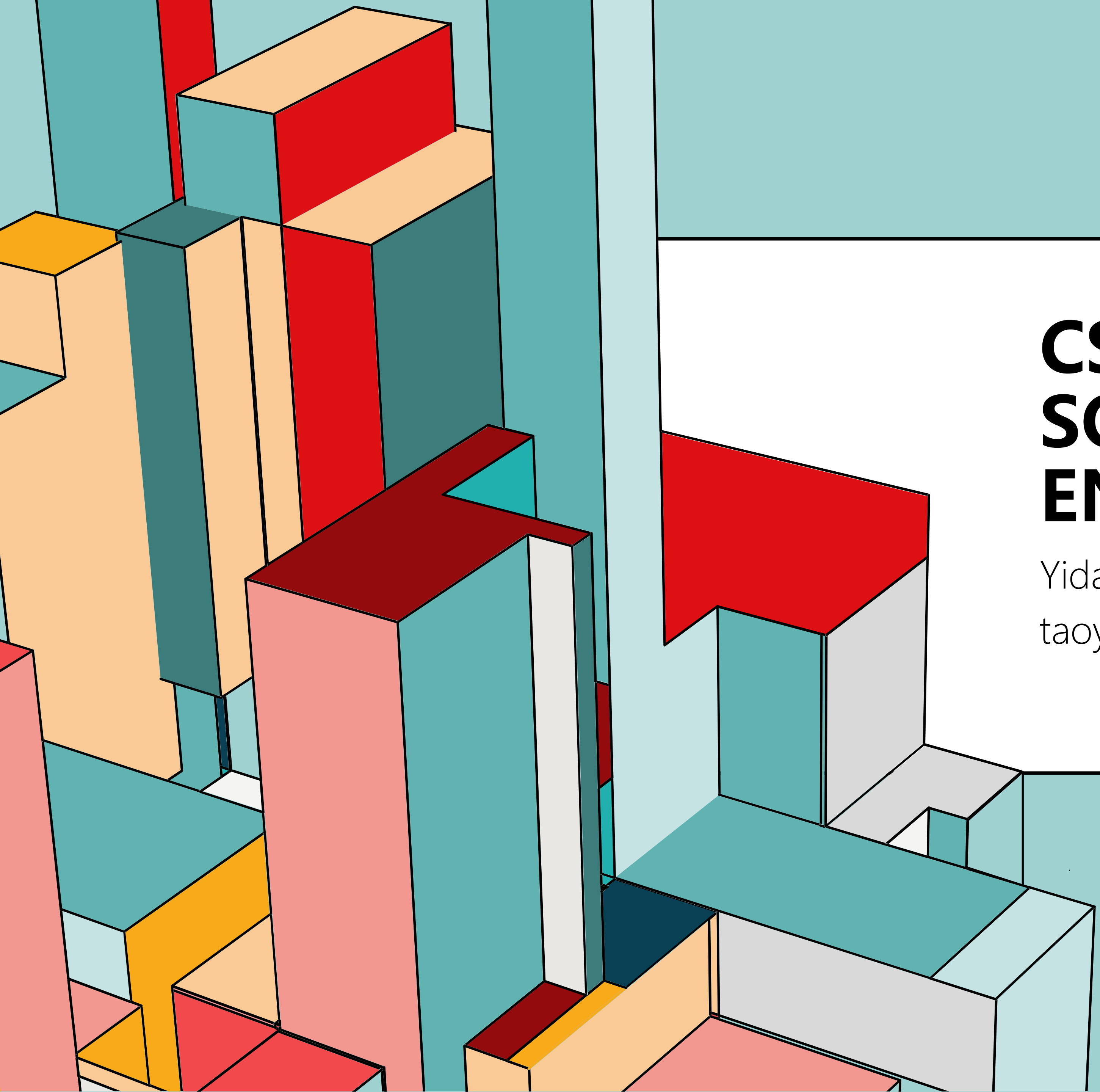
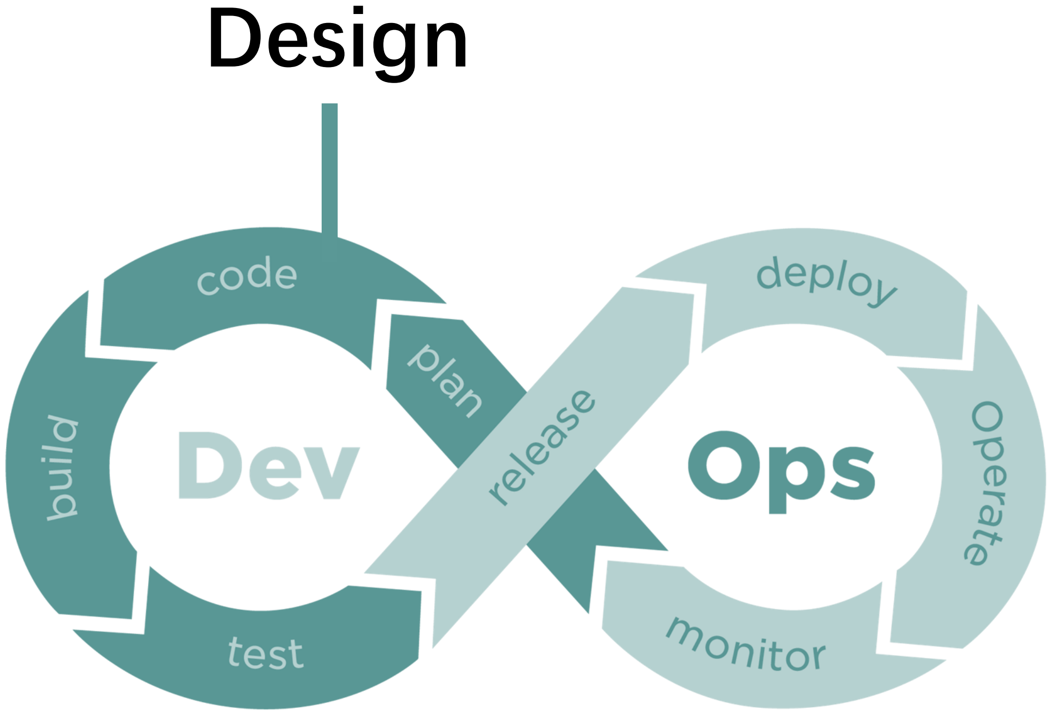


CS304 SOFTWARE ENGINEERING

Yida Tao
taoyd@sustech.edu.cn

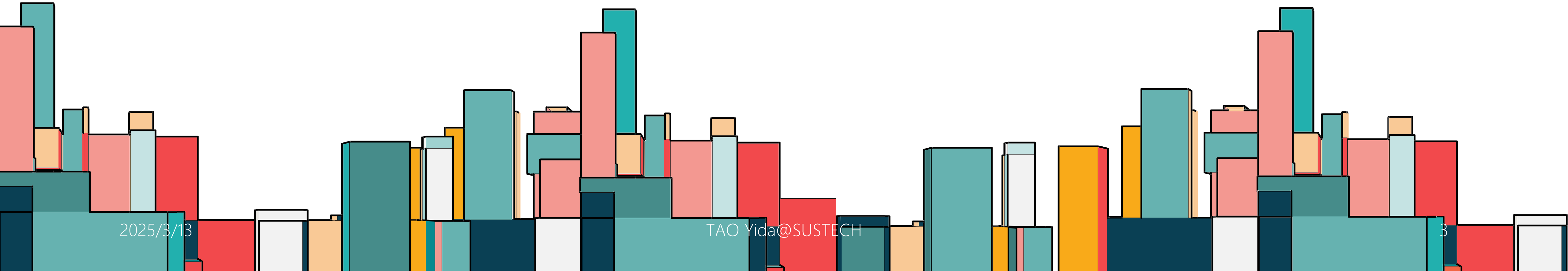
WHERE ARE WE NOW?

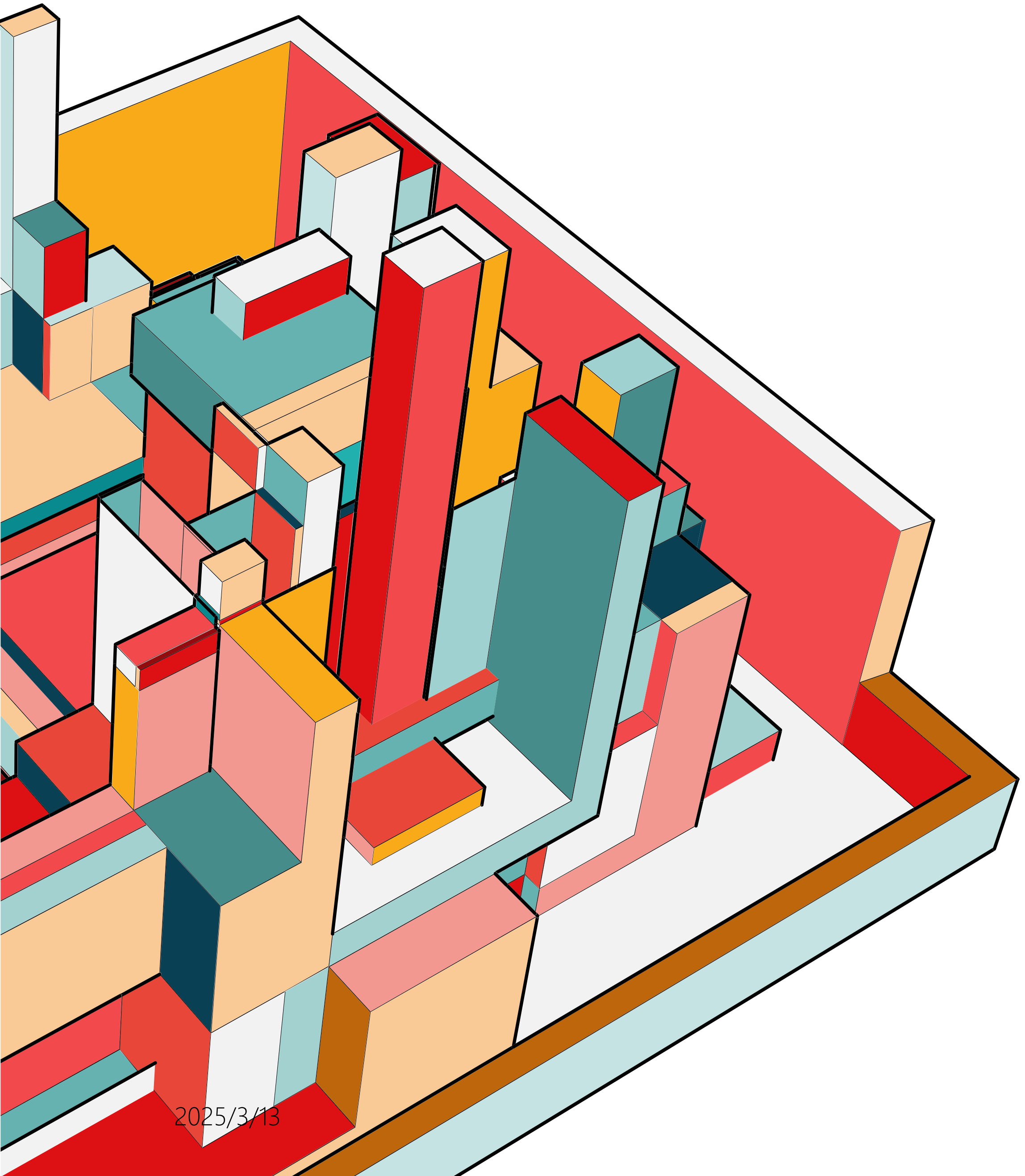


WHAT IS DESIGN?

It’s where you stand with a foot in two worlds—the world of technology and the world of people and human purposes—and you try to bring the two together.

- Mitch Kapor, “software design manifesto”





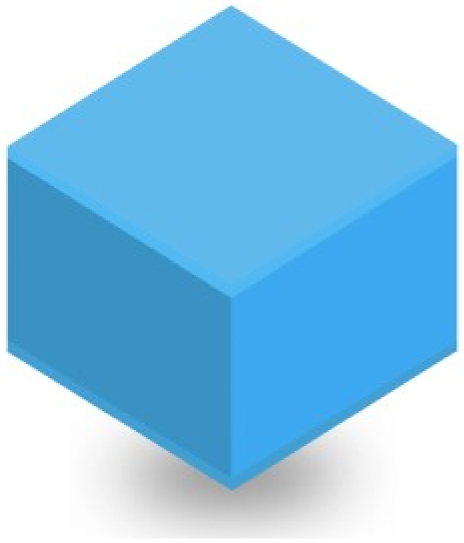
SOFTWARE DESIGN

- **Architectural design**
- **User interface design**
- Data design
- API design

