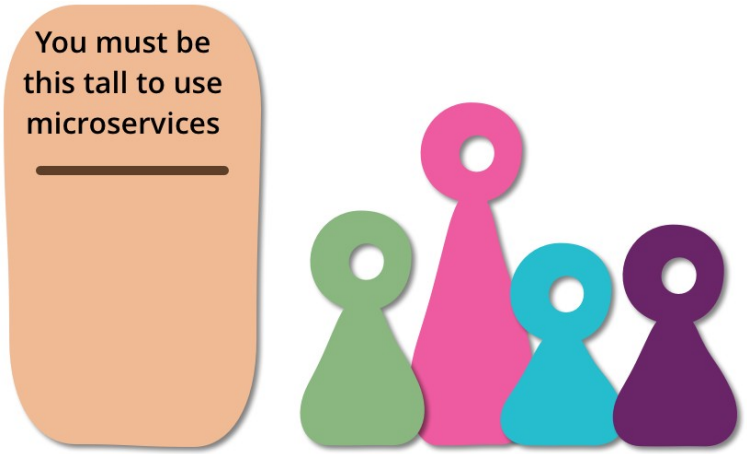
How I learned to stop worrying and love the monolith

Nikola Zivkov
Java Technology Line Manager@Seavus

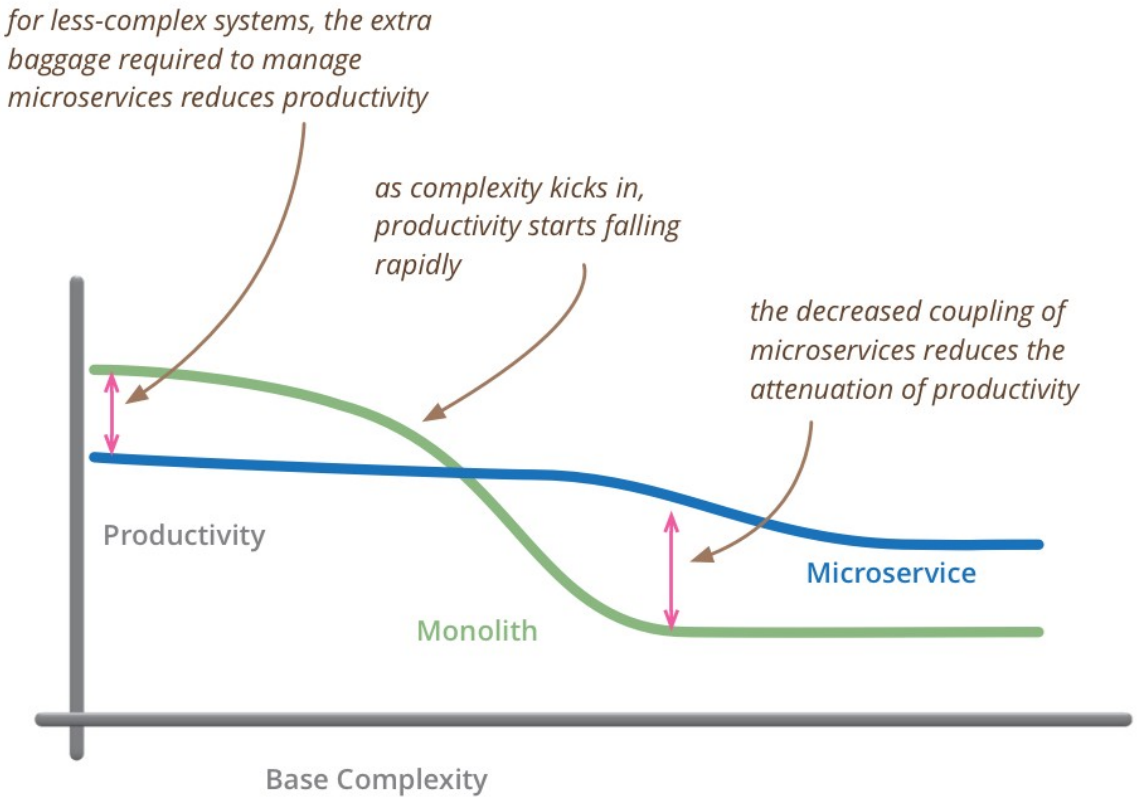
Monolith-first vs. microservices-first

Why monolith-first

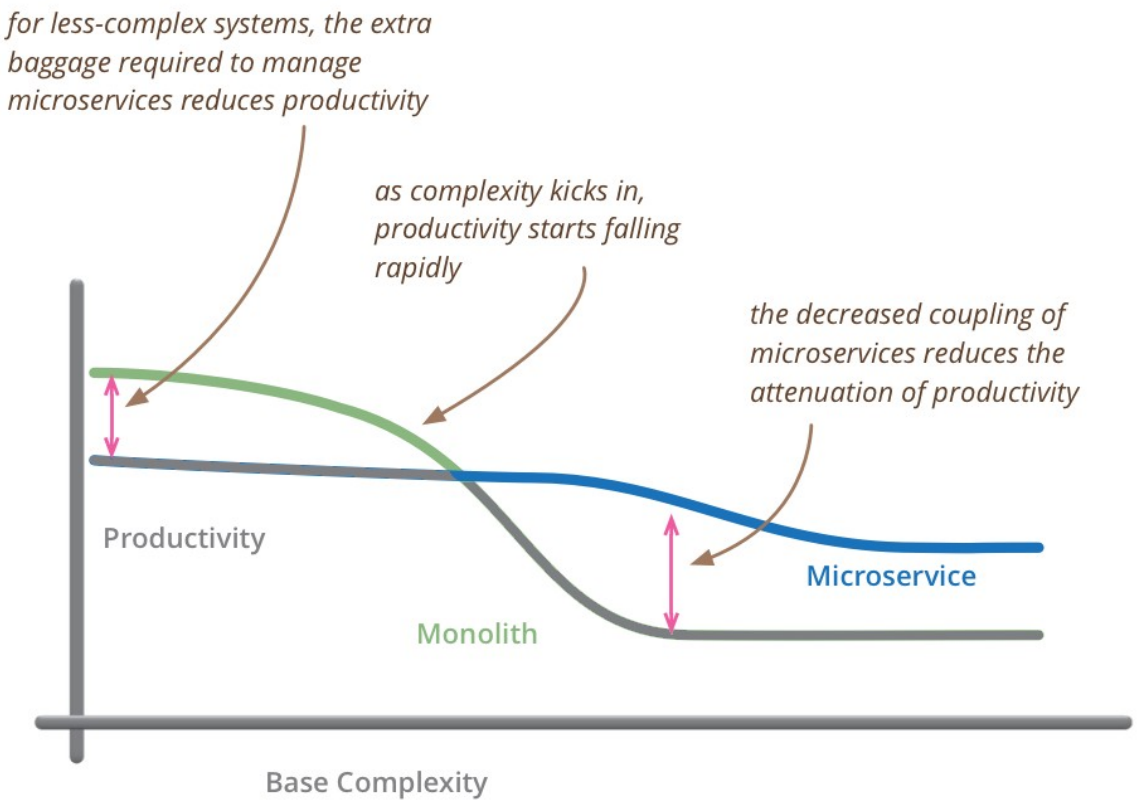
- YAGNI – You Aren't Gonna Need It
