SYZYGY

SYZYGY

Hexagonal Architecture

SYZYGY WARSAW IT SPEAK-UP

IDEA

CHALLENGE

Architecture

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction

Help
About Wikipedia
Community portal

Not lo

Article | Talk | Read | Edit | View

Software architecture

From Wikipedia, the free encyclopedia

Software architecture refers to the high level structures of a software system, the discipline of creating such structures, and the documentation of these structures. These structures are needed to reason about a software system. Each structure comprises software elements, relations among them, and properties of both elements and relations.[1] The *architecture* of a software system is a metaphor, analogous to the architecture of a building.[2]

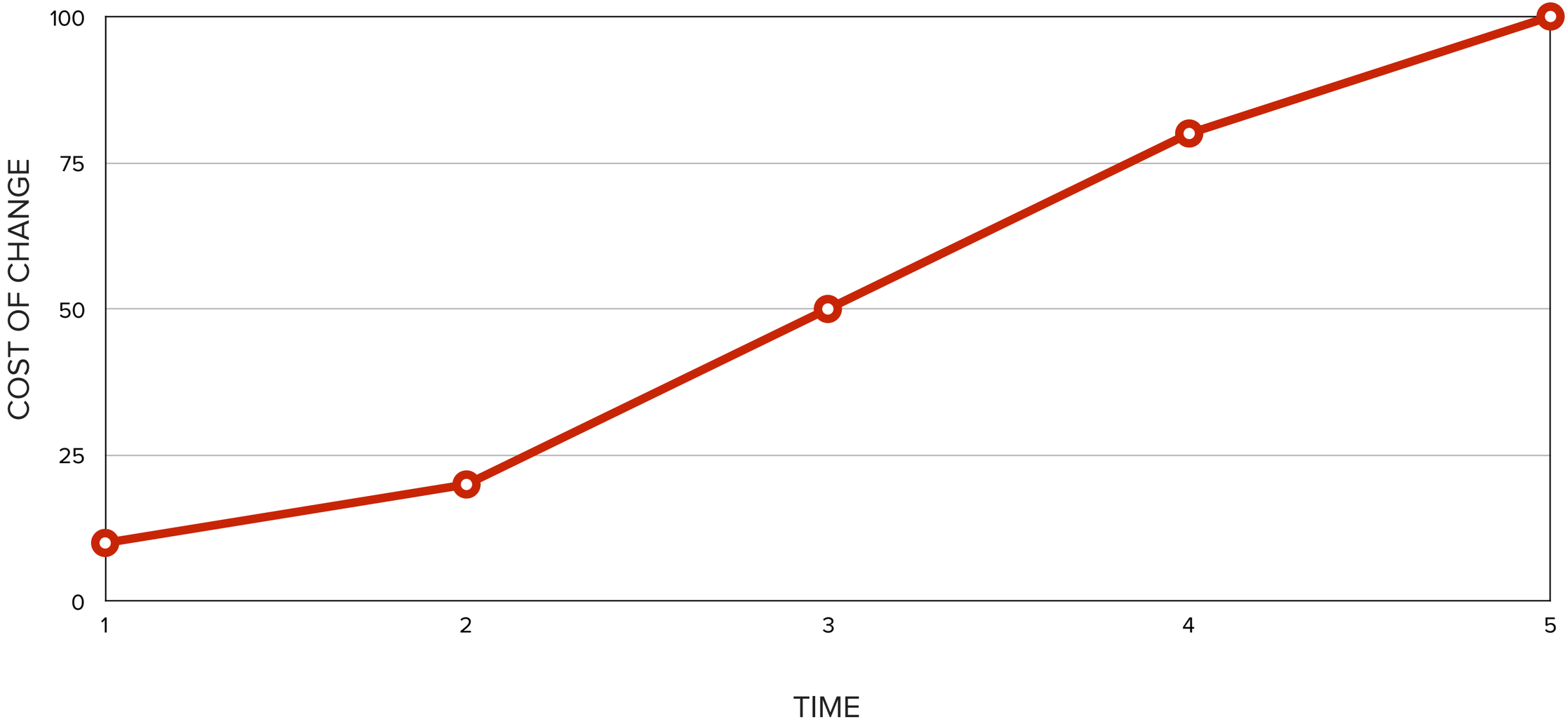
Software architecture is about making fundamental structural choices which are costly to change once implemented. Software architecture choices include specific structural options from possibilities in the design of software. For example, the systems that controlled the space shuttle launch vehicle had the requirement of

