

Microservices zur Architekturmodernisierung

Michael Vitz
Alexander Heusingfeld

innoQ





Alexander Heusingfeld

Senior Consultant @ innoQ

alexander.heusingfeld@innoq.com

@goldstift

Michael Vitz

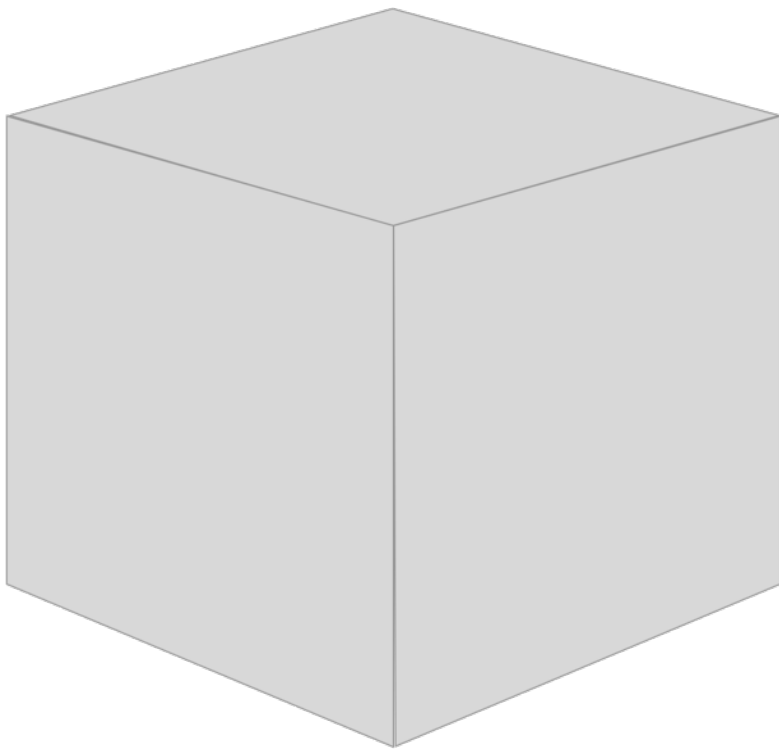
Consultant @ innoQ

michael.vitz@innoq.com

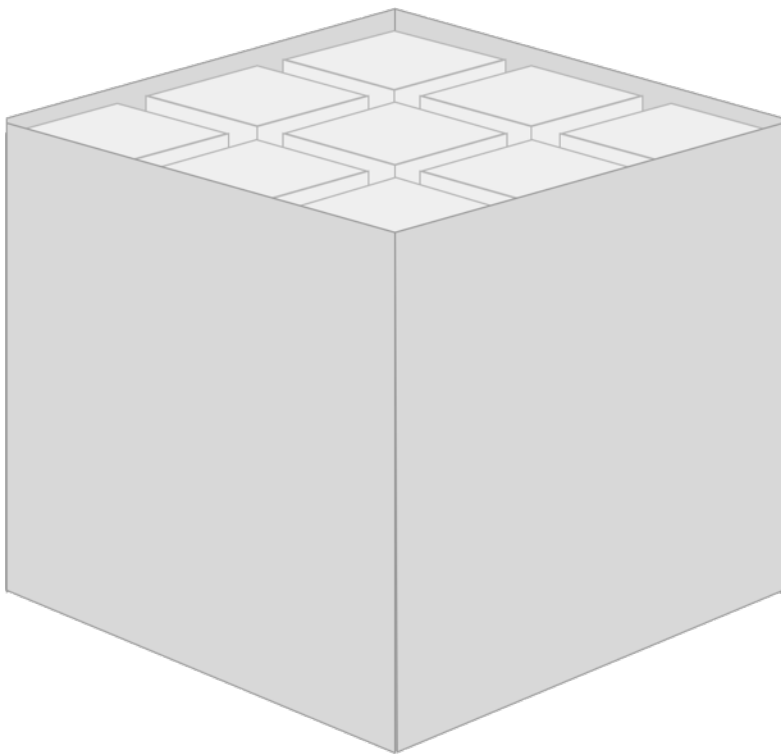
@michaelvitz



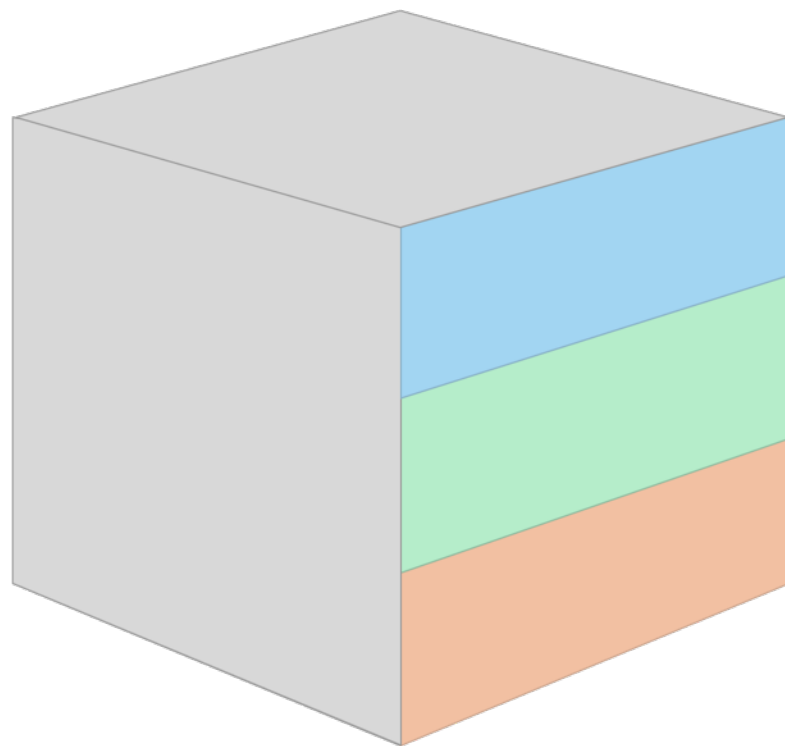
Typical Scenario?!



A monolith contains
numerous things inside
of a single system ...



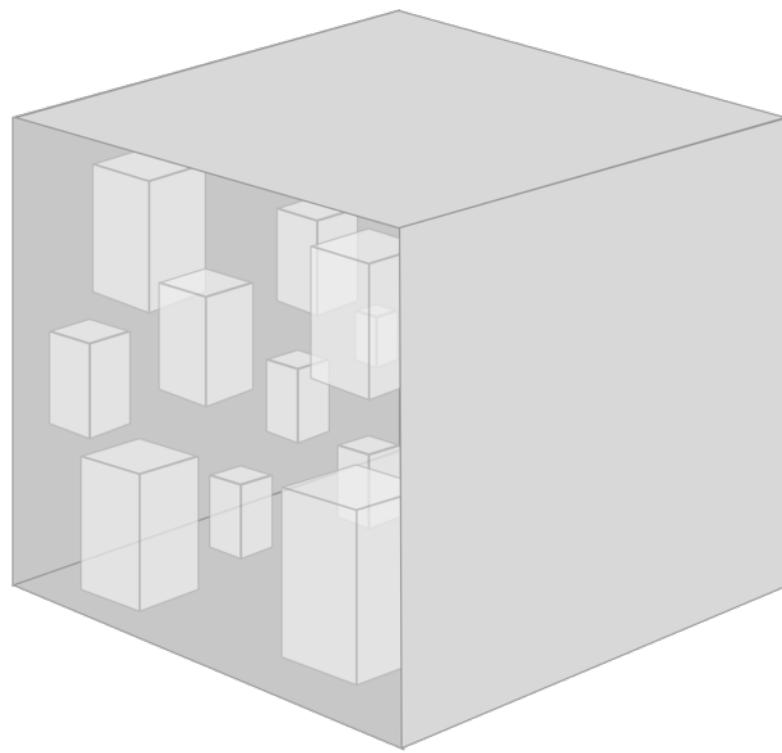
Various Domains



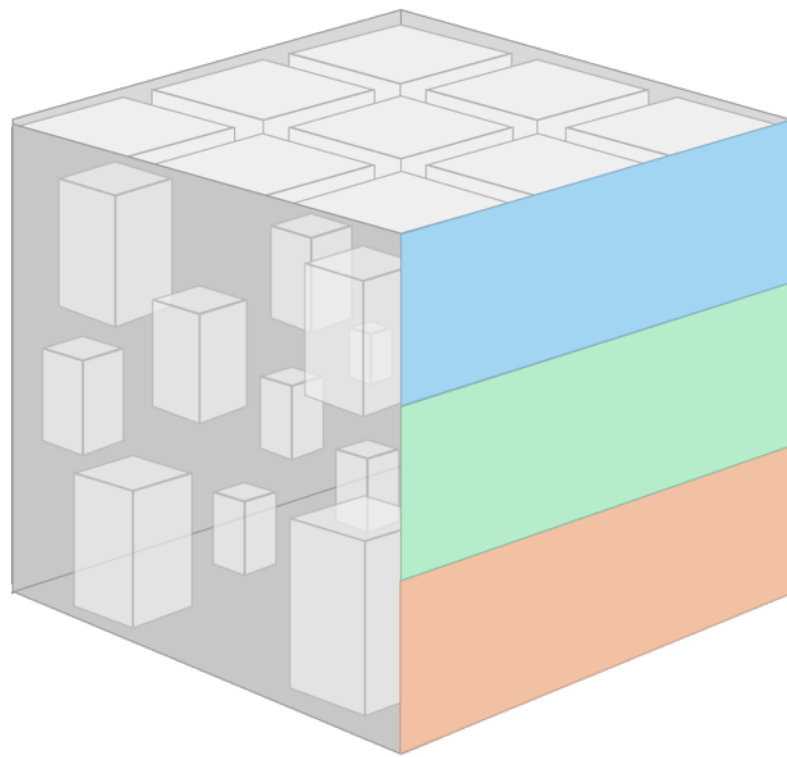
User interface

Business logic

Persistence



... as well as **a lot** of
modules, components,
frameworks and libraries.



With all these layers
in one place, a
monolith tends to
grow.