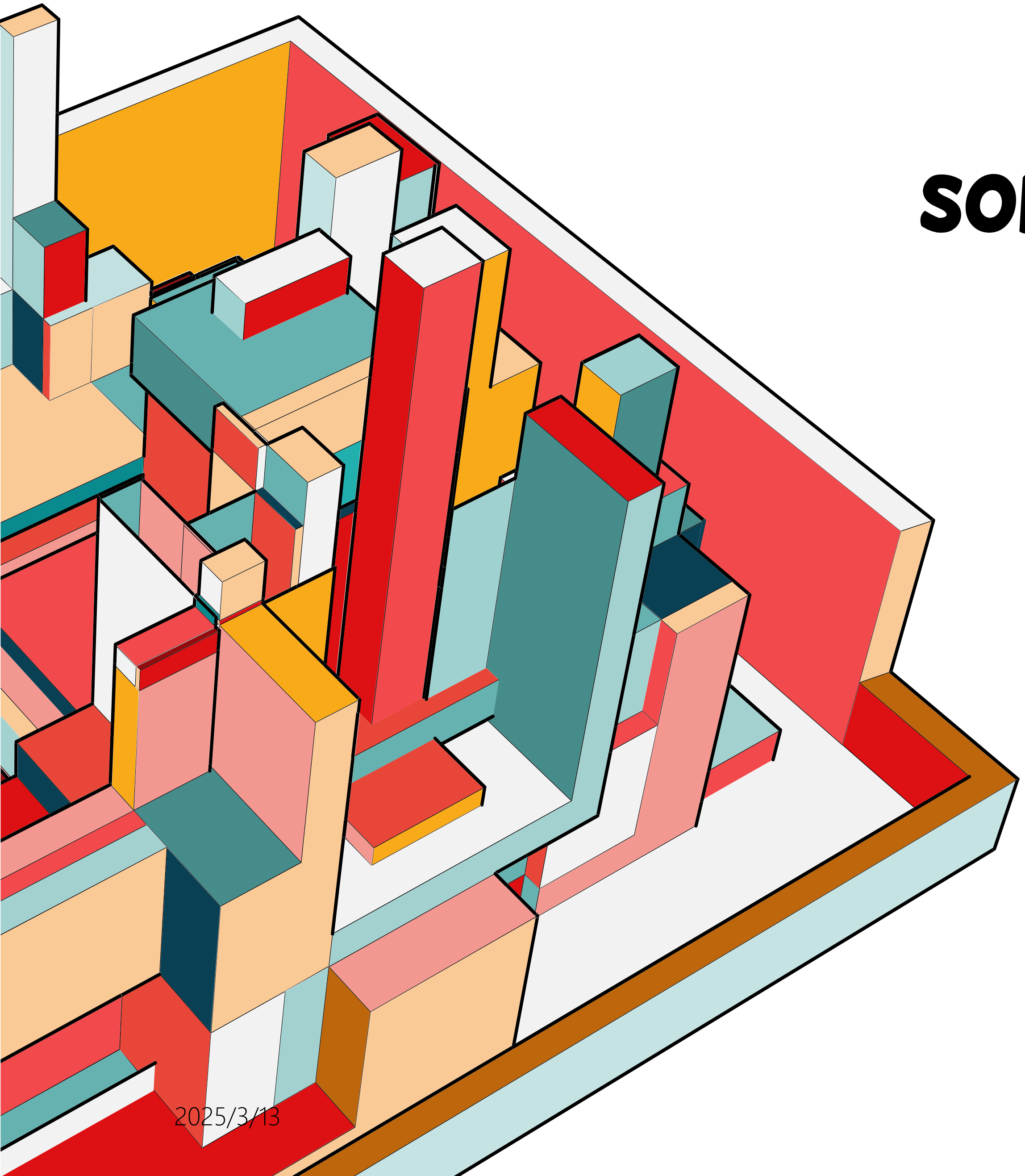
ARCHITECTURAL STYLE

SOFTWARE ARCHITECTURAL STYLE

Classic, Monolithic



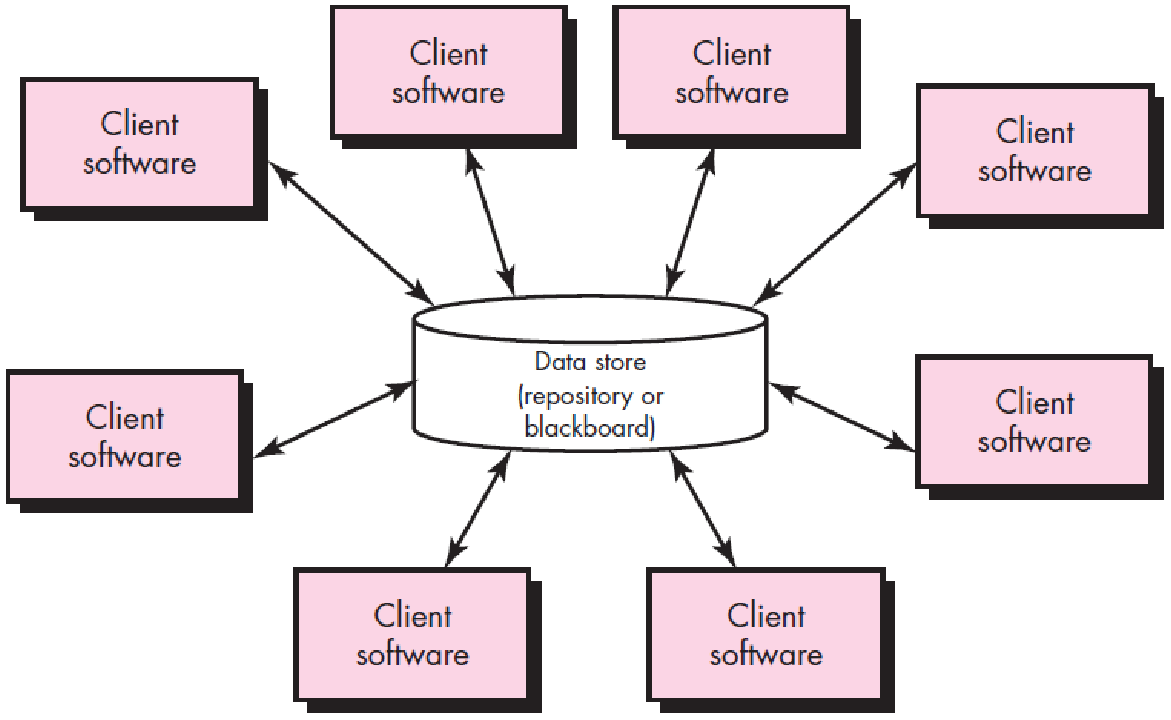
Service-based, Distributed, DevOps



SOFTWARE ARCHITECTURAL STYLE

- Classic, Monolithic
  - Data-centered architecture
  - Data-flow architecture
  - Call-and-return architecture
  - Object-oriented architecture
  - Layered architecture
- Service-based, Distributed, DevOps
  - Microkernel architecture
  - Event-driven architecture
  - Microservice architecture

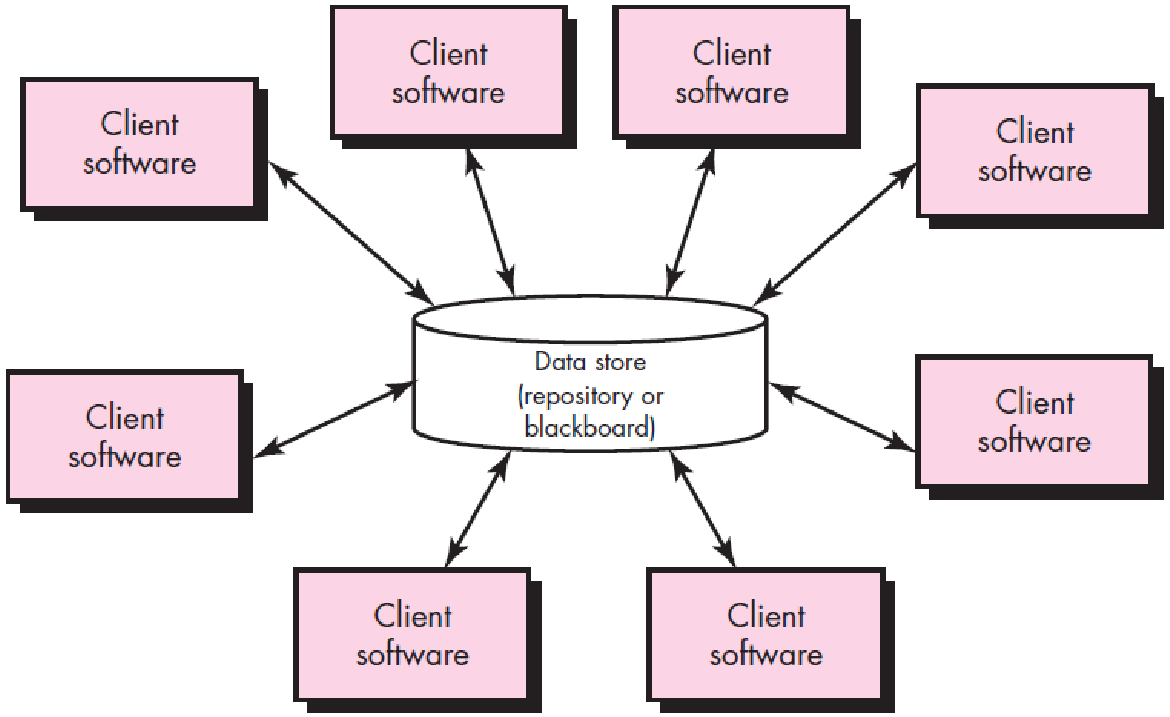
DATA-CENTERED ARCHITECTURES



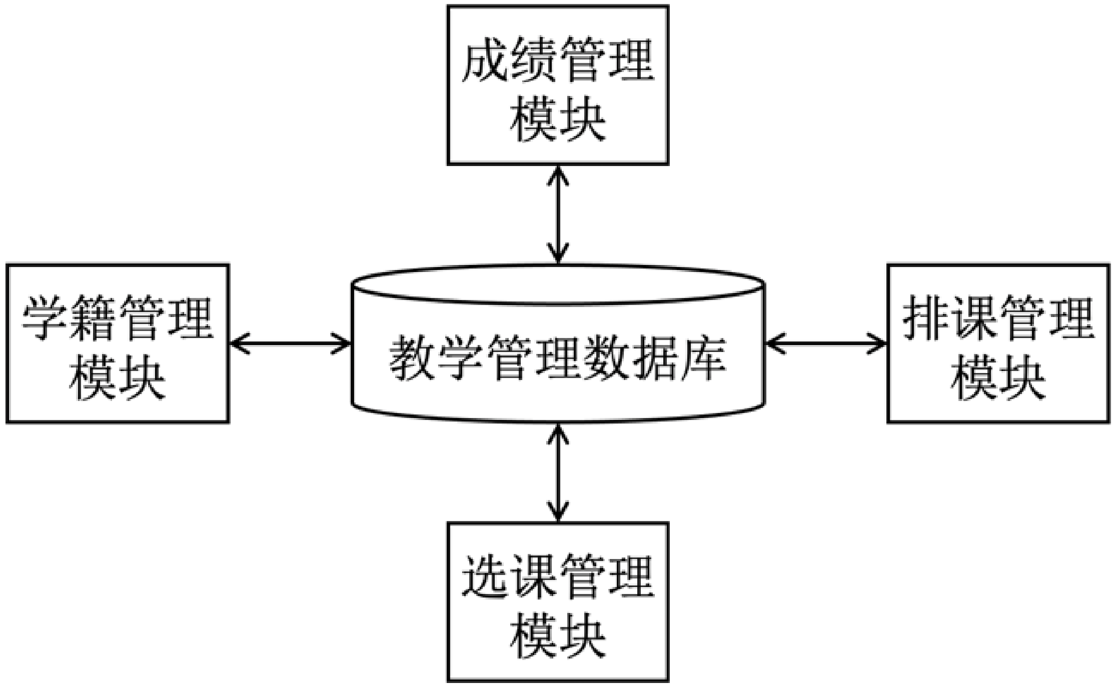
以数据为中心的体系结构

- A data store (e.g., a file or database) resides at the center of this architecture
- The data store is accessed frequently by other components that update, add, delete, or modify data within the store

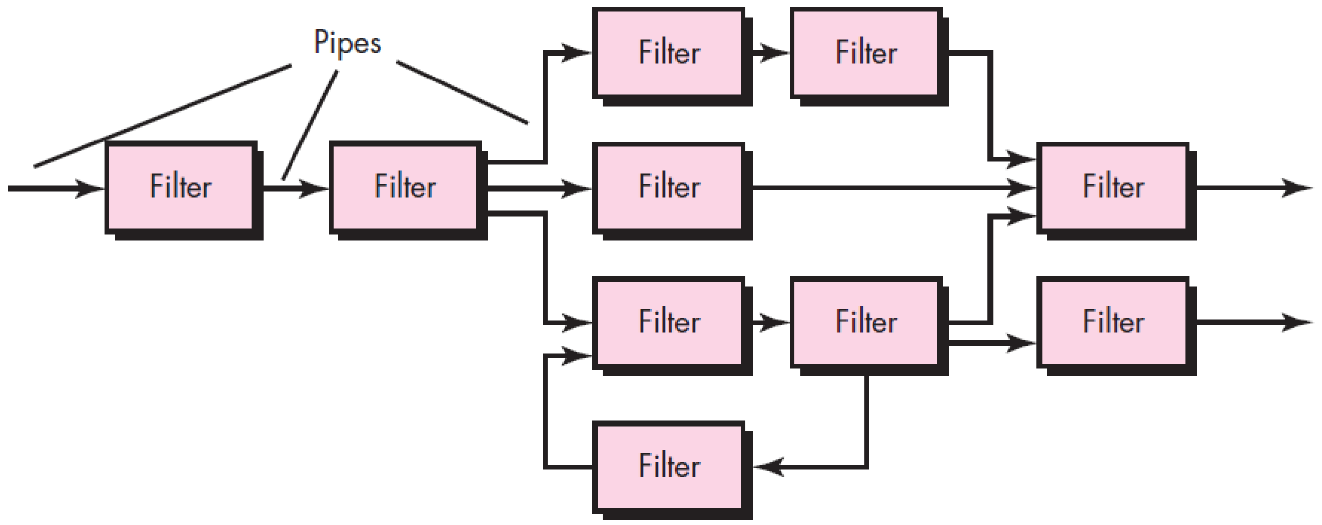
DATA-CENTERED ARCHITECTURES



以数据为中心的体系结构



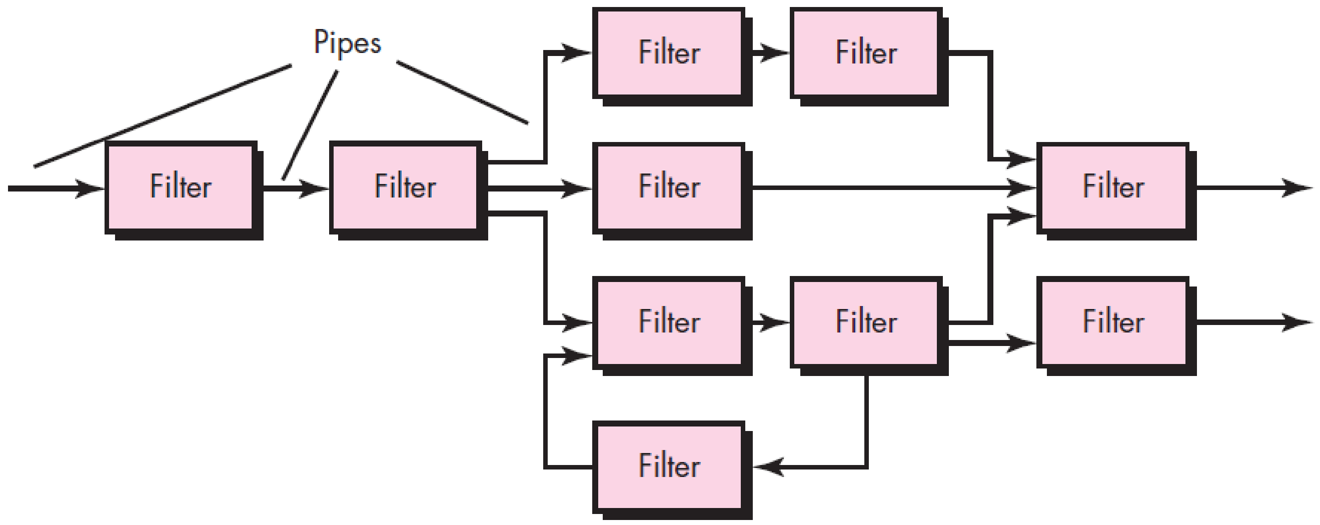
DATA-FLOW ARCHITECTURES



数据流体系结构（管道和过滤器）

This architecture is applied when input data are to be transformed through a series of computational or manipulative components into output data.

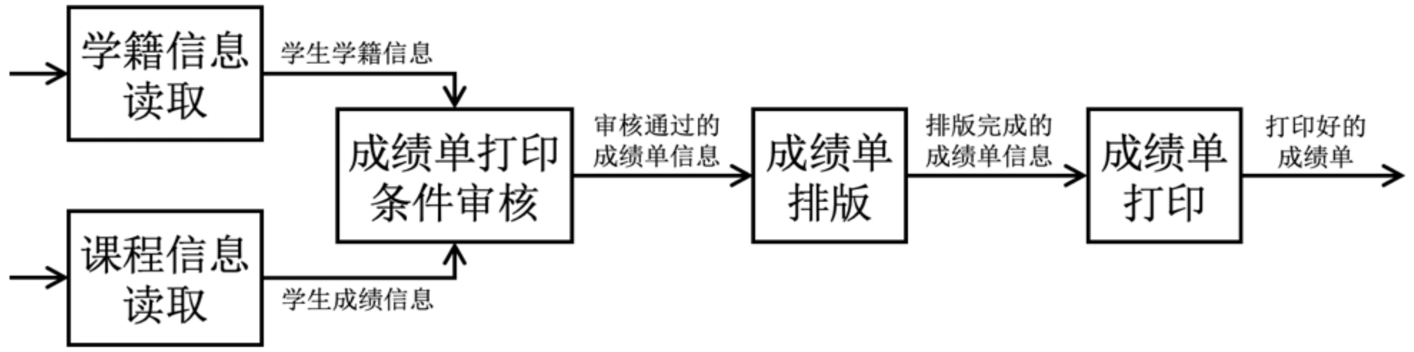
DATA-FLOW ARCHITECTURES



数据流体系结构（管道和过滤器）

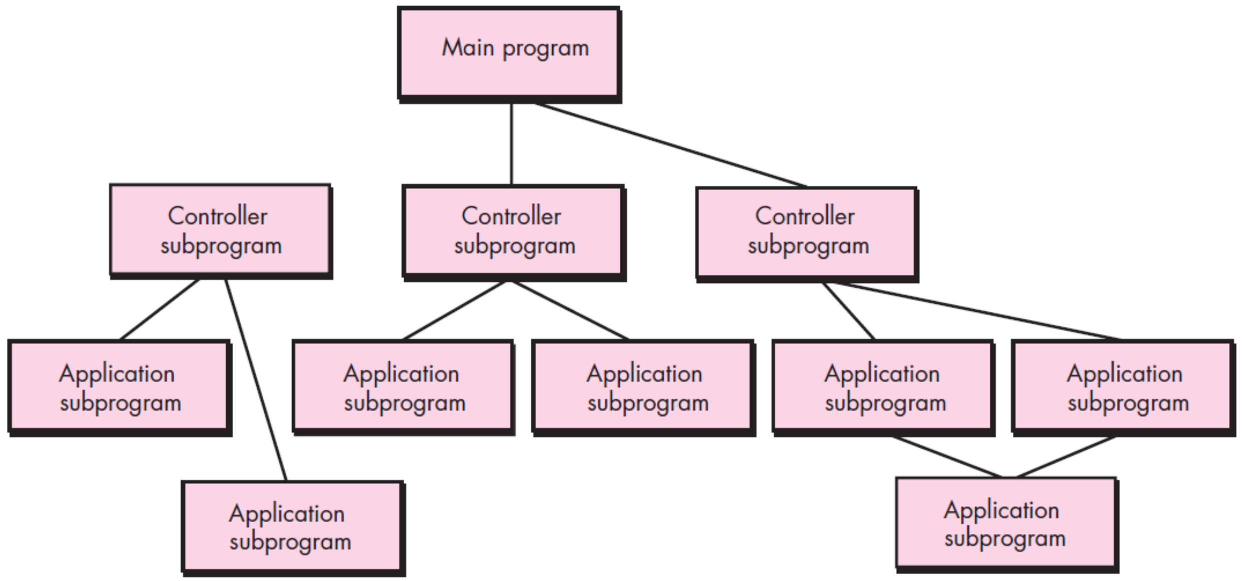
- A pipe-and-filter pattern has a set of components, called filters, connected by pipes that transmit data from one component to the next.
- Each filter works independently of those components upstream and downstream, is designed to expect data input of a certain form, and produces data output (to the next filter) of a specified form.
- The filter does not require knowledge of the workings of its neighboring filters.

