Architecture: Microservices

17-313 Spring 2024

Foundations of Software Engineering

https://cmu-313.github.io

Michael Hilton and Eduardo Feo Flushing

Administrivia

- Project 2B due tonight
  - Next Sprint (2C) due Feb 29
- Teamwork assessments due every Friday
- Reminder: Midterm on February 27 in class
  - We will release sample / practice exams for recitation next week

Smoking Section

- Last **two** full rows

Learning Goals

- Contrast the monolithic application design with a modular design based on microservices.
- Reason about tradeoffs of microservices architectures.
- Principles of microservices: how to benefit and avoid their pitfalls

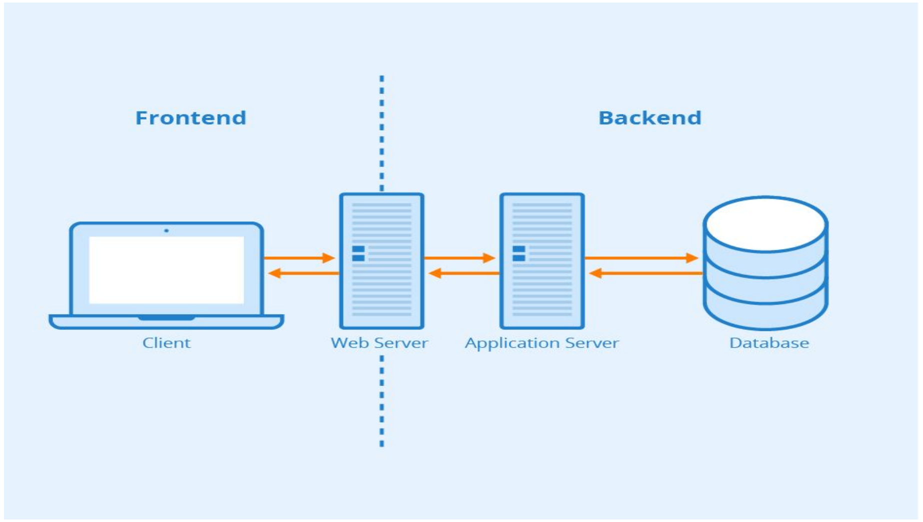
Outline

- From Monoliths to Service Oriented Architecture
  - Case Study: Chrome Web Browser
- Microservices
  - Monolith vs Microservices
  - Advantages
  - Challenges
- Microservices: Principles
- Serverless

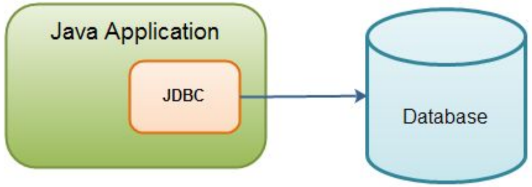
Before we get to microservices…

MONOLITHS

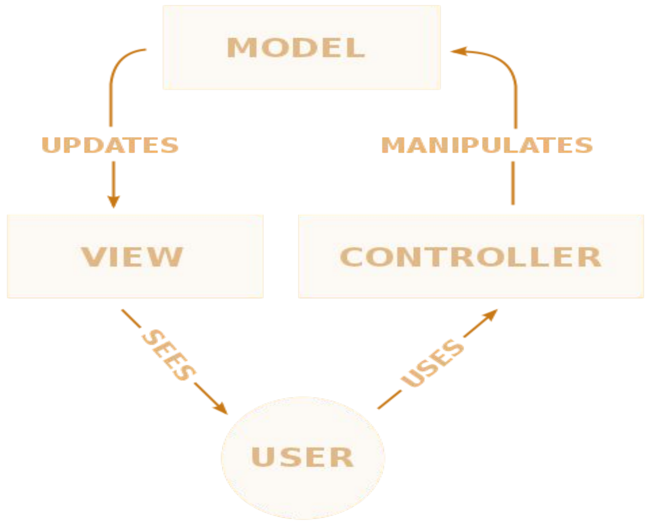
Monolithic styles



Source: https://www.seobility.net (CC BY-SA 4.0)



Monolithic styles: MVC Pattern (e.g., NodeBB)

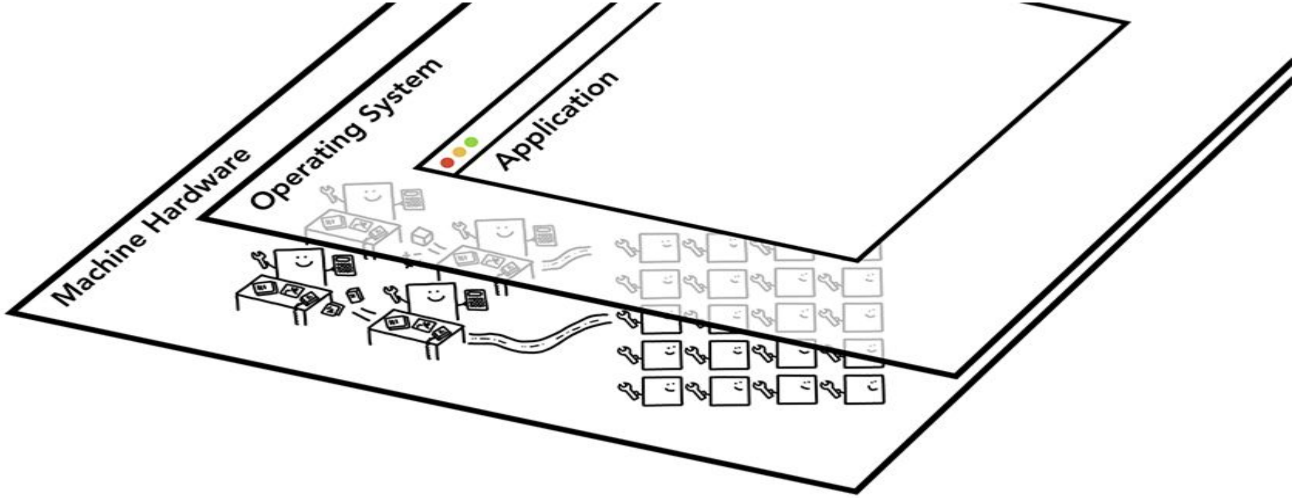


Separation of concerns

SERVICE-BASED ARCHITECTURE

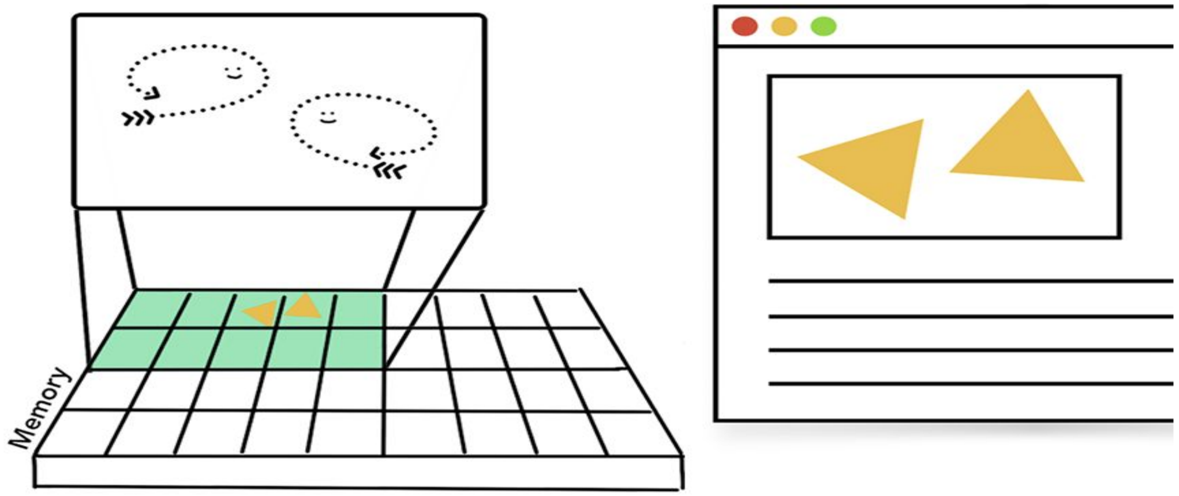
Chrome

Web Browsers



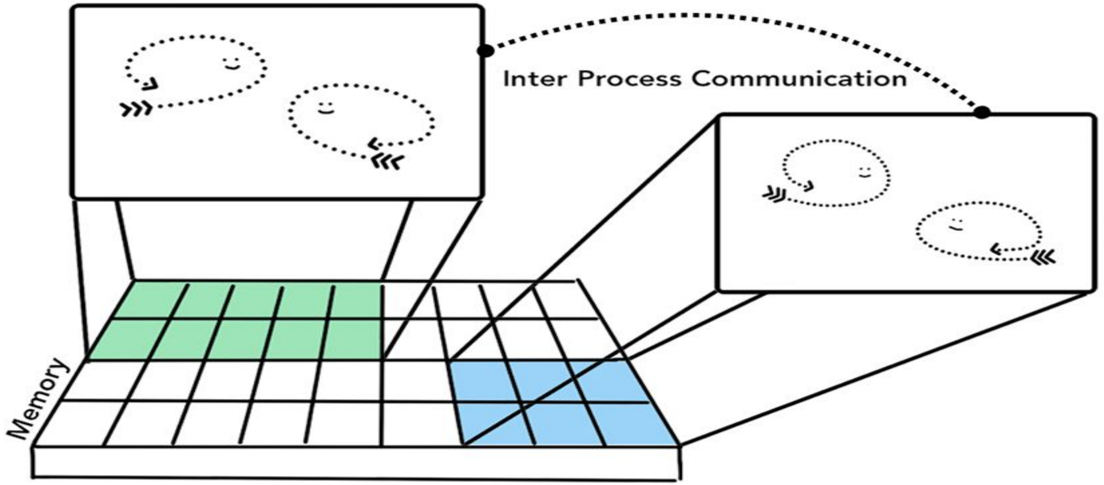
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Browser: A multi-threaded process



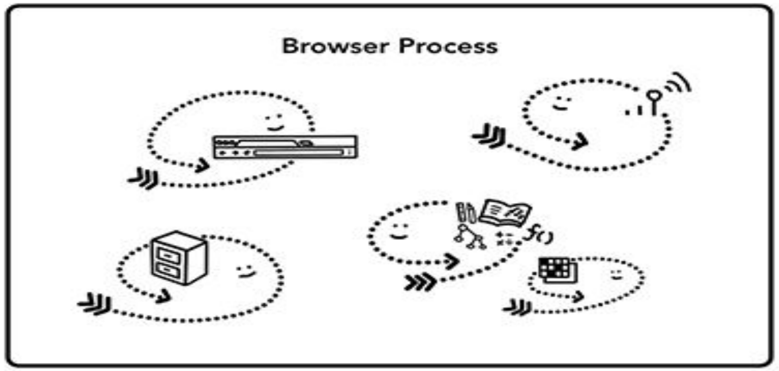
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

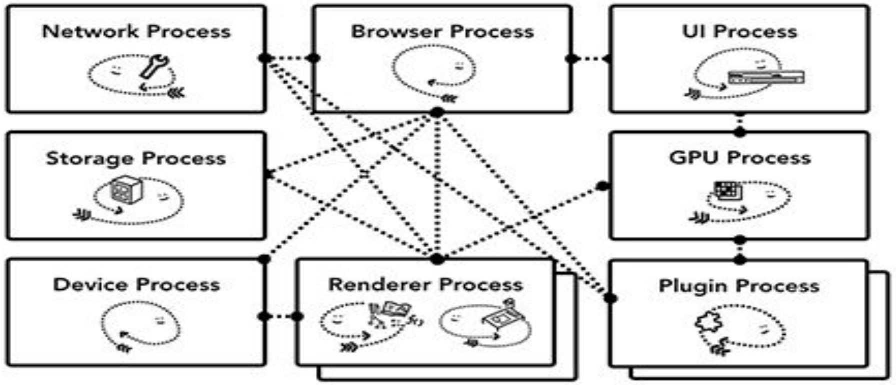
Multi-process browser with IPC



Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

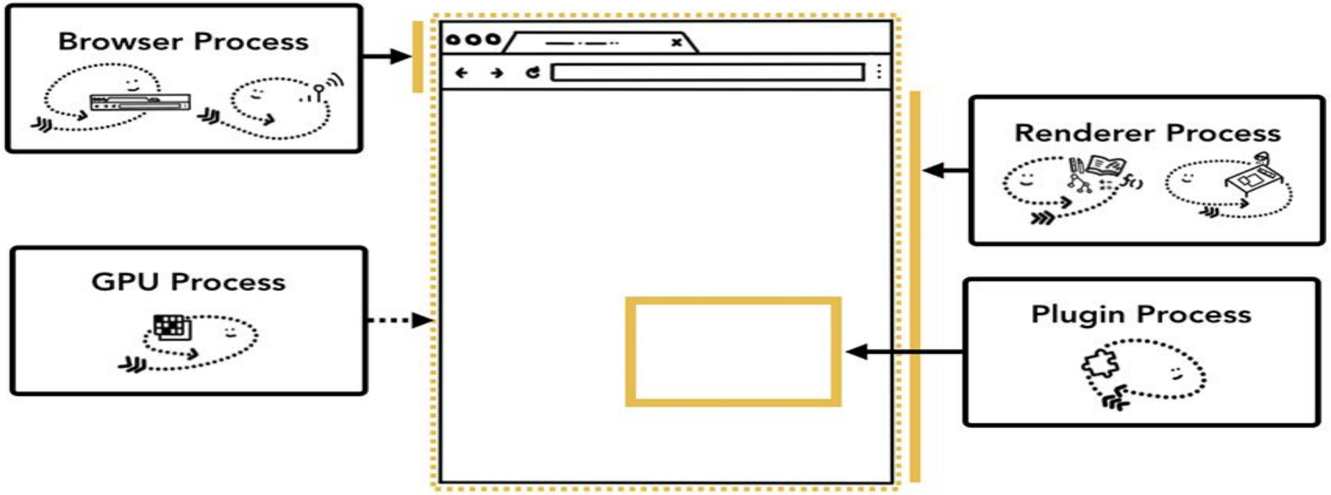
Browser Architectures





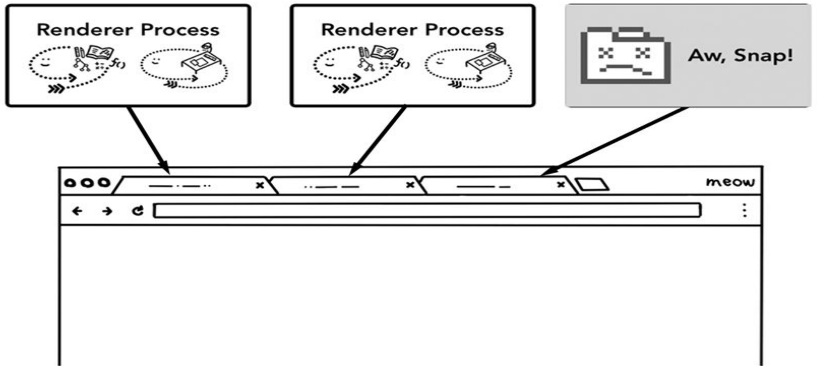
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