Goal

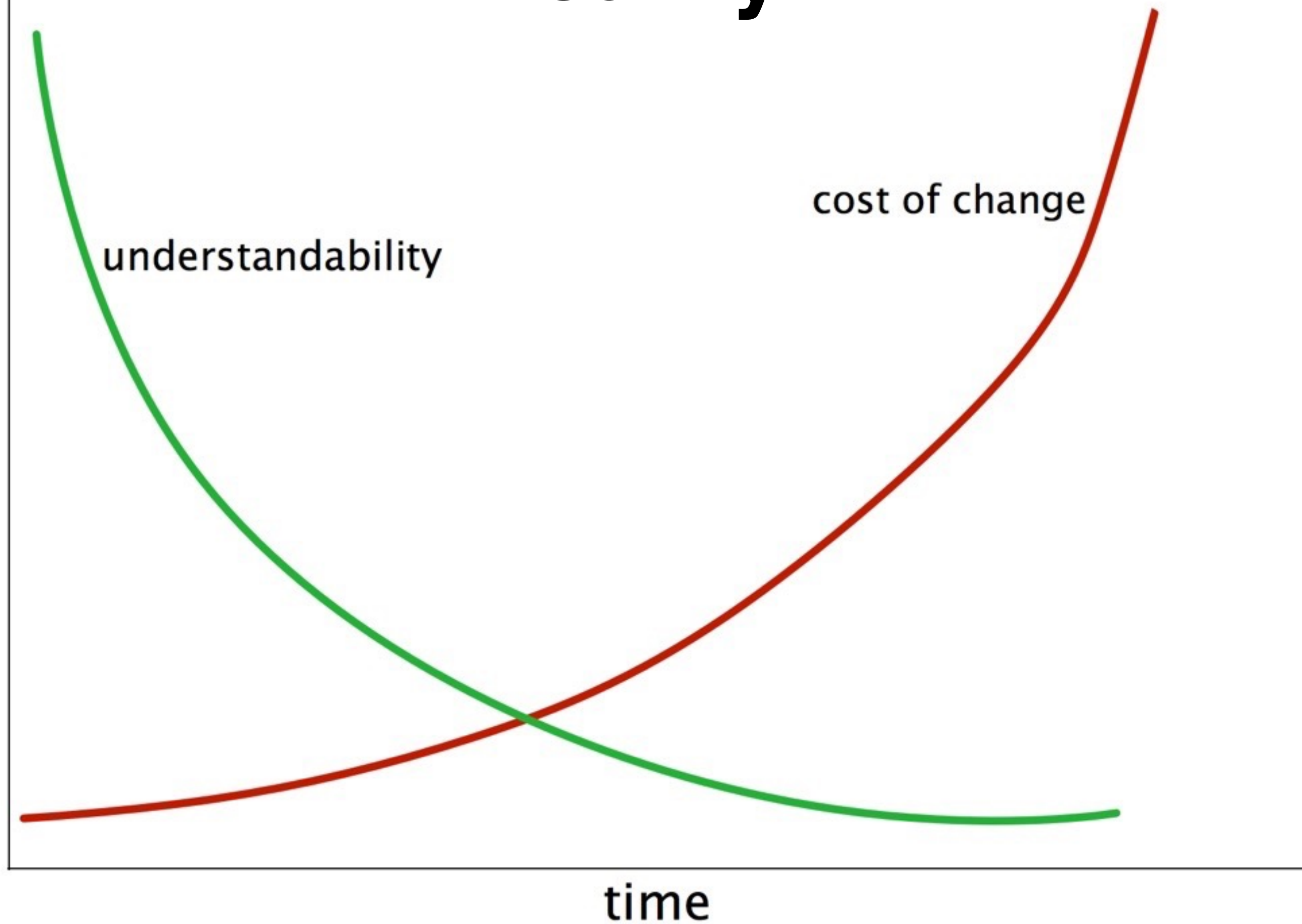
understandability

cost of change

time



Reality



Why?

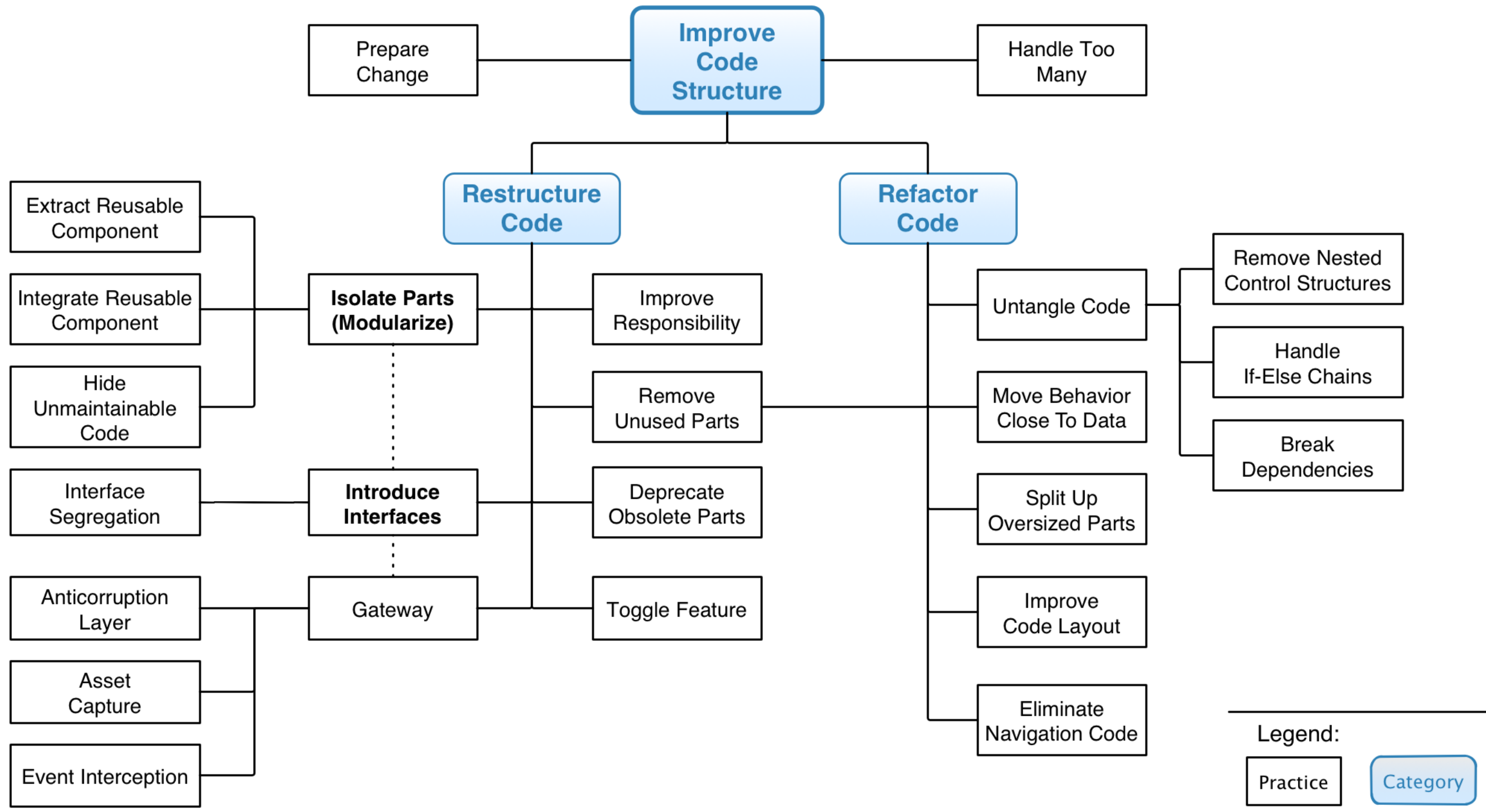
Company X App - Module dependencies

The following is a graph visualizing the dependencies between the OSGi modules in Company X Application, defined via Spring Dynamic Modules XML files.



Typical Reaction?

Code Improvements



Alternatives?

Focus on ~~Technology~~
Business Value

Thesis:

Improvement
of Systems
is more than Refactoring
of single classes



Architecture Improvement Method



- architecture
- code
- runtime
- organization



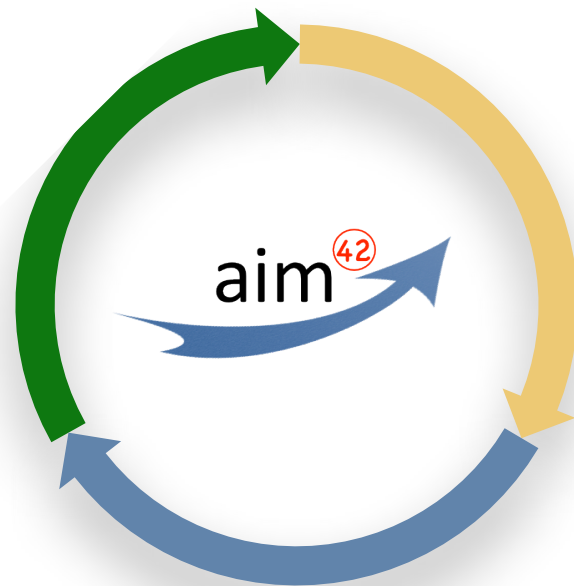
determine „value“ of
problems / risks /
issues and their
improvements





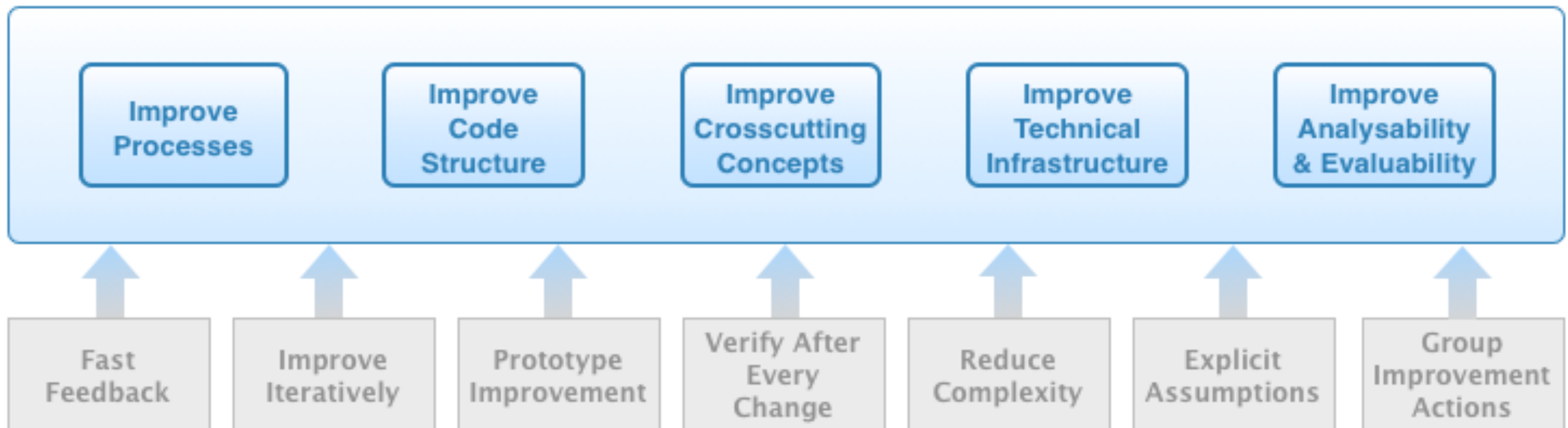
improve

- define improvement strategy
- refactor
- re-architect
- re-organize
- remove debt



