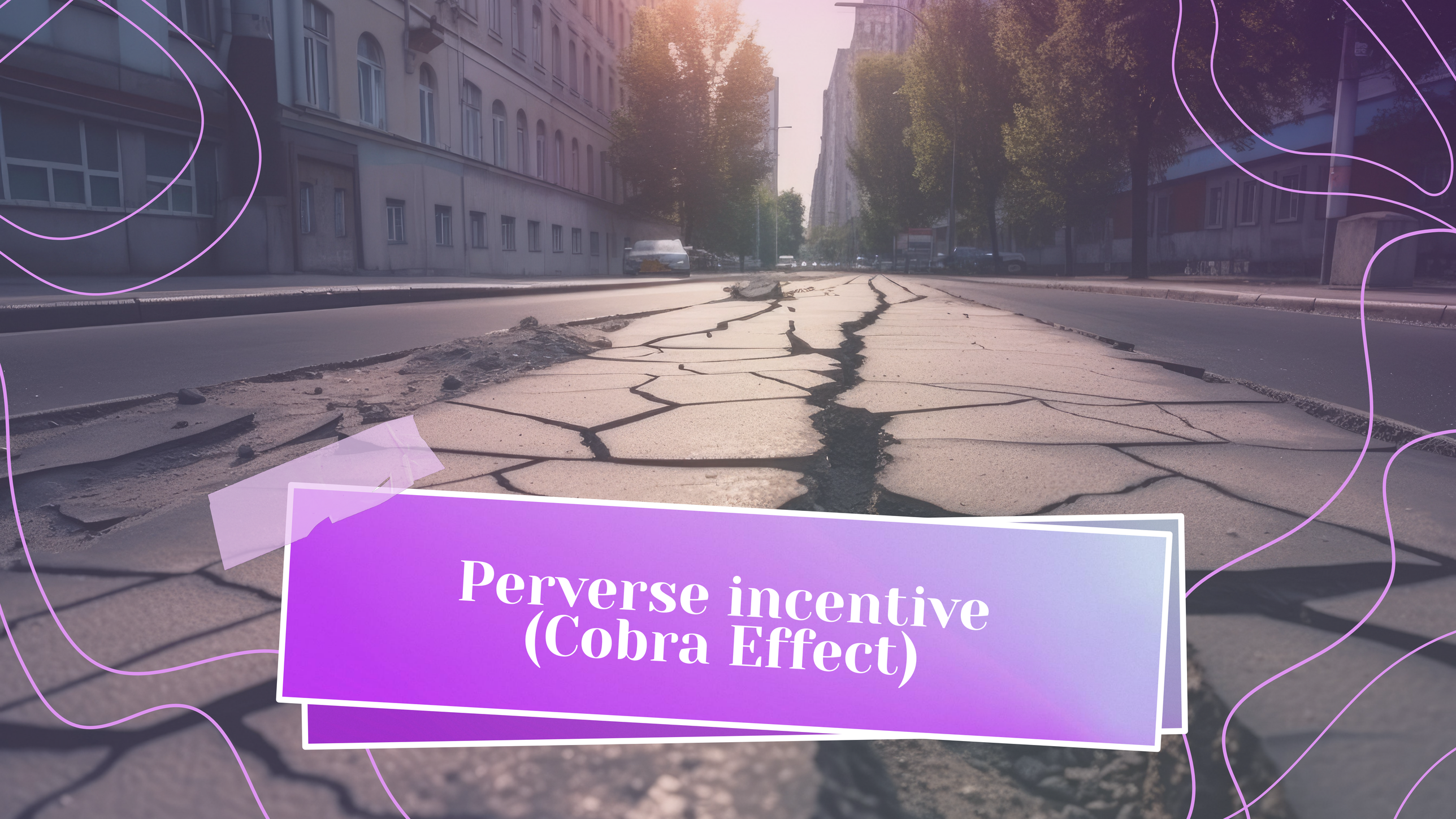
Fitness Function

an objective function used to summarize how close a prospective design solution is to achieving the set aims