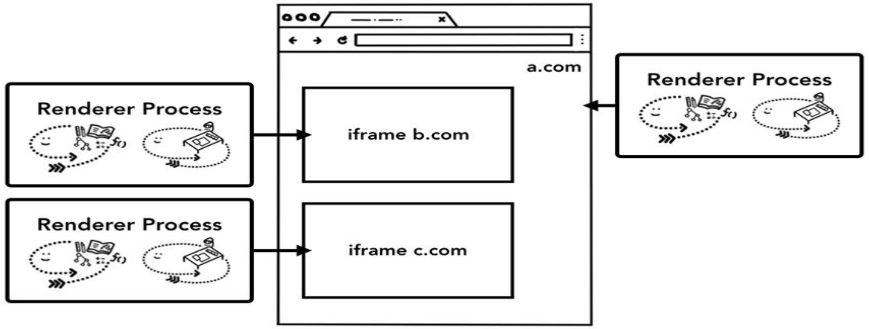
Service-based browser architecture



Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

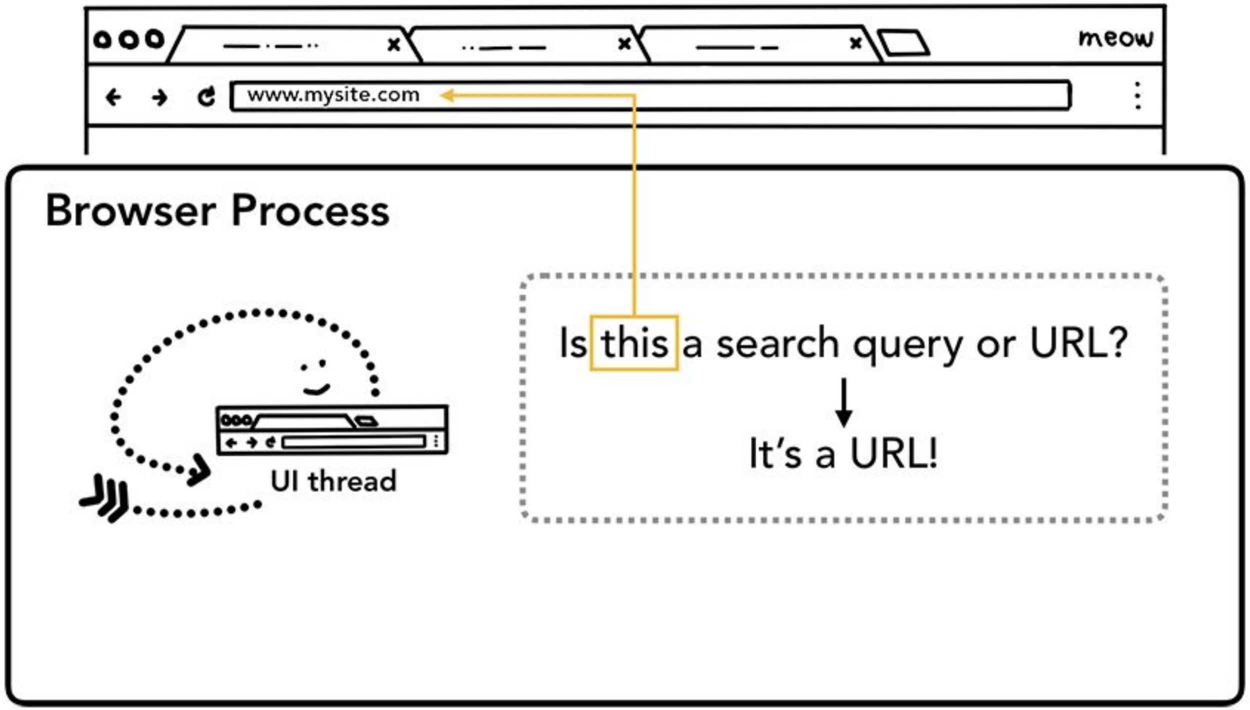
Service-based browser architecture





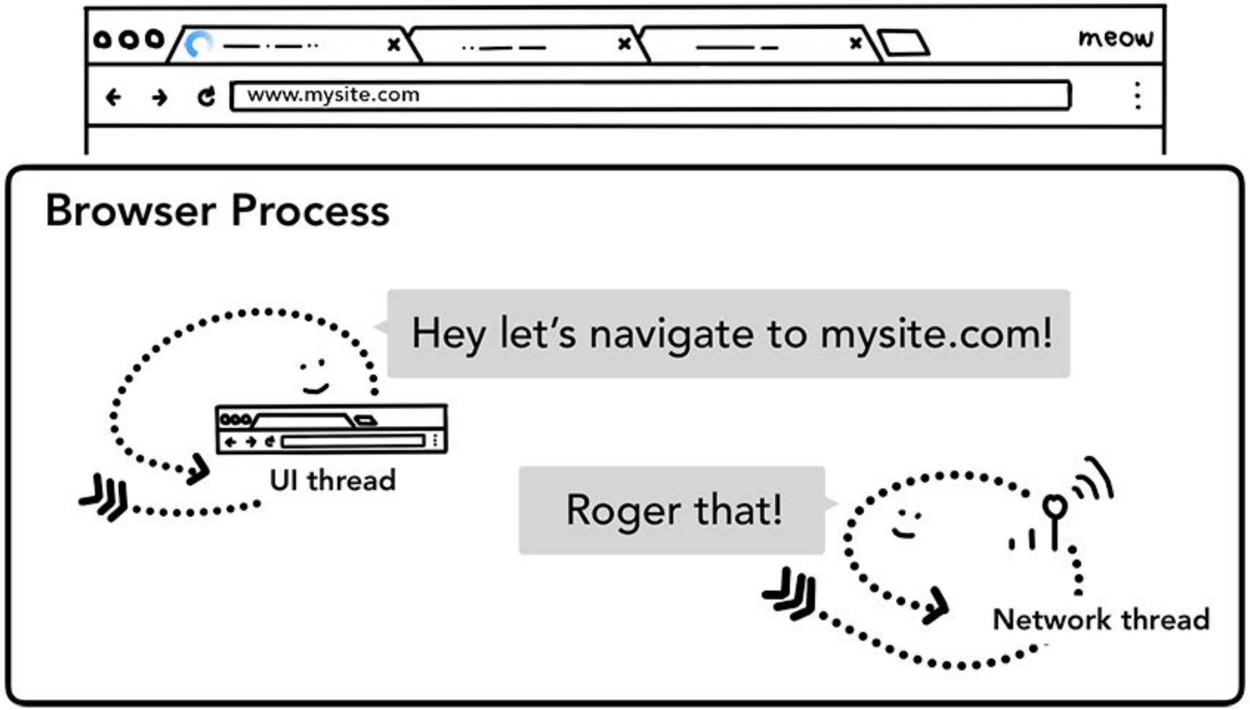
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Navigating to a web site uses service requests



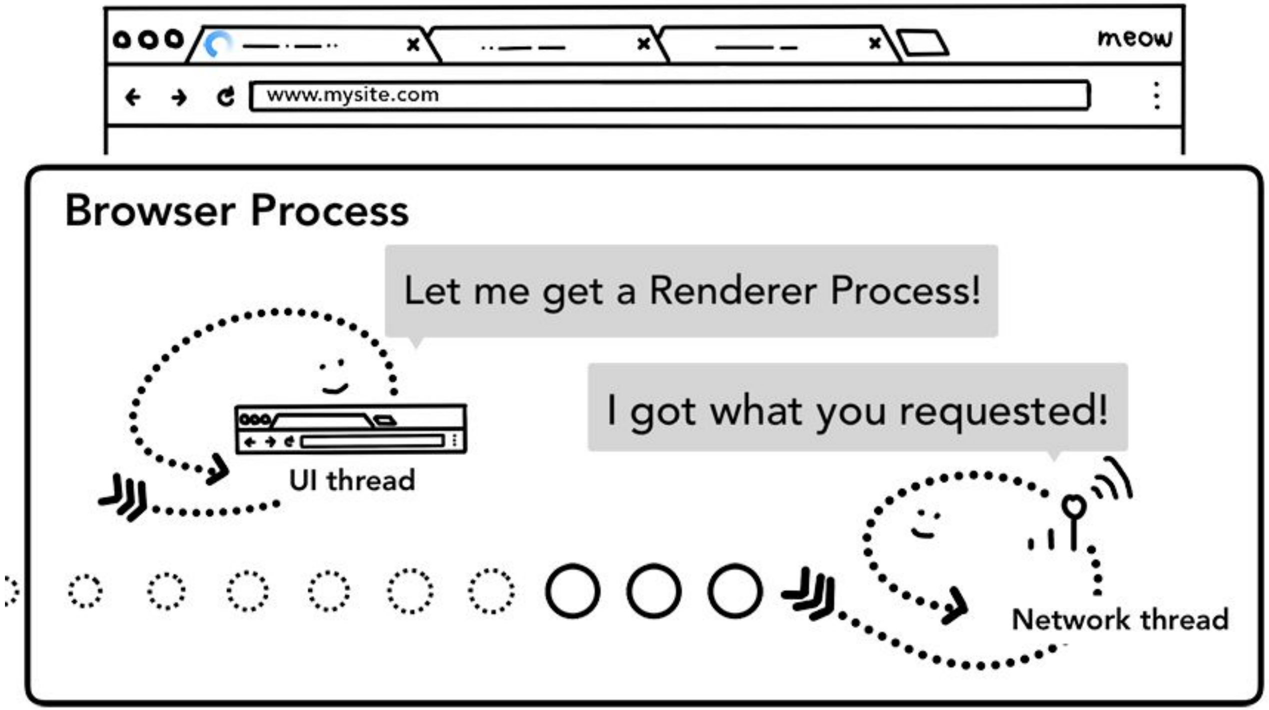
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Navigating to a web site uses service requests



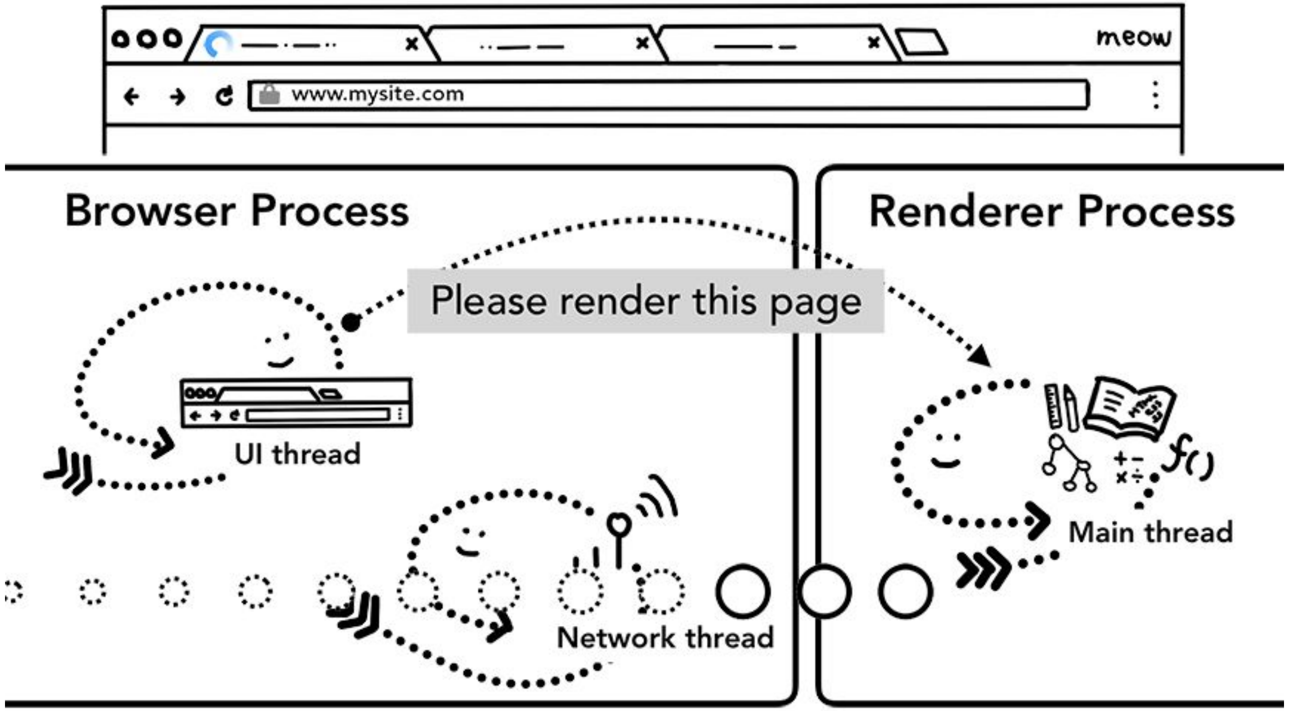
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Navigating to a web site uses service requests



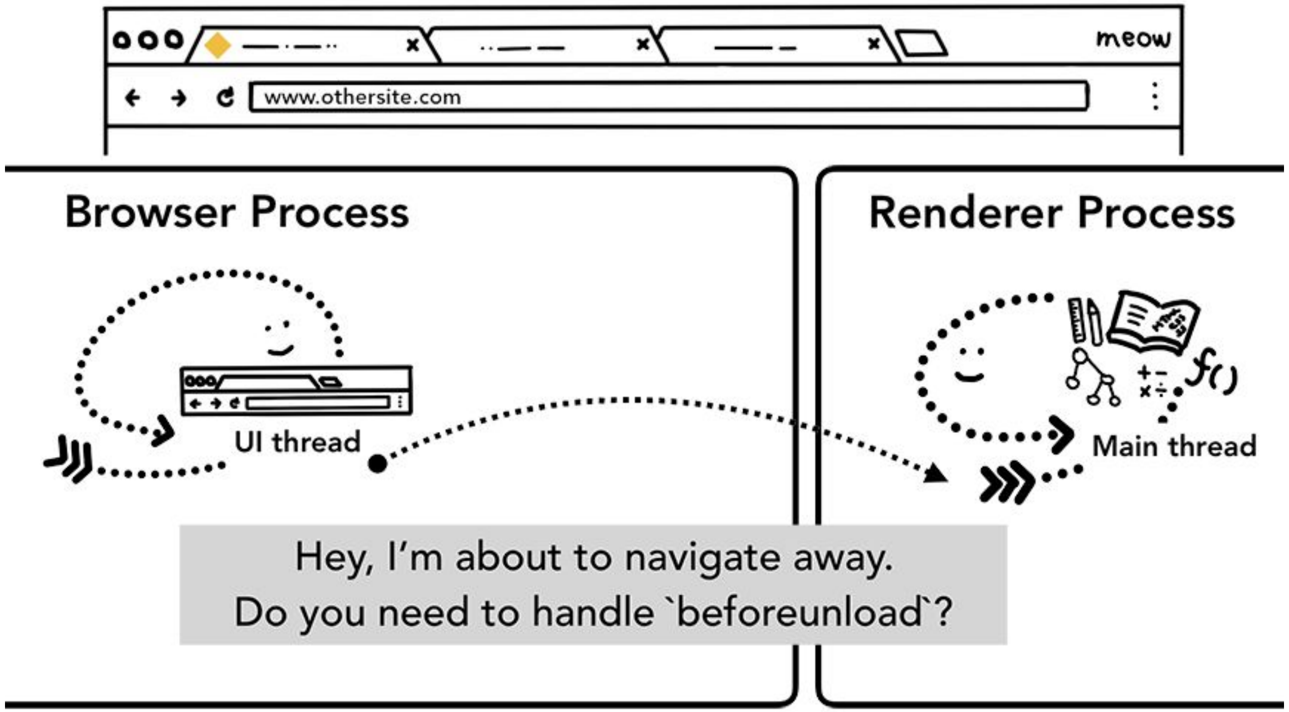
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Navigating to a web site uses service requests