DATA-FLOW ARCHITECTURES



- Data-flow architecture is suitable for automated data analysis and transmission systems
- Such systems contain a series of data analysis components, with almost no user interaction
- Data-flow architecture may not be suitable for GUI intensive systems

CALL AND RETURN ARCHITECTURES

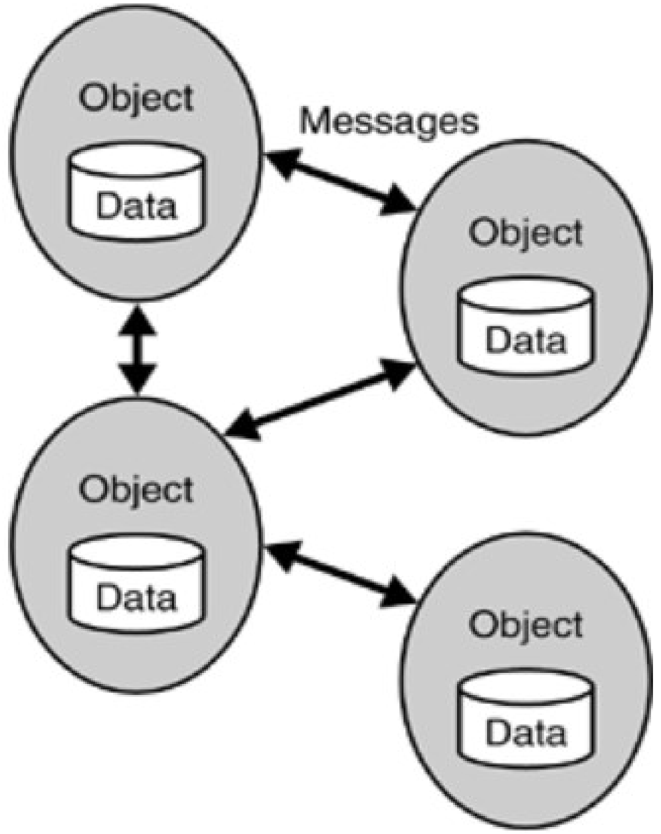


主程序/子程序体系结构

- **Main program/subprogram architecture** decomposes function into a control hierarchy where a "main" program invokes program components that in turn may invoke still other components.
- **Remote procedure call architecture**: The components of a main program and subprogram architecture are distributed across multiple computers on a network.

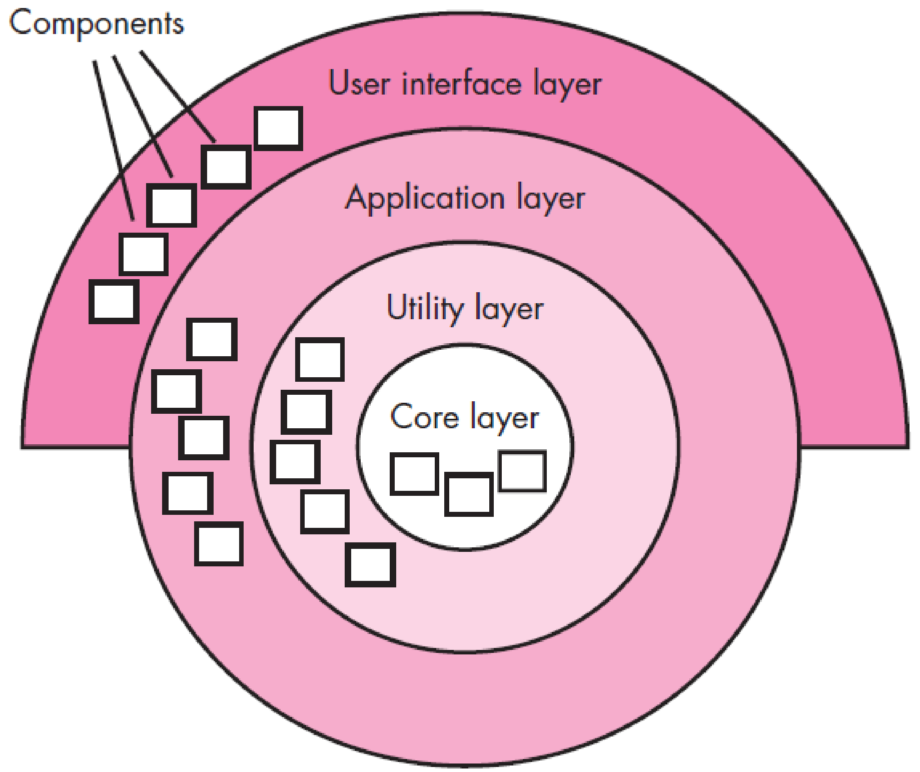
OBJECT-ORIENTED ARCHITECTURES

- The components of a system encapsulate data and the operations that must be applied to manipulate the data.
- Communication and coordination between components are accomplished via message passing

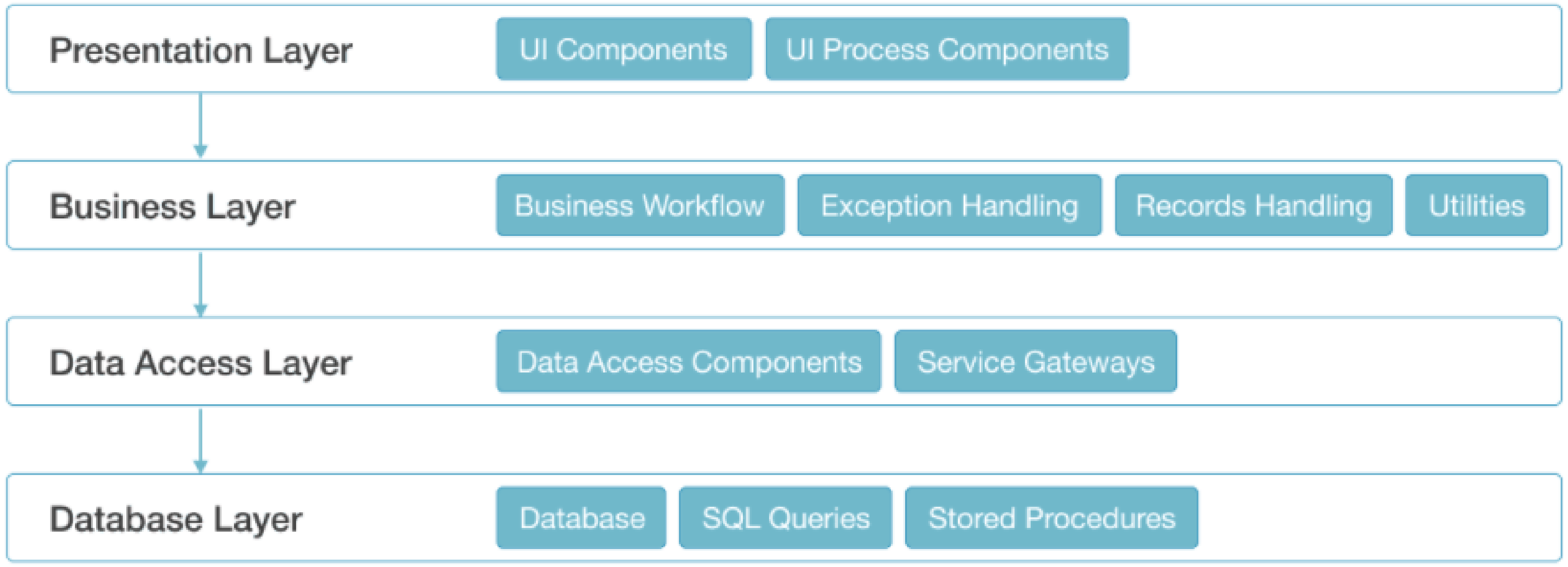


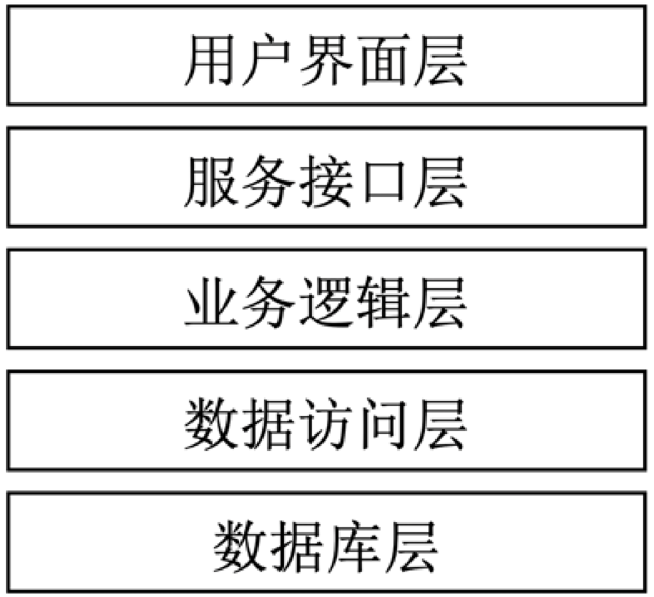
LAYERED ARCHITECTURES

- Different layers are defined, each accomplishing operations that progressively become closer to the machine instruction set.
- At the outer layer, components service user interface operations.
- At the inner layer, components perform operating system interfacing.
- Intermediate layers provide utility services and application software functions.



LAYERED ARCHITECTURES





Layered architecture for web applications

https://www.simform.com/blog/web-application-architecture/

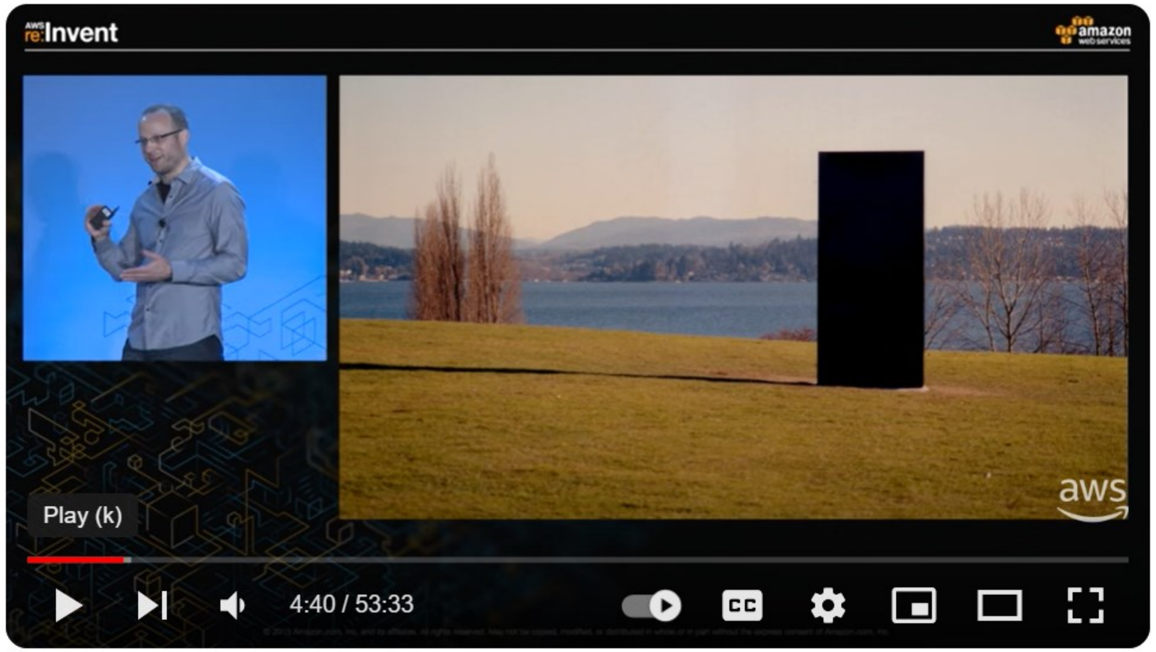
PROBLEMS WITH MONOLITHIC?

DevOps at Amazon: A Look at Our Tools and Processes

Speaker: - Rob Brigham, Senior Product Manager, AWS
- Clare Liguori, Senior Software Engineer, AWS

PROBLEMS WITH MONOLITHIC?

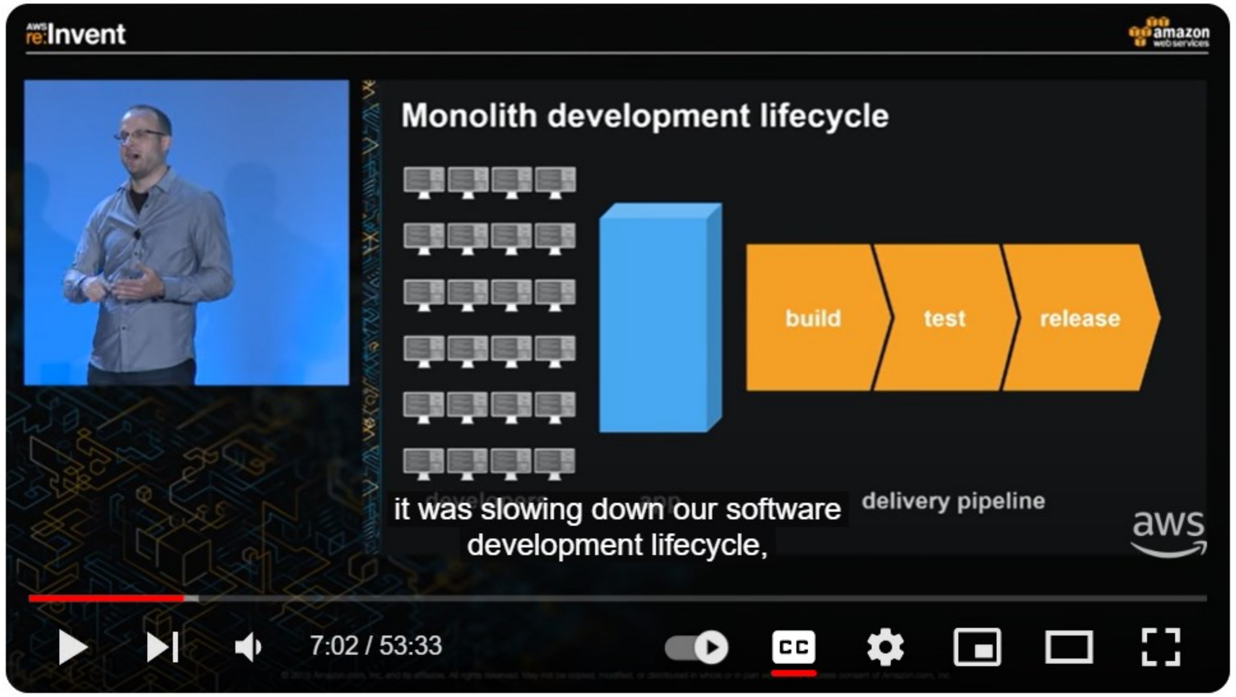
"If you go back to 2001," stated Amazon AWS senior manager for product management Rob Brigham, "the Amazon.com retail website was a large **architectural monolith**."



AWS re:Invent 2015: DevOps at Amazon: A Look at Our Tools and Processes (DVO202)

PROBLEMS WITH MONOLITHIC?