WIKIPEDIA DEFINITION HERE

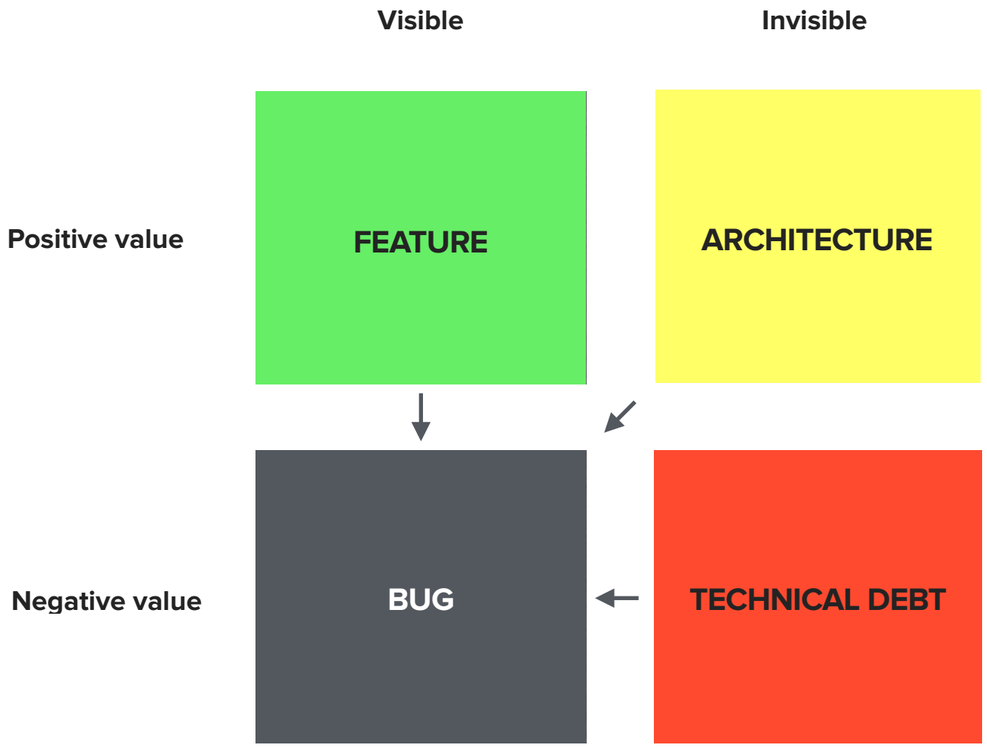
SYZYGY

Why do we even talk about architecture?

COST OF CHANGE GROW MUCH FASTER THAN YOUR CODE BASE



Have you ever seen a bad architecture?



- Code changes become more difficult
- New developers need a very long time to become productive
- Changes in one place result in breakage in an unrelated part of the codebase
- Team agility and responsiveness decreases
- Deadlines are blown
- Budgets are blown