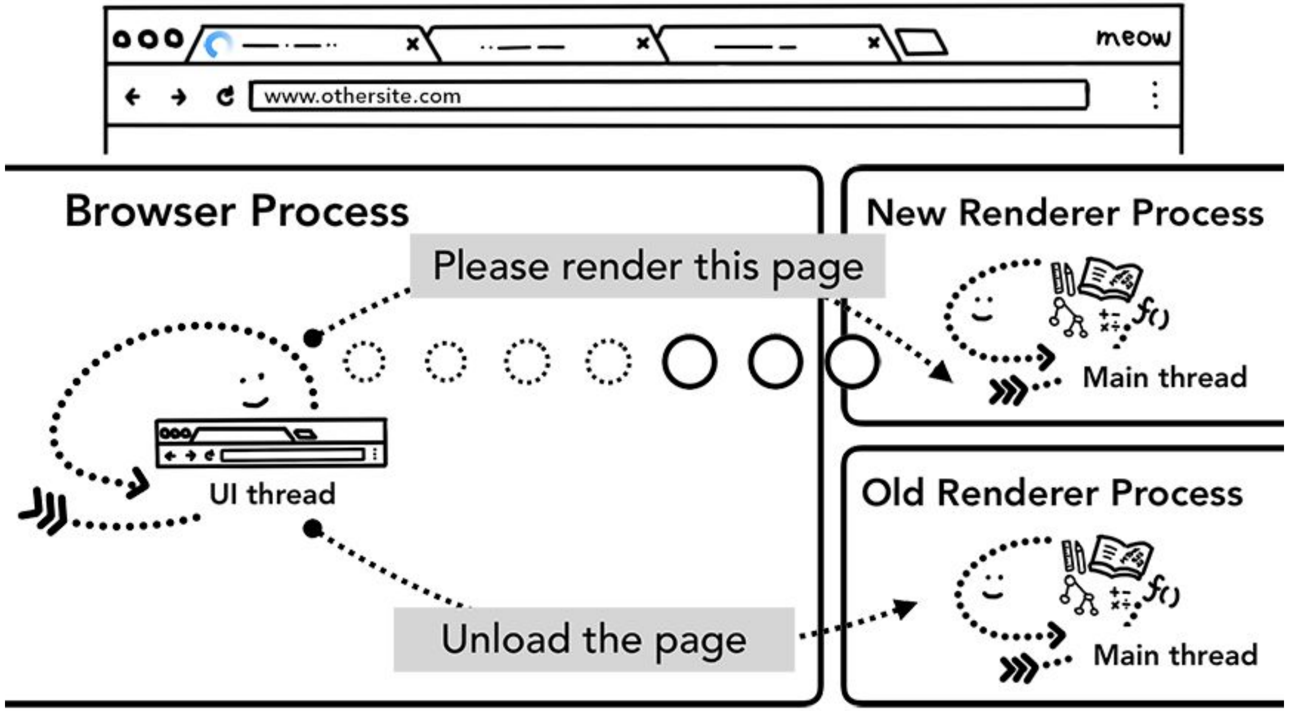
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Navigating to a web site uses service requests

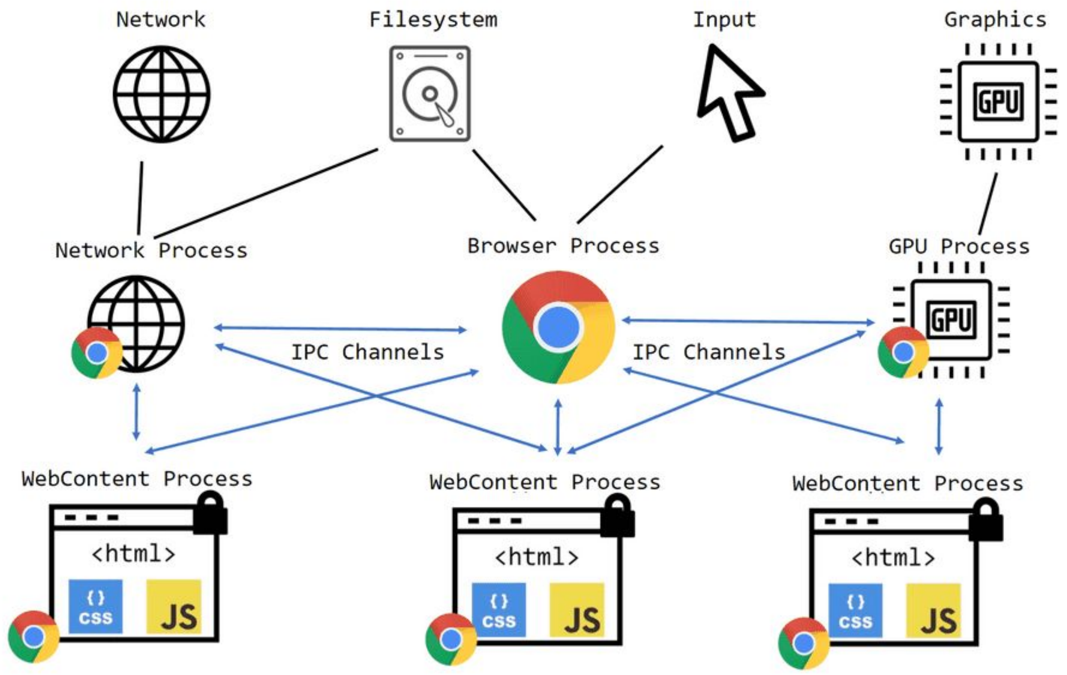


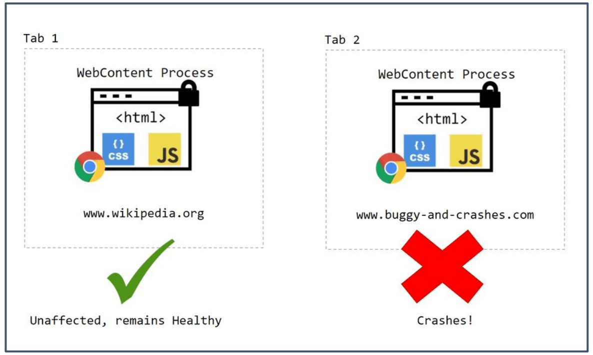
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

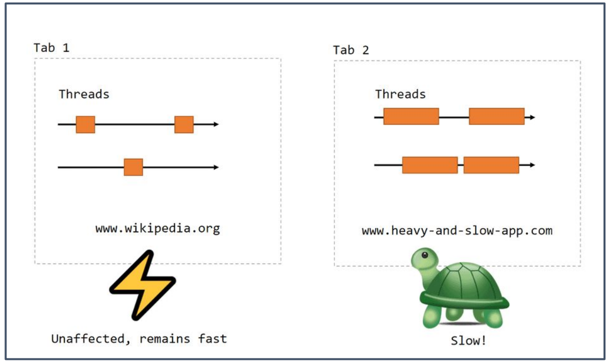
Navigating to a web site uses service requests



Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)







https://webperf.tips/tip/browser-process-model/

Service Oriented Architecture

- Ability to change components independently
- Independent processes (Isolation, Security)
- Focusing on doing one thing well

MICROSERVICES

Microservices
Microservices Everywhere

“*Small <u>autonomous</u> services that work well together*”

Sam Newman

Microservices

COMCAST

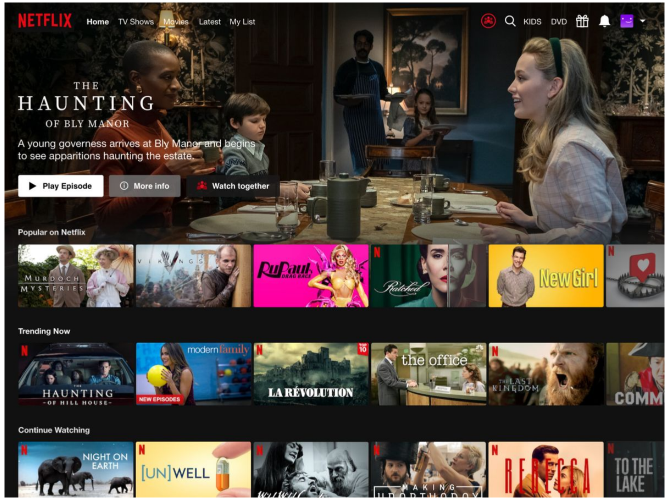
NETFLIX

ebay

UBER

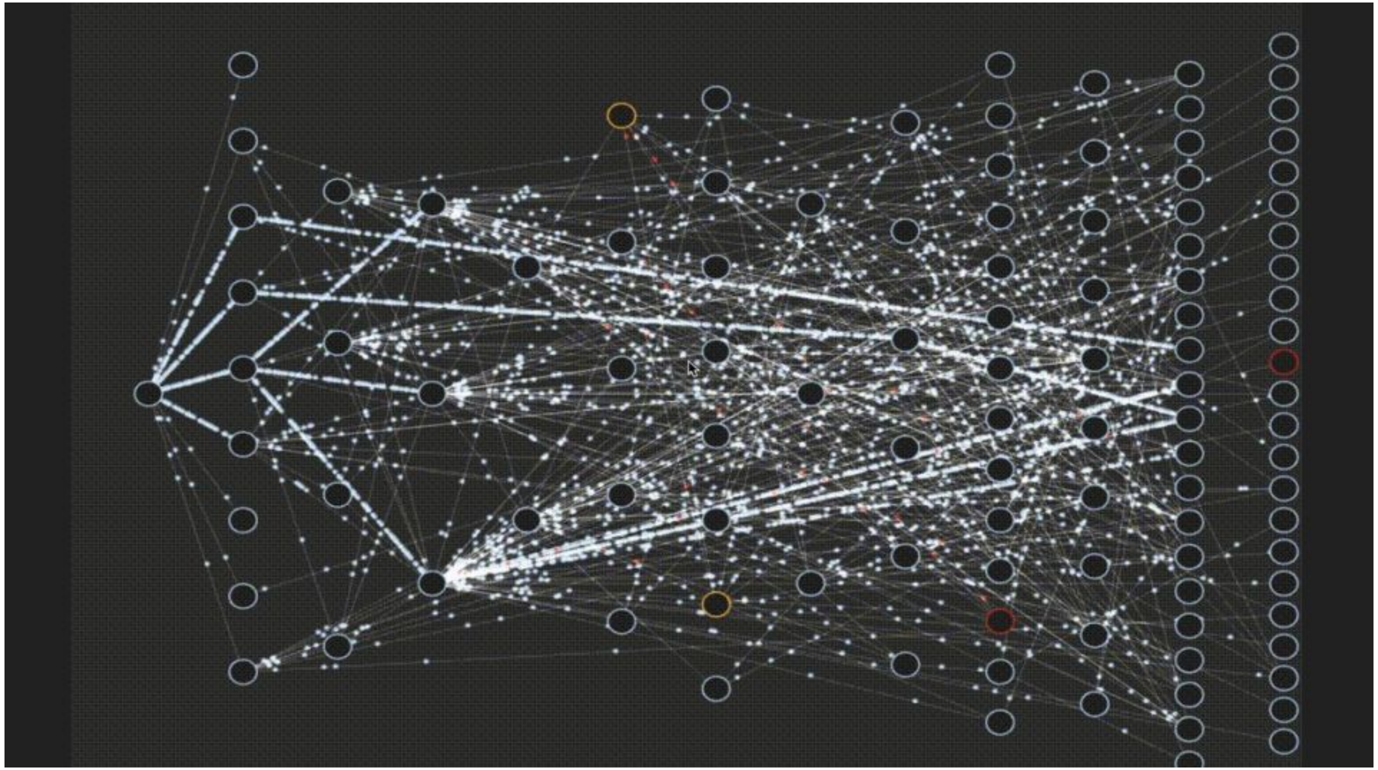
GROUPON®

Netflix Microservices – App Boot



- Recommendations
- Trending Now
- Continue Watching
- My List
- Metrics

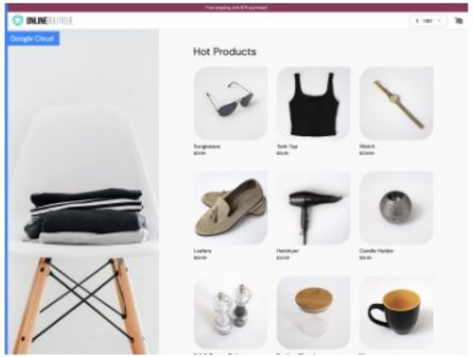
(as of 2016)



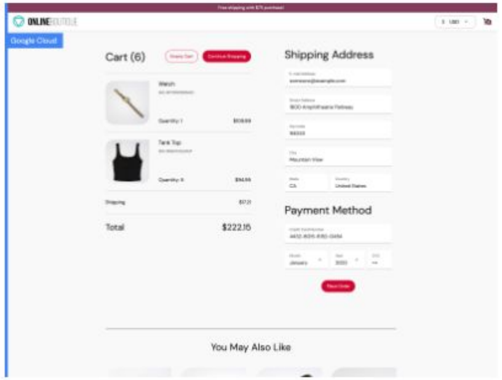
Activity: Mapping Interactions in a Microservices Architecture

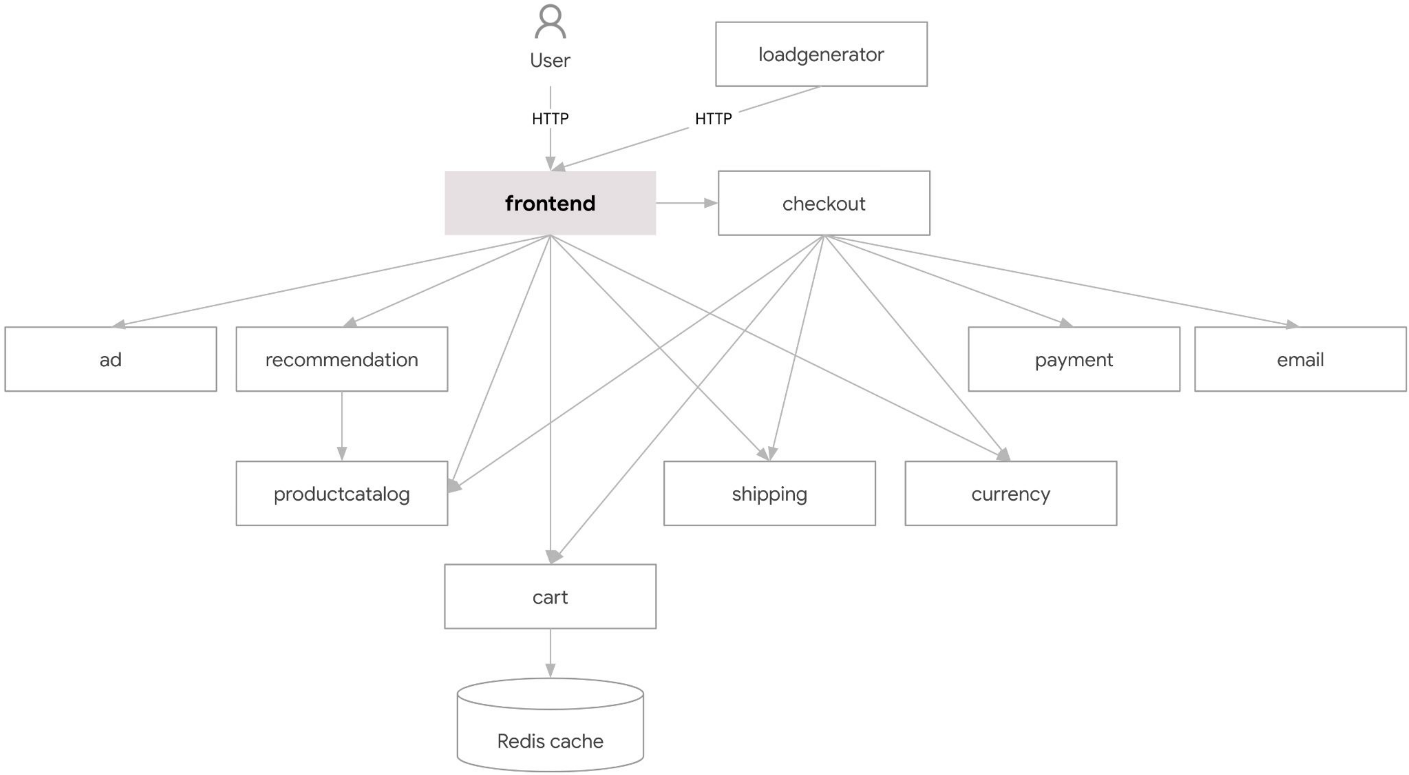
Service	Description
frontend	Exposes an HTTP server to serve the website. Does not require signup/login and generates session IDs for all users automatically.
cartservice	Stores the items in the user's shopping cart in Redis and retrieves it.
productcatalogservice	Provides the list of products from a JSON file and ability to search products and get individual products.
currencyservice	Converts one money amount to another currency. Uses real values fetched from European Central Bank.
paymentservice	Charges the given credit card info (mock) with the given amount and returns a transaction ID.
shippingservice	Gives shipping cost estimates based on the shopping cart. Ships items to the given address (mock)
checkoutservice	Retrieves user cart, prepares order and orchestrates the payment, shipping and the email notification.