- Service boundaries are hard to get at the beginning – refactoring is more expensive
- Microservices require DevOps:
  - Continuous delivery
  - Rapid provisioning
  - Rapid application deployment
  - Central logging
  - Monitoring
  - etc.



Productivity

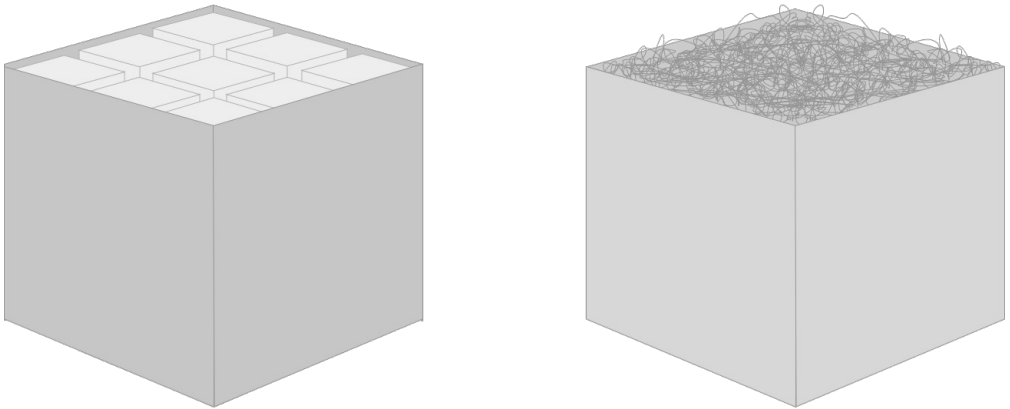


Productivity - Ideally



Why microservices-first?