Definition of good architecture

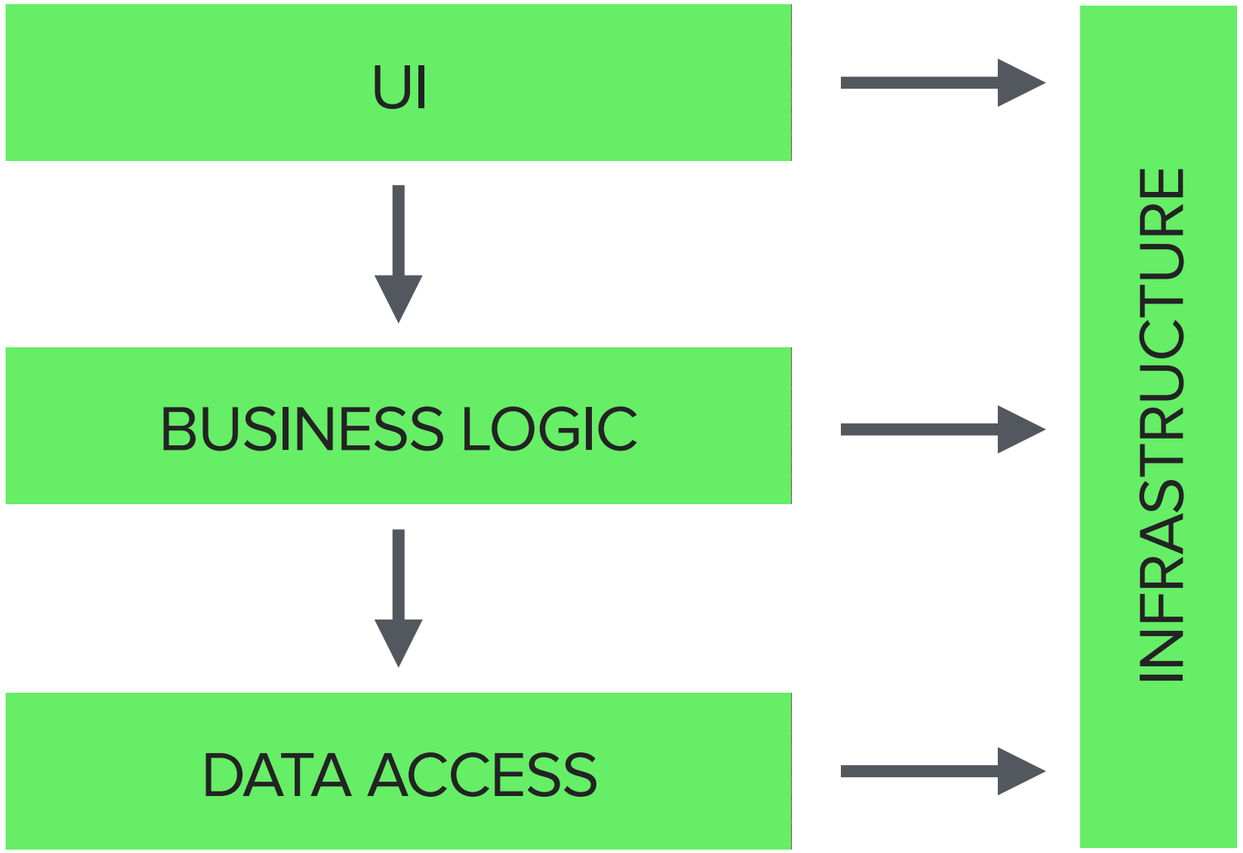
GOOD ARCHITECTURE

High Maintainability

Low Technical Debt

Traditional Layered Architecture

TRADITIONAL LAYERED ARCHITECTURE

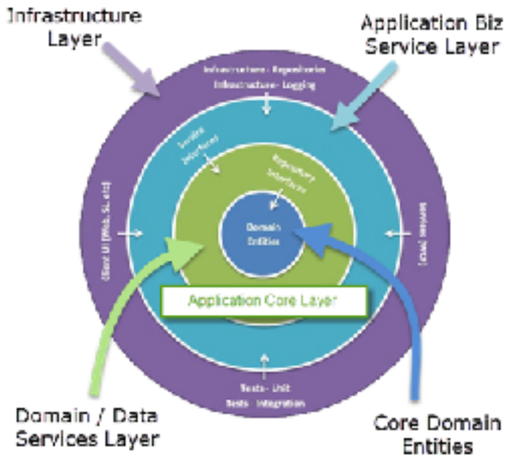


- UI is coupled to data access
- UI can't function if business logic isn't there
- Business logic can't function if data access isn't there
- Each layer is often coupled to various infrastructure concerns *Utils”

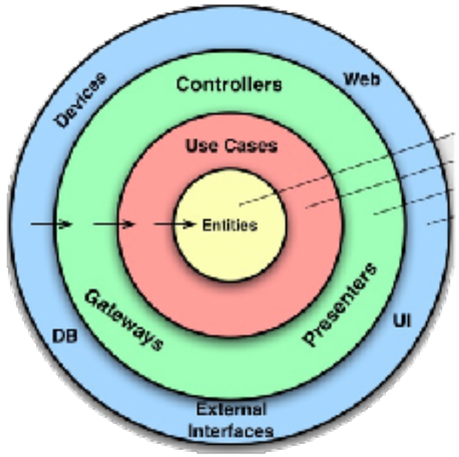
What could be better than that?

ARCHITECTURE

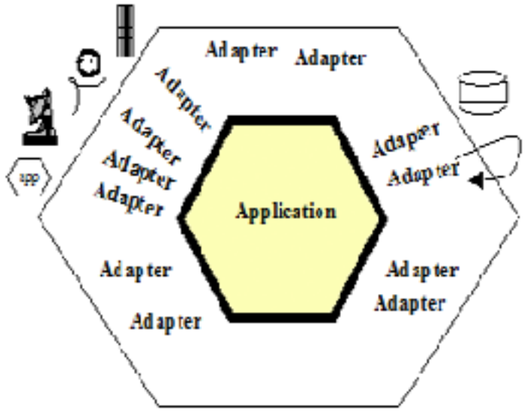
ONION



CLEAN



PORTS & ADAPTERS

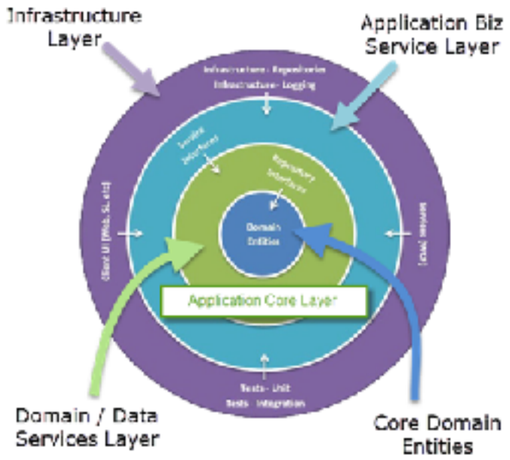


HEXAGONAL

Hexagonal Architecture was presented in 2005 by Alistair Cockburn as a solution to problems with e.g. traditional layering, coupling and entanglement.