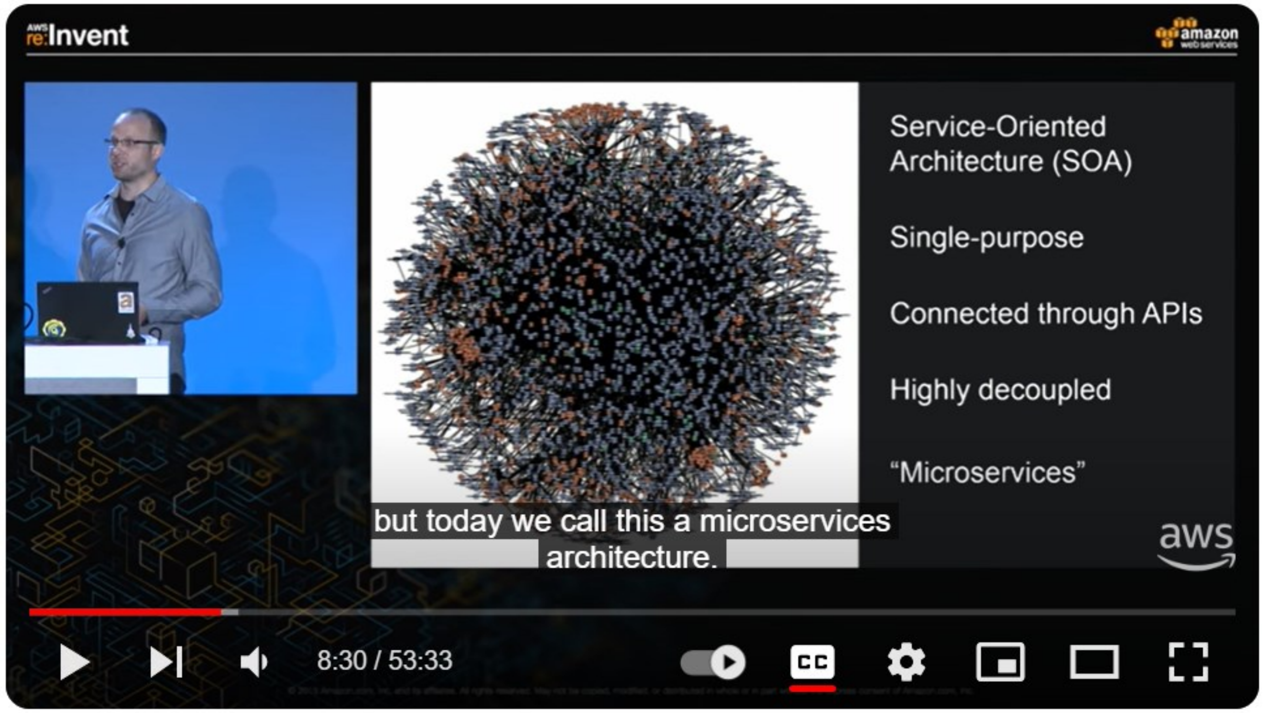
"Monolithic architecture adds large overhead to the process, frustrates developers, and slows down the entire software development lifecycle."


re:Invent
amazon web services
Monolith development lifecycle
build
test
release
delivery pipeline
it was slowing down our software development lifecycle,
aws
7:02 / 53:33

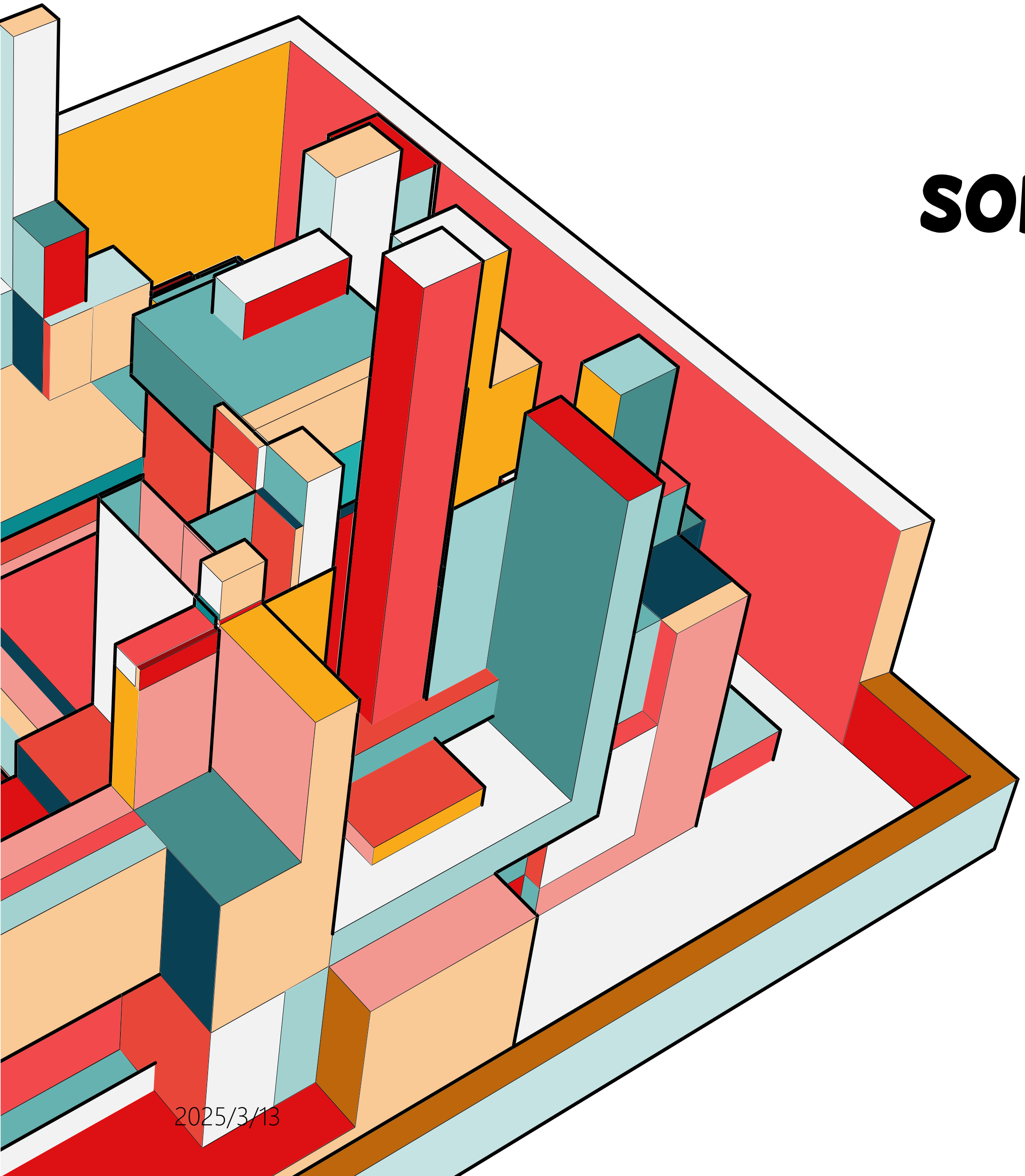
AWS re:Invent 2015: DevOps at Amazon: A Look at Our Tools and Processes (DVO202)

PROBLEMS WITH MONOLITHIC?

"We teased it apart into service-oriented architecture."



AWS re:Invent 2015: DevOps at Amazon: A Look at Our Tools and Processes (DVO202)

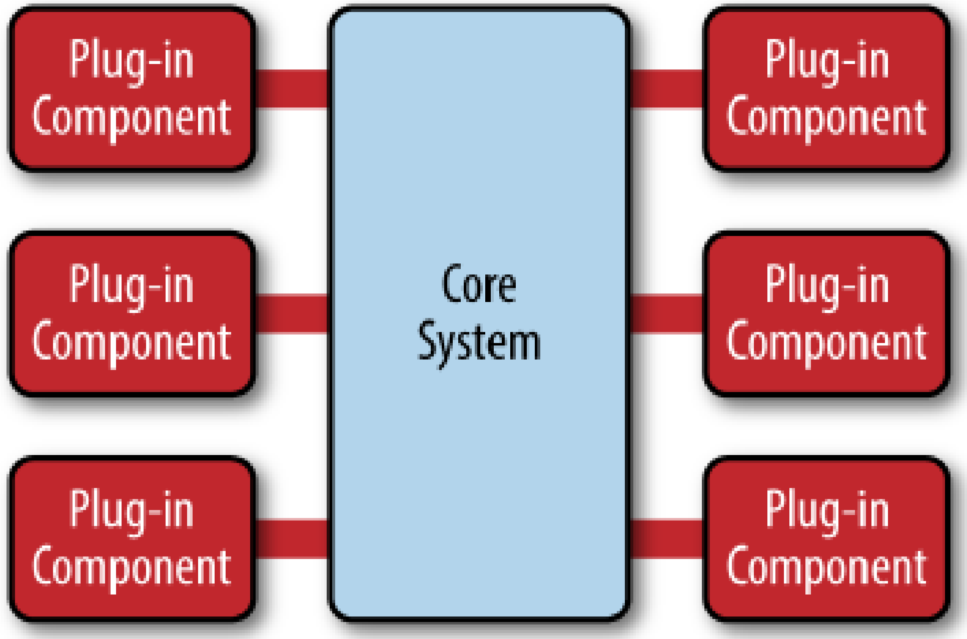


SOFTWARE ARCHITECTURAL STYLE

- Classic, Monolithic
  - Data-centered architecture
  - Data-flow architecture
  - Call-and-return architecture
  - Object-oriented architecture
  - Layered architecture
- Service-based, Distributed, DevOps
  - Microkernel architecture
  - Event-driven architecture
  - Microservice architecture

MICROKERNEL ARCHITECTURES

- **Core system**: only the **minimal functionality** required to make the system operational
- **Plug-in component**: stand-alone, independent components that contain specialized processing, additional features, and custom code that is meant to enhance or extend the core system to produce additional business capabilities.
- Also referred to as the **plug-in architecture pattern**



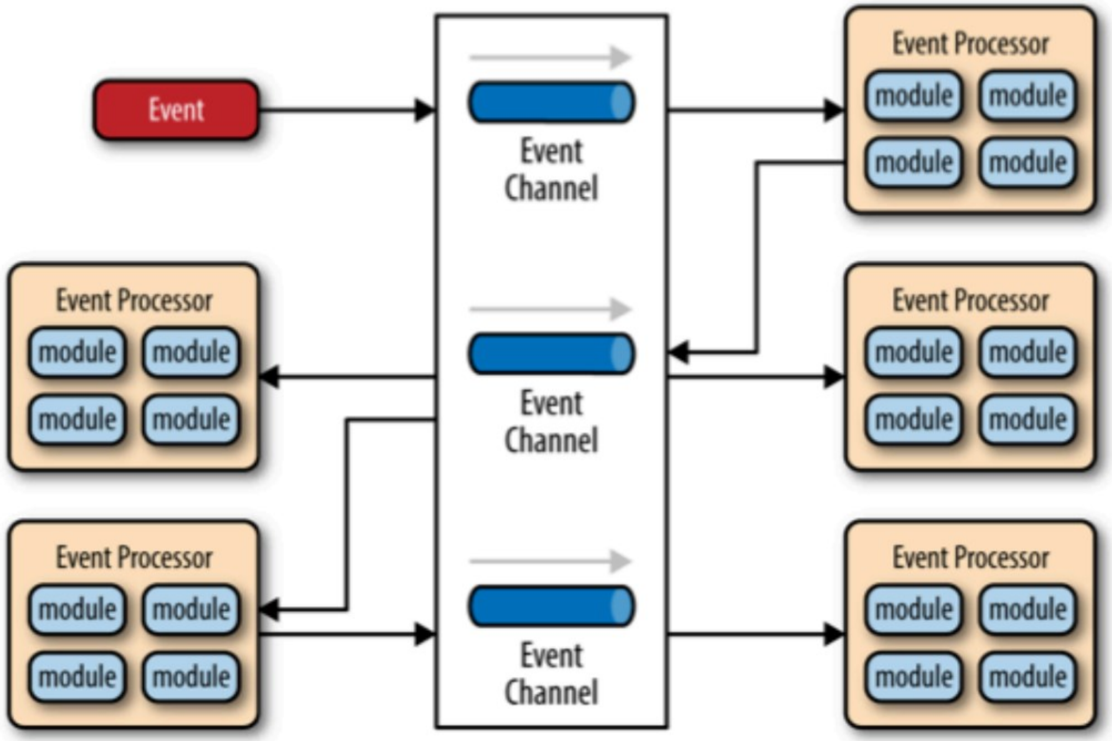
Mark Richards. 2015. Software Architecture Patterns.

MICROKERNEL ARCHITECTURES

eclipse

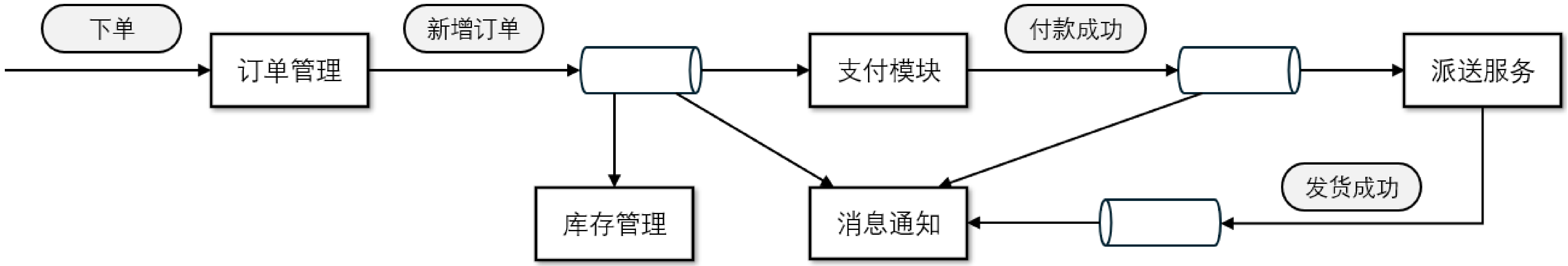
EVENT-DRIVEN ARCHITECTURE

- Core Components:
  - **Event Producer**: Initiates and generates events.
  - **Event Consumer**: Reacts to and processes events.
  - **Event channel/mediator/broker**: orchestration
- Asynchronous and distributed
- Promote the production, detection, consumption, and reaction to events



Mark Richards. 2015. Software Architecture Patterns.

EVENT-DRIVEN ARCHITECTURE



1. **Customer places an order** → Triggers "Order Placed" event
2. **Payment Service** listens → Processes the payment
3. **Inventory Service** listens → Updates stock levels
4. **Notification Service** listens → Sends an order confirmation email

MICROSERVICE ARCHITECTURE

The microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API.

These services are built around business capabilities and independently deployable by fully automated deployment machinery.