It’s hard to cut up a large (BBOM) monolith into microservices

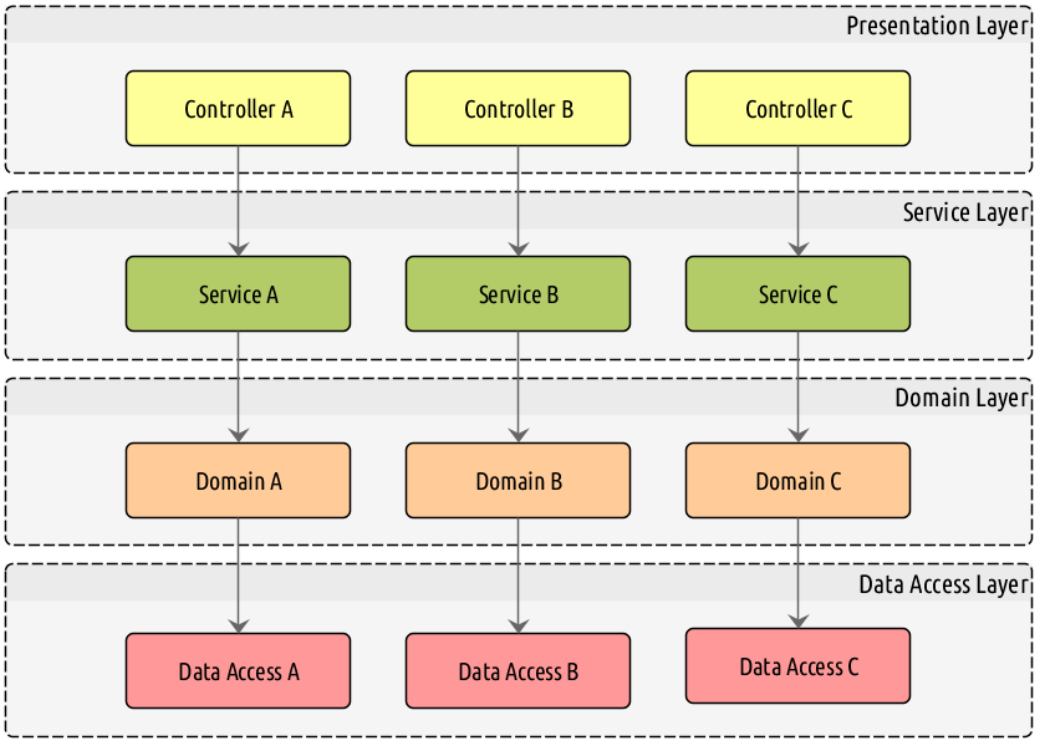


Code entropy

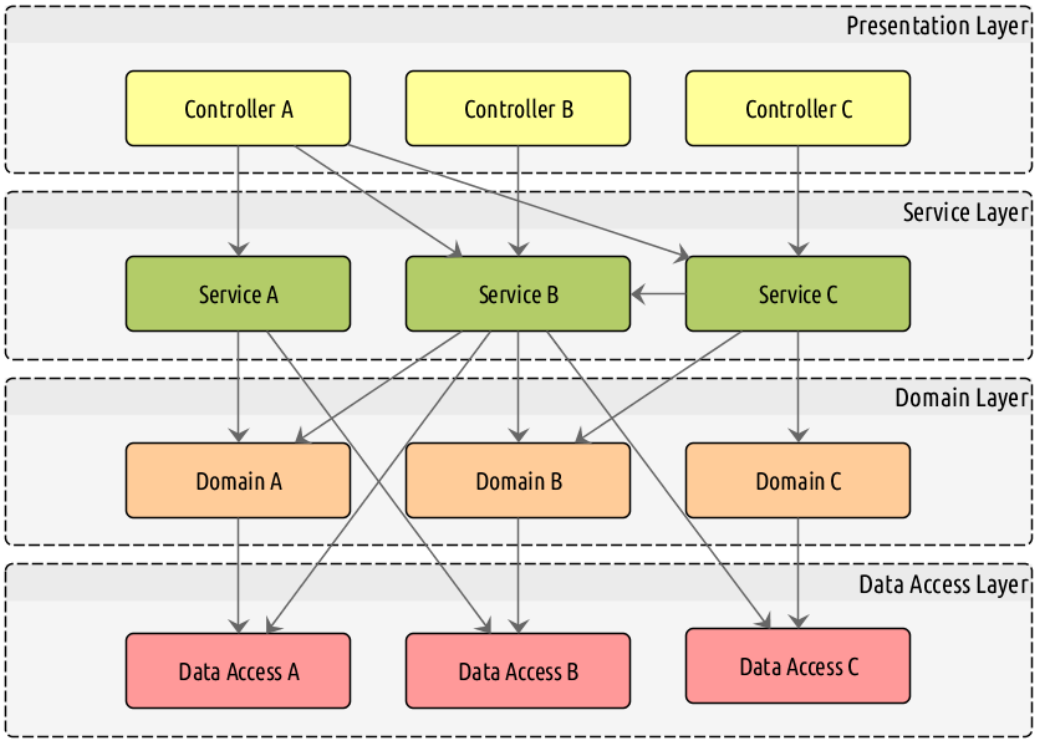
As a system is modified with the inclusion of new functionality, its disorder, or entropy, tends to increase.