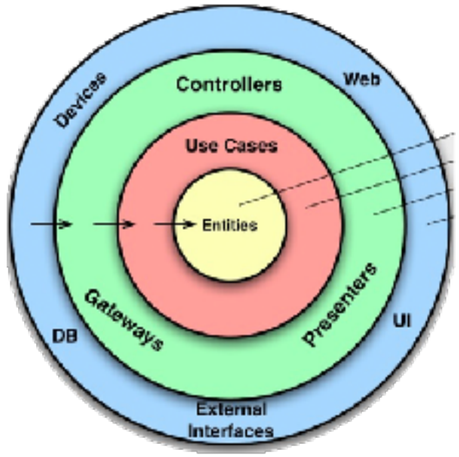
ARCHITECTURE

ONION



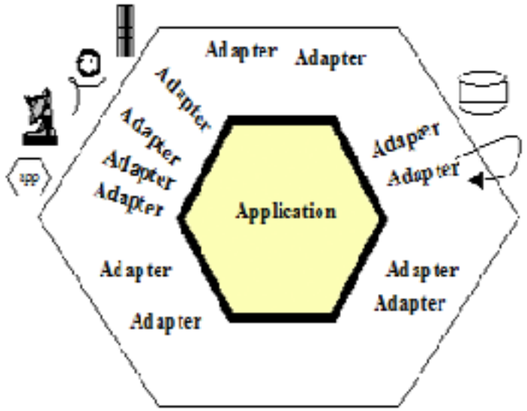
Jeffrey Palermo

CLEAN



Robert Martin

PORTS & ADAPTERS



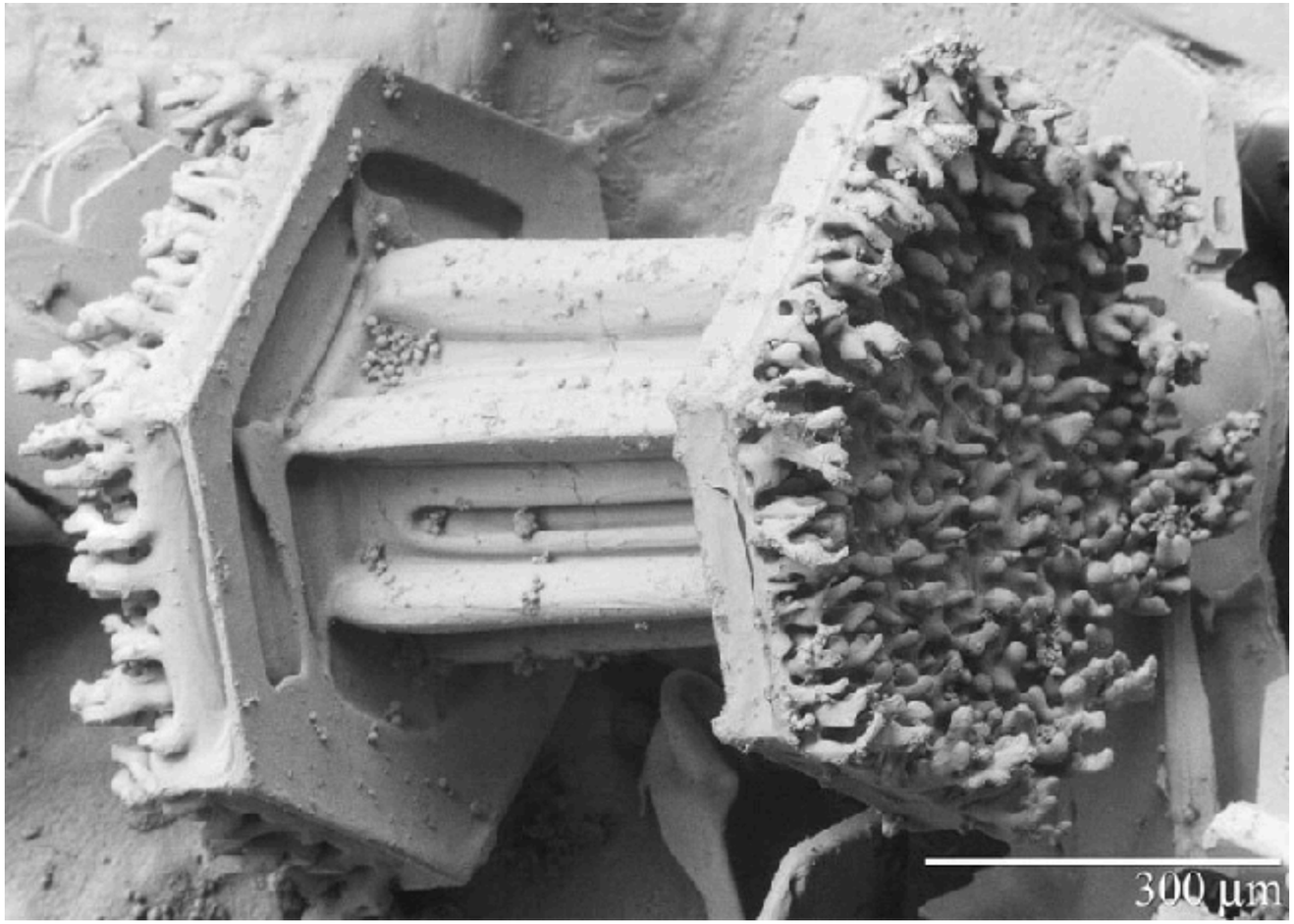
Alistair Cockburn