improve

Fundamentals

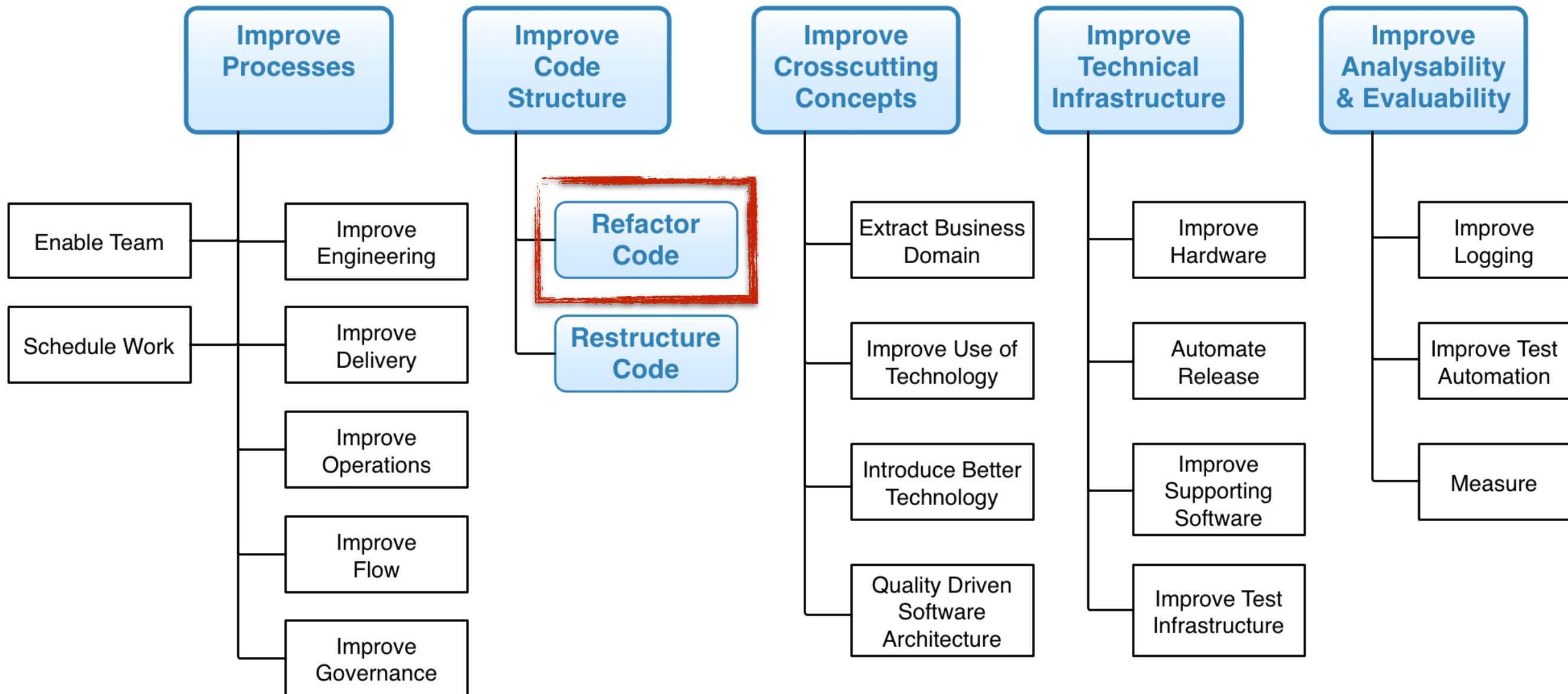


Legend:

fundamental

Category

Practices

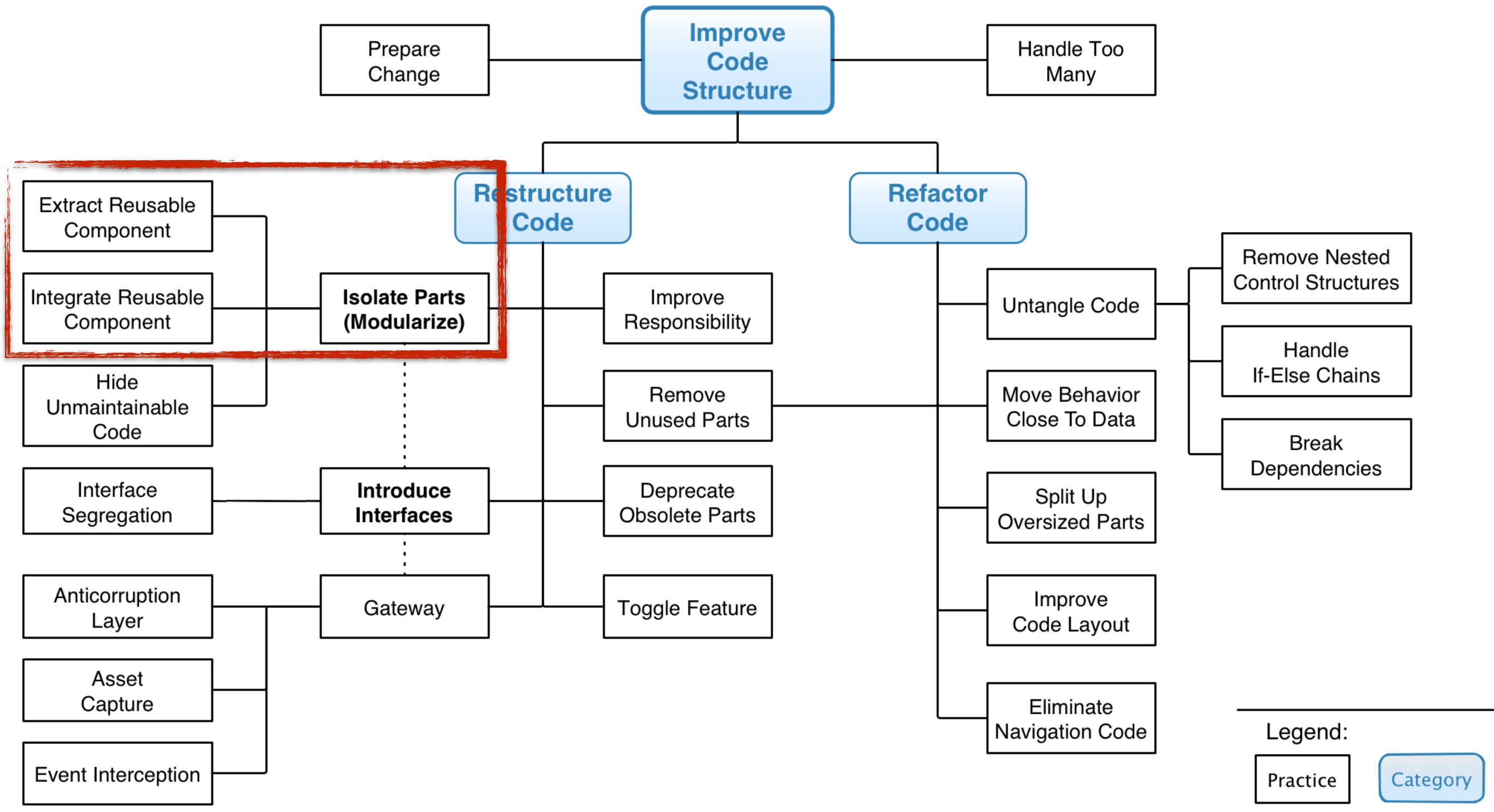


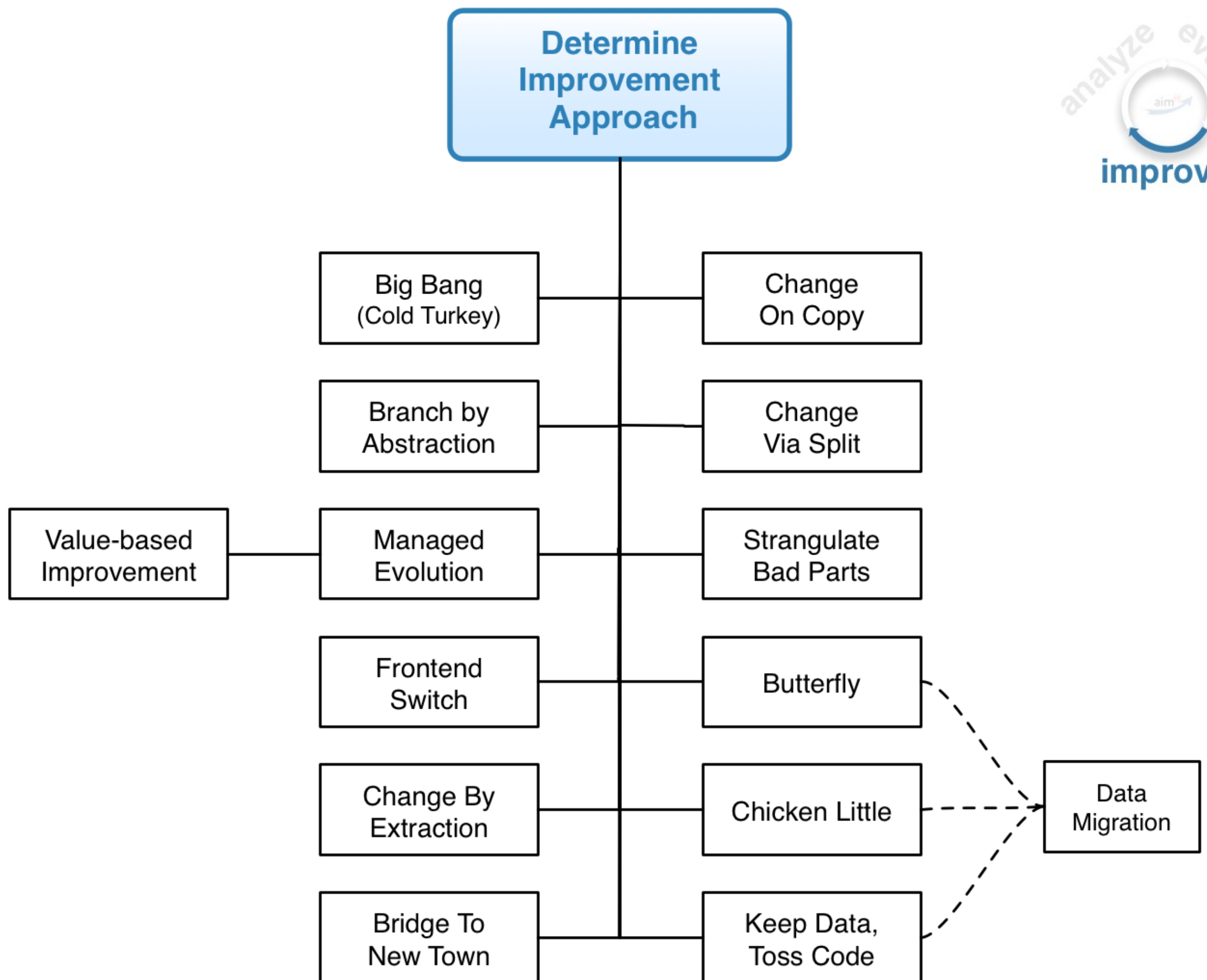
Legend:

Practice

Category

Practices





A smaller Codebase
makes things easier

introduce explicit
boundaries

Just use Microservices

- › Everyone's doing Microservices, so you should, too
- › Everything will be faster with Microservices
- › There are lots of interesting tools to play with, much more interesting than the boring business domain
- › With Microservices we'll be more agile

Business Value?