Home Page



Checkout Screen





Monoliths vs Microservices

Activity: Discussion

What are the consequences of this architecture? On:

- Scalability
- Reliability
- Performance
- Development
- Maintainability
- Testability
- Ownership

Advantages of Microservices

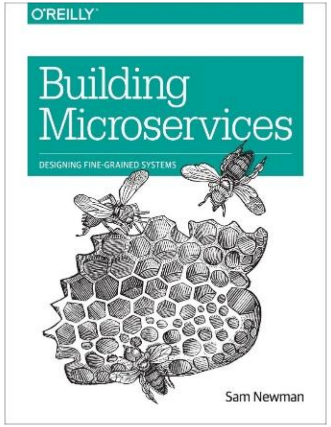
- Better alignment with the organization
- Ship features faster and safer
- Scalability
- Target security concerns
- Allow the interplay of different systems and languages, no commitment to a single technology stack
- Easily deployable and replicable
- Embrace uncertainty, automation, and faults

Microservice challenges

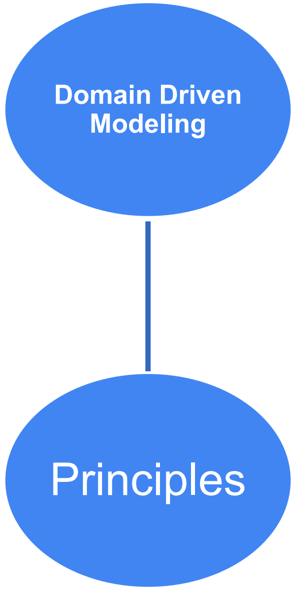
- Too many choices
- Delay between investment and payback
- Complexities of distributed systems
  - network latency, faults, inconsistencies
  - testing challenges
- Monitoring is more complex
- More system states
- More points of failure
- Operational complexity
- Frequently adopted by breaking down a monolithic application

MICROSERVICES: PRINCIPLES

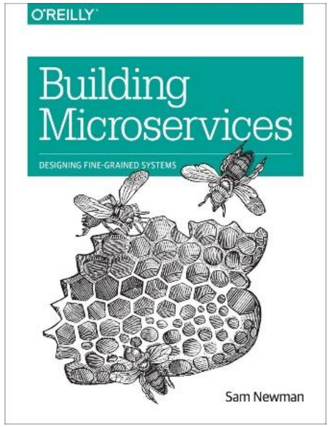
Principles

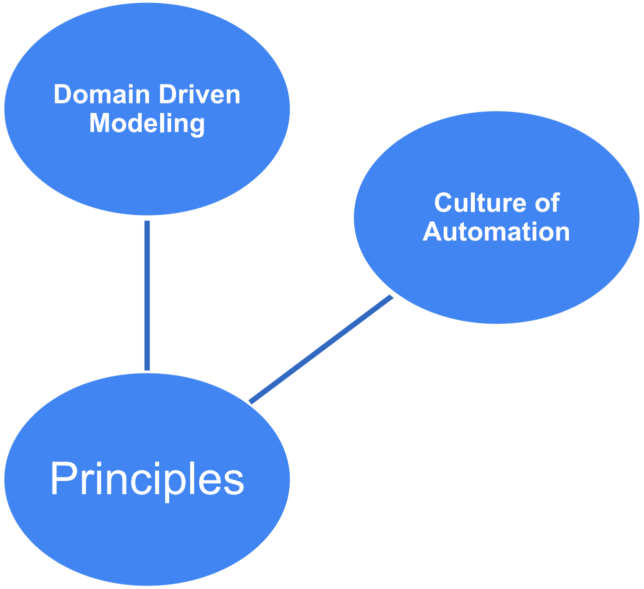


Sam Newman’s Principles of Microservices

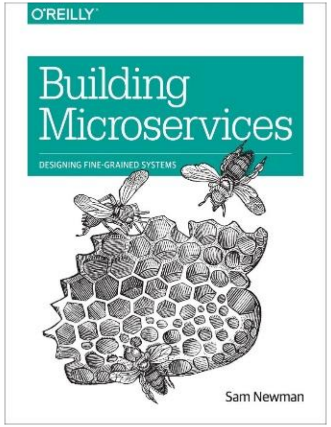


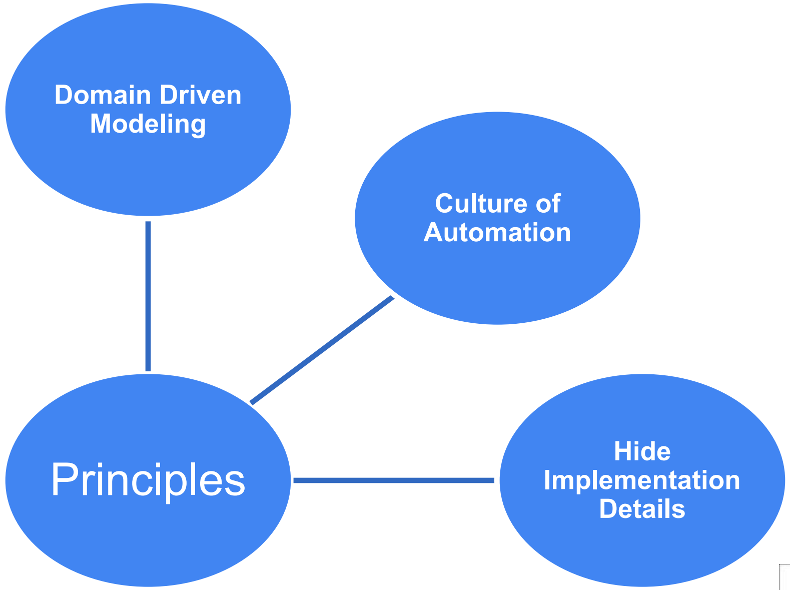
Sam Newman’s Principles of Microservices

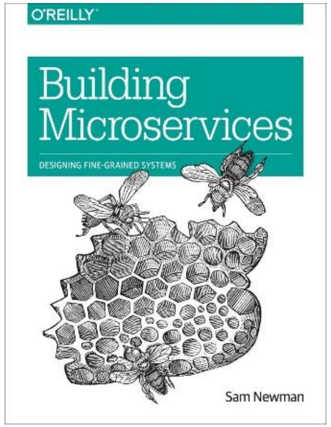




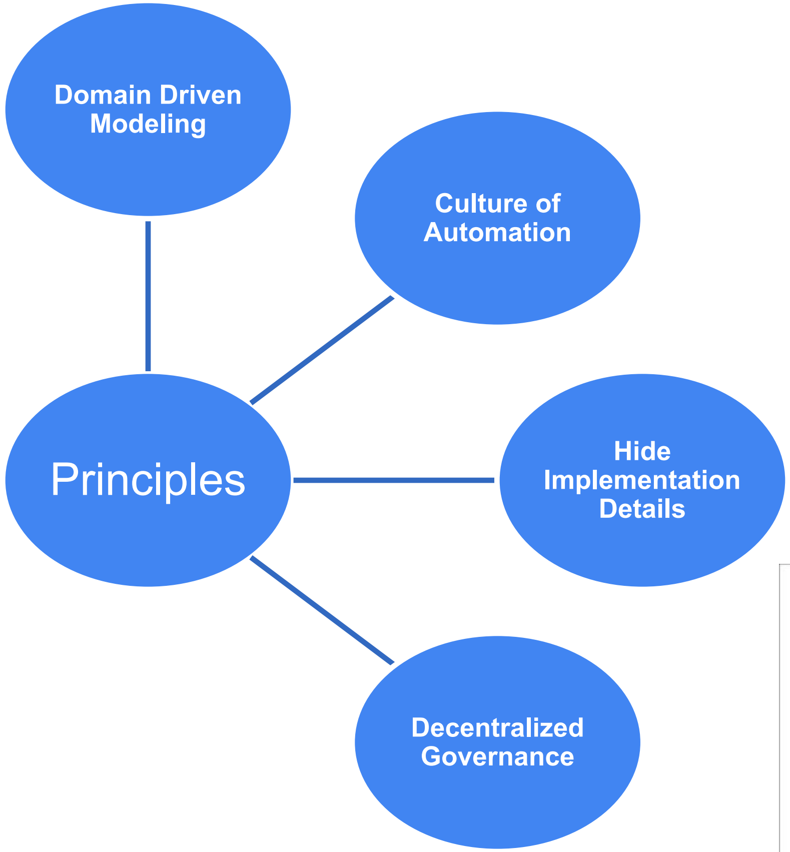
Sam Newman’s Principles of Microservices



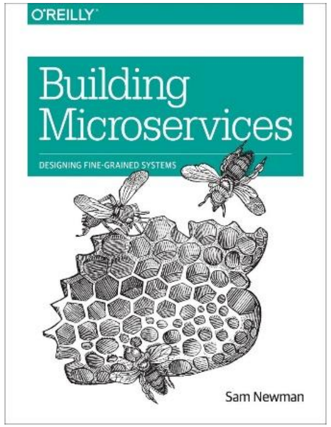


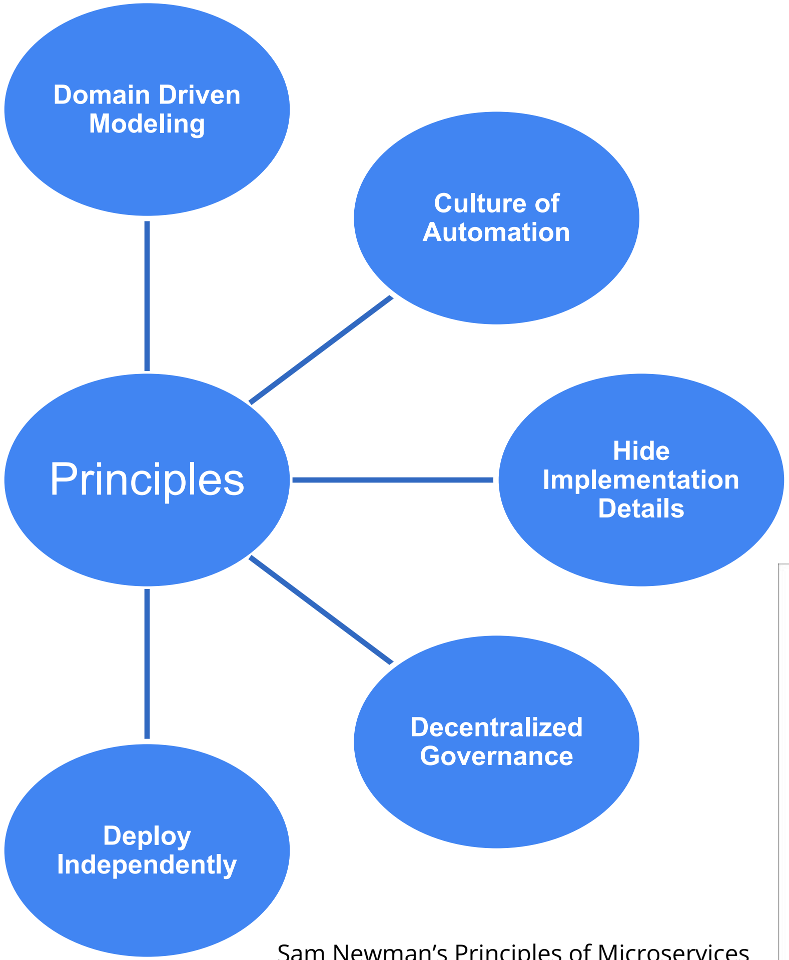


Sam Newman's Principles of Microservices

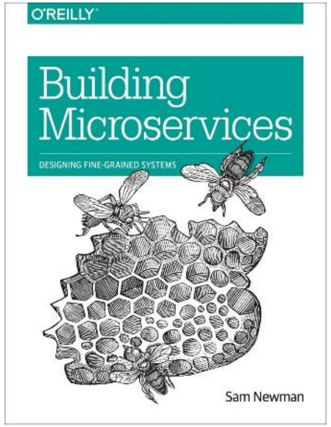


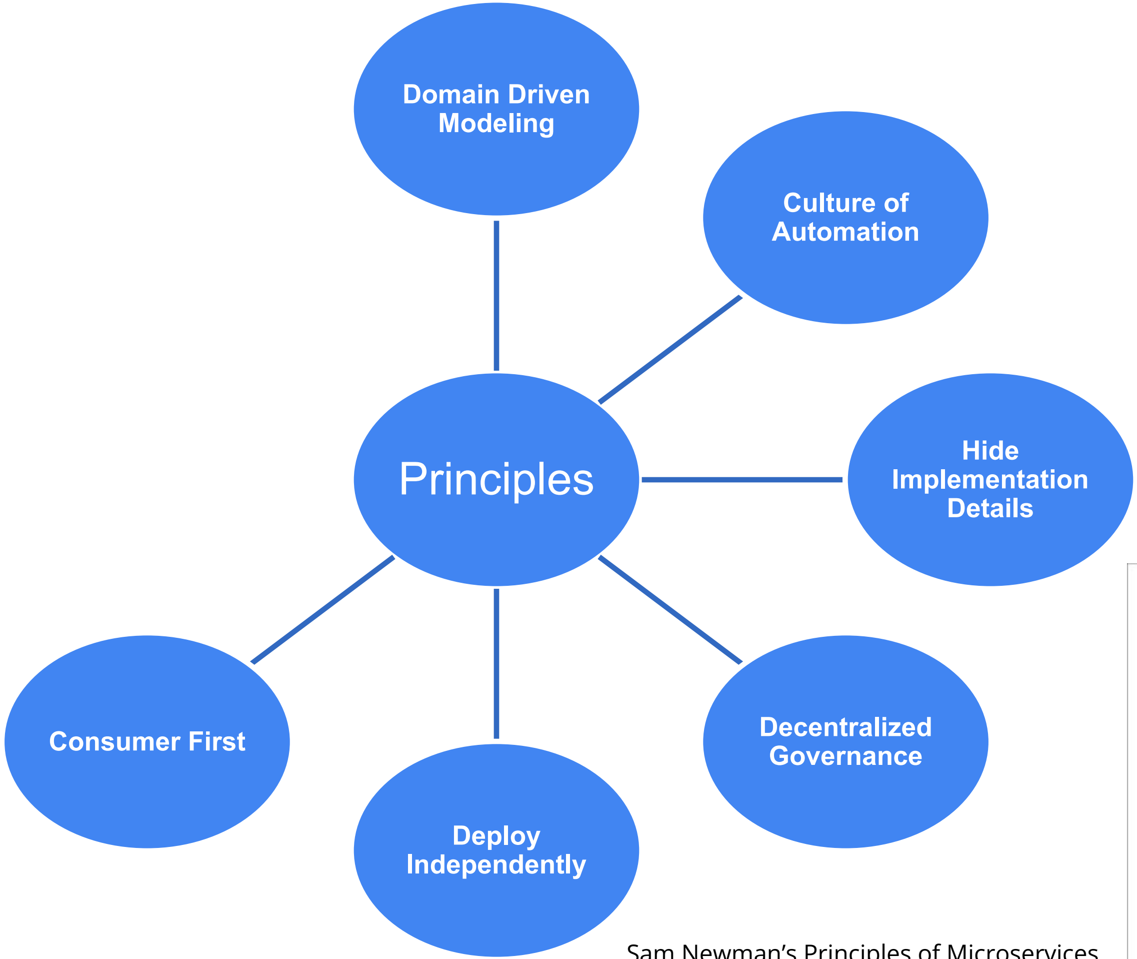
Sam Newman's Principles of Microservices