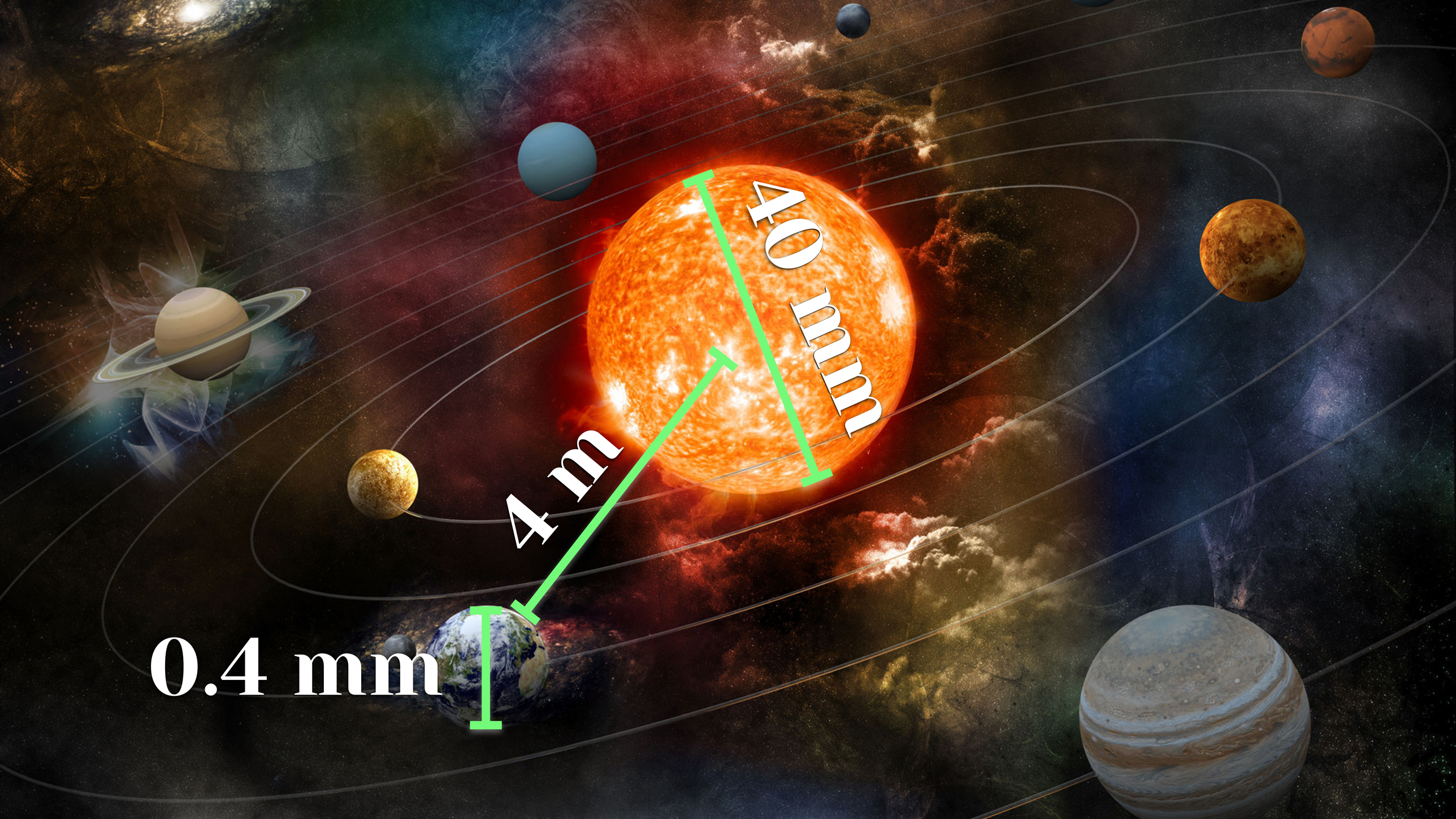
You might use some already!

1. test coverage above 80%
2. p99 response time is below 500 ms
3. below 1% of 500 responses in 5 min windows
4. bundle size is below 500 KiB
5. passes correlation IDs upstream
6. no SSPL tools or libraries

A **Fitness function** is an objective function used to summarize how close a prospective design solution is to achieving the set aims