Microservice Characteristics

small

each running in its own process

lightweight communicating mechanisms (often HTTP)

built around business capabilities

independently deployable

minimum of centralized management

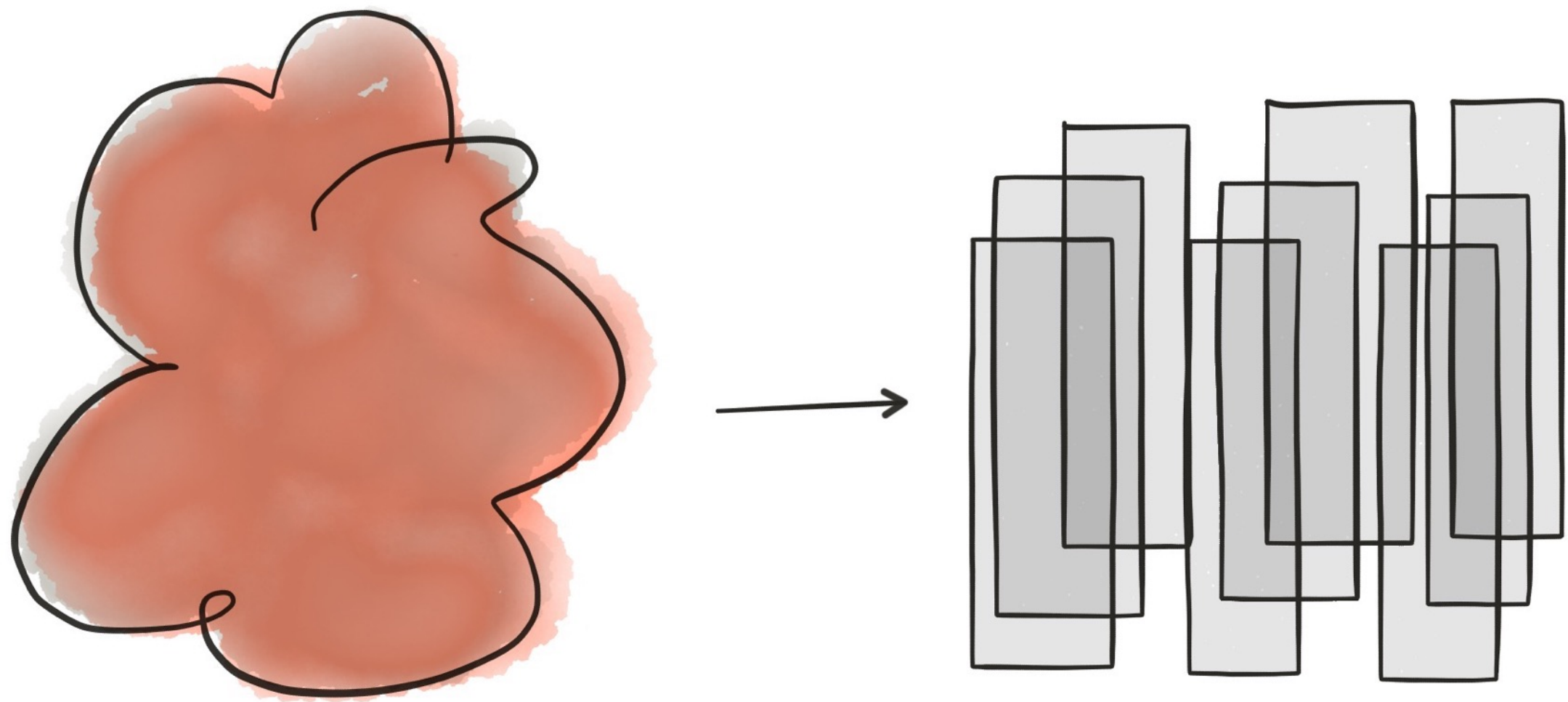
may be written in different programming languages

may use different data storage technologies

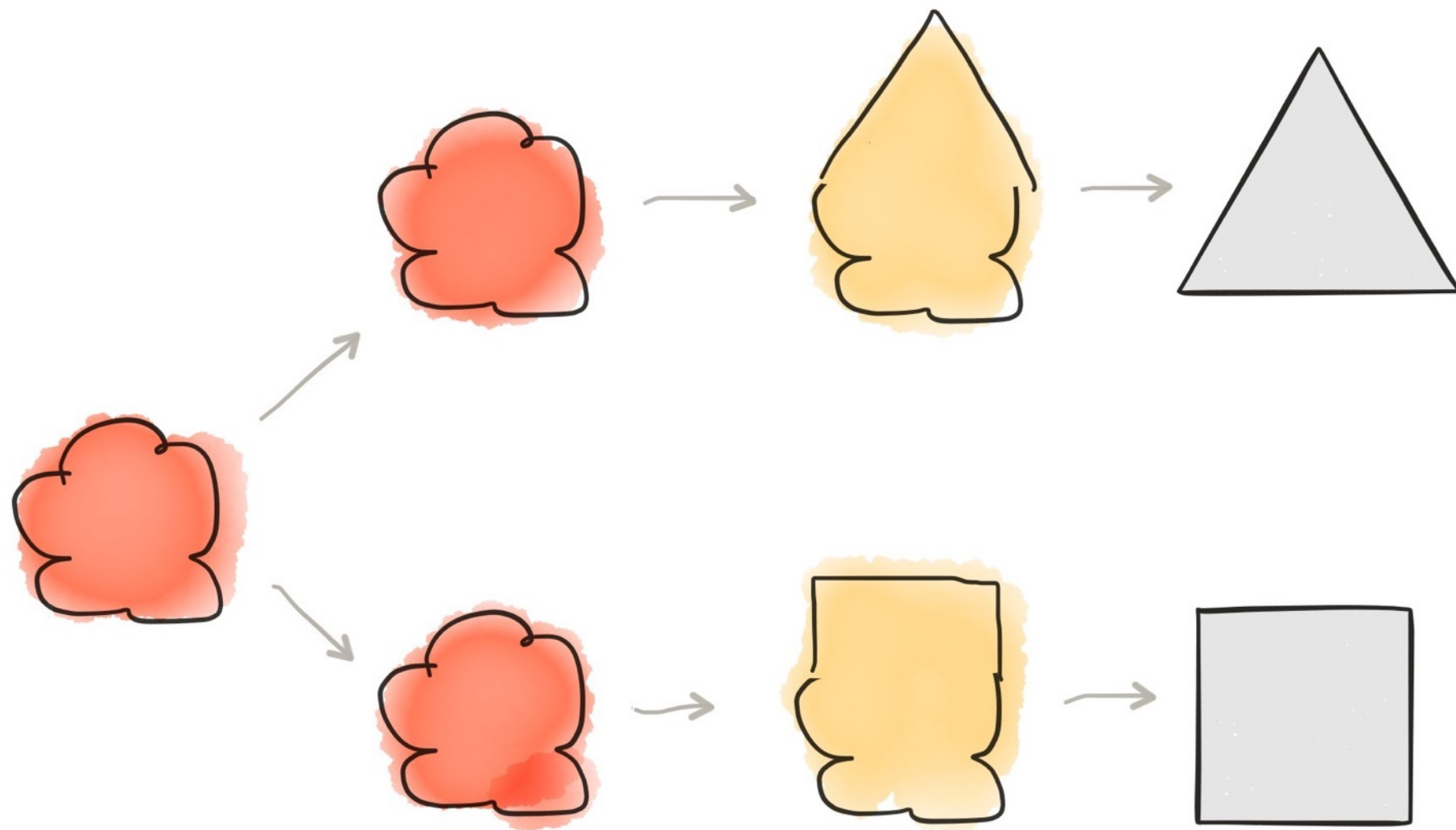
Improvement Approaches applied



Big Bang



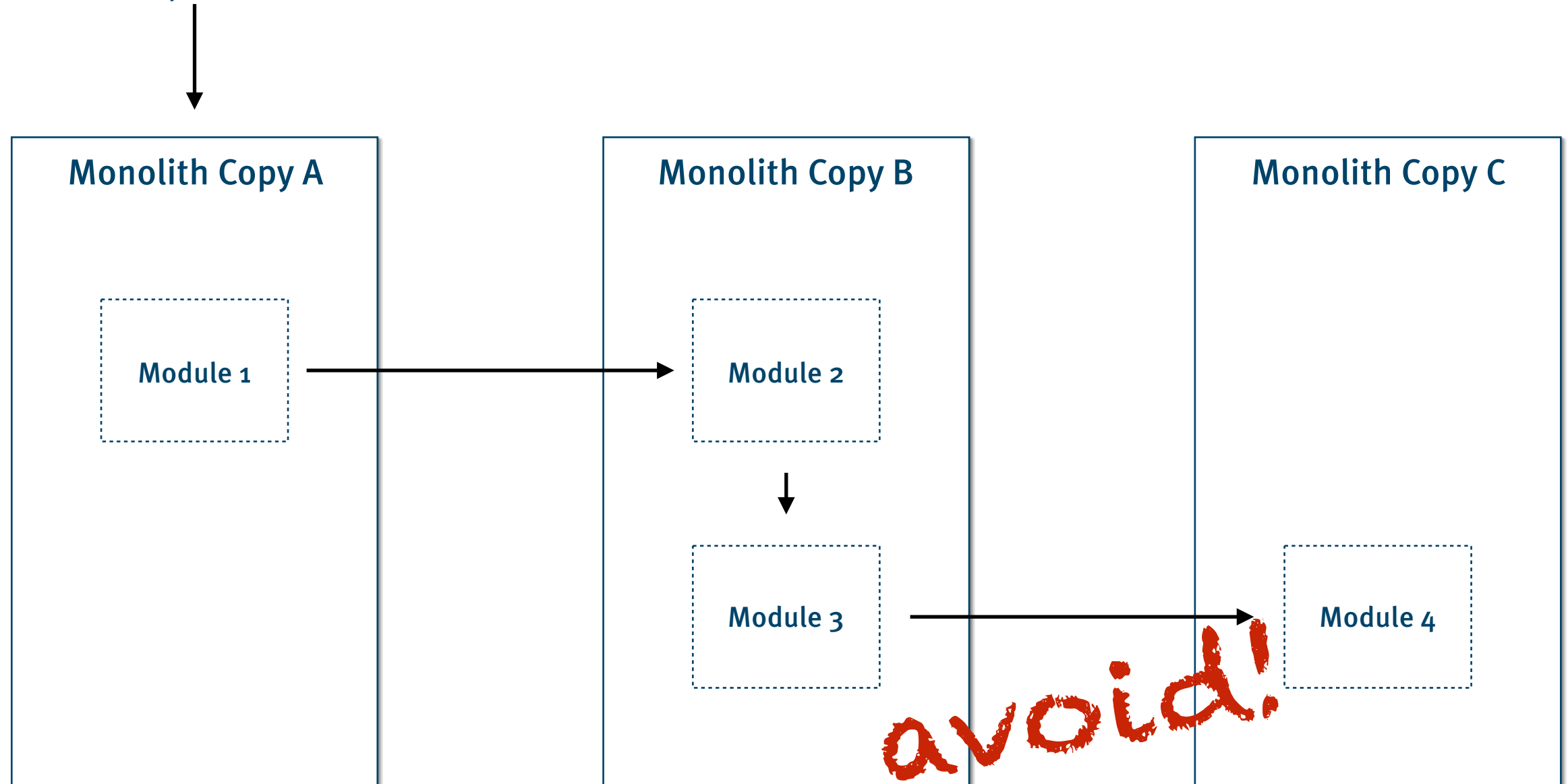
Change on Copy



Integration?