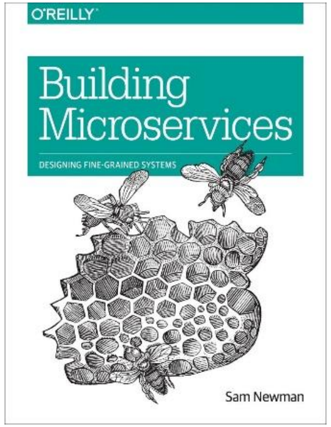


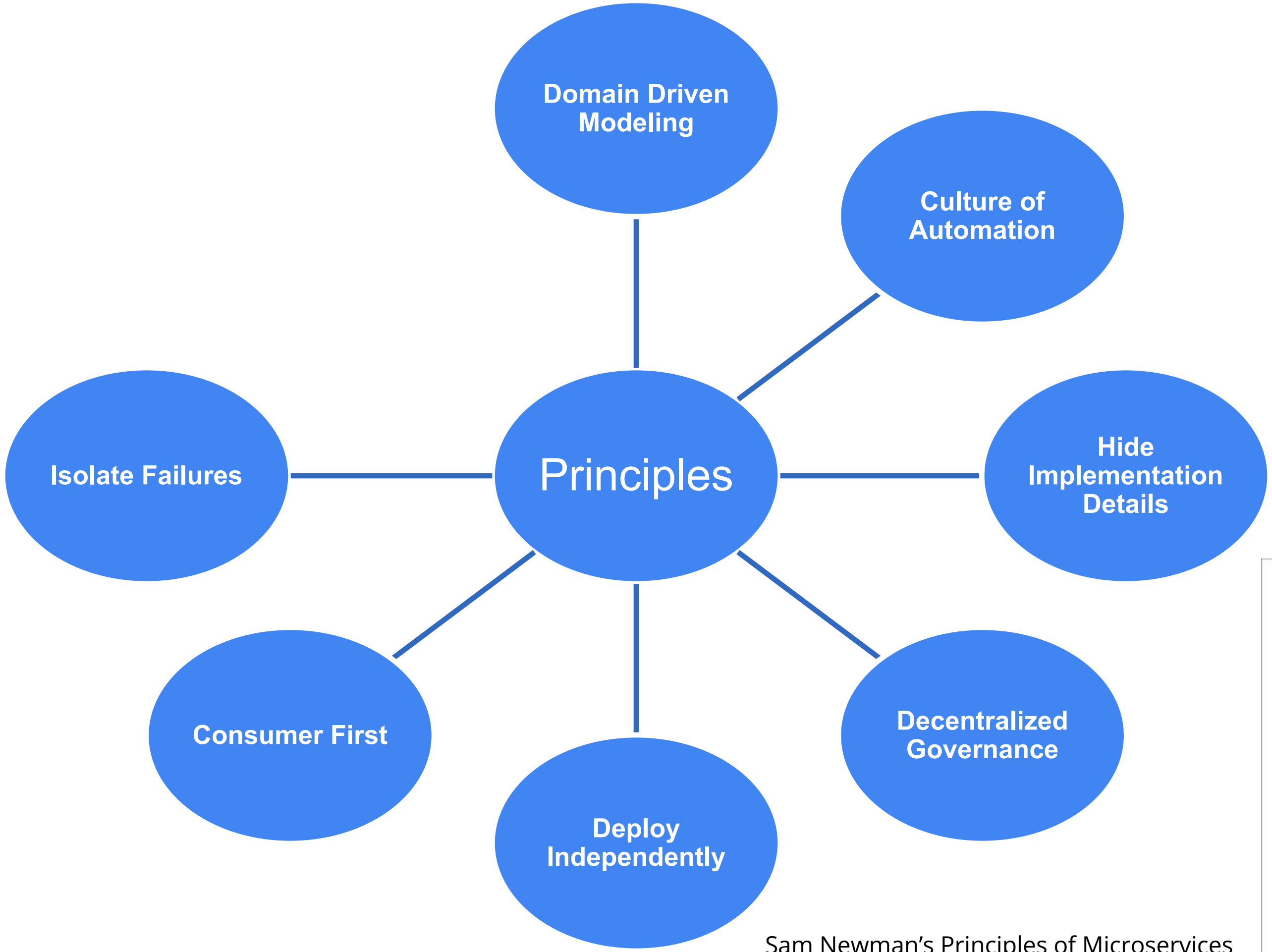
Sam Newman's Principles of Microservices

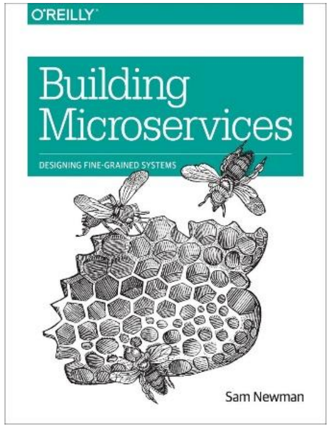




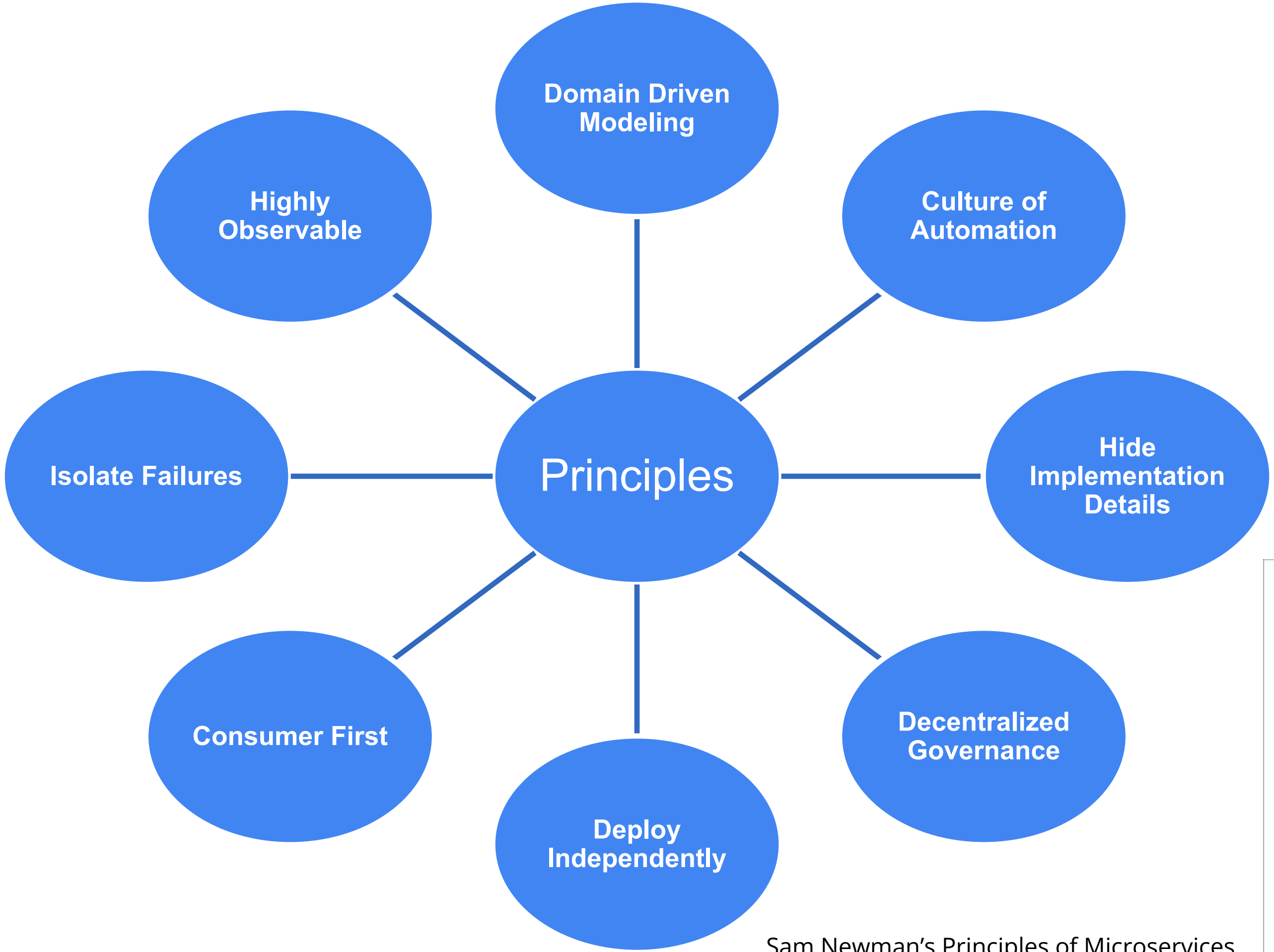
Sam Newman's Principles of Microservices



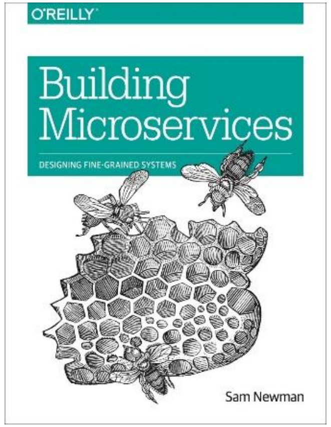




Sam Newman's Principles of Microservices

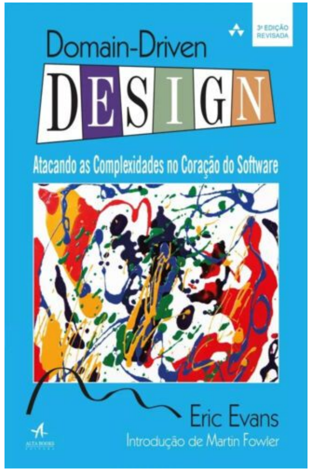


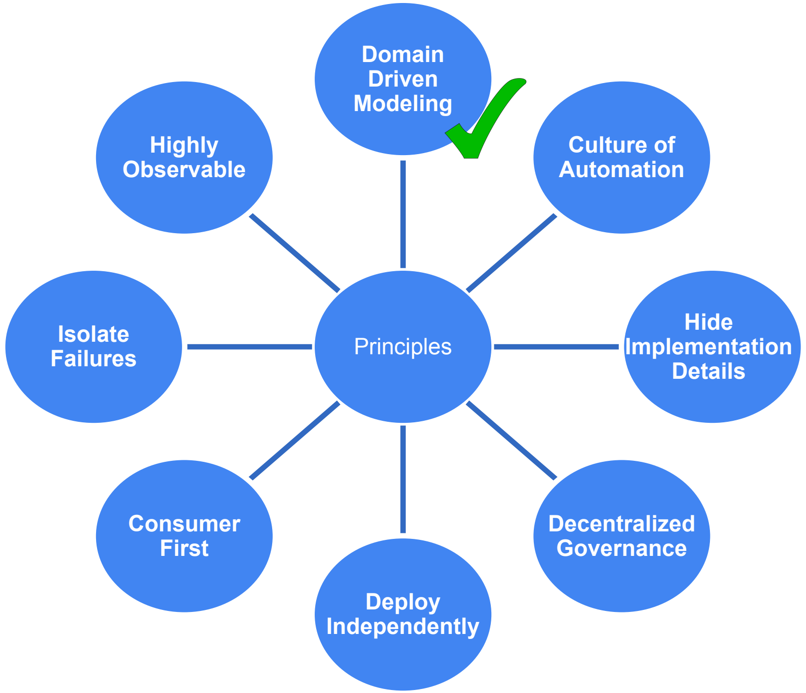
Sam Newman's Principles of Microservices



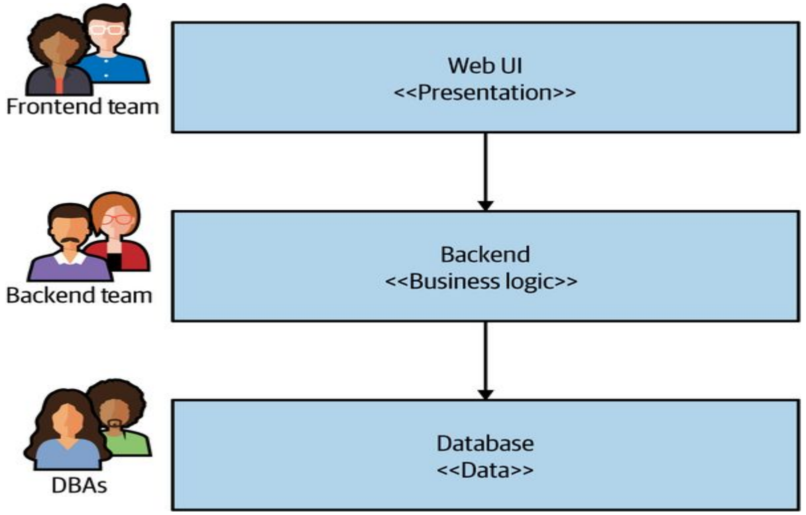
Principle 1: Domain-driven modeling

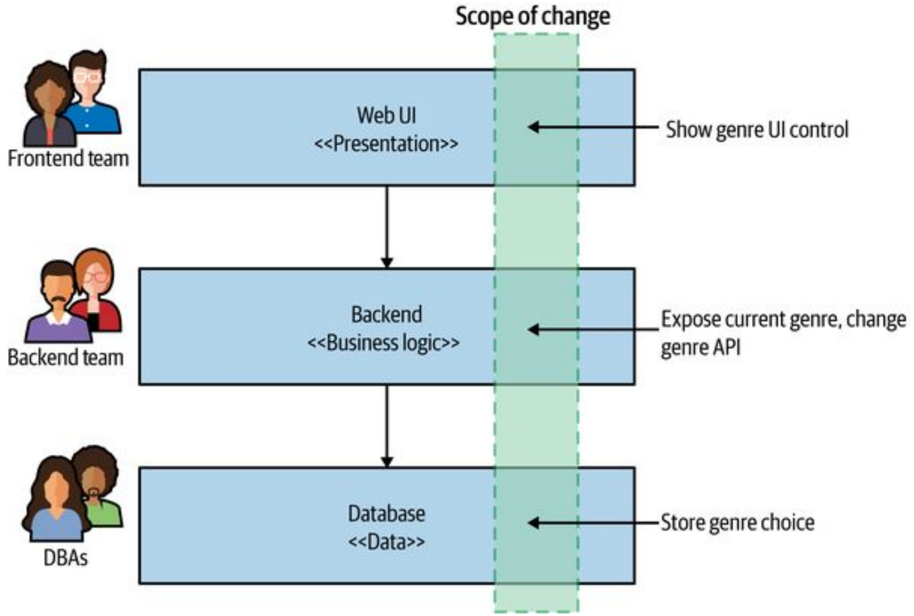
- Model services around business capabilities


Domain-Driven
DESIGN
Atacando as Complexidades no Coração do Software
Eric Evans
Introdução de Martin Fowler