HEXAGONAL

The others

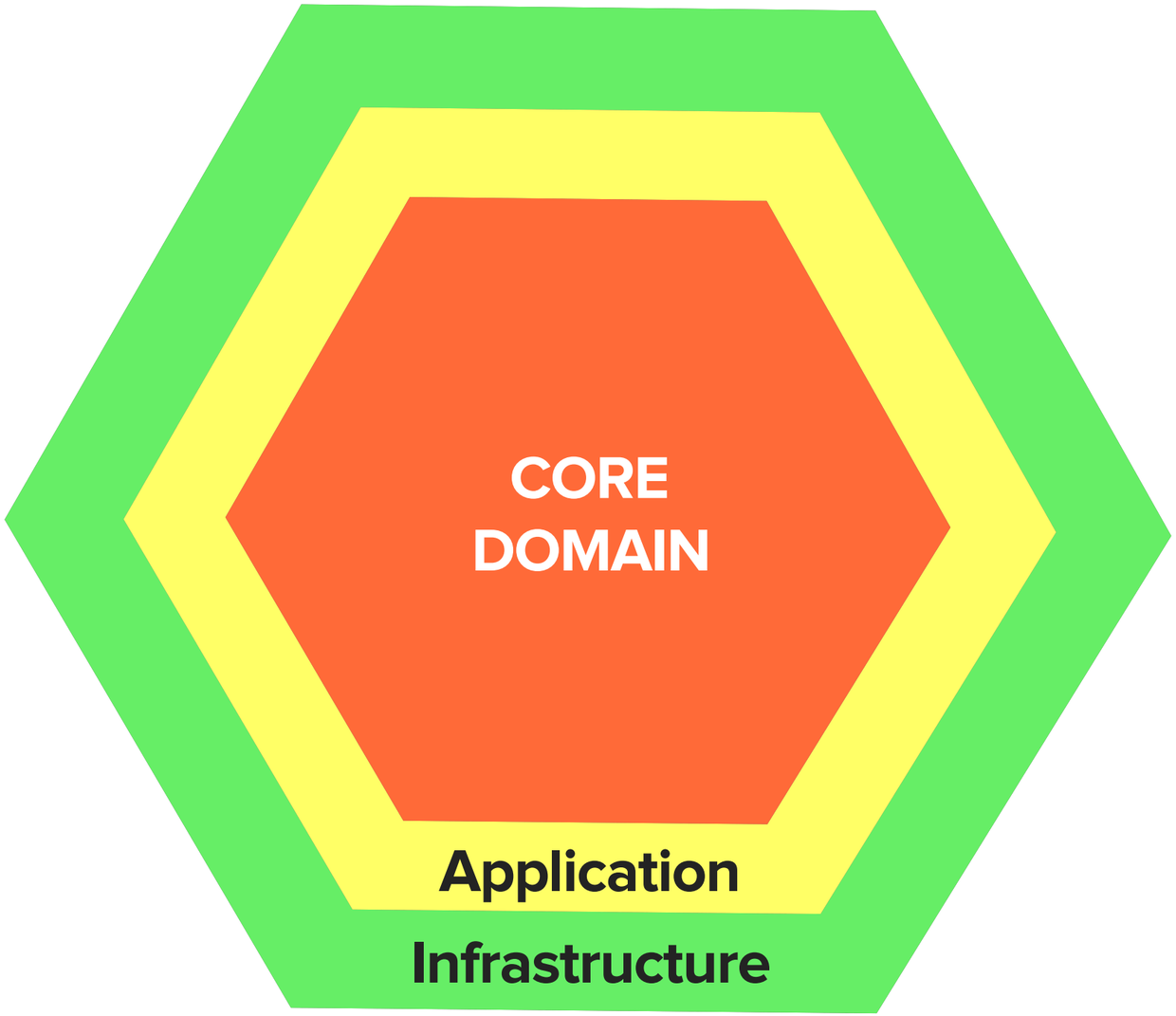
Hexagonal Architecture was presented in 2005 by Alistair Cockburn as a solution to problems with e.g. traditional layering, coupling and entanglement.