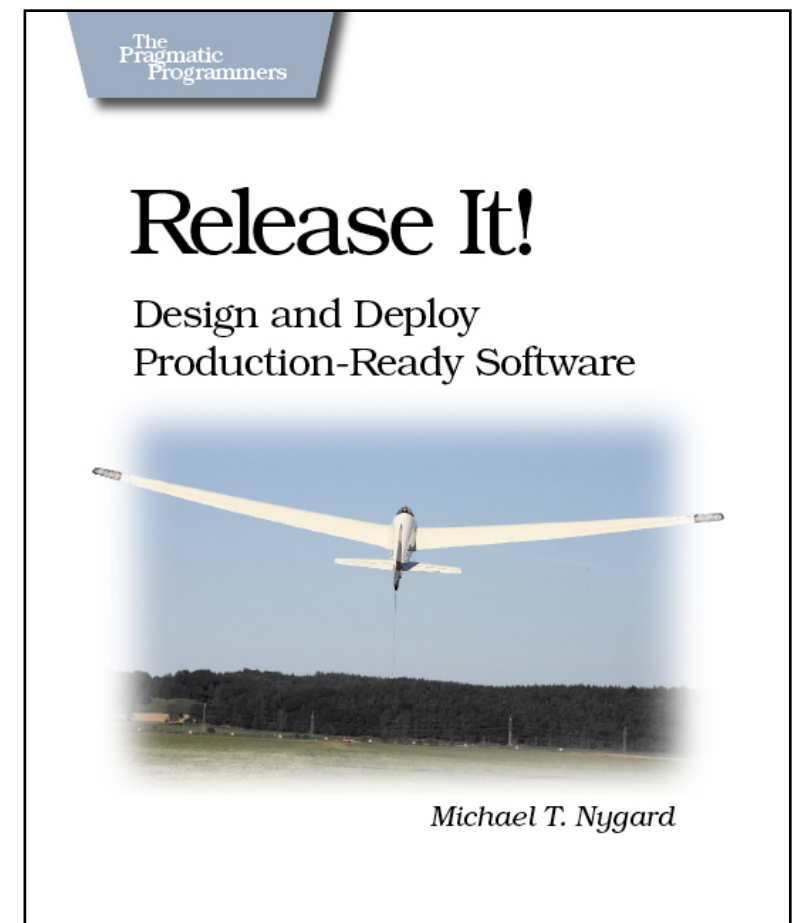
Request Cascades

Customer Request

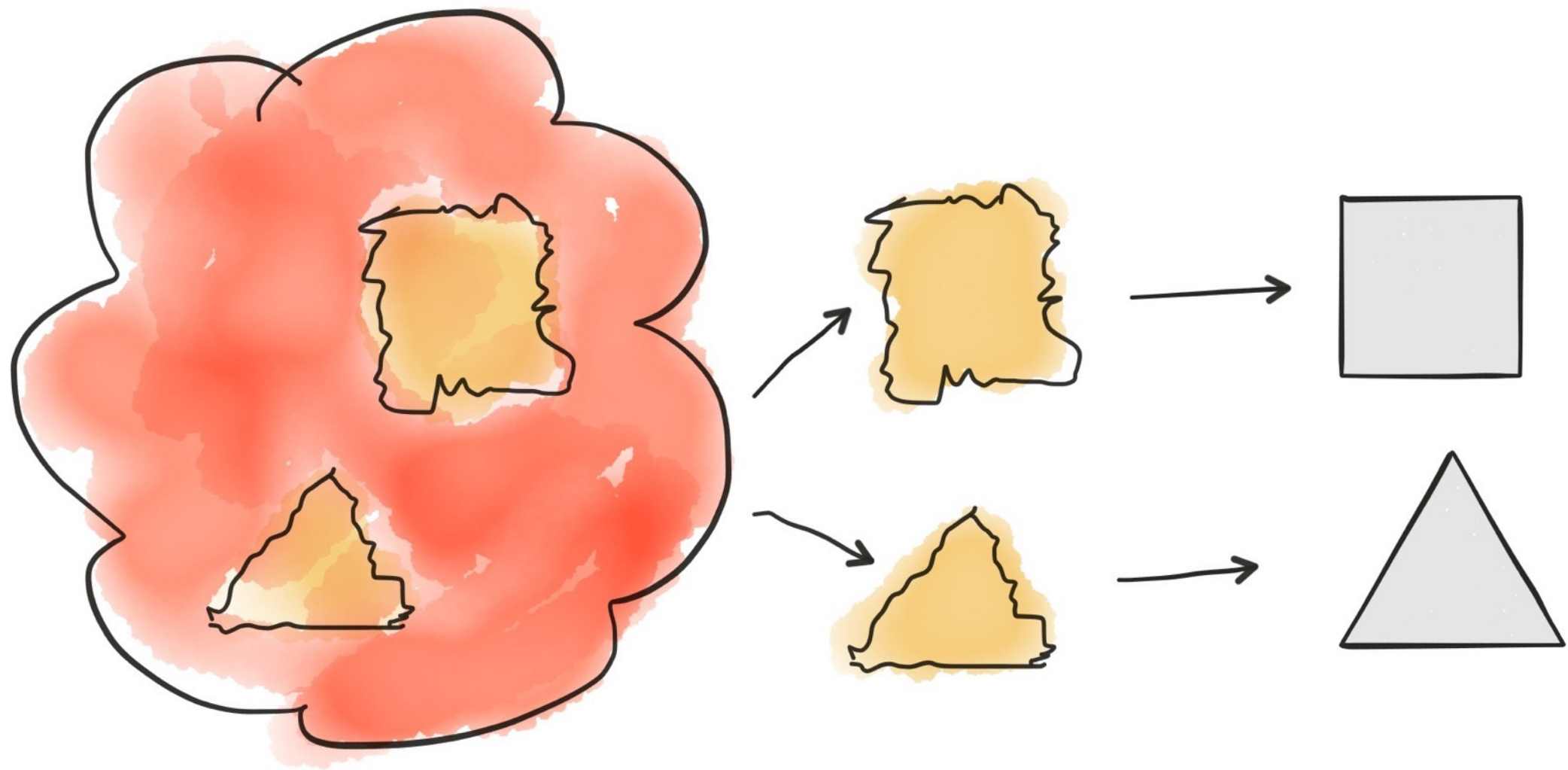


Resilience

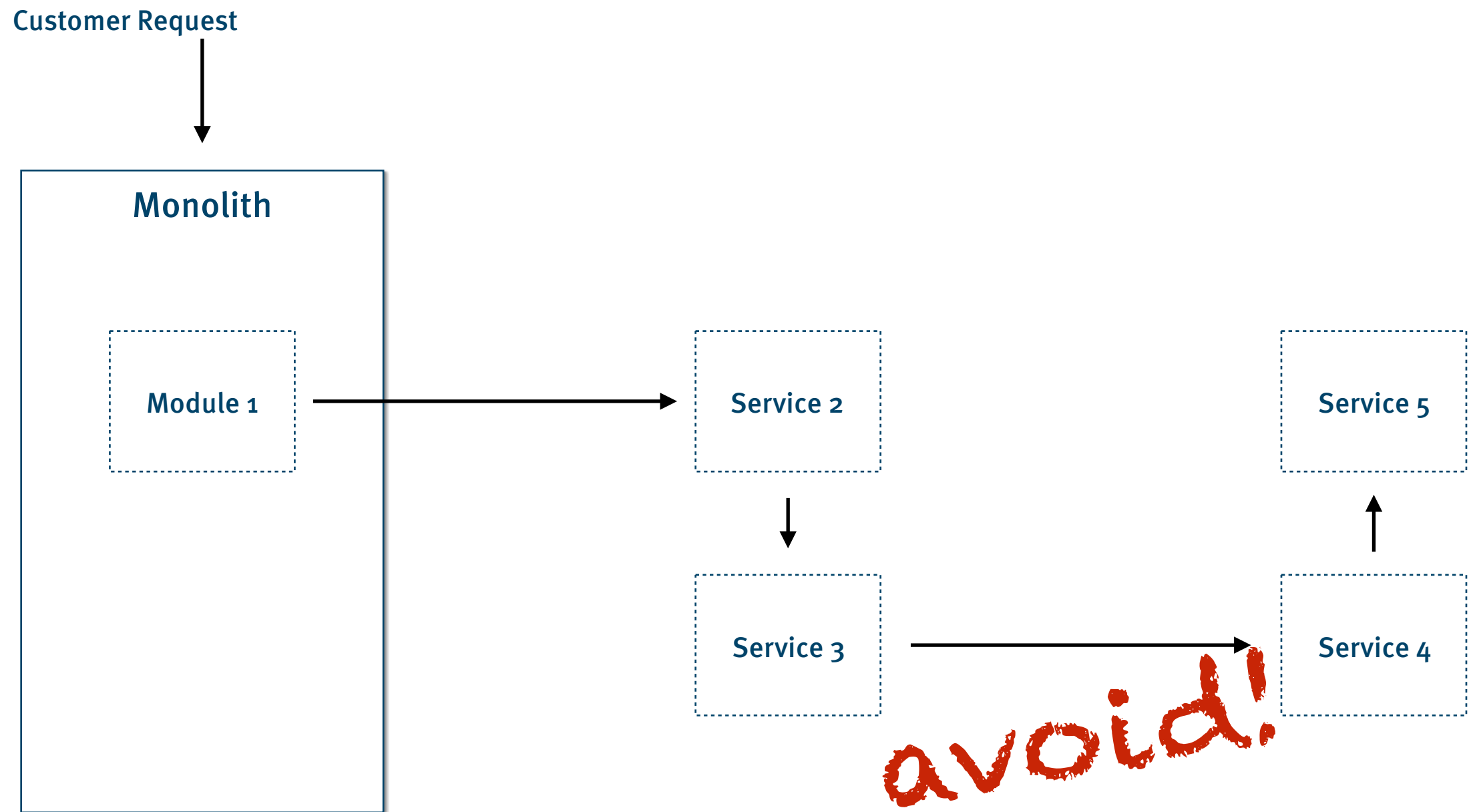
- › isolate Failure
- › apply graceful degradation
- › be responsive in case of failure