- Monolith: new code added to single codebase
- Microservices: new code added as new service – new codebase

Monolith - hope

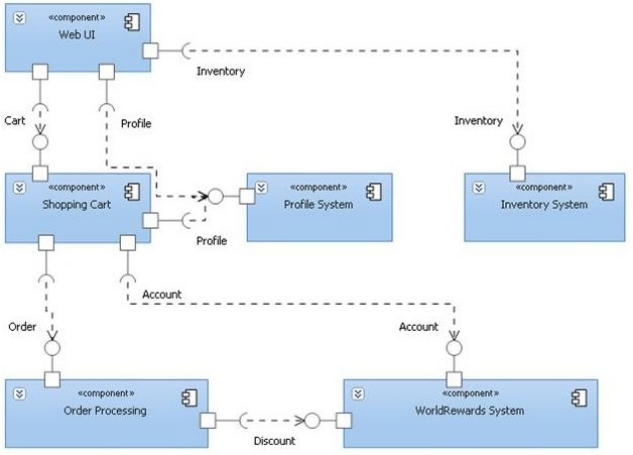


Monolith - reality



The model-code gap

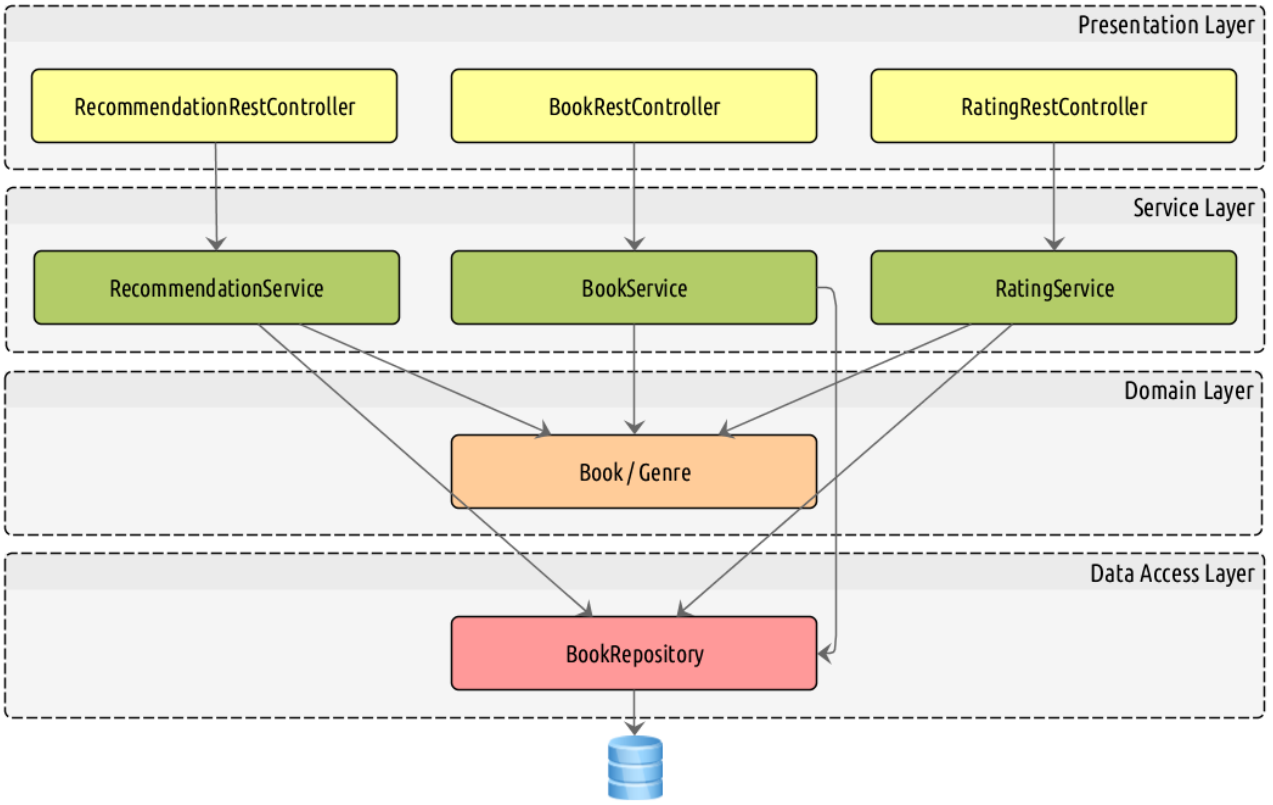
The architecture models include concepts such as component, services, modules, etc. but the code doesn't reflect this - the implementation often happens to be a bunch of classes sitting inside a traditional layered architecture.