NATURAL CURIOSITY



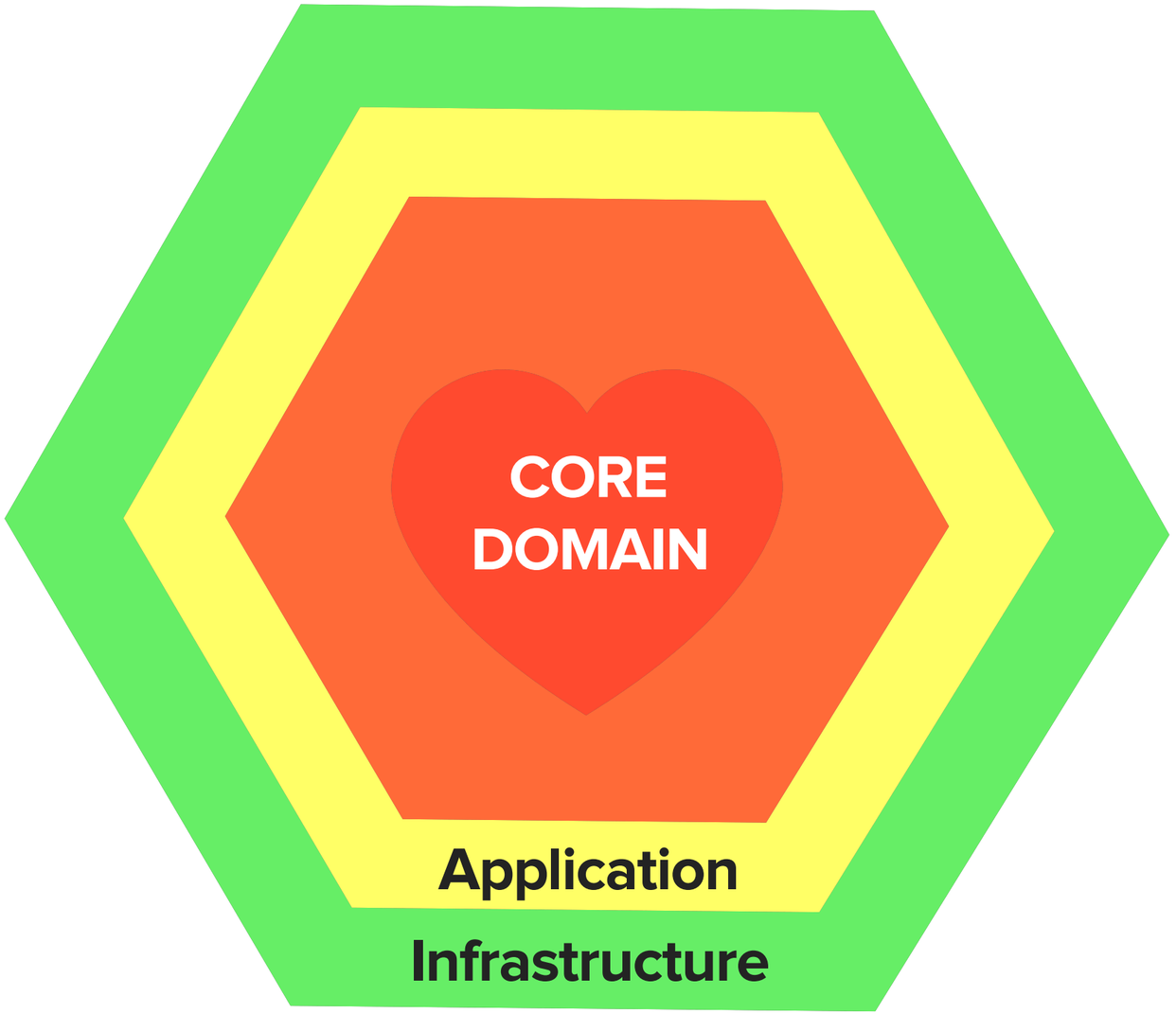
HEXAGONAL ARCHITECTURE

Hexagonal Architecture defines conceptual layers of code responsibility, and then points out ways to decouple code between those layers.



HEXAGONAL ARCHITECTURE

Hexagonal Architecture defines conceptual layers of code responsibility, and then points out ways to decouple code between those layers.