Change via Extraction



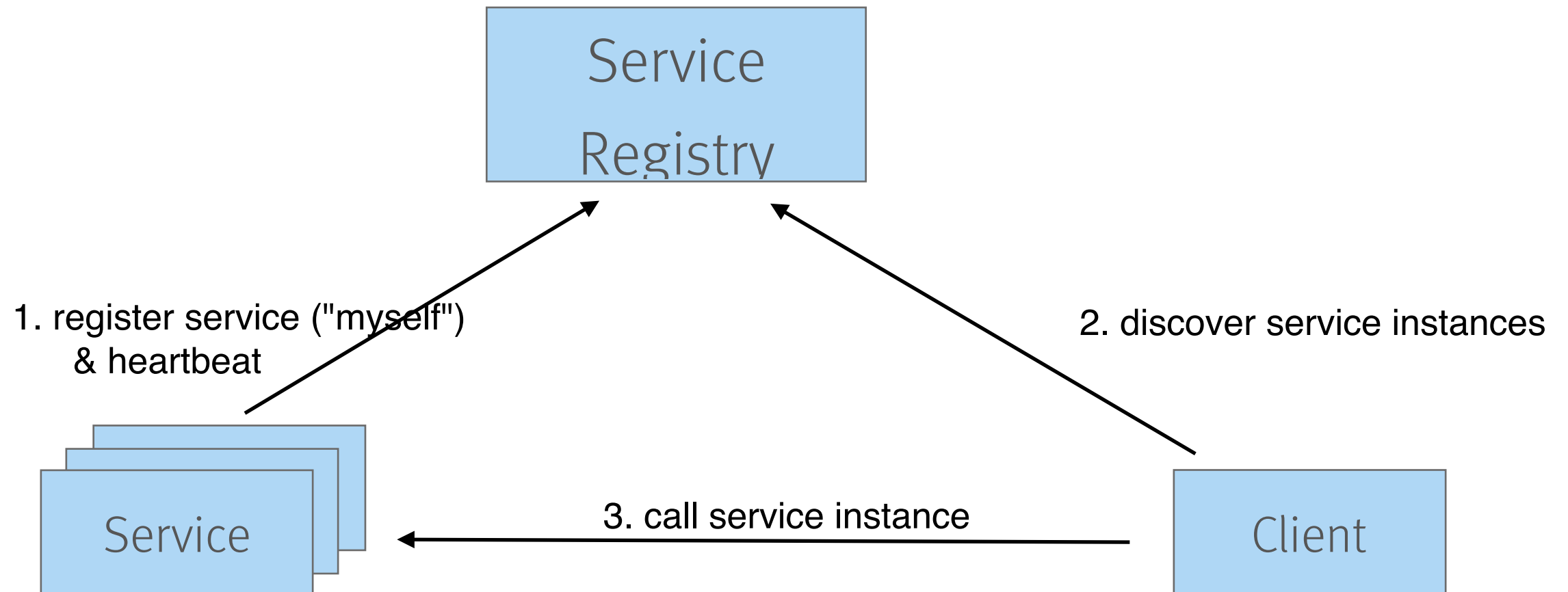
Request Cascades



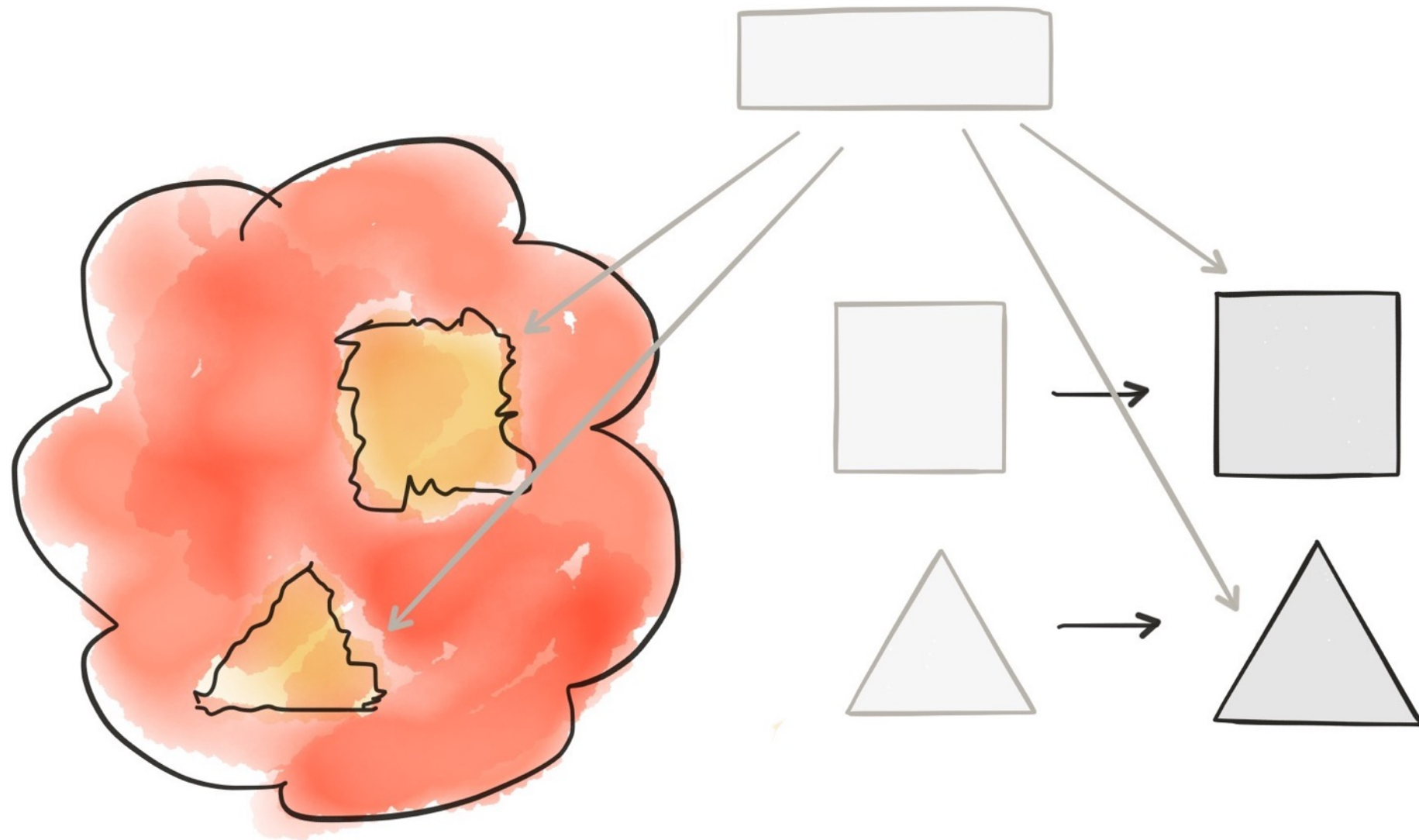
Request Cascades Lower Availability



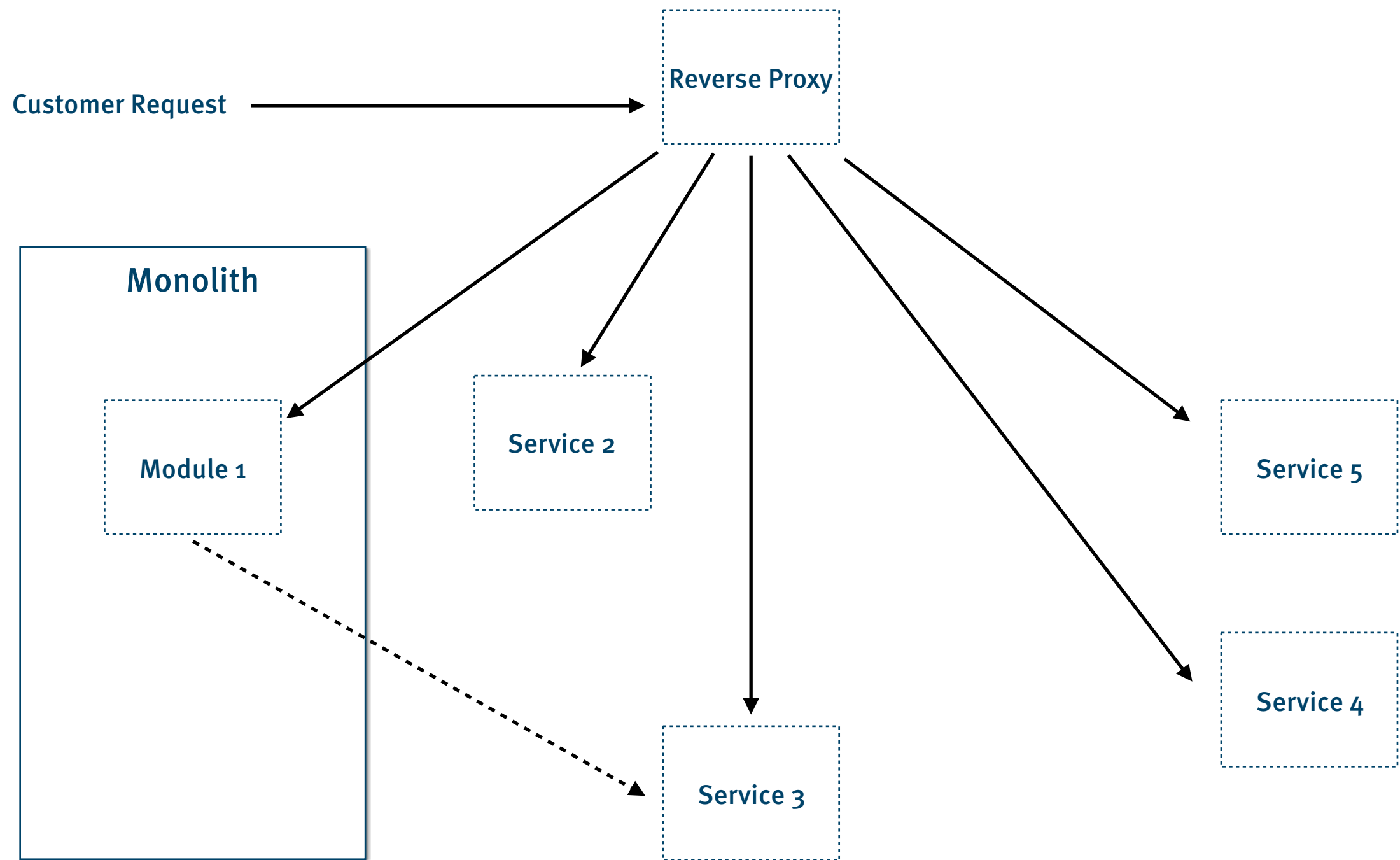
Service Discovery



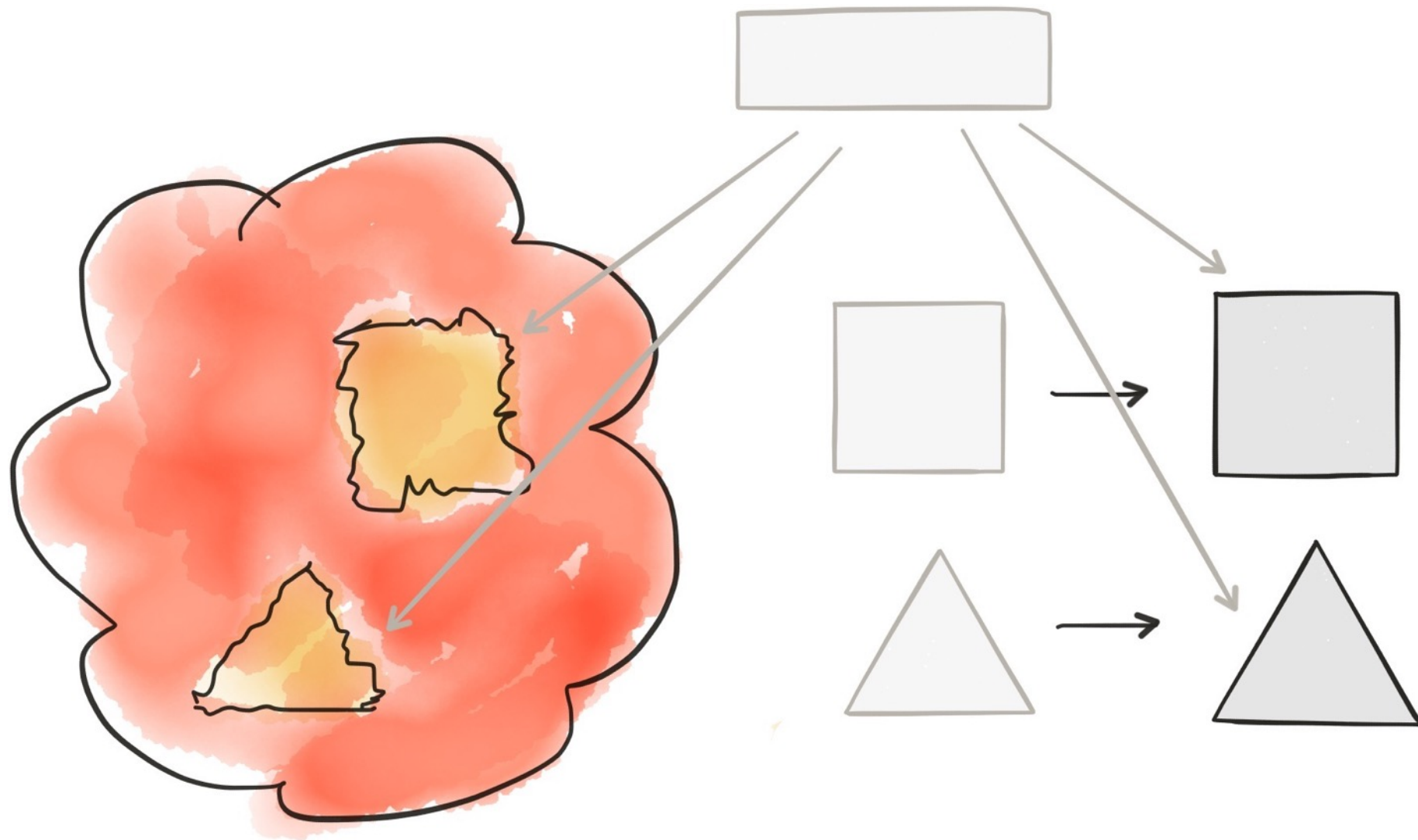
Frontend Switch



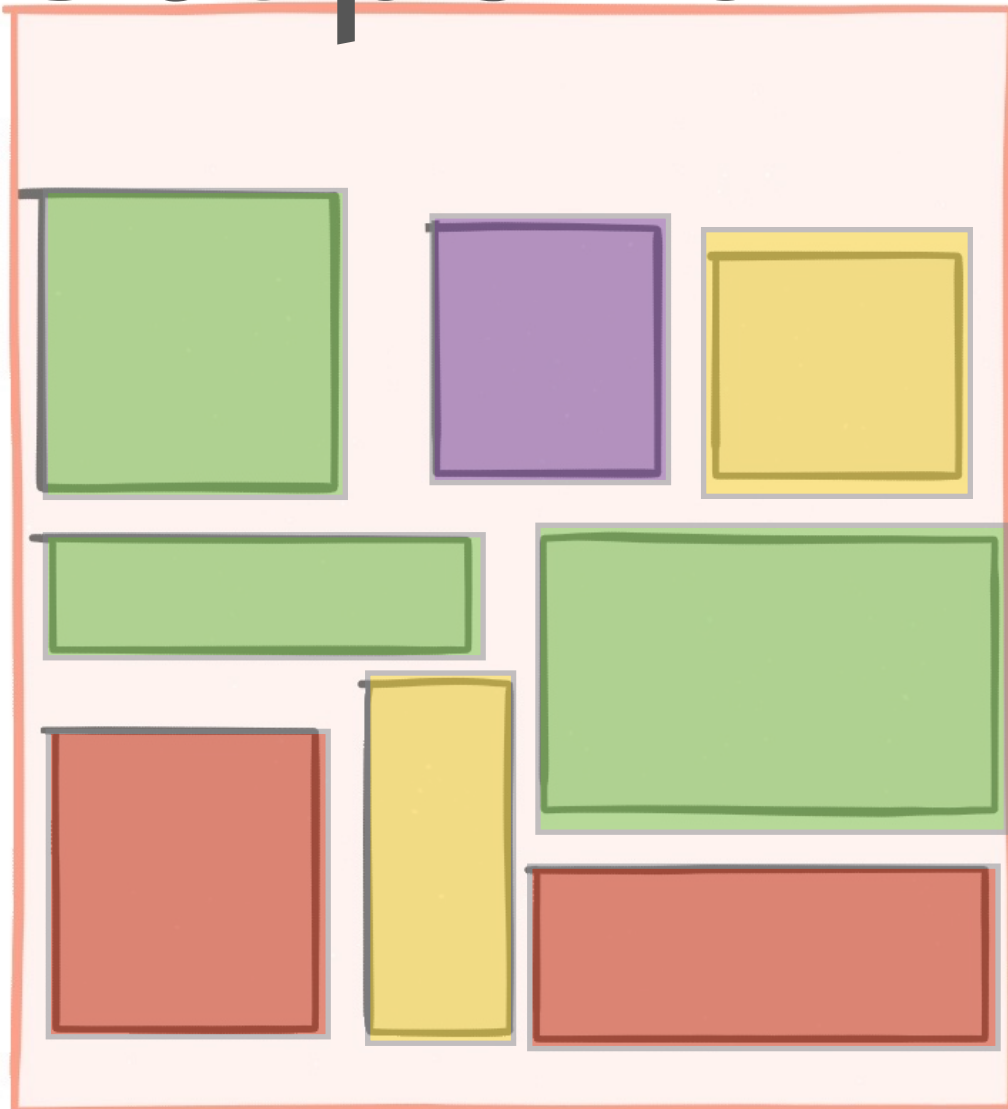
Frontend Switch



Strangulate Bad Parts



Steps for modularisation



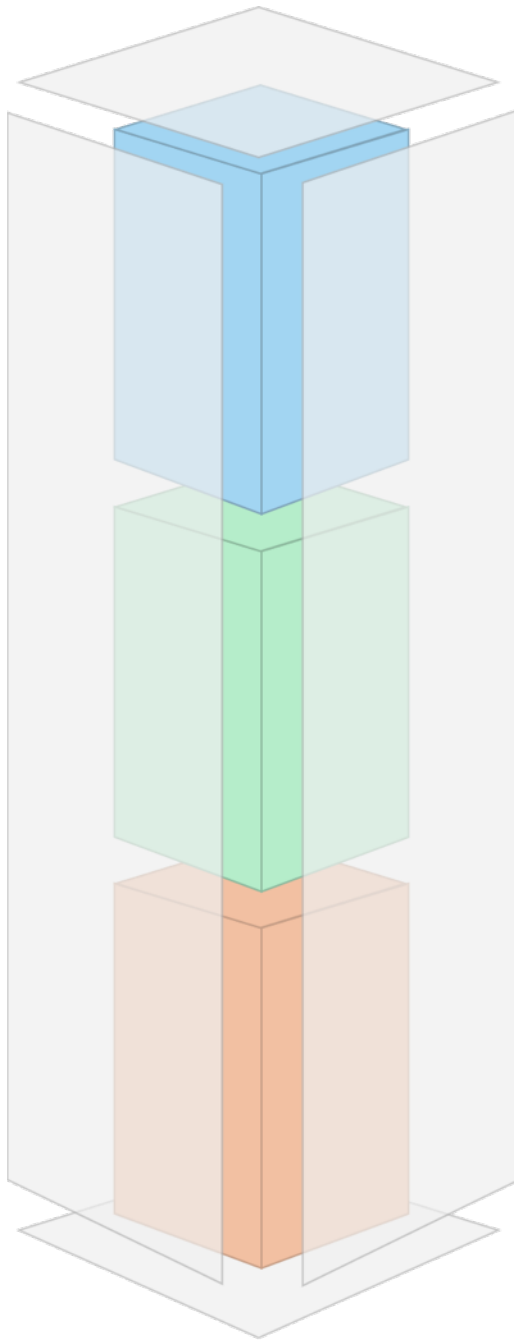
User Management