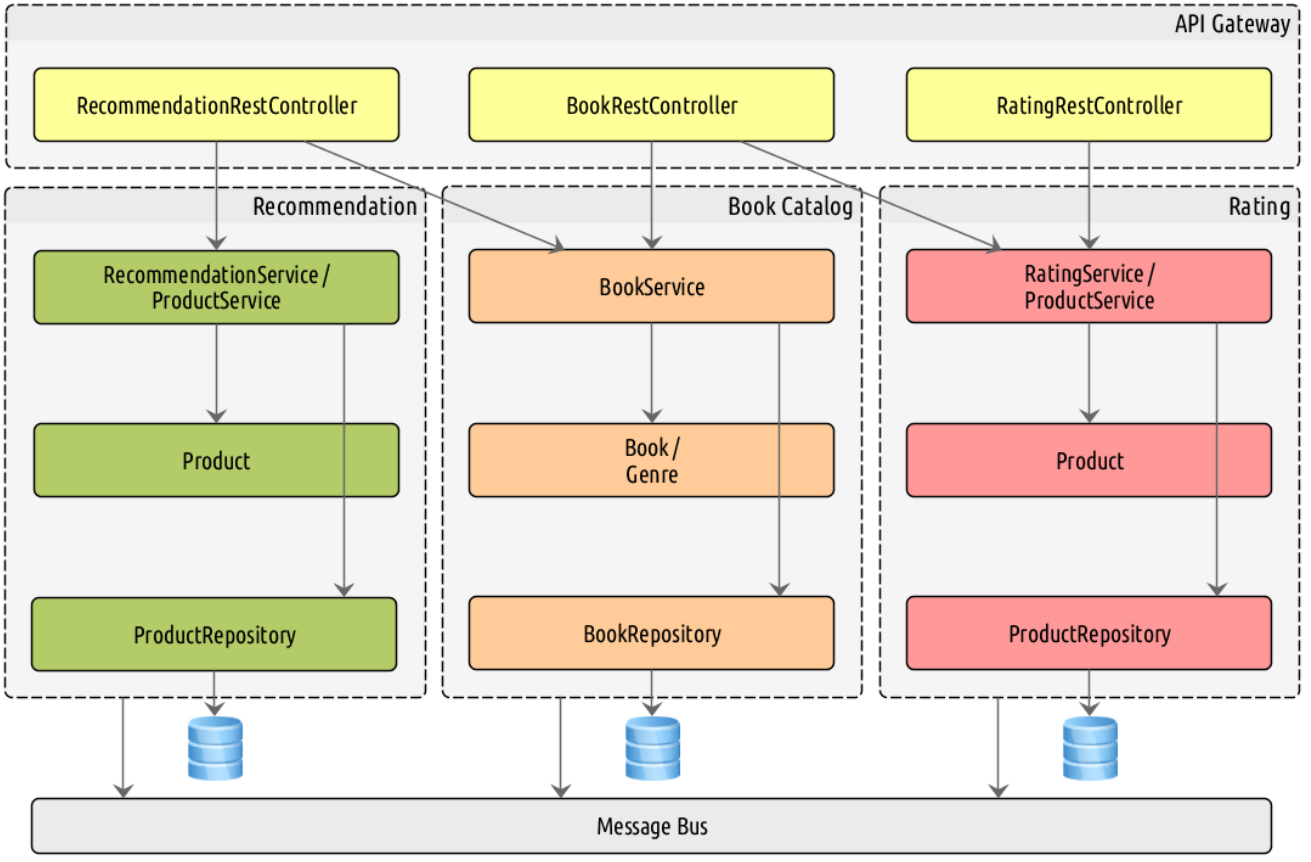
vs.