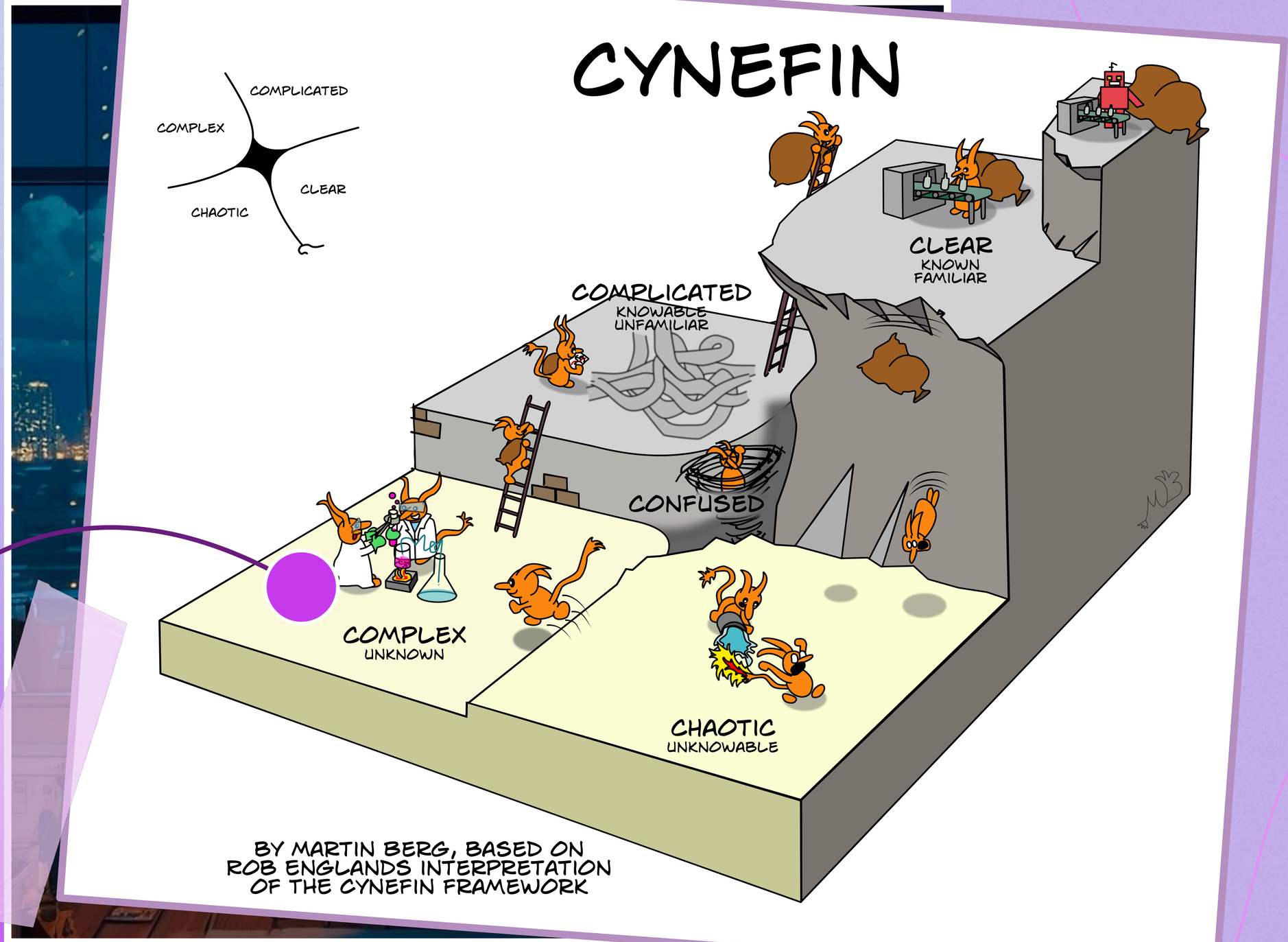
Perverse incentive (Cobra Effect)


40 mm
4 m
0.4 mm

All models are wrong, some are useful

George Box

Socio-technical Systems



Managing complexity (and chaos) in times of crisis

A field guide for decision makers inspired by the Cynefin framework

https://data.europa.eu/doi/10.2760/353

01

Guided

Incremental

Multiple dimensions

Architecture Characteristics Worksheet

performance
The amount of time it takes for the system to process a business request

responsiveness
The amount of time it takes to get a response to the user

availability
The amount of uptime of a system; usually measured in 9's (e.g., 99.9%)

fault tolerance
When fatal errors occur, other parts of the system continue to function

scalability
A function of system capacity and growth over time; as the number of users or requests increase in the system, responsiveness, performance, and error rates remain constant

elasticity
The system is able to expend and respond quickly to unexpected or anticipated extreme loads (e.g., going from 20 to 250,000 users instantly)

data integrity
The data across the system is correct and there is no data loss in the system

data consistency
The data across the system is in sync and consistent across databases and tables

adaptability
The ease in which a system can adapt to changes in environment and functionality

concurrency
The ability of the system to process simultaneous requests, in most cases in the same order in which they were received; implied when scalability and elasticity are supported

interoperability
The ability of the system to interface and interact with other systems to complete a business request

extensibility
The ease in which a system can be extended with additional features and functionality

deployability
The amount of ceremony involved with releasing the software, the frequency in which releases occur, and the overall risk of deployment

testability
The ease of and completeness of testing

abstraction
The level at which parts of the system are isolated from other parts of the system (both internal and external system interactions)

workflow
The ability of the system to manage complex workflows that require multiple parts (services) of the system to complete a business request