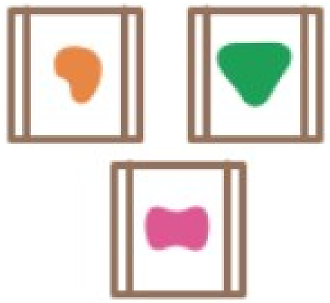
https://martinfowler.com/articles/microservices.html

MICROSERVICE ARCHITECTURE

Services

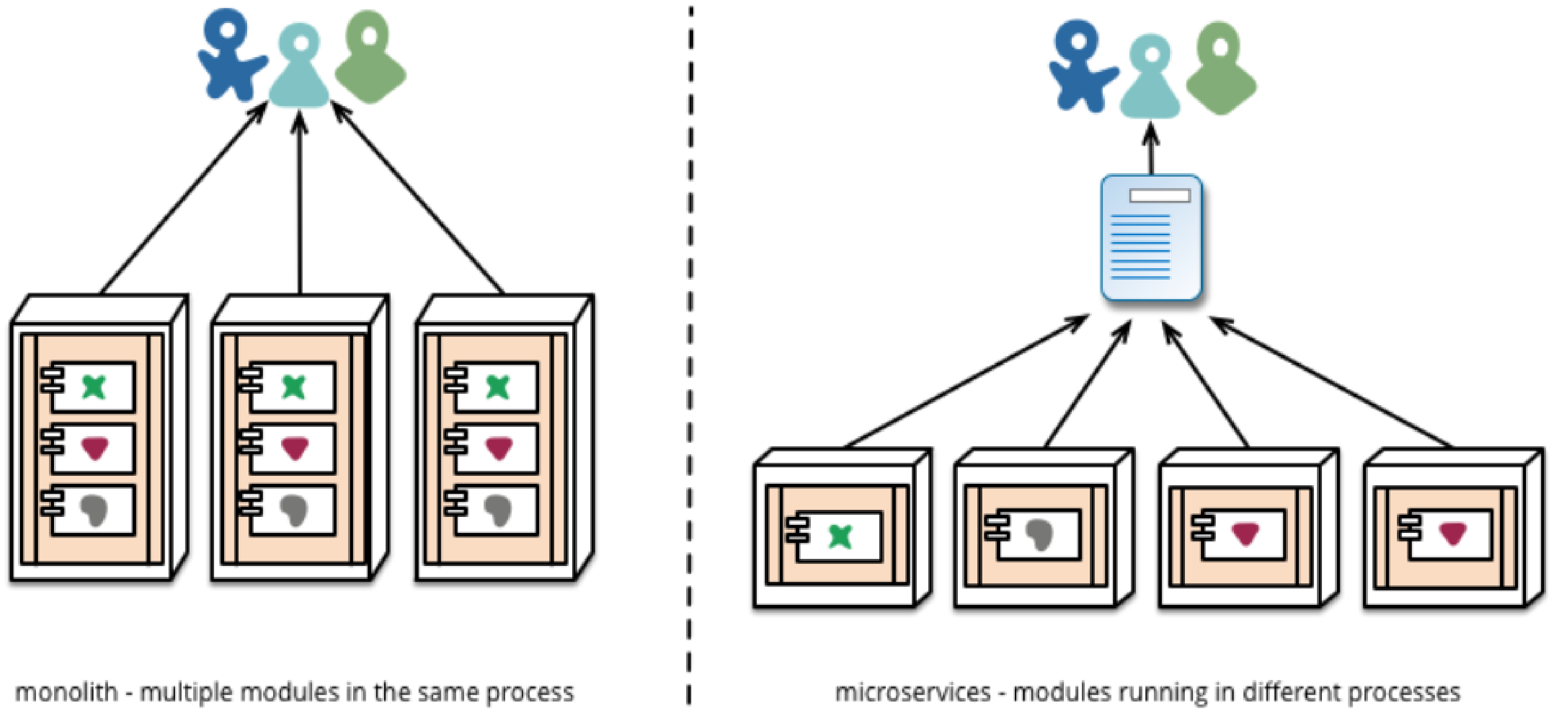
A monolithic application puts all its functionality into a single process...

A microservices architecture puts each element of functionality into a separate service...



https://martinfowler.com/articles/microservices.html

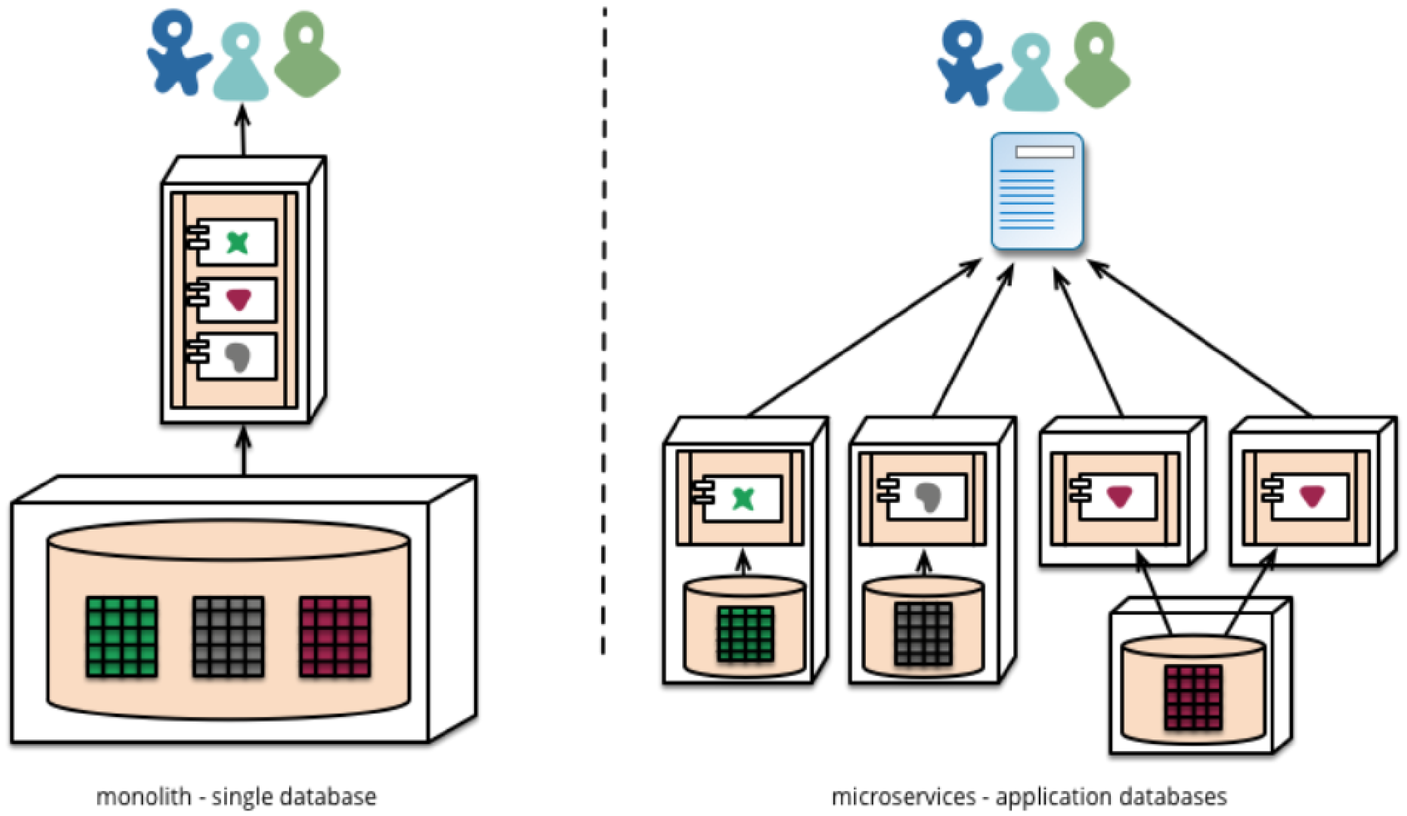
Deployment


monolith - multiple modules in the same process
microservices - modules running in different processes

MICROSERVICE ARCHITECTURE

https://martinfowler.com/articles/microservices.html

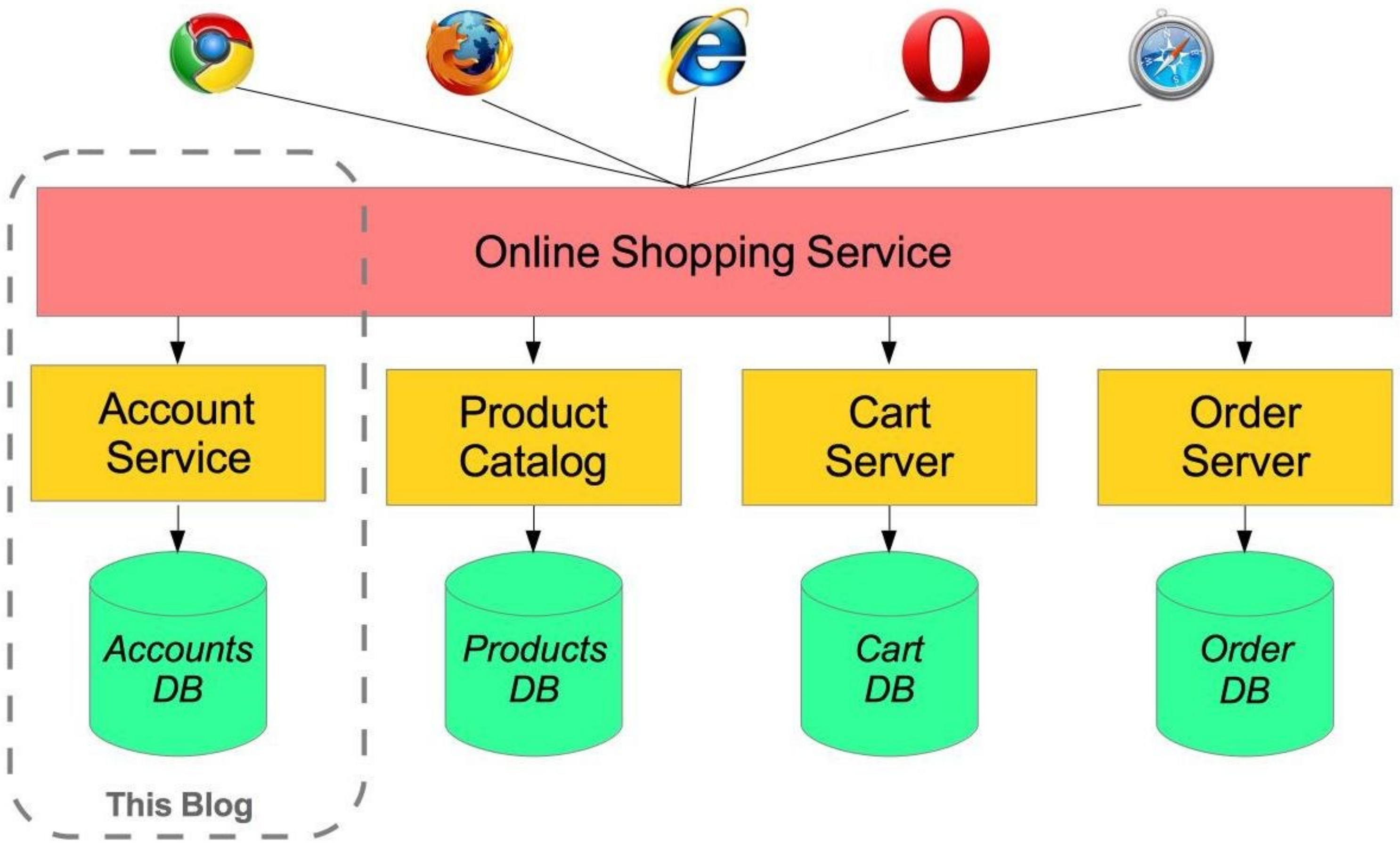
Decentralized Data Management


monolith - single database
microservices - application databases

MICROSERVICE ARCHITECTURE

https://martinfowler.com/articles/microservices.html

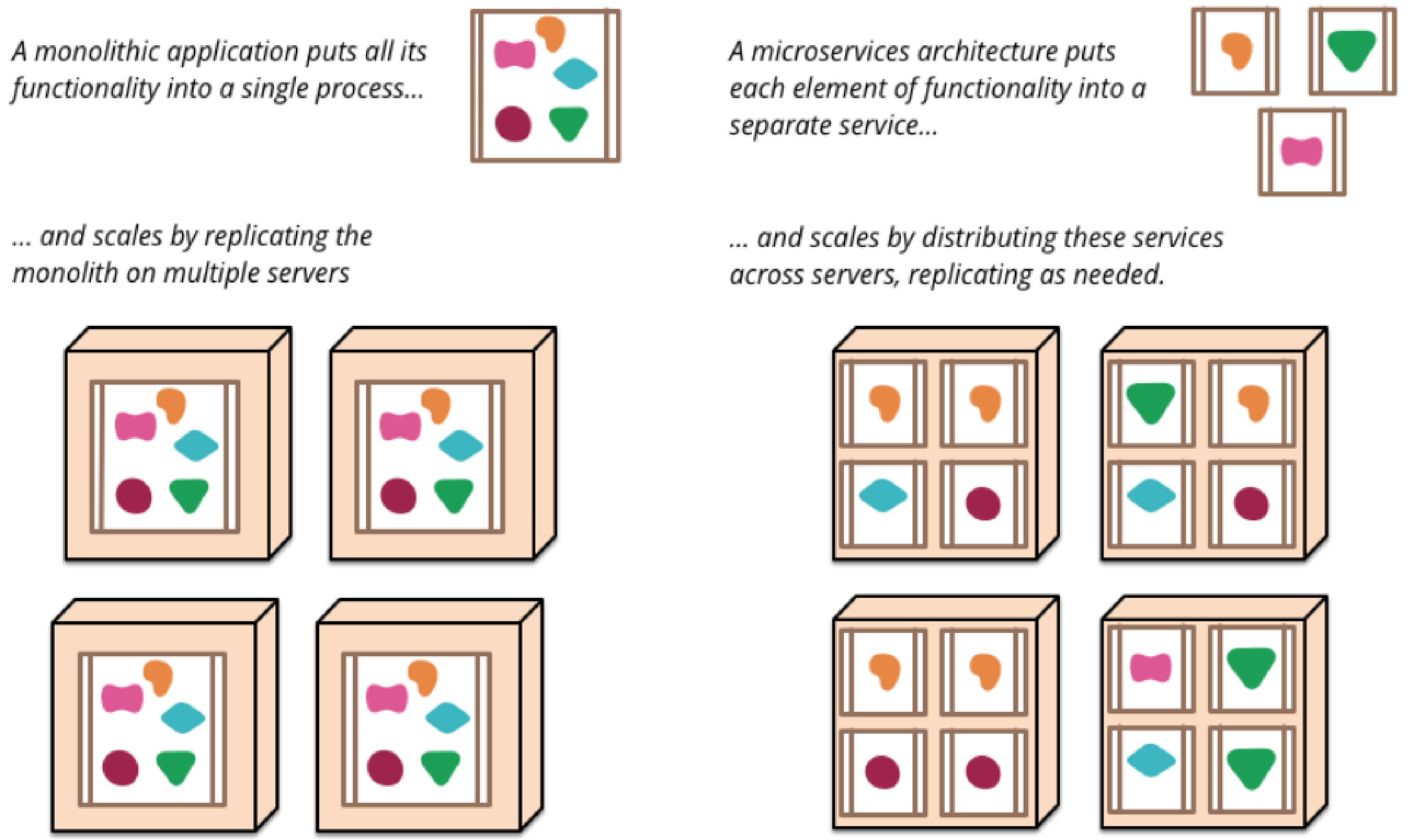
Example: online shopping



https://spring.io/blog/2015/07/14/microservices-with-spring

MICROSERVICE ARCHITECTURE

Scale


A monolithic application puts all its functionality into a single process...

... and scales by replicating the monolith on multiple servers

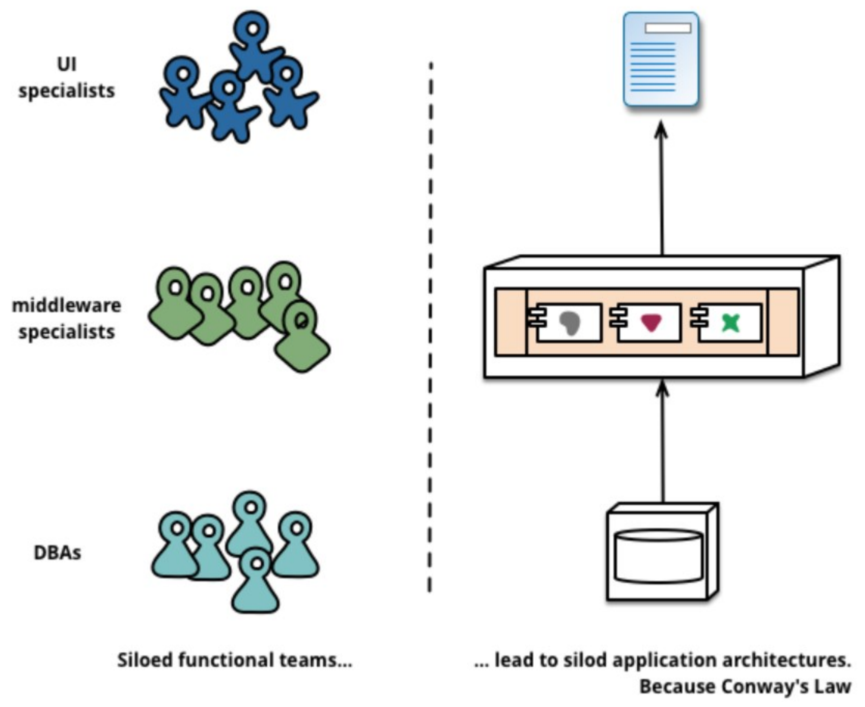
A microservices architecture puts each element of functionality into a separate service...

... and scales by distributing these services across servers, replicating as needed.