Bookstore

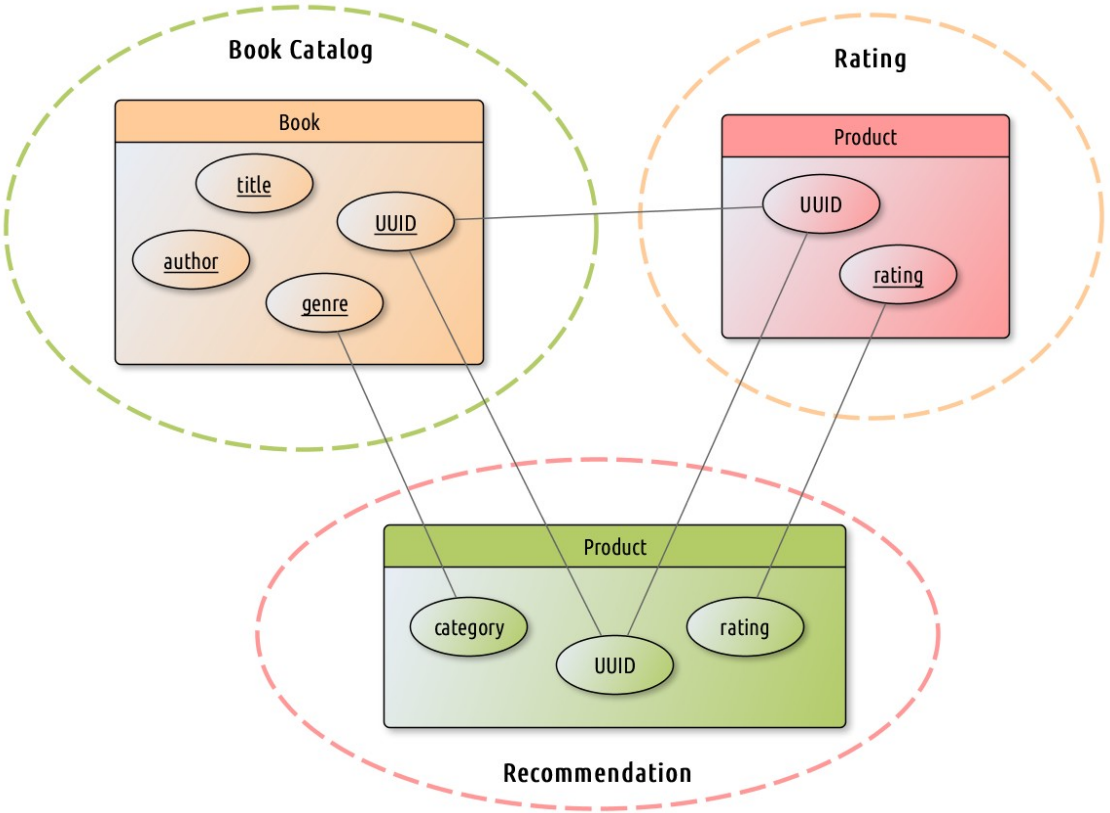
Bookstore monolith



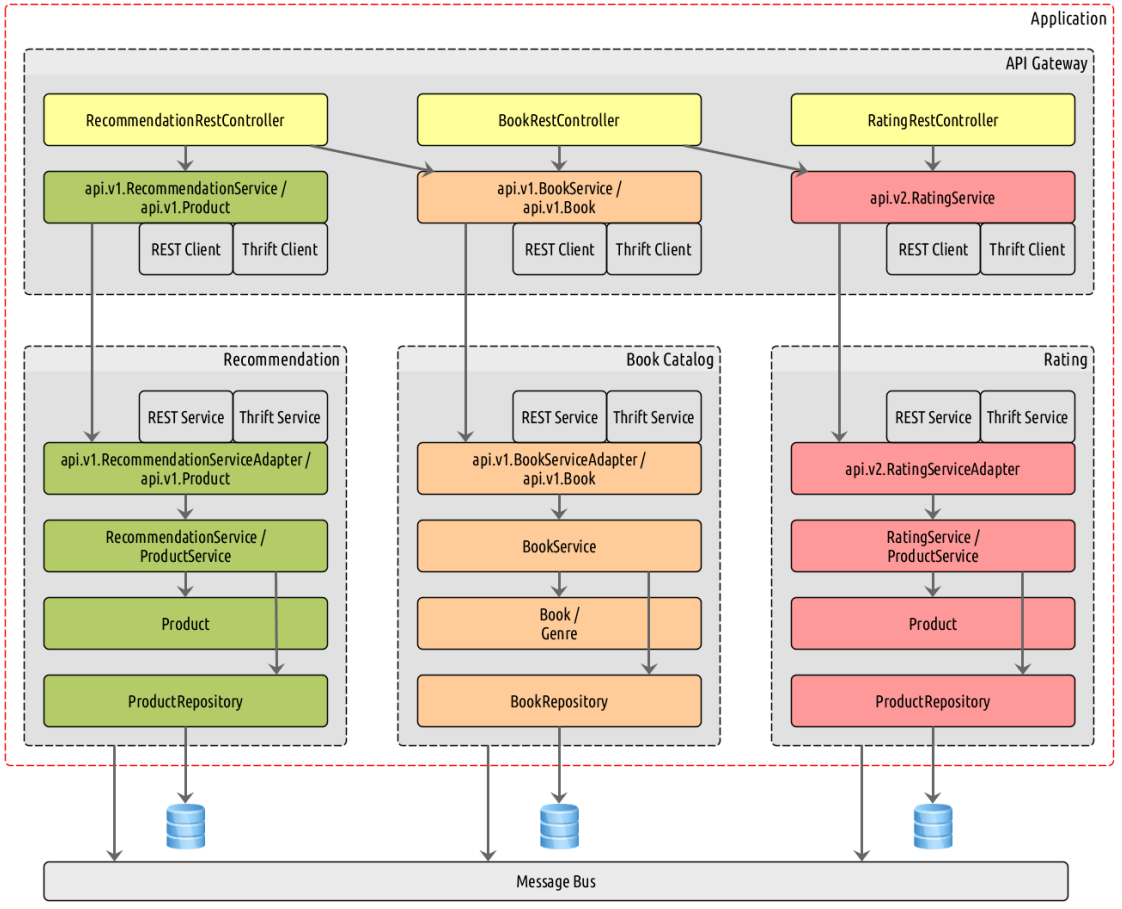
Bookstore monolith restructured



