Architectural Skills

CSSE6400

Richard Thomas

May 22, 2023

Quote

Architecture is the stuff you can't Google.
    – Mark Richards [Richards and Ford, 2020]

Quote

There are no right or wrong answers in architecture—only trade-offs.
– Neal Ford *[Richards and Ford, 2020]*

Architectural Design

Architects use knowledge and experience to analyse trade-offs to design architectures appropriate to the system context.

Developers – Technical Depth [Richards and Ford, 2020]

Architects – Technical Breadth [Richards and Ford, 2020]

What are the benefits of a monolith architecture?

What are the benefits of a monolith architecture?

- Simple deployment

What are the benefits of a monolith architecture?

- Simple deployment
- Simple communication between modules

What are the benefits of a monolith architecture?

- Simple deployment
- Simple communication between modules
- Simple system testing & debugging

Why do monoliths have a bad name?

Why do monoliths have a bad name?

- Many legacy system nightmares were monoliths

Why do monoliths have a bad name?

- Many legacy system nightmares were monoliths
- Easy to defeat modularity

Why do monoliths have a bad name?

- Many legacy system nightmares were monoliths
- Easy to defeat modularity
- Cannot scale components of system

Why do monoliths have a bad name?

- Many legacy system nightmares were monoliths
- Easy to defeat modularity
- Cannot scale components of system
- Monolith databases scale poorly

What can be done if a monolith architecture is no longer suitable?

What can be done if a monolith architecture is no longer suitable?

- Greenfields replacement

What can be done if a monolith architecture is no longer suitable?

- Greenfields replacement
- Migrate to another architecture

How do I migrate a monolith to a new architecture?

How do I migrate a monolith to a new architecture?

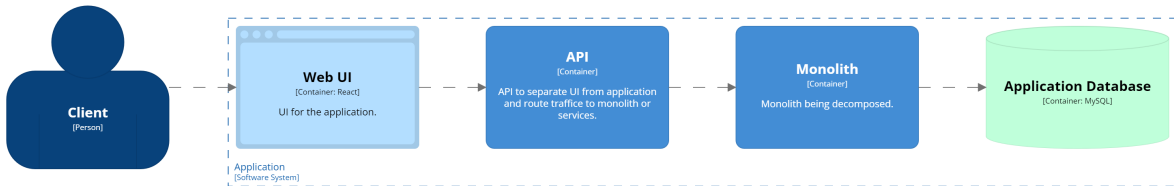Decompose the monolith into services.

Strangler Fig Pattern

- Develop API for application's UI
- Proxy intercepts API calls
  - Proxy directs calls to application or new services
- Implement a service
  - Redirect calls to service
- Progressively replace monolith
- Shadow & Blue-Green Deployment

Monolith Deployment

Client
[Person]

Application
[Software System]

Web UI
[Container: React]

UI for the application.

API
[Container]

API to separate UI from application and route traffice to monolith or services.

Monolith
[Container]

Monolith being decomposed.

Application Database
[Container: MySQL]

Monolith Decompose: Step 1

Monolith Decompose: Step 2

- Identify bounded-contexts
- Simple first service
  - e.g. Authentication
- Minimise dependency from services to monolith
  - Monolith may use services

Decomposition Process

- Reduce coupling between bounded- contexts
  - e.g. Customer account management
    - Profile, Wish List, Payment Preferences – separate services
- Decouple vertically
  - Service delivers entire bounded-context
    - Data is decoupled from monolith

- Focus on pain points
  - Bottlenecks
  - Frequently changing behaviour
- Rewrite, don't reuse
  - Redesign for new infrastructure
  - Reuse complex logic
    - e.g. Discounts based on customer loyalty and behaviour, bundle offers, . . .

- Refactor monolith
  - Use service to deliver application functionality
    - Monolith may need to invoke service
  - Remove service logic from monolith

Stepwise Decomposition

Replace application functionality one service at a time.

Definition 1. Macroservice

Separate service, but may span more than one domain or share a database with the monolith or other services.

Definition 2. Nanoservice

Service that depends on other services and cannot be deployed independently – its context is too small.

Definition 3. Conway's Law

Organisations design systems whose structure is inevitably a copy of the organisation's communication structure[Conway, 1968][MacCormack et al., 2012].

- Business Process Management
- Microservices to reflect organisation structure
- Teams formed around services

Team insularity – more loyal to team than organisation.

- Cross-cutting concerns
  - e.g. Security
- Organisation structure should align with market structure
- Physical location of teams

Evidenced-Based Software Engineering

Don't follow fads, seek evidence for good practice.

Software Engineering's Greatest Hits

**what we actually know about software development
and why we believe it's true**

Greg Wilson

https://youtu.be/HrVtA-ue-x0

References

[Conway, 1968] Conway, M. E. (1968).
How do committees invent?
Datamation.

[MacCormack et al., 2012] MacCormack, A., Baldwin, C., and Rusnak, J. (2012).
Exploring the duality between product and organizational architectures: A test
of the "mirroring" hypothesis.
Research Policy, 41(8):1309–1324.

[Richards and Ford, 2020] Richards, M. and Ford, N. (2020).
Fundamentals of Software Architecture: An Engineering Approach.
O'Reilly Media, Inc.