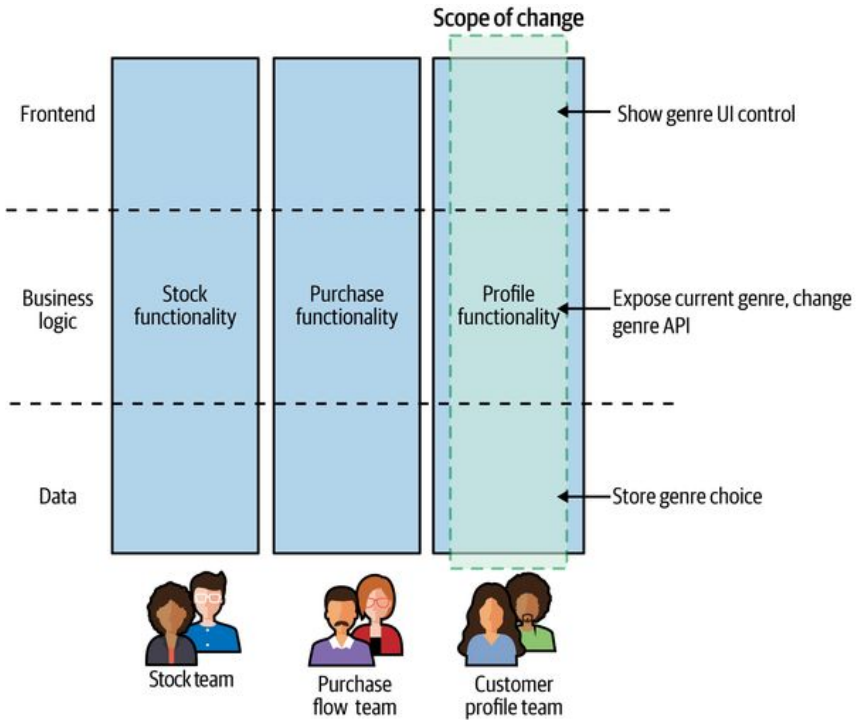


Principle 1: Domain-driven modeling



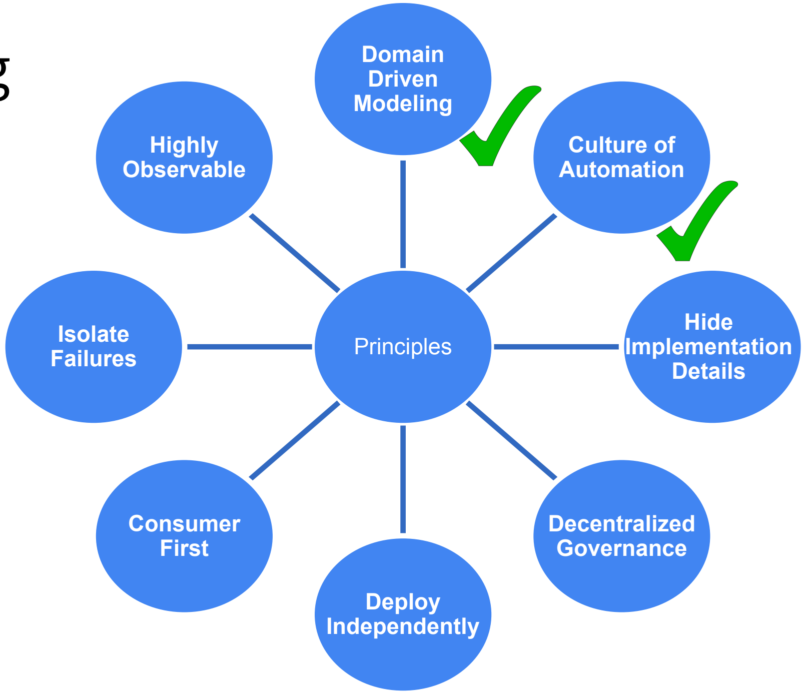


Principle 1: Domain-driven modeling



Principle 2: Culture of Automation

- API-Driven Machine Provisioning
- Continuous Delivery
- Automated Testing



API-Driven Machine Provisioning

Example: Infrastructure as code (IaC)

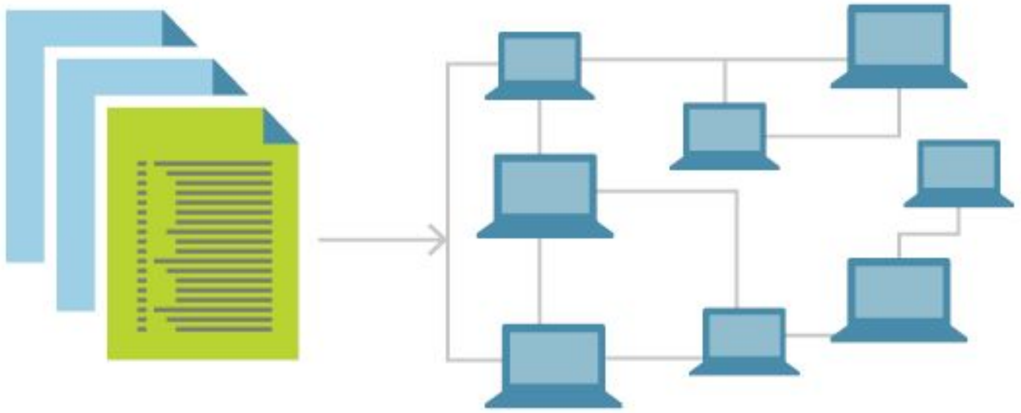


Image source: https://learn.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code

Continuous Delivery

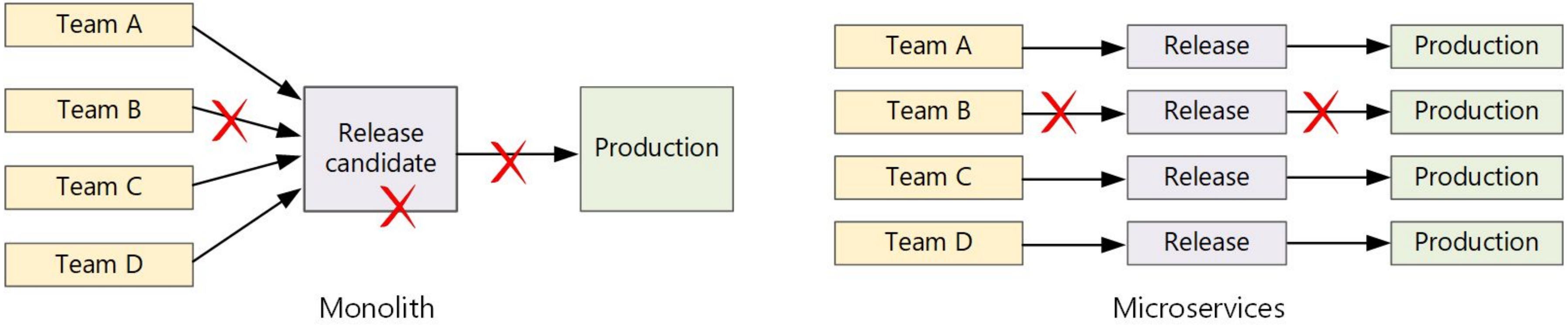
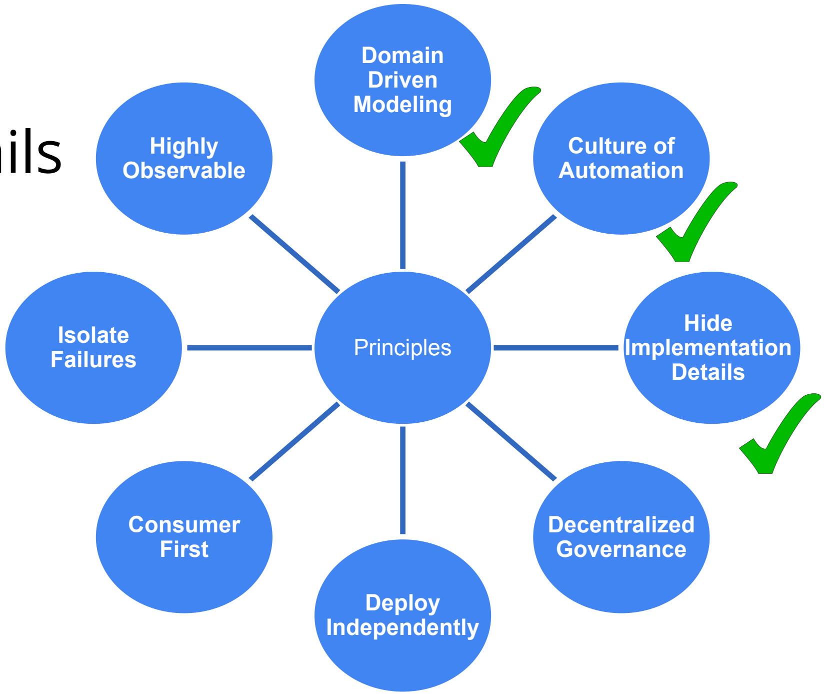


Image Source: https://learn.microsoft.com/en-us/azure/architecture/microservices/ci-cd

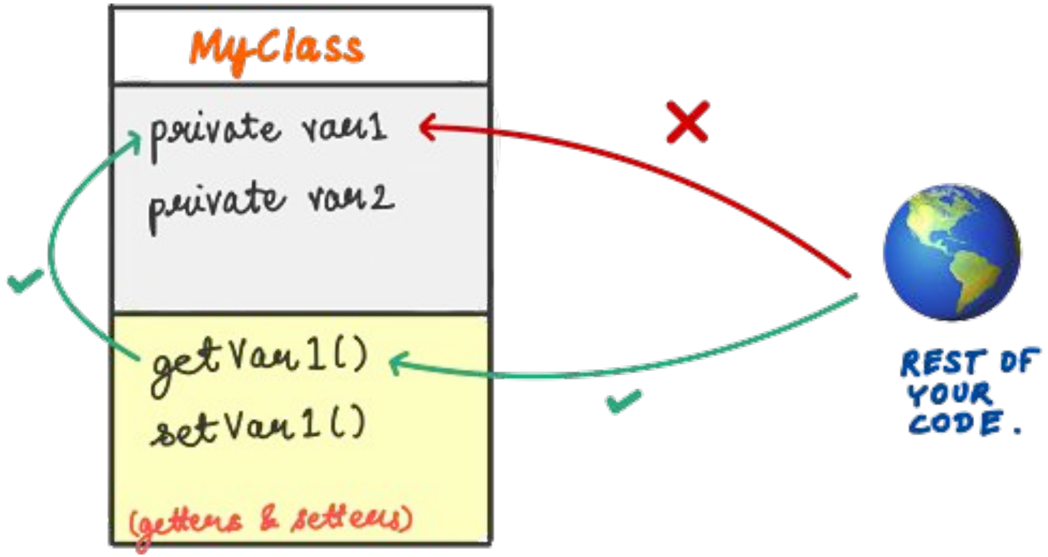
Principle 3: Hide implementation details

- Design carefully your APIs
- It's easier to expose some details later than hide them
- Do not share your database!

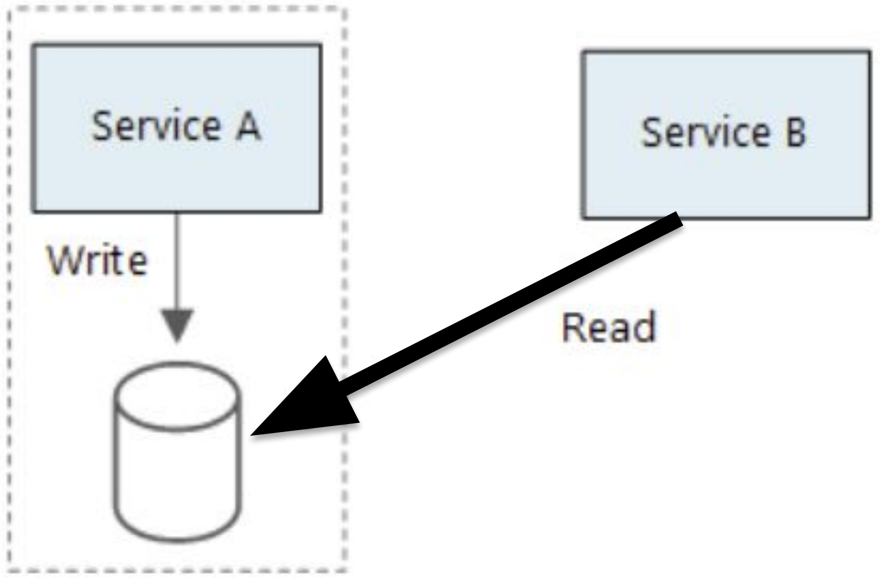


Principle 3: Hide implementation details

Recall: Encapsulation in OOP

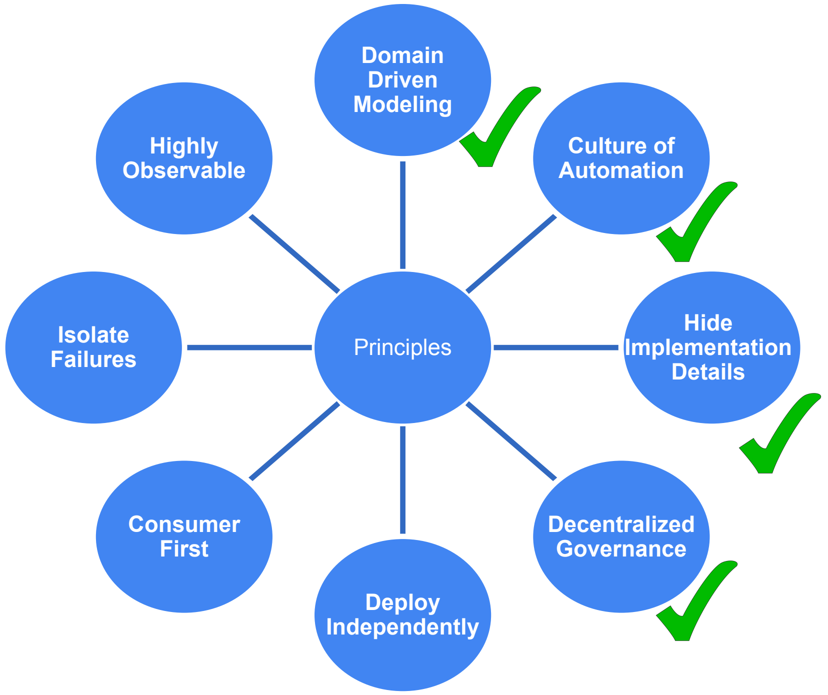


Sharing database: Anti-pattern

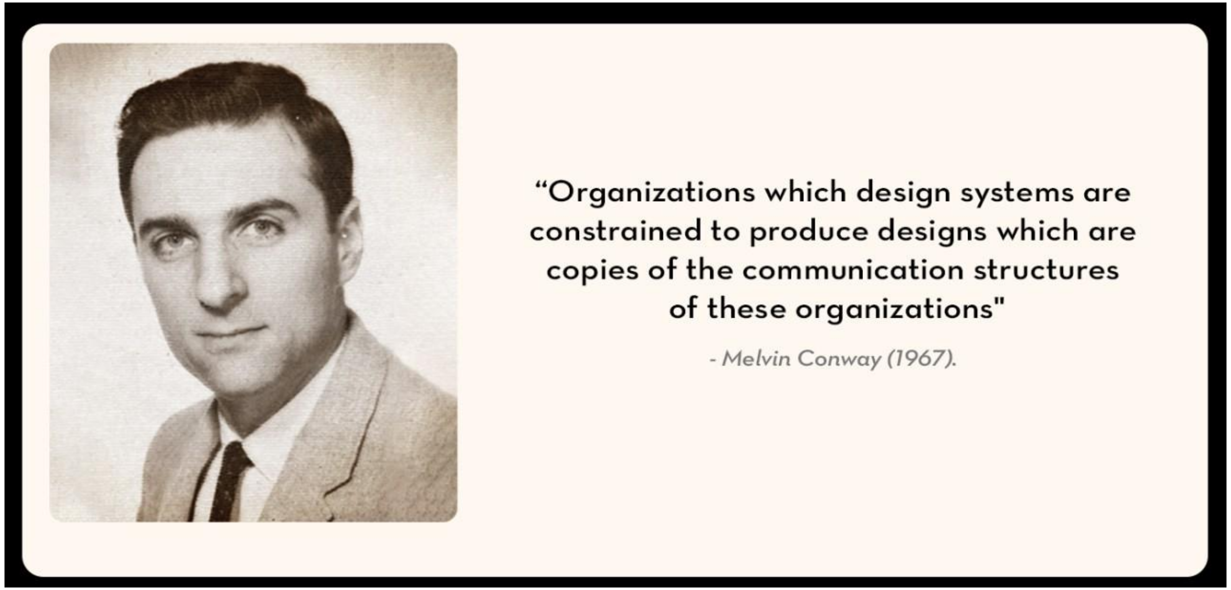


Principle 4: Decentralized Governance

- Mind Conway's Law
- You Build It, You Run It
- Embrace team autonomy
- Internal Open Source Model