MICROSERVICE ARCHITECTURE

https://martinfowler.com/articles/microservices.html

Team organization

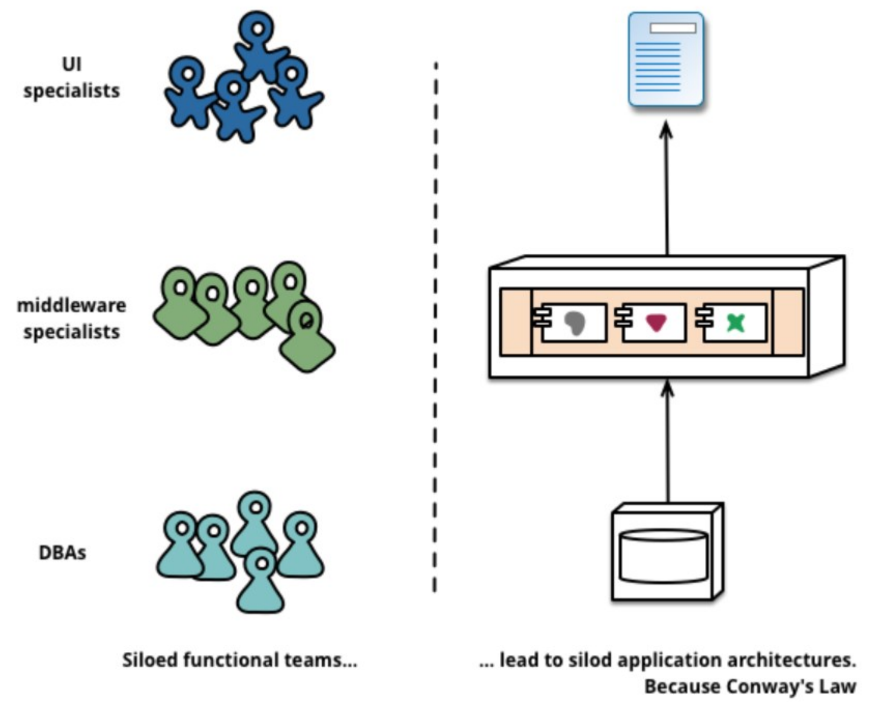

UI specialists
middleware specialists
DBAs
Siloed functional teams...
... lead to silod application architectures. Because Conway's Law

Monolithic (layered)

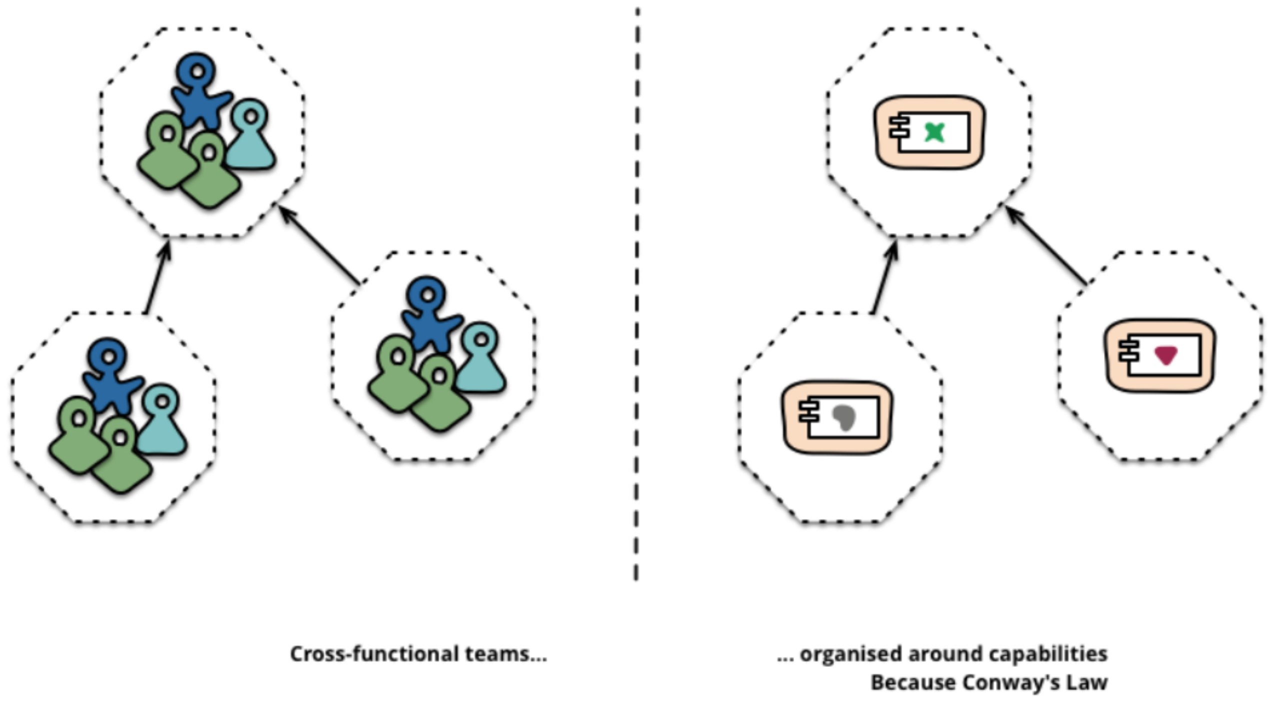
Conway’s Law
“Any organization that designs a system will produce a design whose structure is a copy of the organization's communication structure.”

https://martinfowler.com/articles/microservices.html

Team organization


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

Monolithic (layered)


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

Microservice

https://martinfowler.com/articles/microservices.html

HOW DO MICROSERVICES COMMUNICATE?

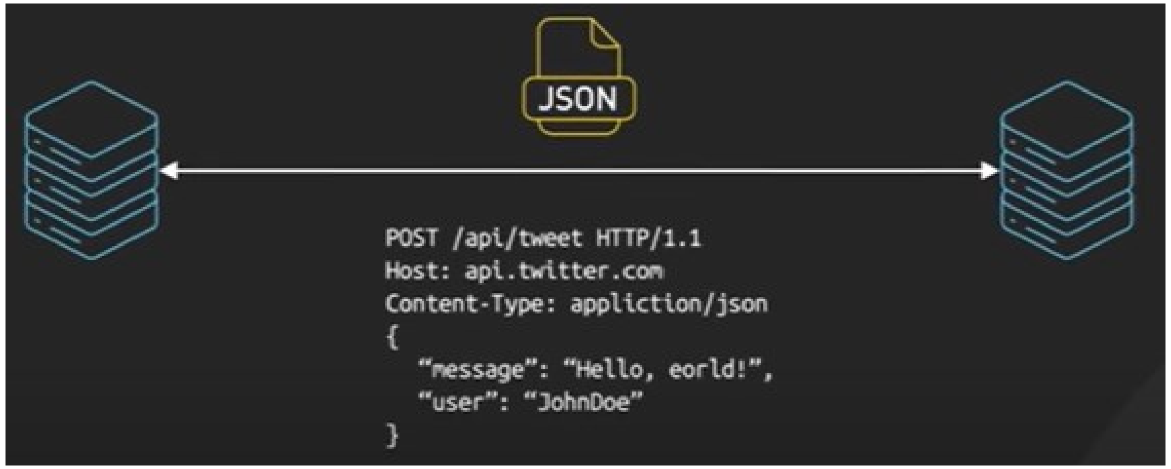
- Synchronous
  - The client expects a timely response from the service and might even block while it waits.
  - RESTful API, gRPC

- Asynchronous:
  - The client doesn't block, and the response, if any, isn't necessarily sent immediately.
  - The Messaging Model

Microservice Patterns: with Examples in Java. Chris Richardson

RESTFUL API

Resource-based API for web servers



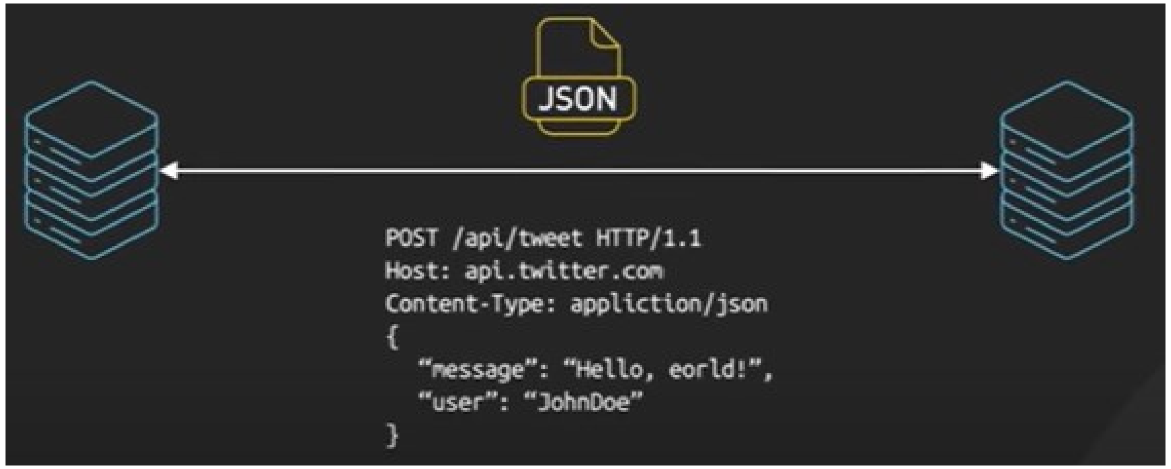
```
POST /api/tweet HTTP/1.1
Host: api.twitter.com
Content-Type: appliction/json
{
  "message": "Hello, eorld!",
  "user": "JohnDoe"
}
```

https://www.youtube.com/@ByteByteGo

- Client-server: A client-server architecture made up of clients, servers, and resources (info like text, image, video)
- Resources could be accessed using URL
- Stateless: Resource requests should be made independently of one another
- Requests are made using HTTP protocol: GET, POST, PUT, DELETE
- Used by X (Twitter), Youtube, etc.

RESTFUL API

Resource-based API for web servers



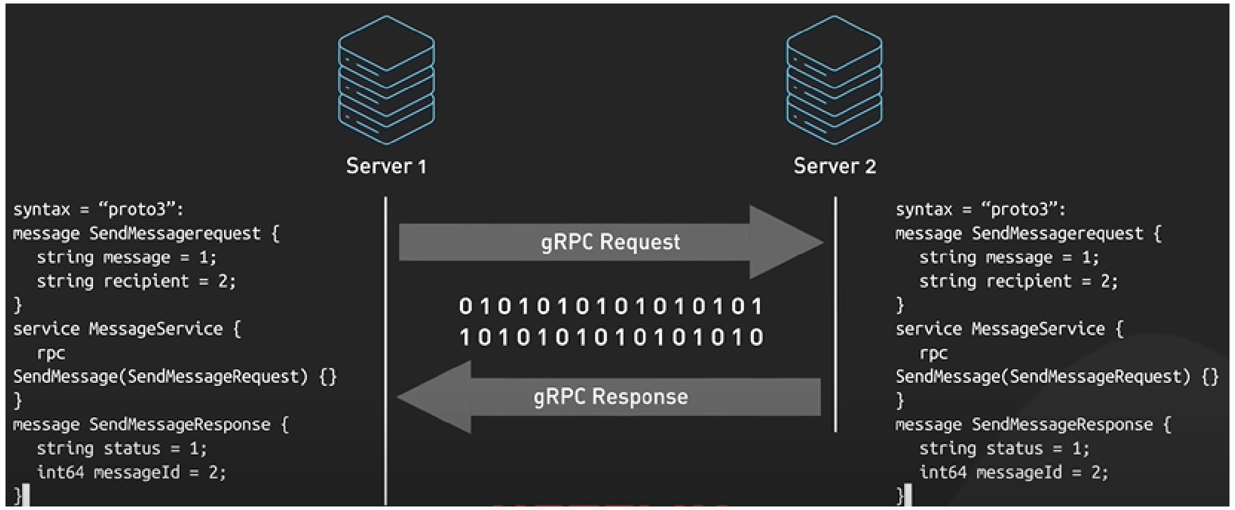
```
POST /api/tweet HTTP/1.1
Host: api.twitter.com
Content-Type: appliction/json
{
  "message": "Hello, eorld!",
  "user": "JohnDoe"
}
```

https://www.youtube.com/@ByteByteGo

- REST API is best suited for applications with simple data sources where resources are well-defined.
- REST API is not suitable for fetching multiple resources in a single request
- Alternatives
  - GraphQL: Meta (Facebook)
  - Netflix Falcor

GRPC

- gRPC is a binary message-based protocol, facilitating efficient communication between distributed systems through Remote Procedure Calls (RPC).
- Support more operations (verbs) than REST
- Best suited for high-performance or data-heavy microservice architectures.
- Used by Netflix, Google, etc.



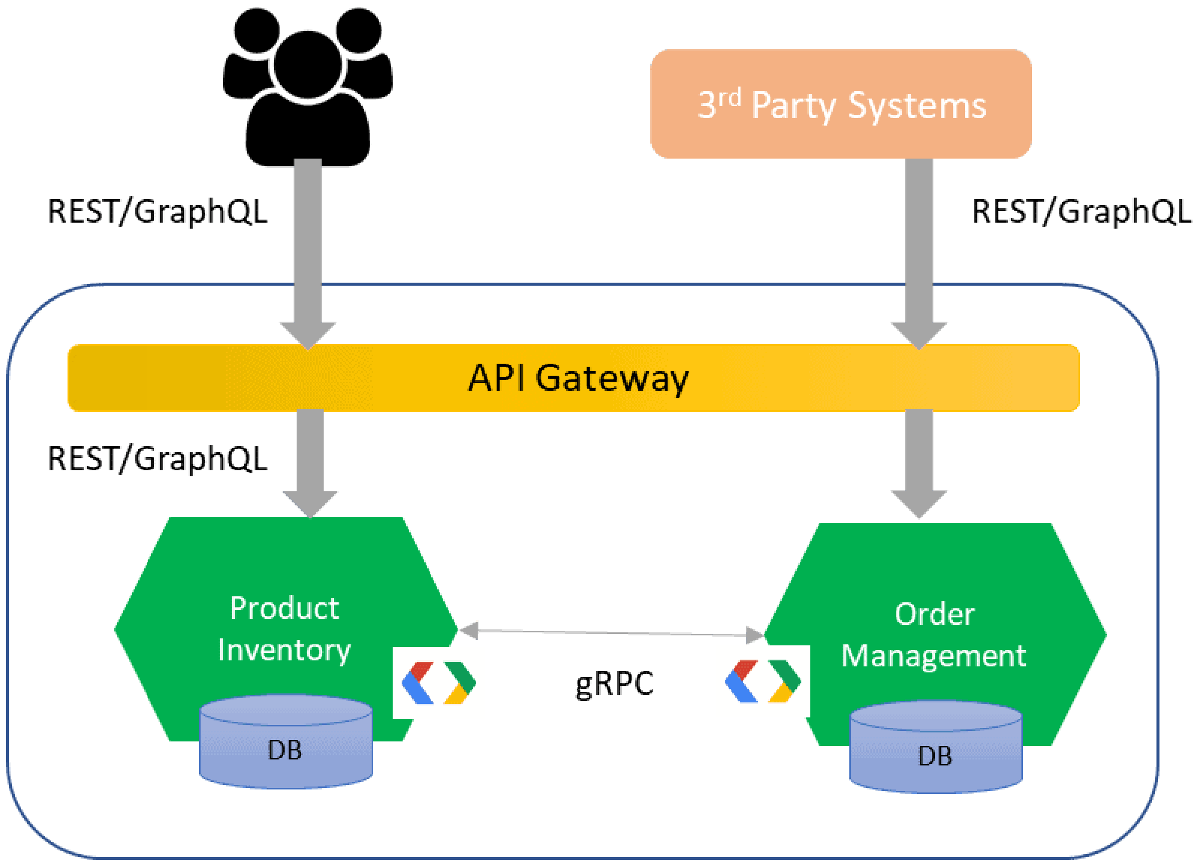
```
syntax = "proto3":
message SendMessagerequest {
  string message = 1;
  string recipient = 2;
}
service MessageService {
  rpc
SendMessage(SendMessageRequest) {}
}
message SendMessageResponse {
  string status = 1;
  int64 messageId = 2;
}
```