Created by Mark Richards, DeveloperToArchitect.com

Architecture Styles Worksheet

System/Project: _____

Architect/Team: _____ Date: _____

Selected Architecture(s):

	☐ layered	☐ modular monolith	☐ microkernel	☐ microservices	☐ service-based	☐ service-oriented	☐ event-driven	☐ space-based
agility	★	★★	★★★	★★★★★	★★★★	★	★★★	★★
abstraction	★	★	★★★	★	★	★★★★★	★★★★	★
configurability	★	★	★★★★	★★★	★★	★	★★	★★
cost	★★★★★	★★★★★	★★★★★	★	★★★★	★	★★★	★★
deployability	★	★★	★★★	★★★★★	★★★★	★	★★★	★★★
domain part.	★	★★★★★	★★★★★	★★★★★	★★★★★	★	★	★★★★★
elasticity	★	★	★	★★★★★	★★	★★★	★★★★	★★★★★
evolvability	★	★	★★★	★★★★★	★★★	★	★★★★★	★★★
fault-tolerance	★	★	★	★★★★★	★★★★	★★★	★★★★★	★★★
integration	★	★	★★★	★★★	★★	★★★★★	★★★	★★
interoperability	★	★	★★★	★★★	★★	★★★★★	★★★	★★
performance	★★★	★★★	★★★	★★	★★★	★★	★★★★★	★★★★★
scalability	★	★	★	★★★★★	★★★	★★★	★★★★★	★★★★★
simplicity	★★★★★	★★★★★	★★★★	★	★★★	★	★	★
testability	★★	★★	★★★	★★★★★	★★★★	★	★★	★
workflow	★	★	★★	★	★	★★★★★	★★★★★	★