Ports & Adapters

PORTS & ADAPTERS

USB Type-A

USB Type-C

Infrastructure
Application
Domain
Action, Form,
Input Validation
Repository
Command
Command Handler
Query
Factory
Value Object
Entitiy
Event
RepositoryInterface
Service

Dependency Inversion Principle

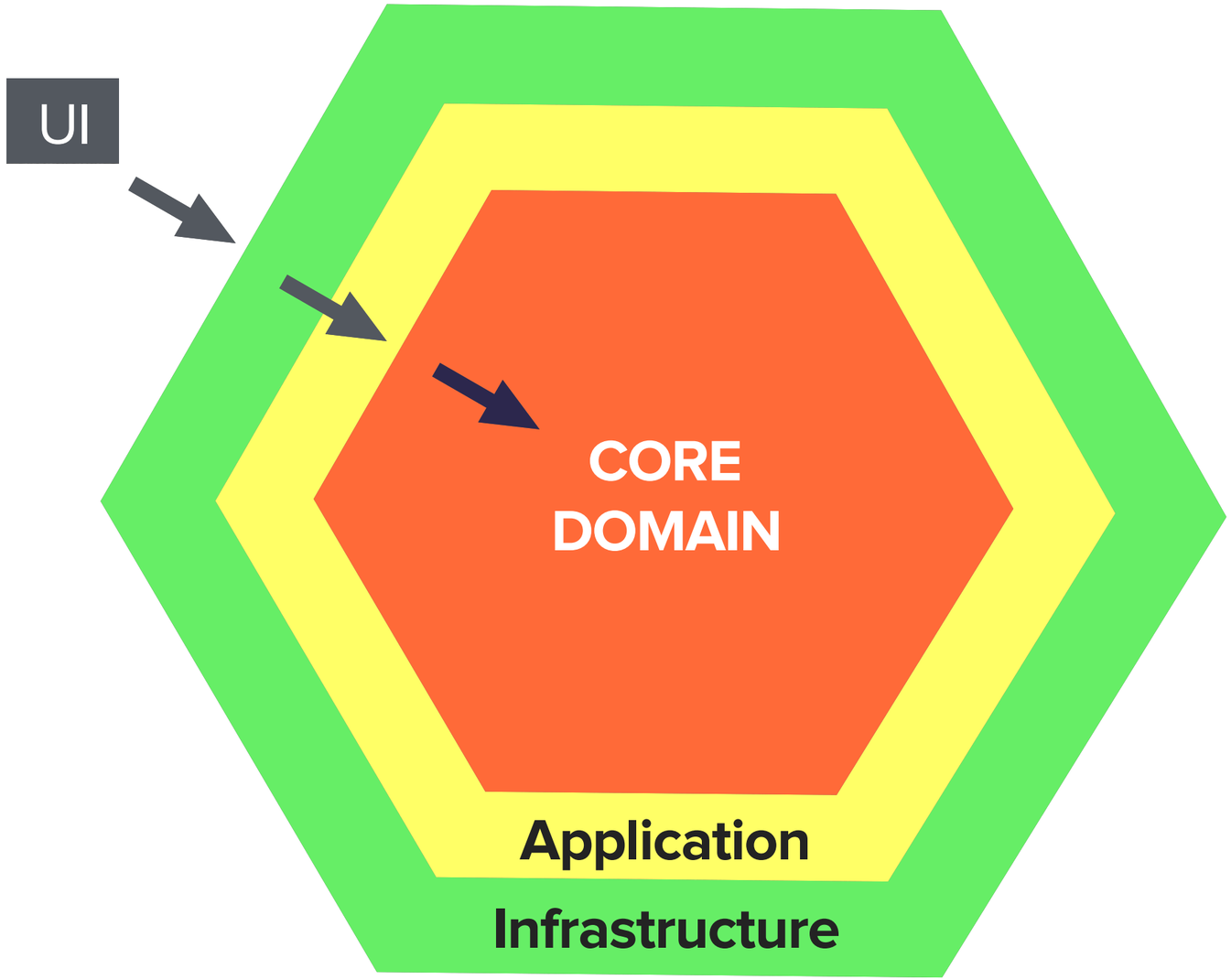
High-level modules should not depend on low-level modules.
Both should depend on abstractions.