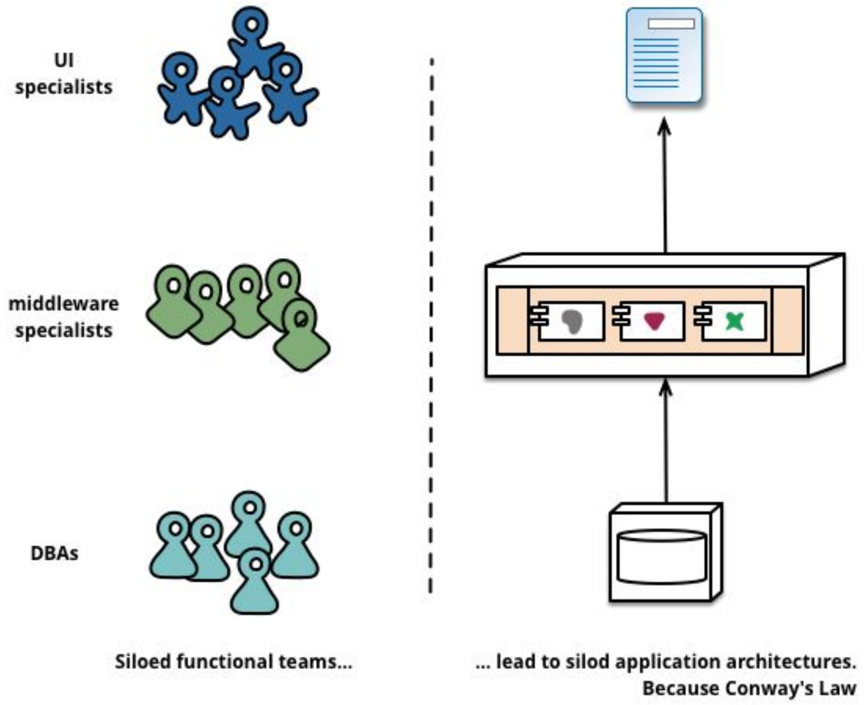
Mind Conway’s Law

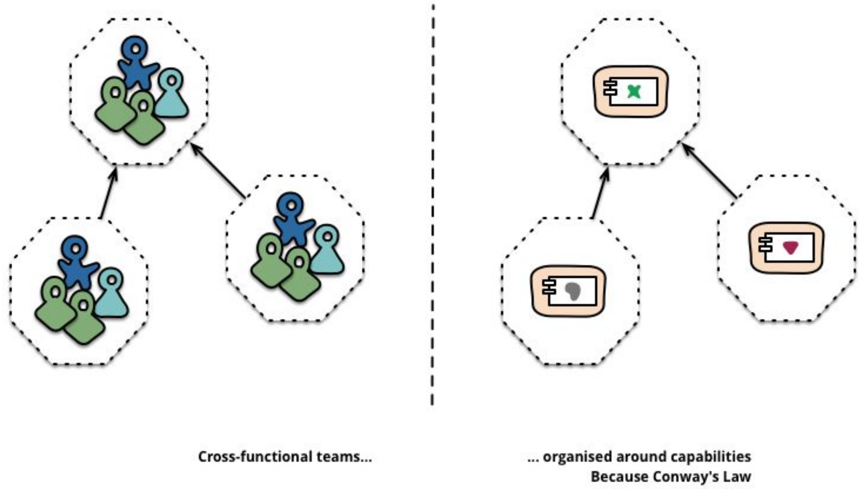


“Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations"

- Melvin Conway (1967).

Mind Conway's Law


UI specialists
middleware specialists
DBAs
Siloed functional teams...
... lead to silod application architectures. Because Conway's Law


Cross-functional teams...
... organised around capabilities Because Conway's Law

"Products" not "Projects"

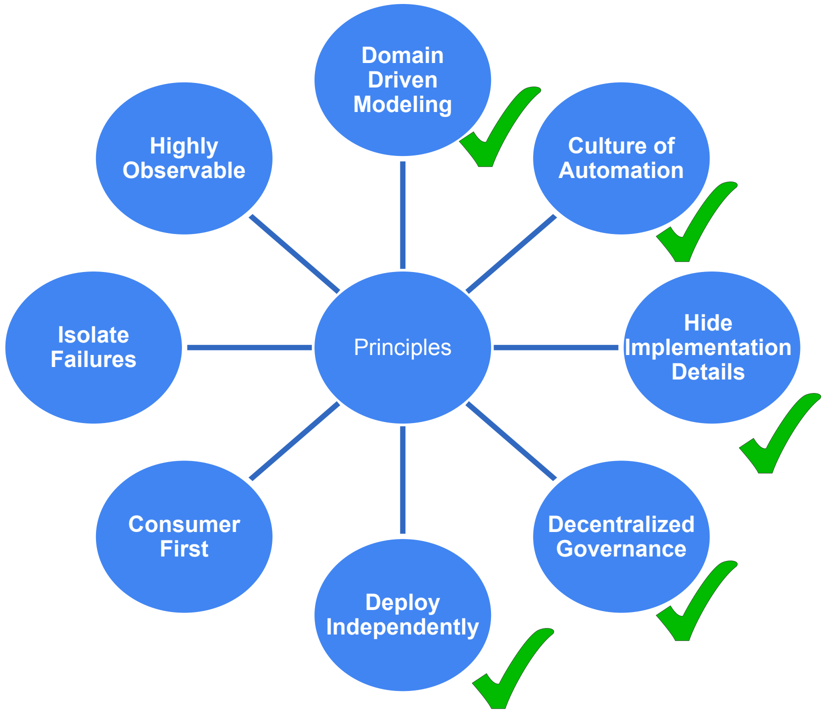
YOU BUILD IT YOU RUN ~~AWAY~~ IT

"The traditional model is that you take your software to the wall that separates development and operations, and throw it over and then forget about it. Not at Amazon. You build it, you run it. This brings developers into contact with the day-to-day operation of their software. It also brings them into day-to-day contact with the customer. This customer feedback loop is essential for improving the quality of the service."

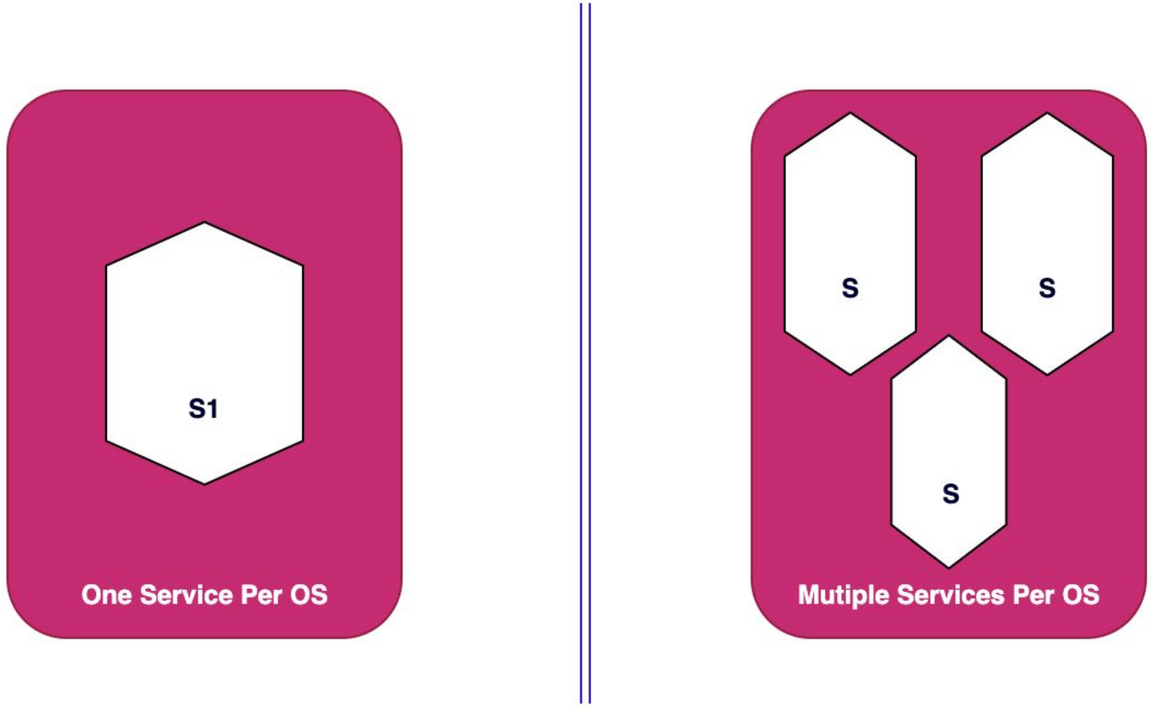
-- Werner Vogels in "A conversation with Werner Vogels" in ACM Queue, May 2006

Principle 5: Deploy Independently

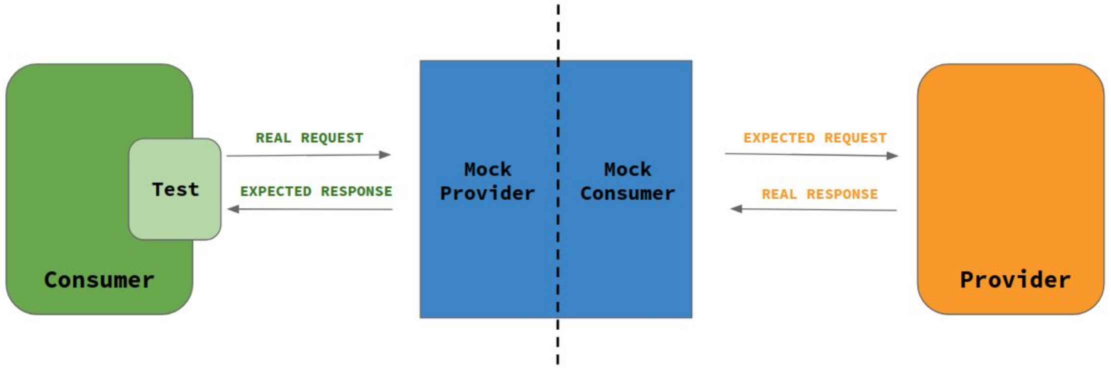
- One Service Per OS
- Consumer-Driven Contracts
- Multiple coexisting versions