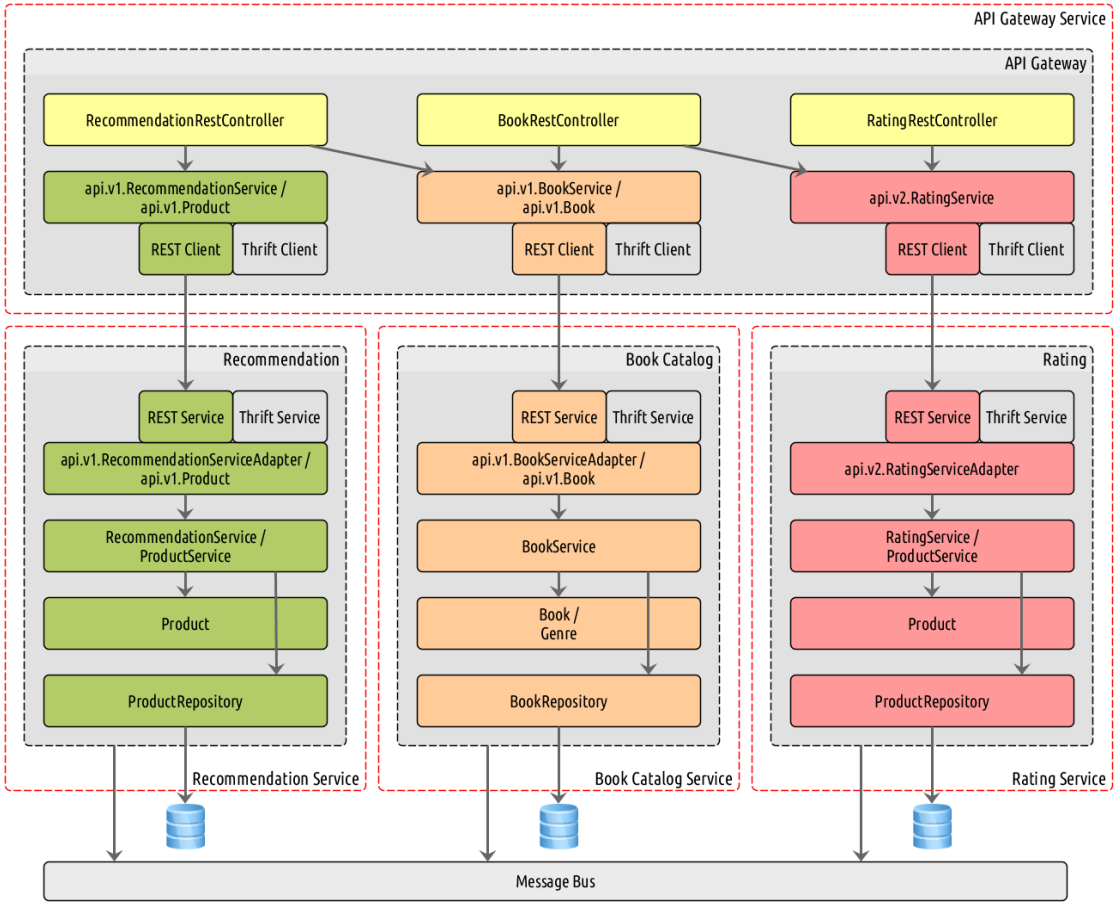
Context map


Book Catalog
Book
title
UUID
author
genre
Rating
Product
UUID
rating
Product
category
UUID
rating
Recommendation