Product Management

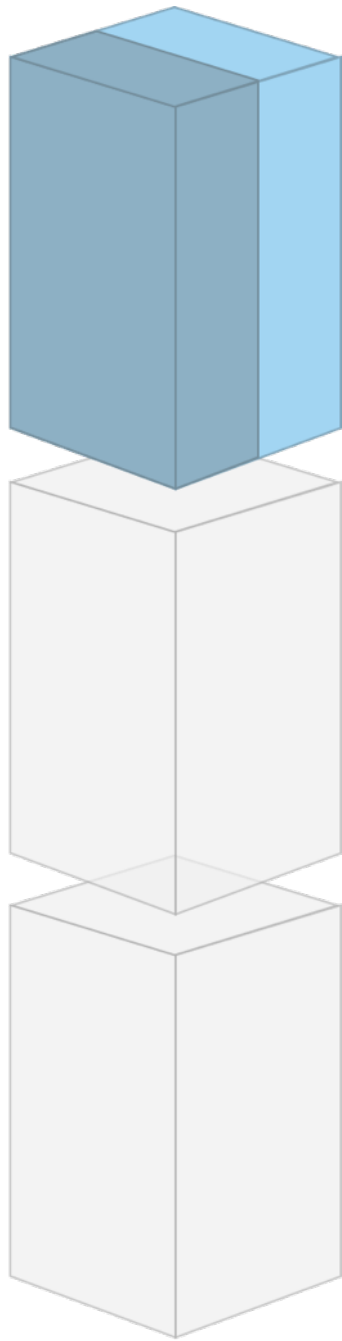
Payment

- identify domains
- group teams by domain
- agree on macro architecture
- focus delivery pipeline on end-to-end features
- team decides migration approach case-by-case

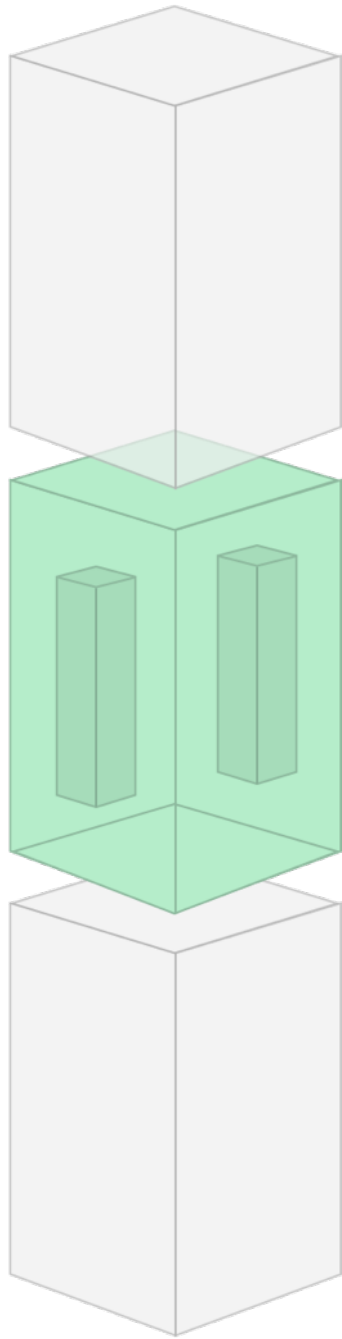
Self-Contained System (SCS)



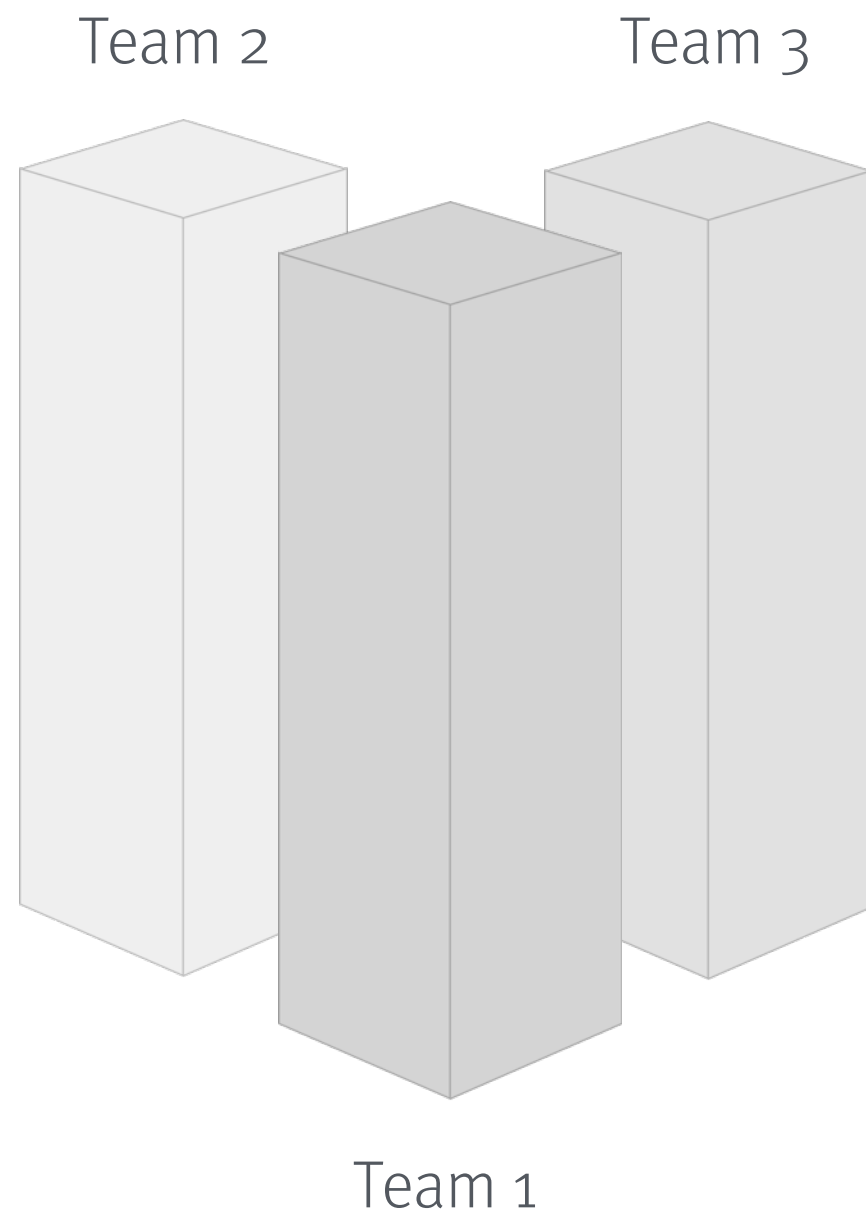
An SCS contains its own
user interface, specific
business logic and
separate data storage