SOFTWARE DESIGN BERLIN MEETUP

Wed, Jun 12

18:30

Thoughtworks
Event Space
Revaler Str. 31
10245 Berlin

Architectural Kata #3

Sponsored by

/thoughtworks

02

Guided

Incremental

Multiple dimensions

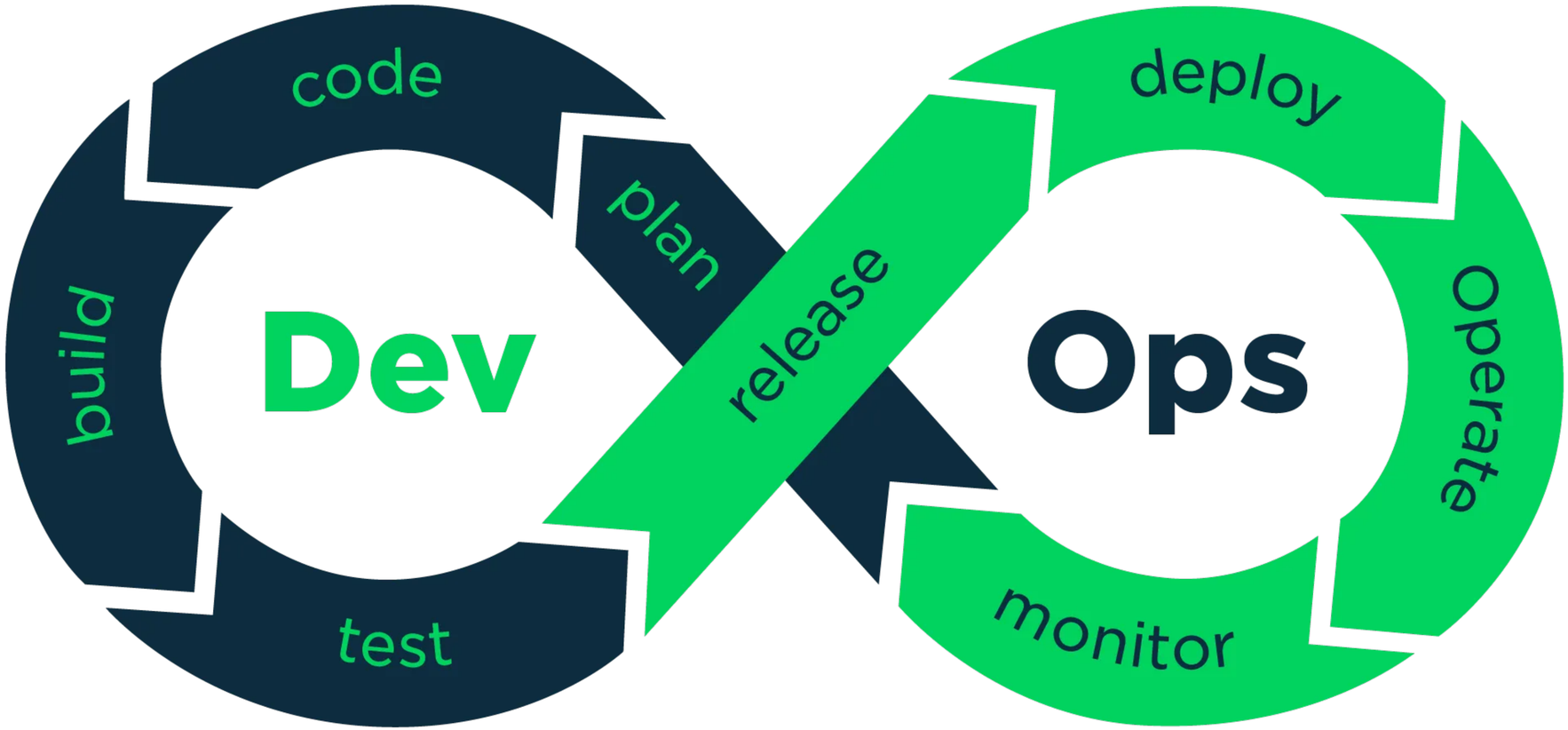
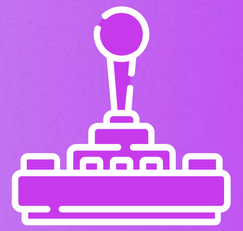


Image by Irma Harlann: https://neonrocket.medium.com/devops-is-a-culture-not-a-role-be1bed149b0