Abstractions should not depend on details.
Details should depend on abstractions.

DEPENDENCIES

Source code dependencies can only point inwards.

Nothing in an inner layer can know anything at all about something in an outer layer



WEB PORT
Action, Form,
Input Validation
COMMAND BUS
Command
Command Handler
Factory
Value Object
Entitiy
Infrastructure
Application
Domain
PERSISTENCE PORT
Repository
REPOSITORY INTERFACE
Query
Event
RepositoryInterface
Service

Implementation example

Code’s what Tiggers like the best.

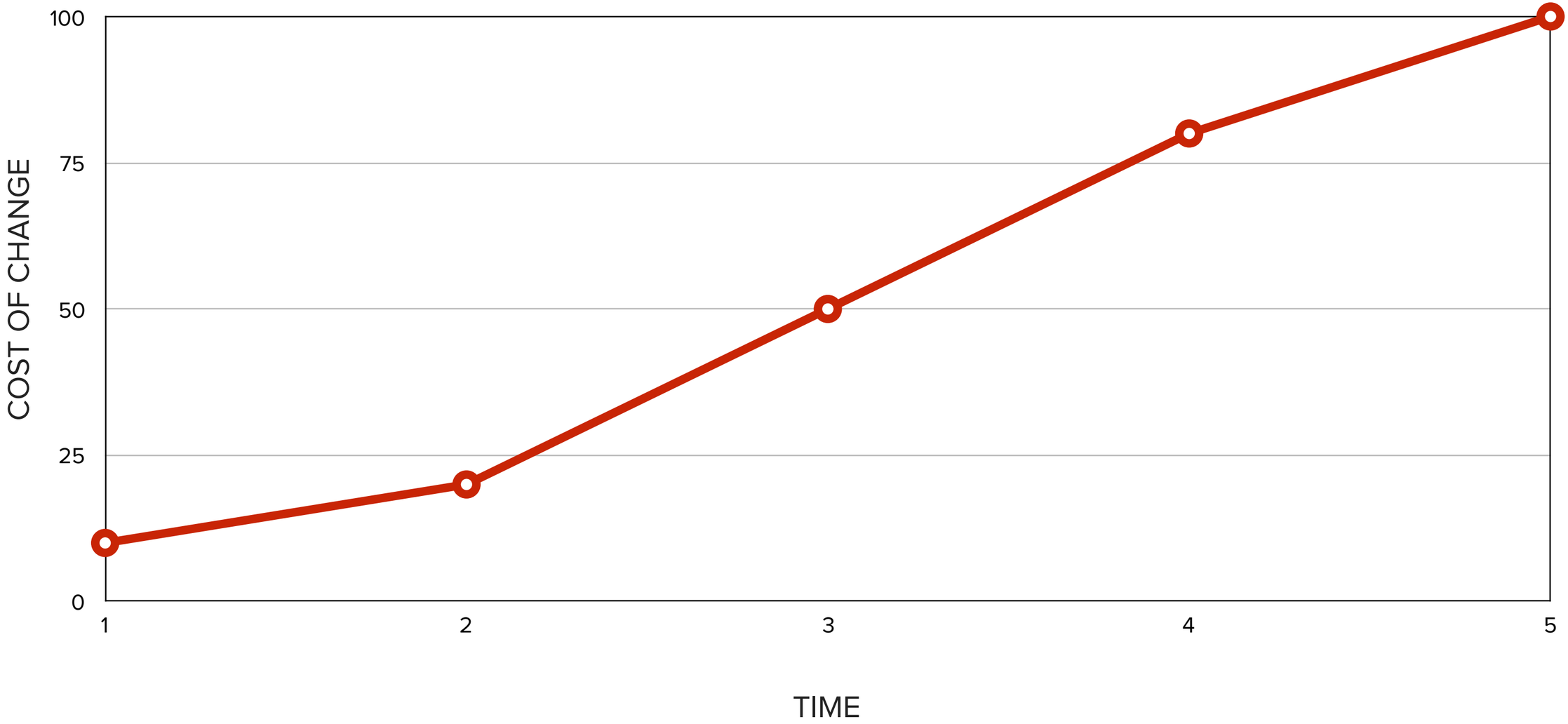
LET’S RECAP

1. Business Logic is the most important (DDD)
2. One direction dependencies
3. Extremely fast testing
4. Loos coupling
5. High cohesion - well defined job of class
6. Database is only implementation detail

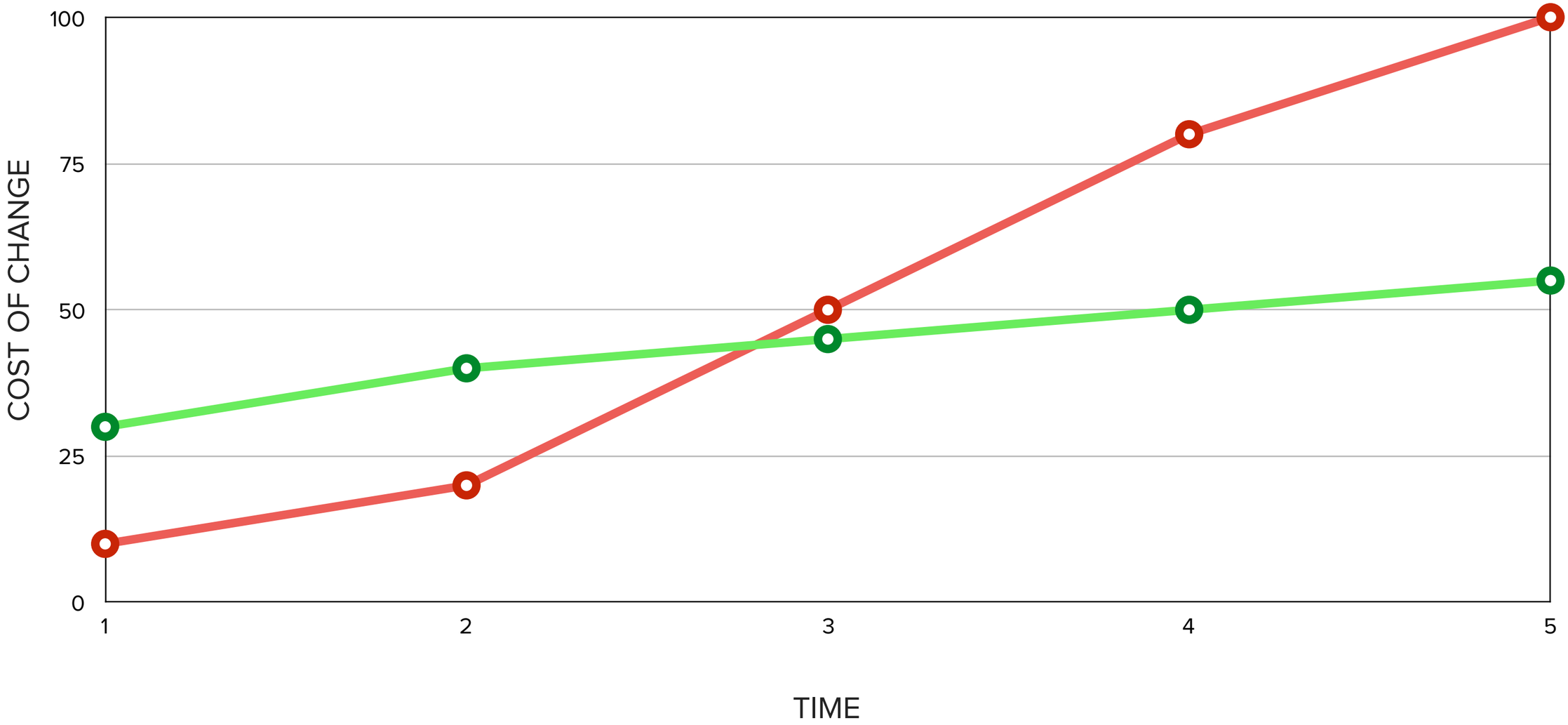
Why?

We want our applications to be easy to work with. We want to make future changes easy.

COST OF CHANGE GROW MUCH FASTER THAN YOUR CODE BASE



I HAVE A DREAM...



SIZIGI

Time of confession

SYZYGY

Time for discussion

What's existing state of things?
What's wrong with our architecture?
What can we do better?