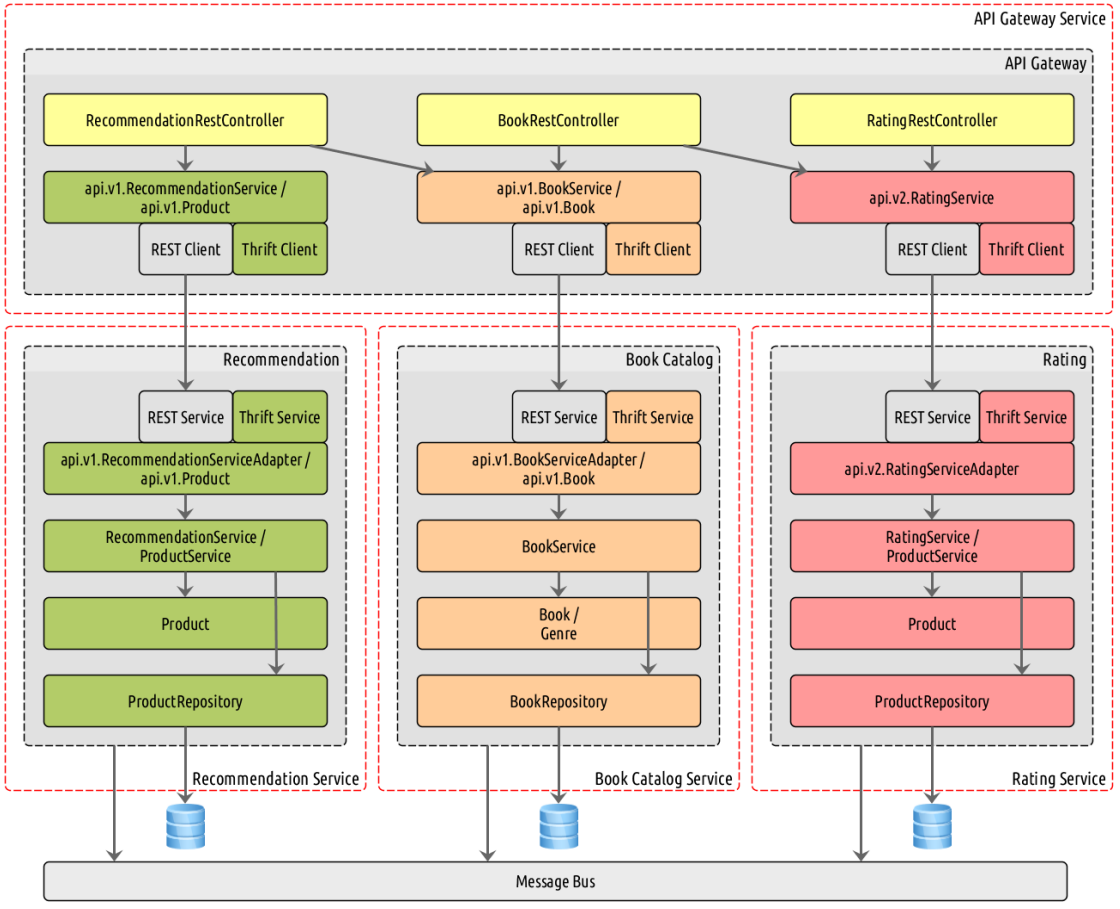
Deployment as monolith



Deployment as microservices

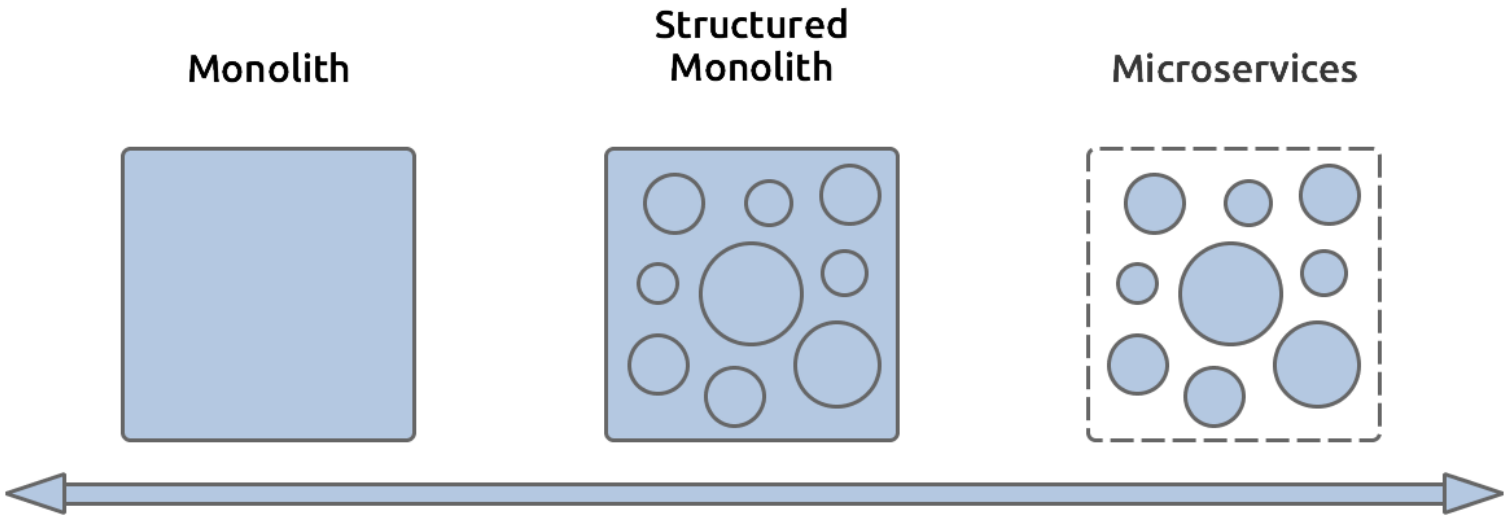


Deployment as microservices



Monolith to microservices

Monolith to microservices



Monolith codebase: YES
Monolith deployment: YES

Monolith codebase: NO
Monolith deployment: YES

- Components
- High cohesion
- Low coupling

Monolith codebase: NO
Monolith deployment: NO

+ Polyglot programming
+ Heterogeneous technology stack
+ Individually deployable, upgradeable, replaceable, scalable

(Microservices) architectural prerequisites

- Forget about layers – your package structure should depict functional concepts
- Use Domain-Driven Design's bounded contexts to isolate and model components
- Use database per component to achieve full decoupling
- Propagate data changes between components asynchronously using a message bus
- Use ports and adapters when components needs to talk synchronously (RPC) – carefully consider when to use this

To conclude

Simon Brown
@simonbrown

I'll keep saying this ... if people can't build monoliths properly, microservices won't help. #qconlondon #DesignThinking #Modularity

RETWEETS 244 LIKES 100

10:49 AM - 4 Mar 2015

Monolithic

Microservices

Thank You

Time for some Q&A

eventory

YOUR OPINION IS VERY IMPORTANT TO ME.

- FIND MY LECTURE ON THE SCHEDULE IN THE EVENTORY APP.
- RATE AND COMMENT MY PERFORMANCE.

THANKS TO YOUR FEEDBACK, I WILL KNOW WHAT TO IMPROVE!