Reinforcement learning on Pokemon Red

github.com/PWhiddy/PokemonRedExperiments

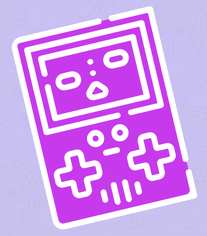
03

Guided

Incremental

Multiple dimensions

Multiple dimensions



Tech

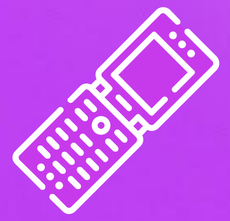
languages
frameworks
libraries

Data

DB schemata
optimization

Security

policies
guidelines



Ops

servers
cloud resources

Fitness Function

Cyclomatic Complexity

Limit the number of linearly independent paths

Coupling and Cohesion

Low dependencies between cohesive modules

Performance

Response time, throughput, or memory usage against thresholds

Chaos Monkey

Randomly terminate instances in production

Directionality of imports

Ensures correct flow among modules, no oriented circles

Compliance

Adherence to legal and regulatory requirements

.eslintrc.json

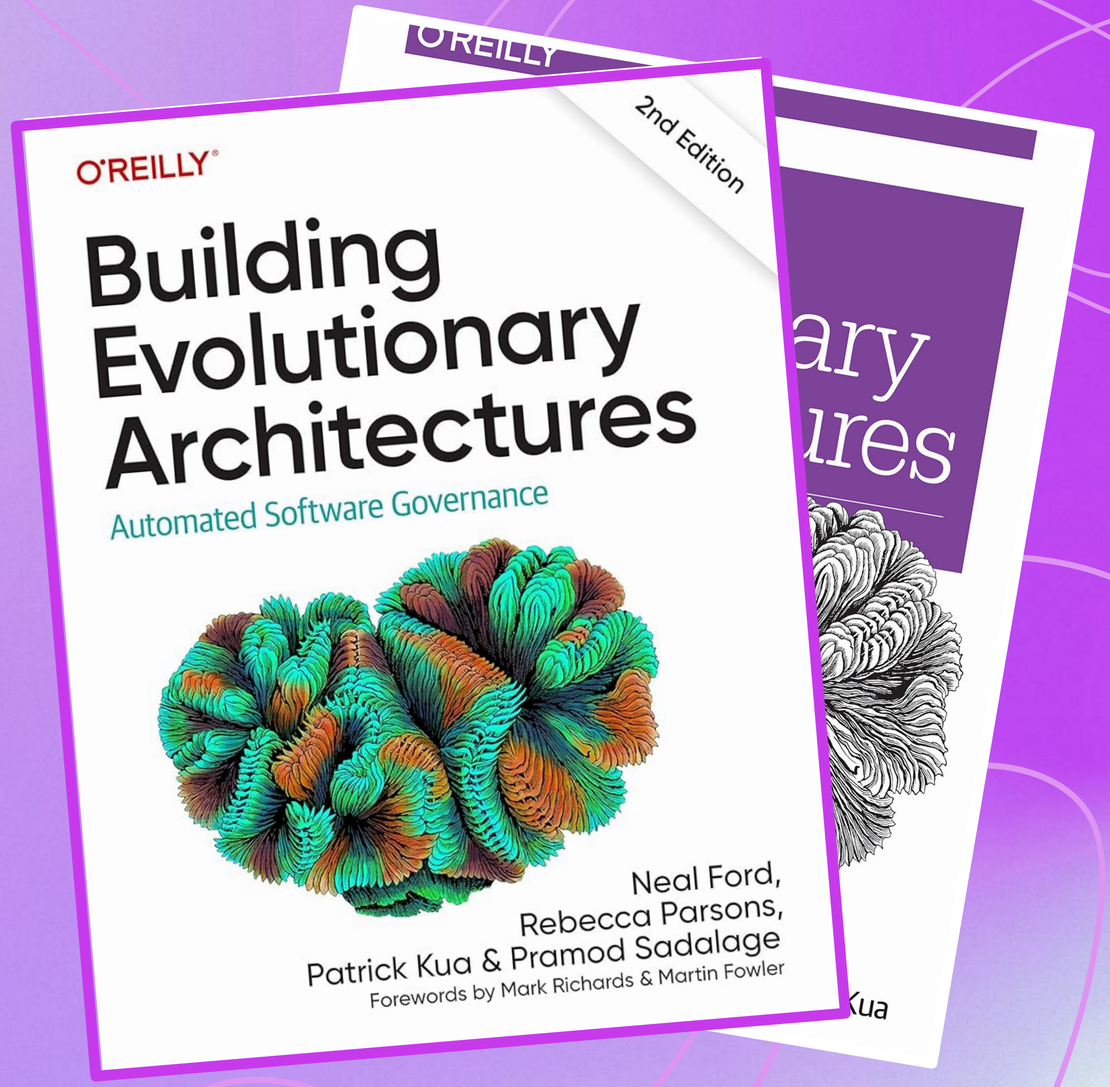
```
{
  "plugins": ["@nx"],
  "@nx/enforce-module-boundaries": [
    "error",
    {
      "allow": [],
      "depConstraints": [
        {
          "sourceTag": "controllers",
          "onlyDependOnLibsWithTags": ["controllers", "services"]
        },
        {
          "sourceTag": "services",
          "onlyDependOnLibsWithTags": ["services", "persistence"]
        },
        {
          "sourceTag": "persistence",
          "onlyDependOnLibsWithTags": ["persistence"]
        }
      ]
    }
  ]
}
```

LayeredArchitectureTest.java

```java
@AnalyzeClasses(packages = "com.tngtech.archunit.example.layers")
public class LayeredArchitectureTest {
  @ArchTest
  static final ArchRule layer_dependencies_are_respected =
    layeredArchitecture().consideringAllDependencies()
      .layer("Controllers")
        .definedBy("com.tngtech.archunit.example.layers.controller..")
      .layer("Services")
        .definedBy("com.tngtech.archunit.example.layers.service..")
      .layer("Persistence")
        .definedBy("com.tngtech.archunit.example.layers.persistence..")

      .whereLayer("Controllers")
        .mayNotBeAccessedByAnyLayer()
      .whereLayer("Services")
        .mayOnlyBeAccessedByLayers("Controllers")
      .whereLayer("Persistence")
        .mayOnlyBeAccessedByLayers("Services");
}
```

An evolutionary architecture supports **guided**, **incremental** change as a first principle across **multiple dimensions**



Move from solutions and implementations to outcomes

Outcomes

Define what characteristics are important for you

Fitness functions

Write a test to validate those outcomes

Forget

Use your brain power for something else

Outcomes

Fitness functions

Forget

robinpokorny.com
me@robinpokorny.com
@robinpokorny

Fitness Functions