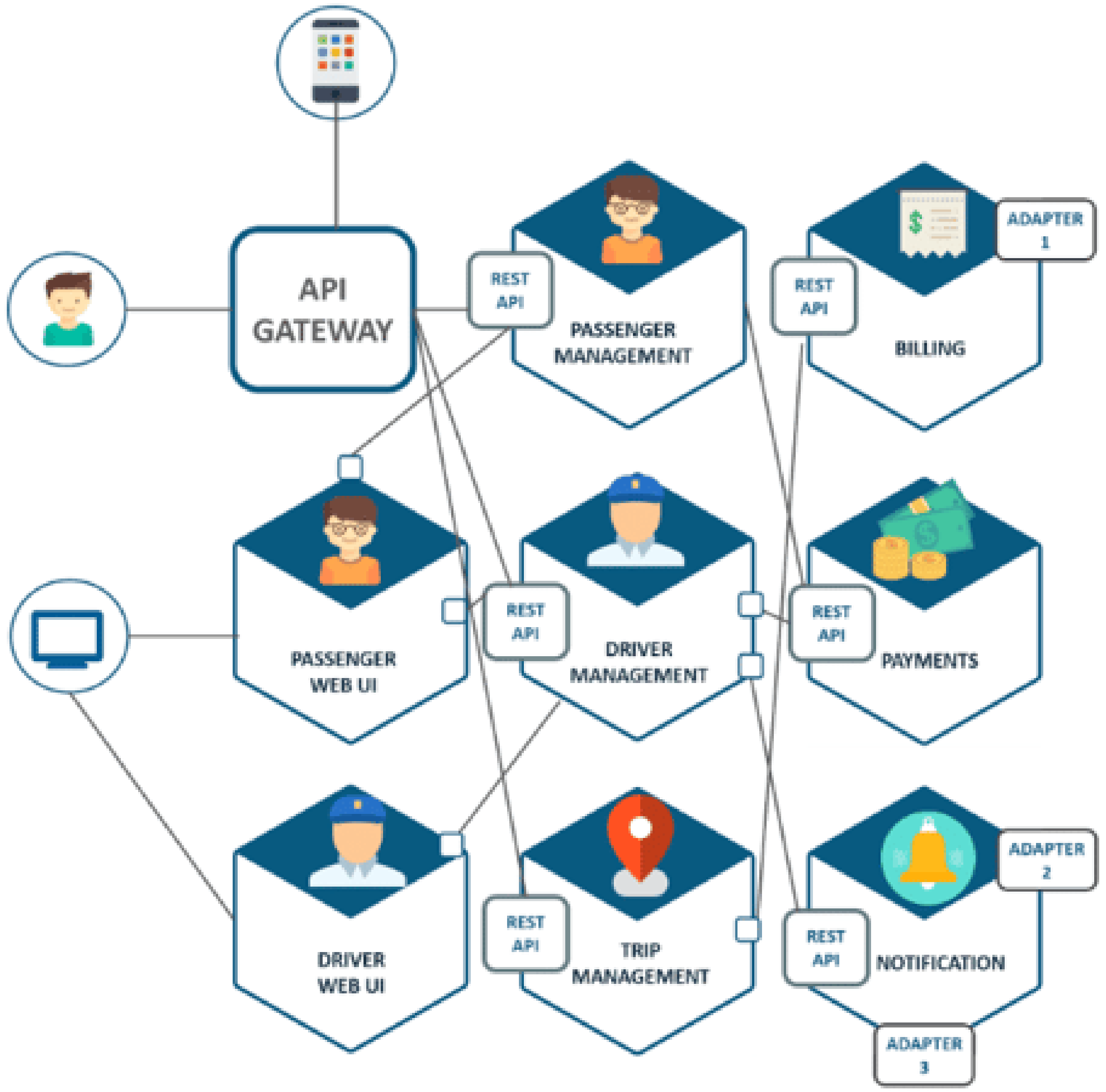
```
syntax = "proto3":
message SendMessagerequest {
  string message = 1;
  string recipient = 2;
}
service MessageService {
  rpc
SendMessage(SendMessageRequest) {}
}
message SendMessageResponse {
  string status = 1;
  int64 messageId = 2;
}
```

https://www.youtube.com/@ByteByteGo

EXAMPLE

- Client-facing service: REST
- Inter-service communication: gRPC





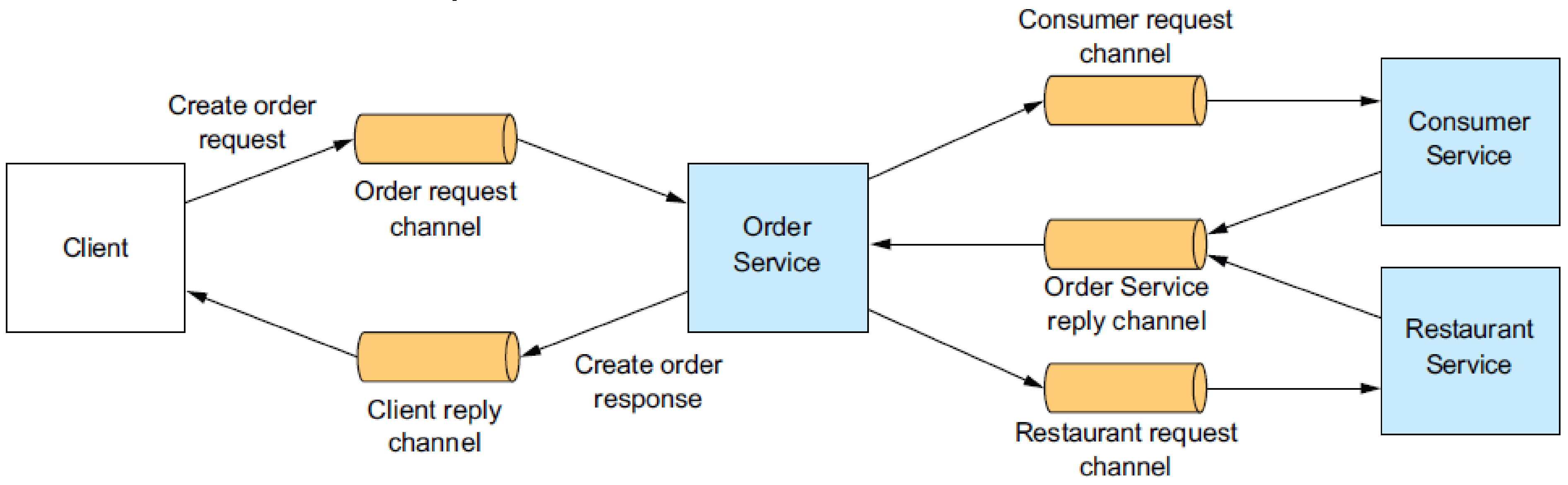
CASE STUDY

Uber's microservice architecture

https://dzone.com/articles/microservice-architecture-learn-build-and-deploy-a

MESSAGING

- In the messaging model, messages are exchanged over message channels.
  - A sender (e.g., a service) writes a message to a channel
  - a receiver reads messages from a channel.
- Open source: RabbitMQ, Apache Kafka



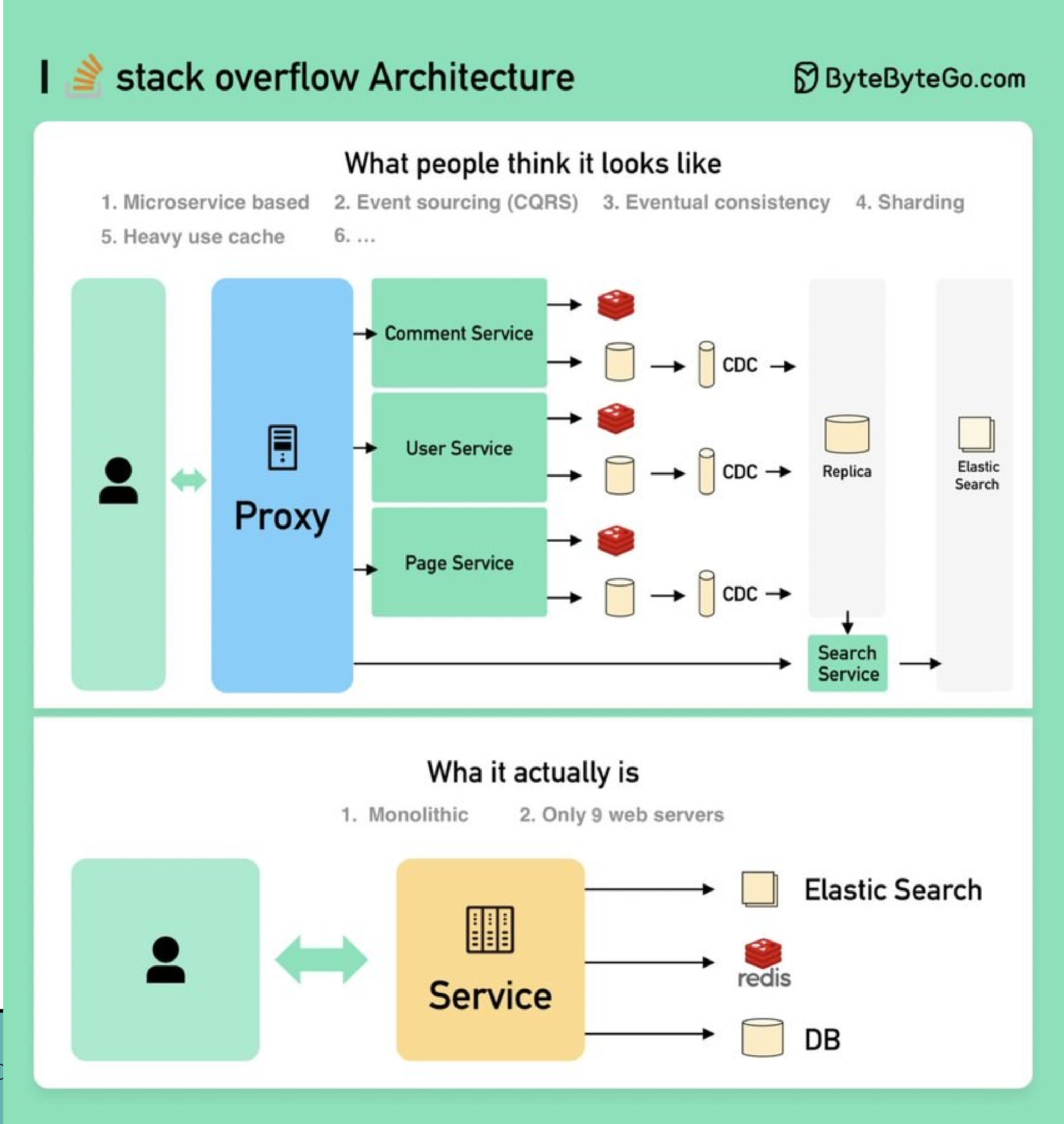
Microservice Patterns: with Examples in Java. Chris Richardson

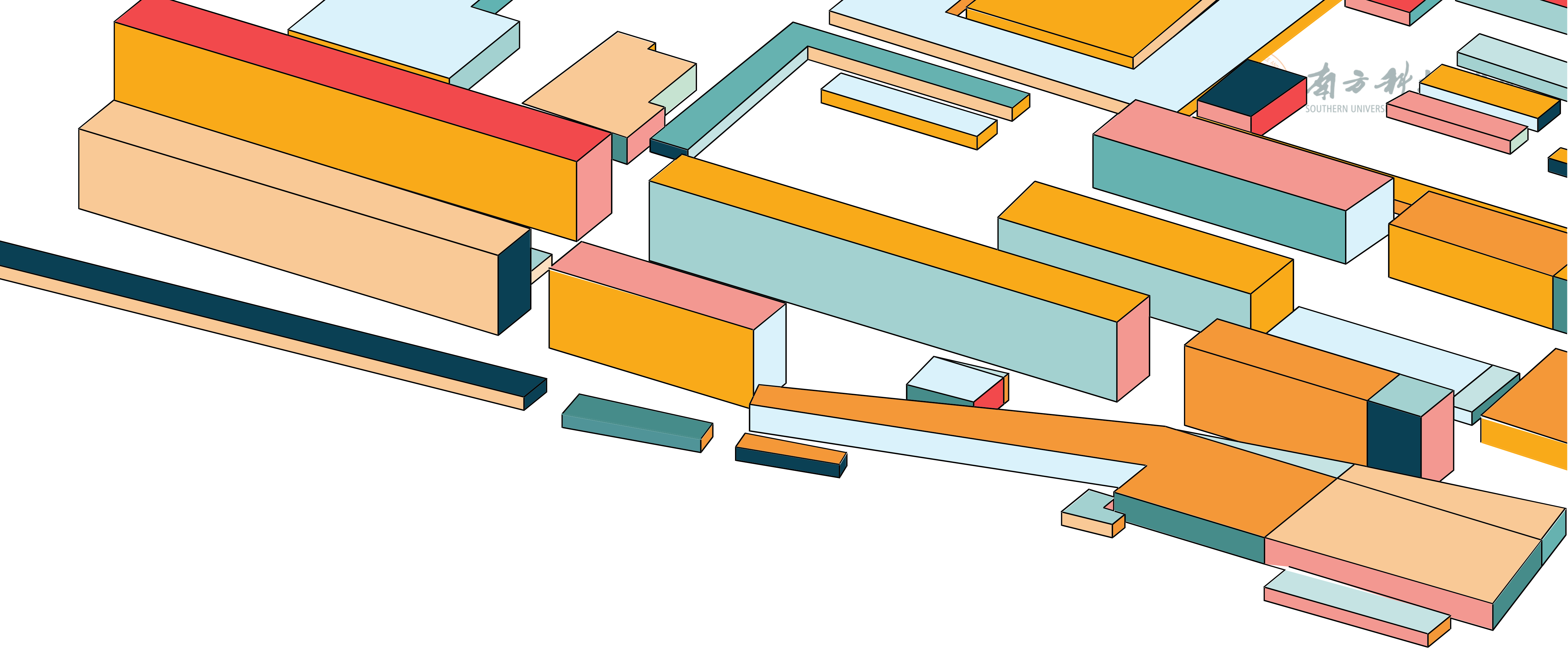
CHOOSING ARCHITECTURAL STYLES

- We've introduced only a subset of available architectural styles
- Once requirements engineering uncovers the characteristics and constraints of the system to be built, the architectural style that best fits those characteristics and constraints can be chosen.
- Different architectural styles are NOT mutually exclusive; instead, they are often applied in combination (e.g., a layered style can be combined with a data-centered architecture in many database applications.)

CHOOSING ARCHITECTURES

- No good or bad architecture (We are not saying microservice is better than monolithic)
- Choose what's suitable based on your application, resources, user base, business style, team structure, etc.