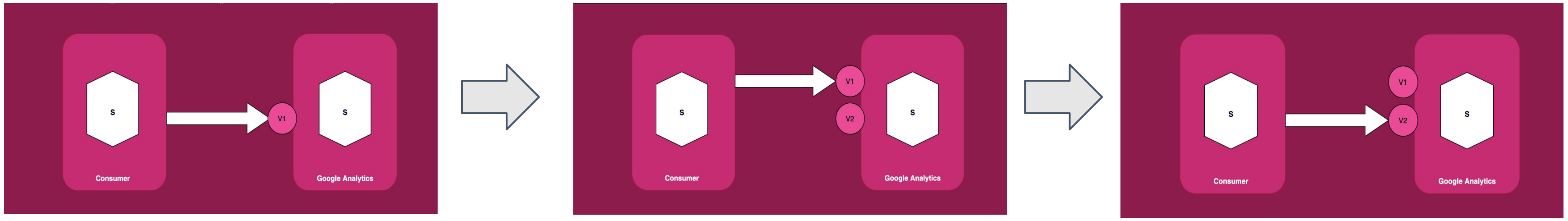
One Service Per OS



Consumer-Driven Contracts

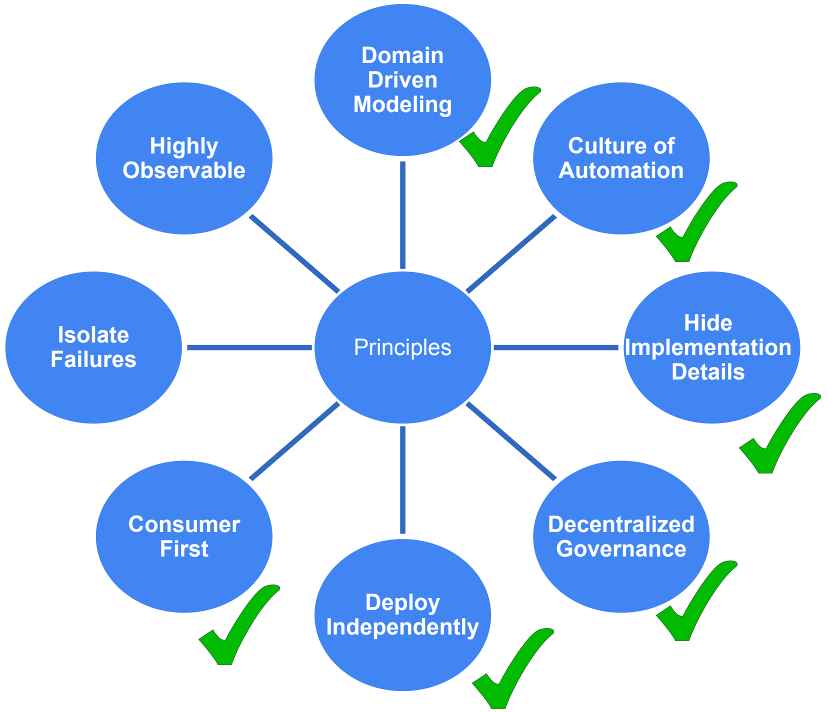


Multiple coexisting versions



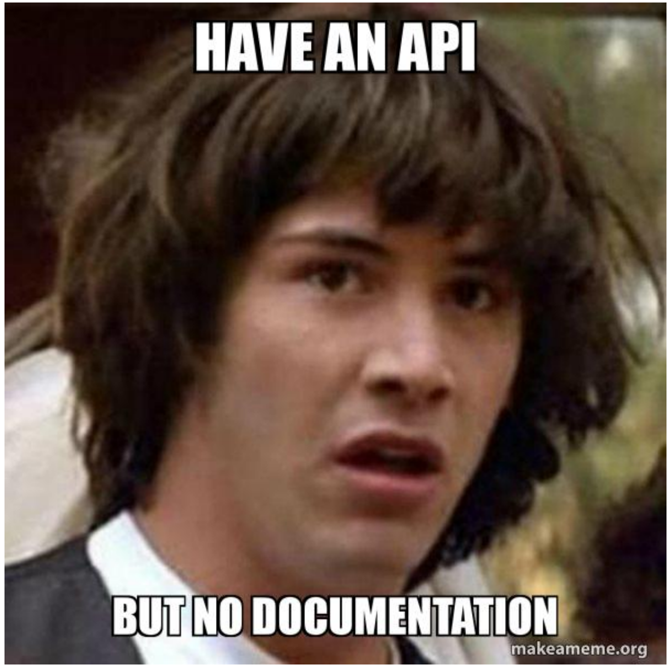
Principle 6: Consumer First

- Encourage conversations
- API Documentation
- Service Discovery



Encourage conversations

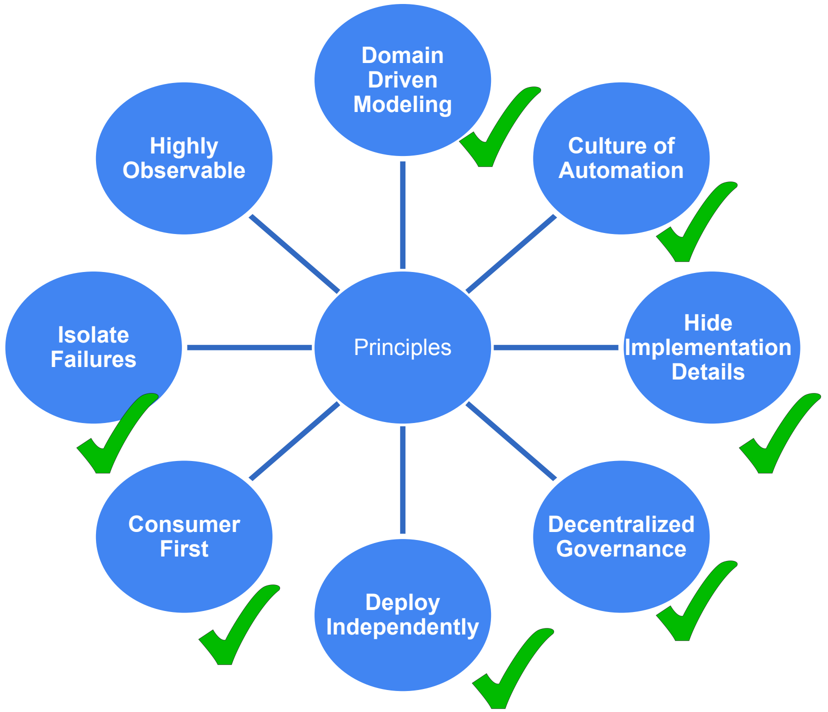
API Documentation

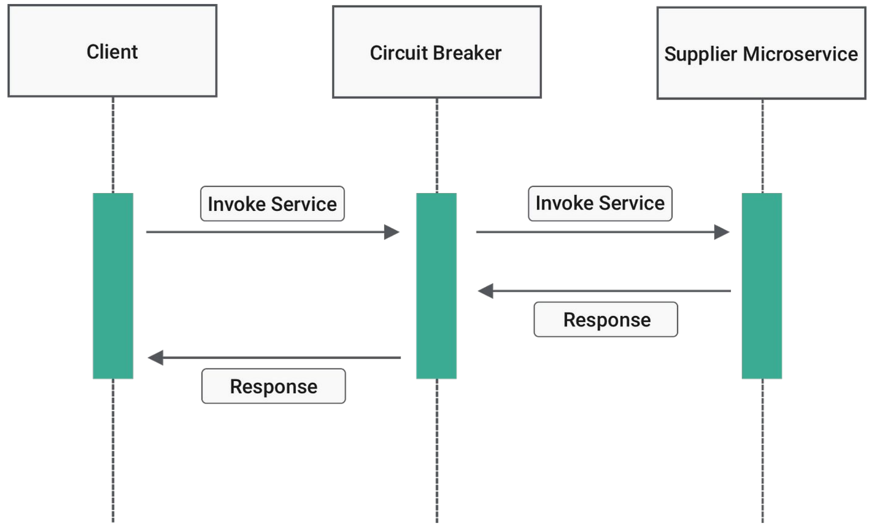


Swagger™

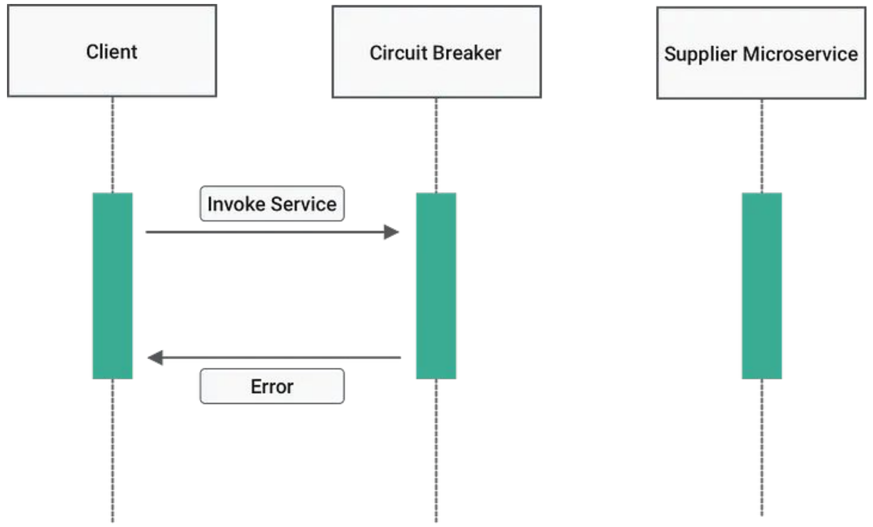
Principle 7: Isolate Failure

- Avoid cascading failures
- Timeouts between components
- **Fail fast** aka *Design for Failure*
    - Bulkheading / Circuit breakers





Closed circuit

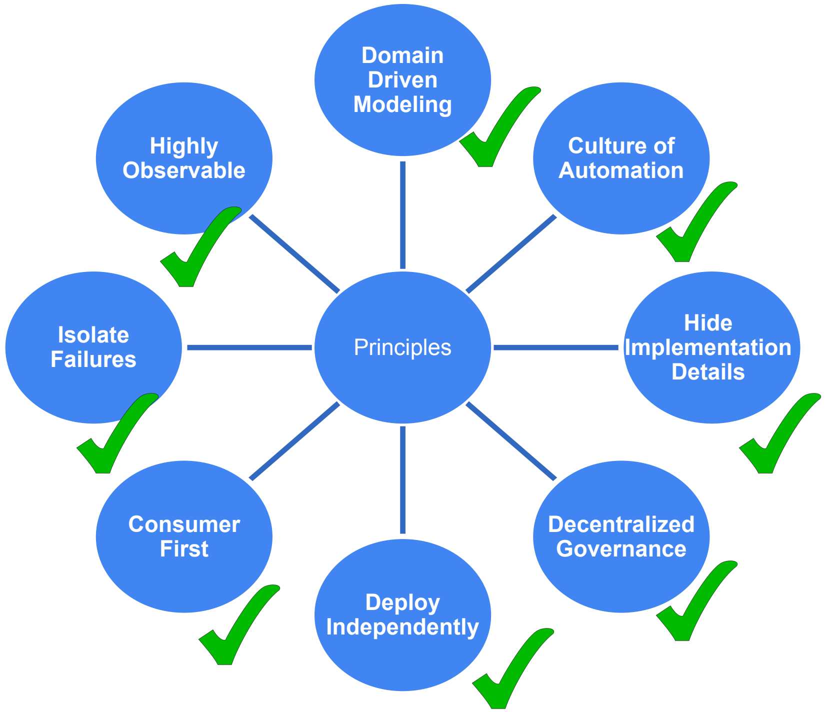


Open circuit

Image source: blogs.halodoc.io

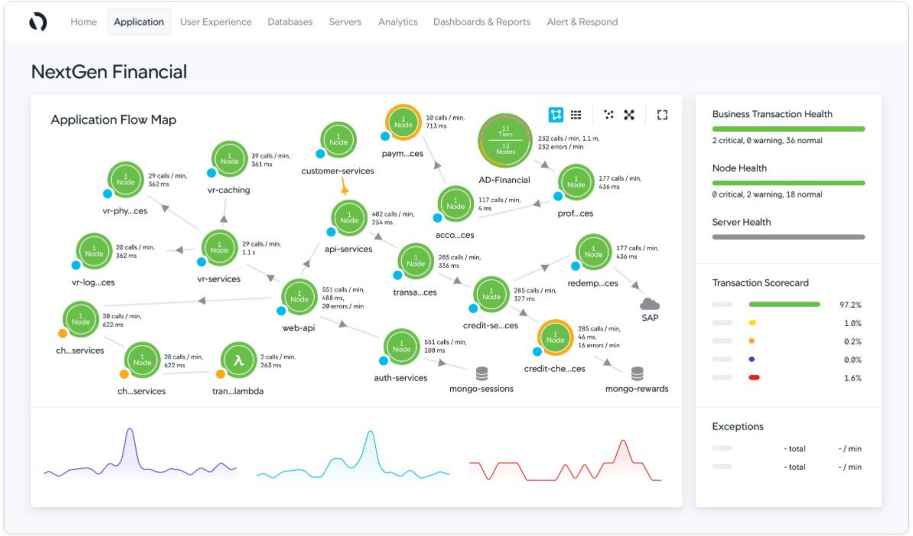
Principle 8: Highly Observable

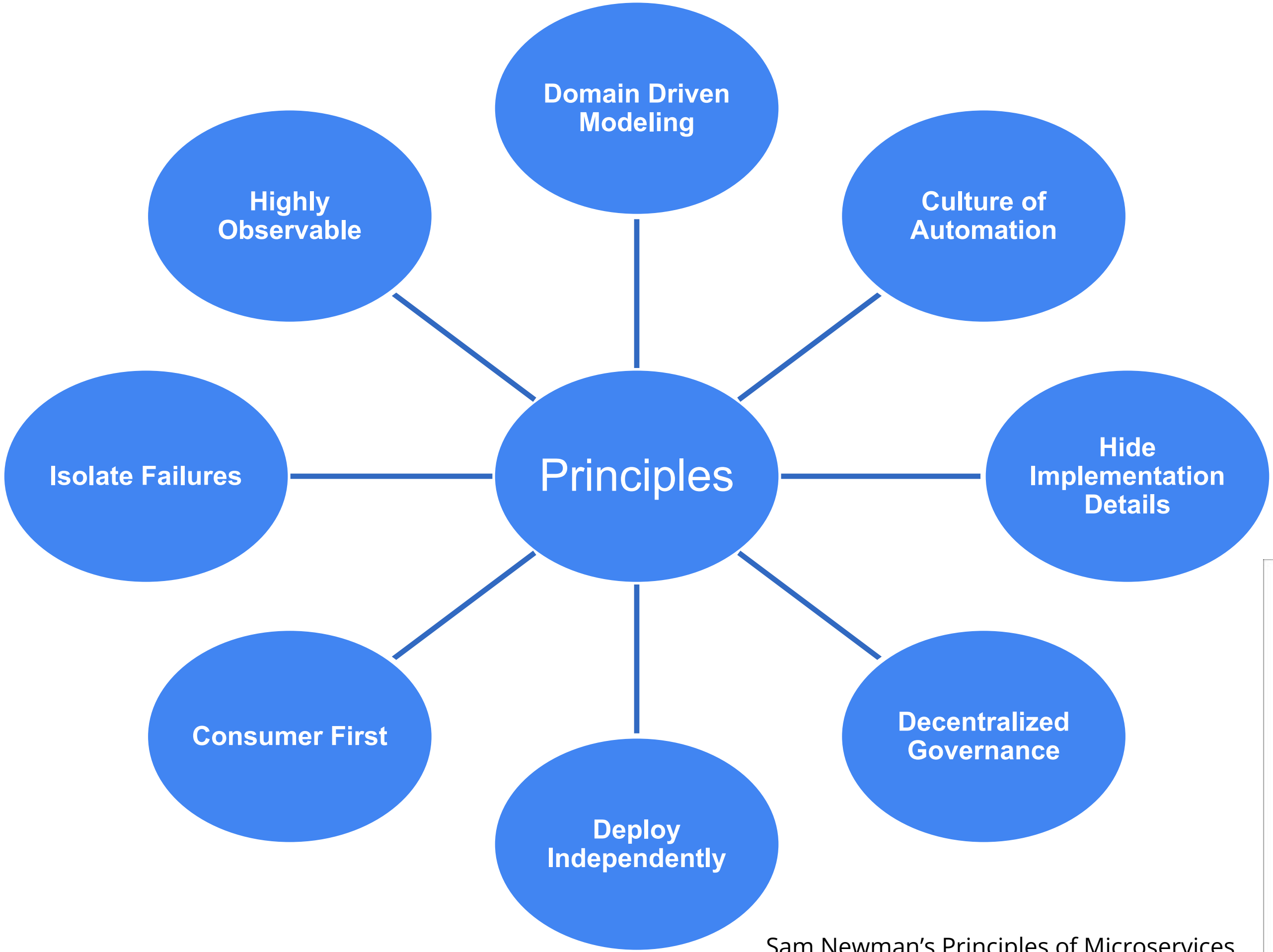
- Standard Monitoring
- Health-Check Pages
- Log and Stats aggregation
- Downstream monitoring

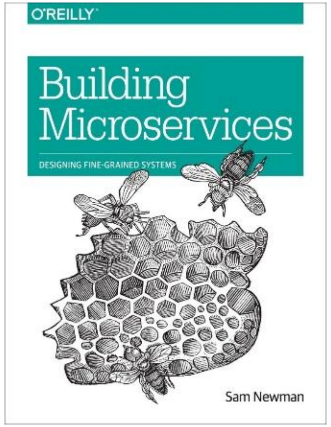


Principle 8: Highly Observable

- Standard Monitoring
- Health-Check Pages
- Log and Stats aggregation
- Downstream monitoring



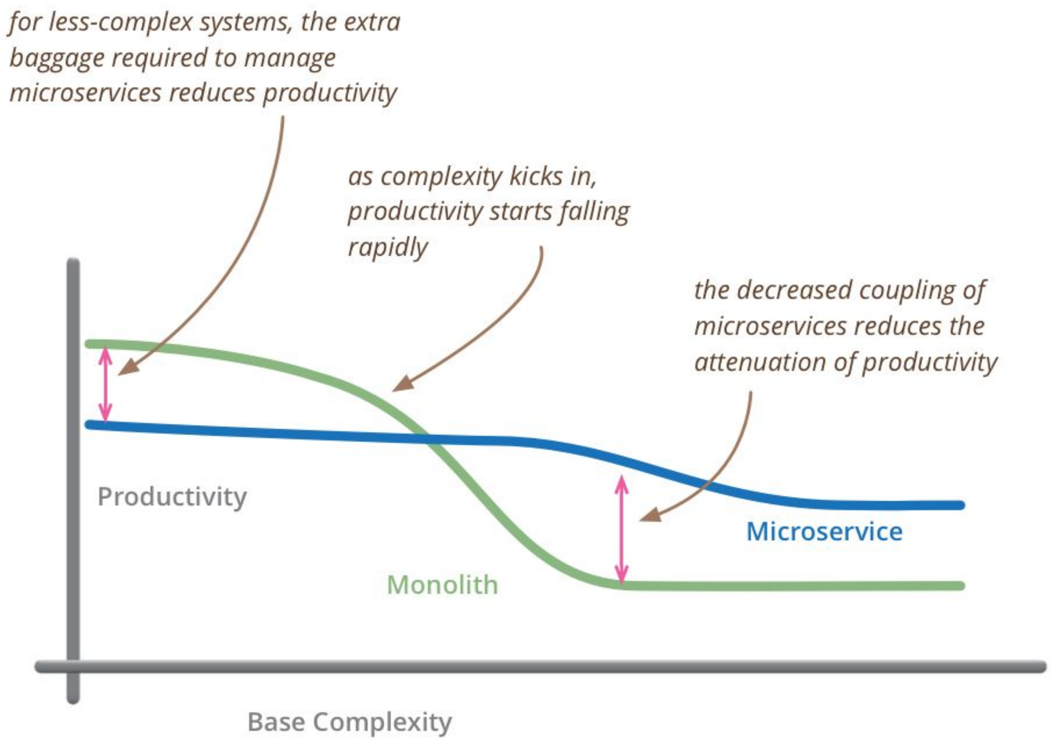




Sam Newman's Principles of Microservices

Are microservices always the right choice?

Microservices overhead



Taking it to the extreme

SERVERLESS

Taken to the extreme… Serverless (Functions-as-a-Service)

- Instead of writing minimal services, write just functions
- No state, rely completely on cloud storage or other cloud services
- Pay-per-invocation billing with elastic scalability
- Drawback: more ways things can fail, state is expensive
- Examples:
  AWS lambda, CloudFlare workers, Azure Functions
- What might this be good for?