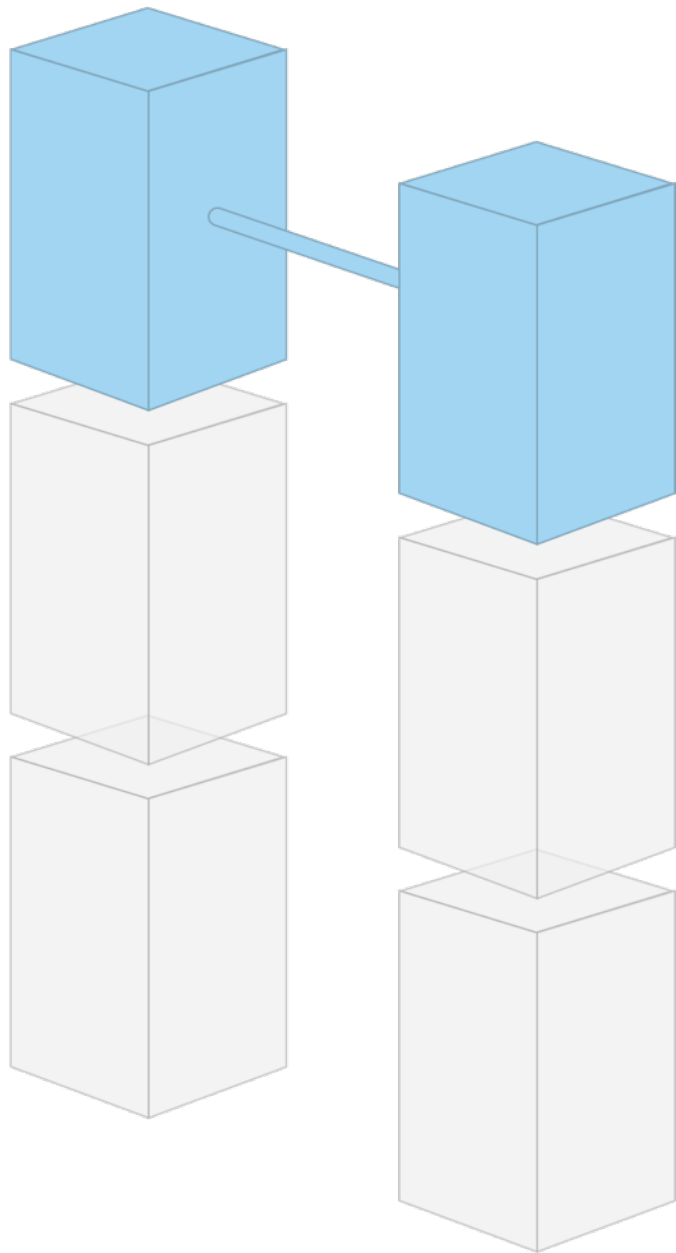
Besides a web interface a self-contained system can provide an **optional API**.



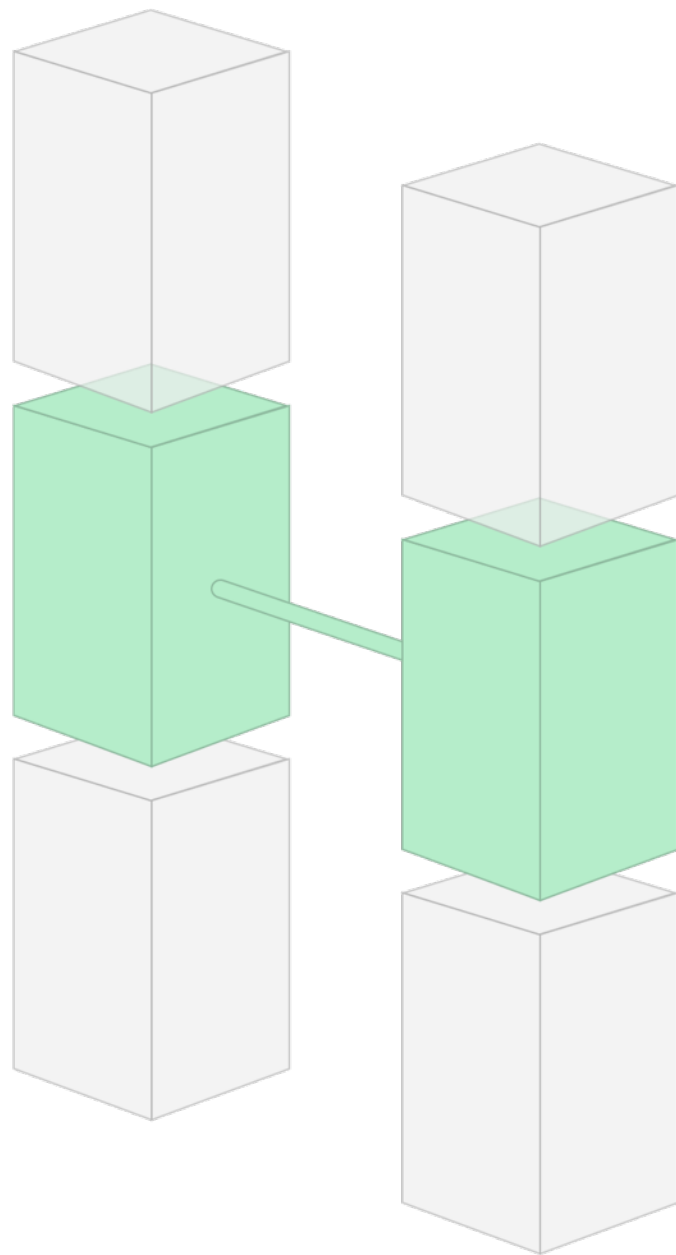
The business logic can consist of **microservices** to solve domain specific problems.



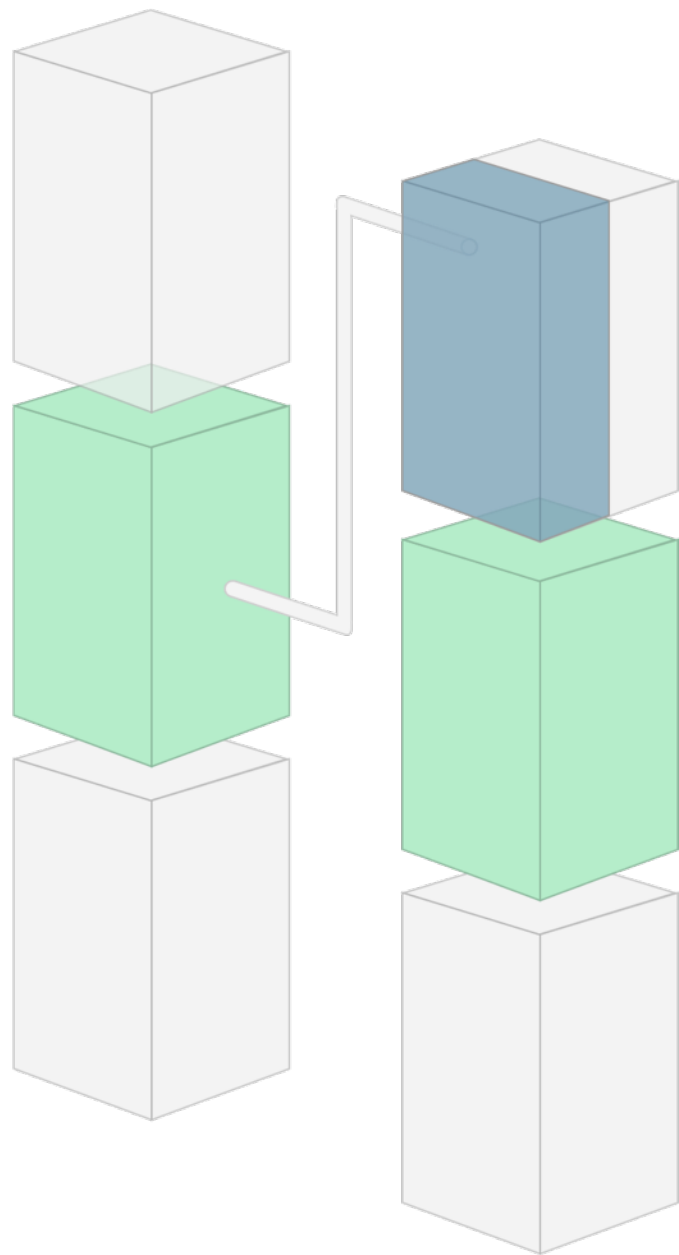
The manageable domain specific scope enables the development, operation and maintenance of an SCS by a **single team**.



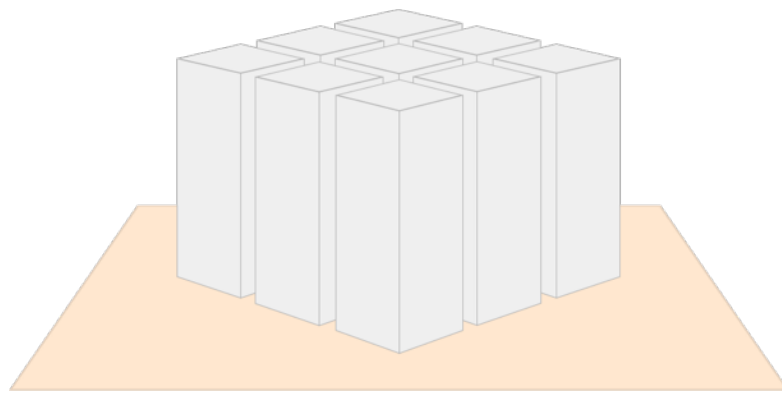
Self-contained Systems
should be integrated over their
web interfaces to minimize
coupling to other systems.



To further minimize coupling to other systems, synchronous remote calls inside the business logic should be **avoided**.



Instead remote API calls should be handled **asynchronously** to reduce dependencies and prevent error cascades.



more information on
self-contained systems
(SCS) can be found at

<http://scs-architecture.org/>

conclusion

Summary

- › aim42 provides structure for software modernization
- › SCSs are a reasonable approach to Microservices
- › Not everyone who **wants** microservices is immediately **capable** to establish them
- › **Don't overwhelm people**, change one thing at a time

Thank you!

Questions?

Comments?

<https://www.innoq.com/en/talks/>

Alexander Heusingfeld, @goldstift

alexander.heusingfeld@innoq.com

<https://www.innoq.com/people/alexander-heusingfeld>

Michael Vitz, @michaelvitz

michael.vitz@innoq.com

<https://www.innoq.com/people/michael-vitz>



www.innoq.com

innoQ Deutschland GmbH

Krischerstr. 100

40789 Monheim am Rhein

Germany

Phone: +49 2173 3366-0

Offices in:

Berlin

Offenbach

München

innoQ Schweiz GmbH

Gewerbestr. 11

CH-6330 Cham

Switzerland

Phone: +41 41 743 0116

info@innoq.com