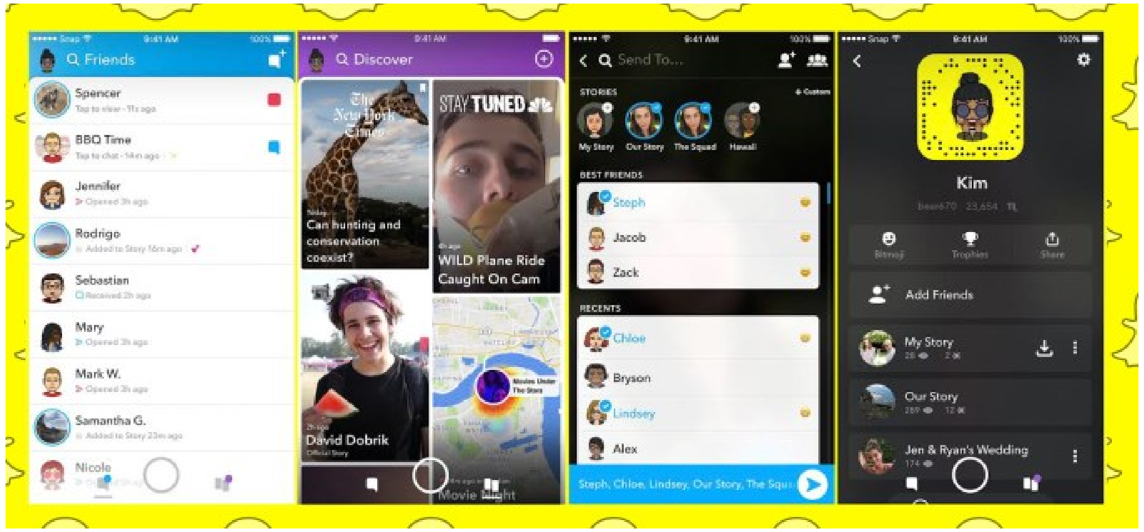


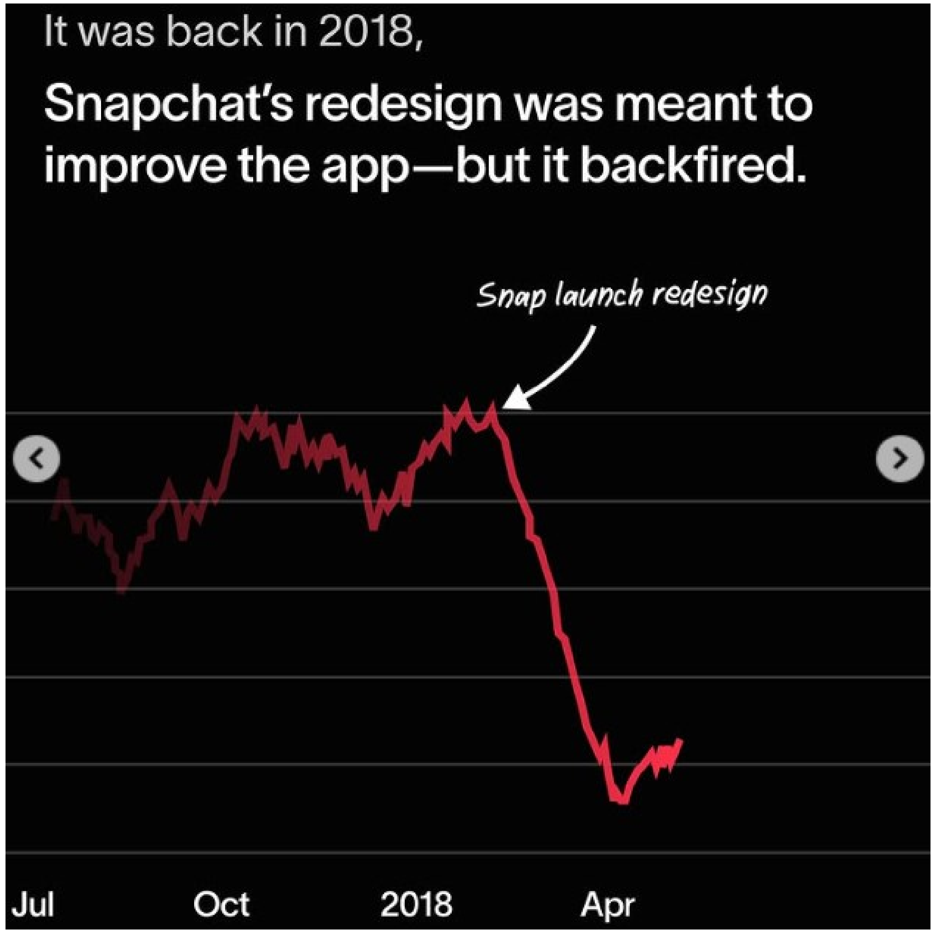


SOFTWARE UI DESIGN

UI DESIGN FAILURE

In 2018, Snapchat's UI redesign faced significant backlash and was widely considered a failure.

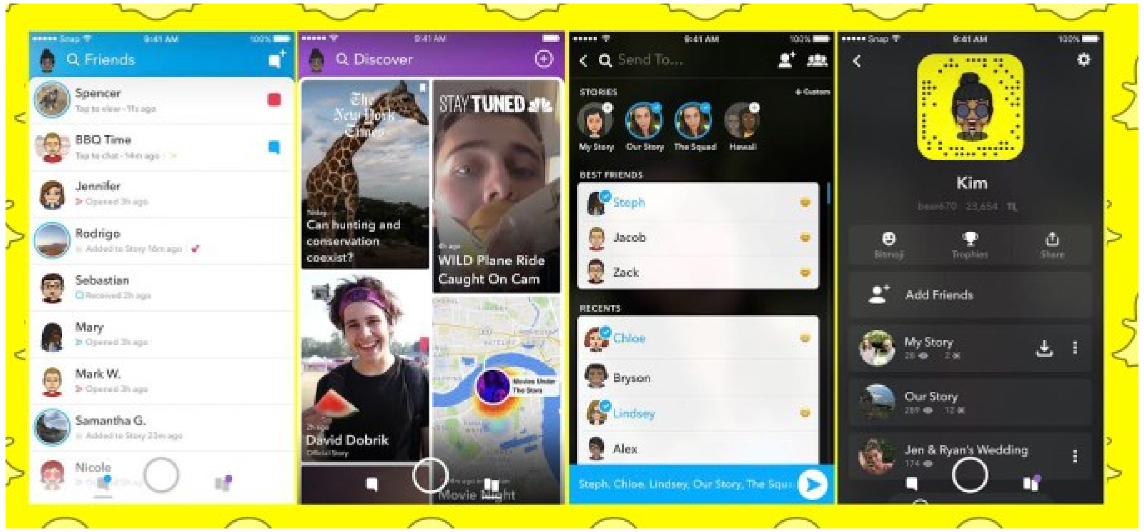


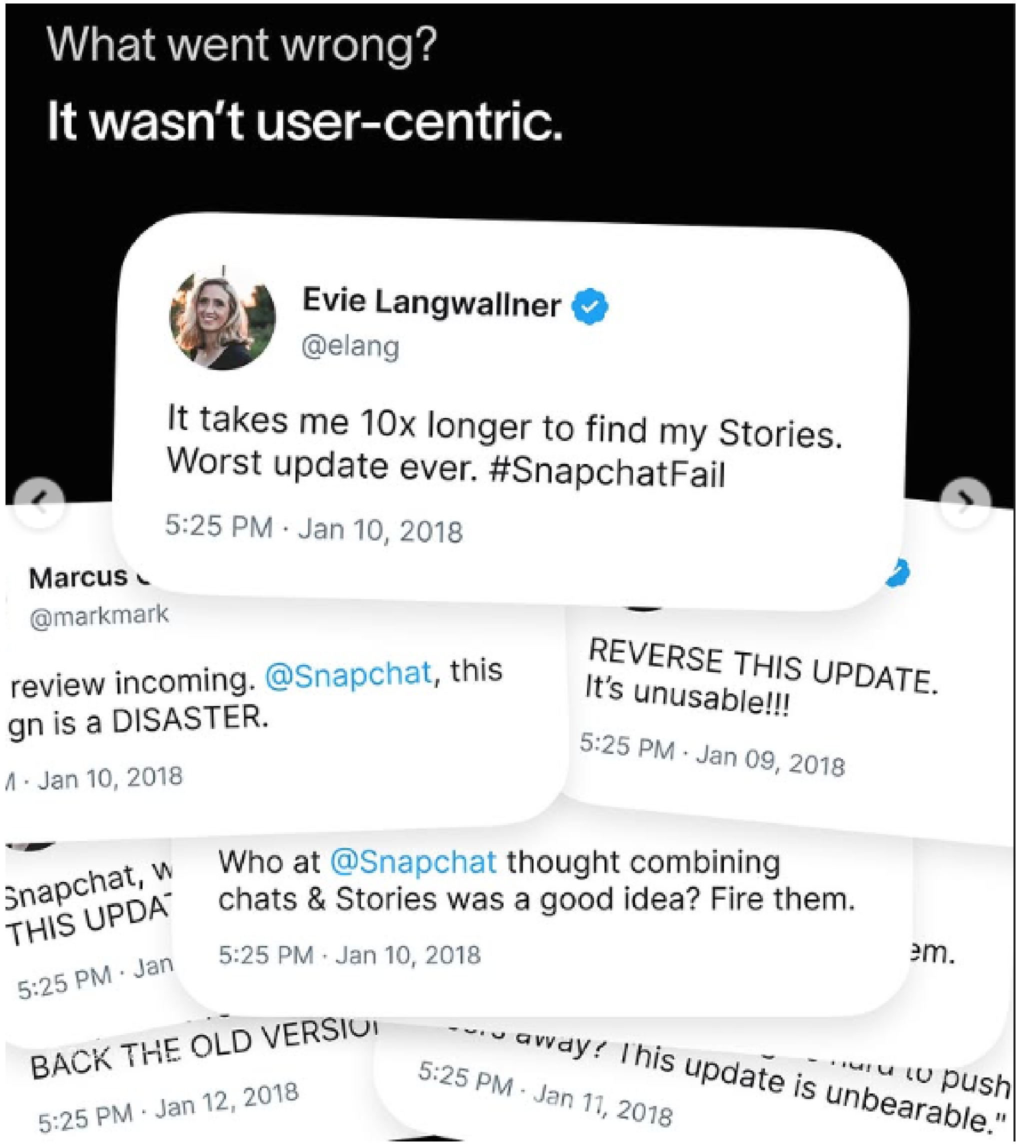


Source: https://www.instagram.com/won.agency/p/DEzluNrJK6B/?img_index=2

UI DESIGN FAILURE

Over 1.2 million users signed a petition requesting the company to revert the design changes





What went wrong?

It wasn't user-centric.

Evie Langwallner @elang

It takes me 10x longer to find my Stories. Worst update ever. #SnapchatFail

5:25 PM · Jan 10, 2018

Marcus @markmark

review incoming. @Snapchat, this gn is a DISASTER.

M · Jan 10, 2018

REVERSE THIS UPDATE. It's unusable!!!

5:25 PM · Jan 09, 2018

Snapchat, w THIS UPDA

5:25 PM · Jan

Who at @Snapchat thought combining chats & Stories was a good idea? Fire them.

5:25 PM · Jan 10, 2018

em.

BACK THE OLD VERSIO

5:25 PM · Jan 12, 2018

away? This update is unbearable."

5:25 PM · Jan 11, 2018

Source: https://www.instagram.com/won.agency/p/DEzluNrJK6B/?img_index=2

UI DESIGN

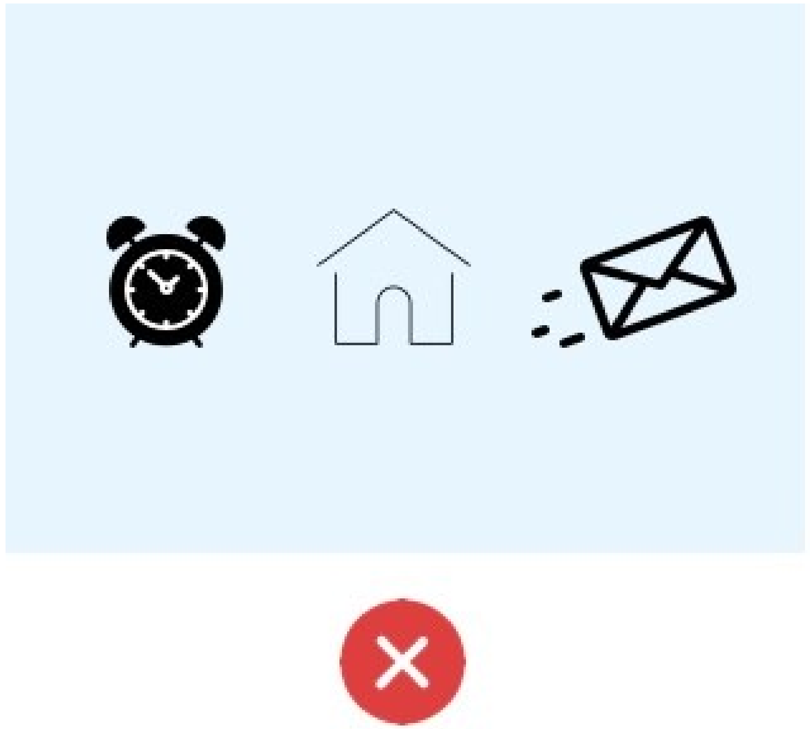
Visual Design

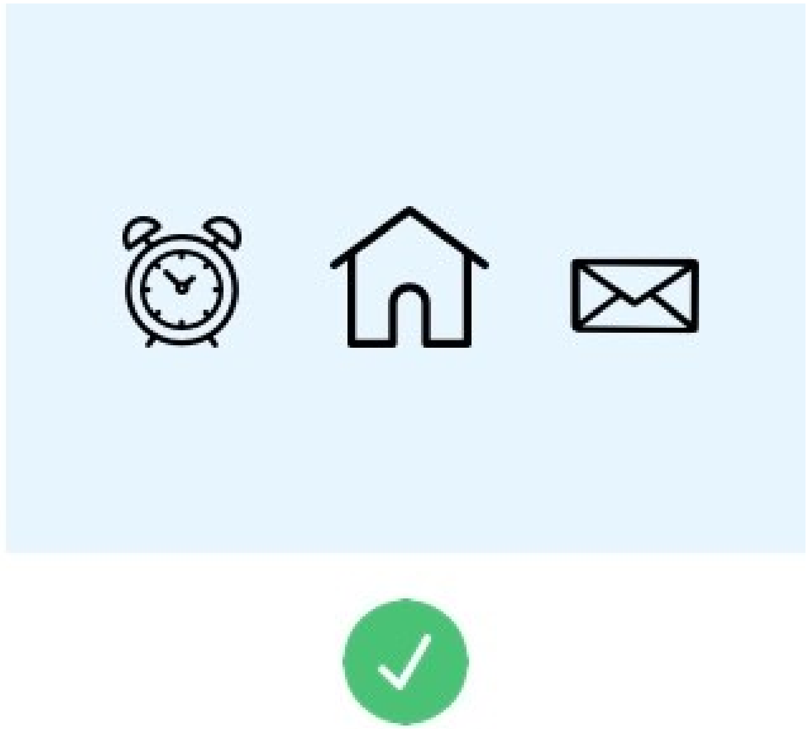
- Focuses on aesthetics, layout, color, typography, and visual hierarchy.
- Enhance usability and user experience.

Interaction Design

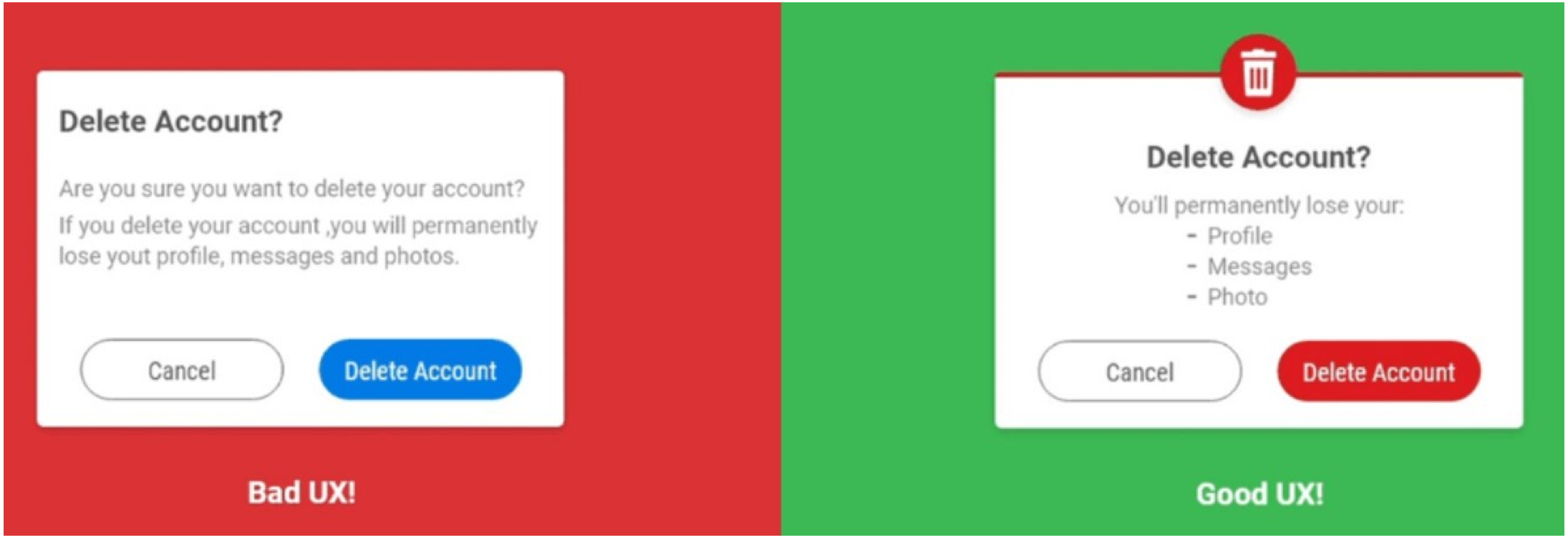
- How users interact with UI elements (buttons, forms, navigation).
- Goal: Minimize friction and cognitive load.

CONSISTENCY





CLARITY


Delete Account?

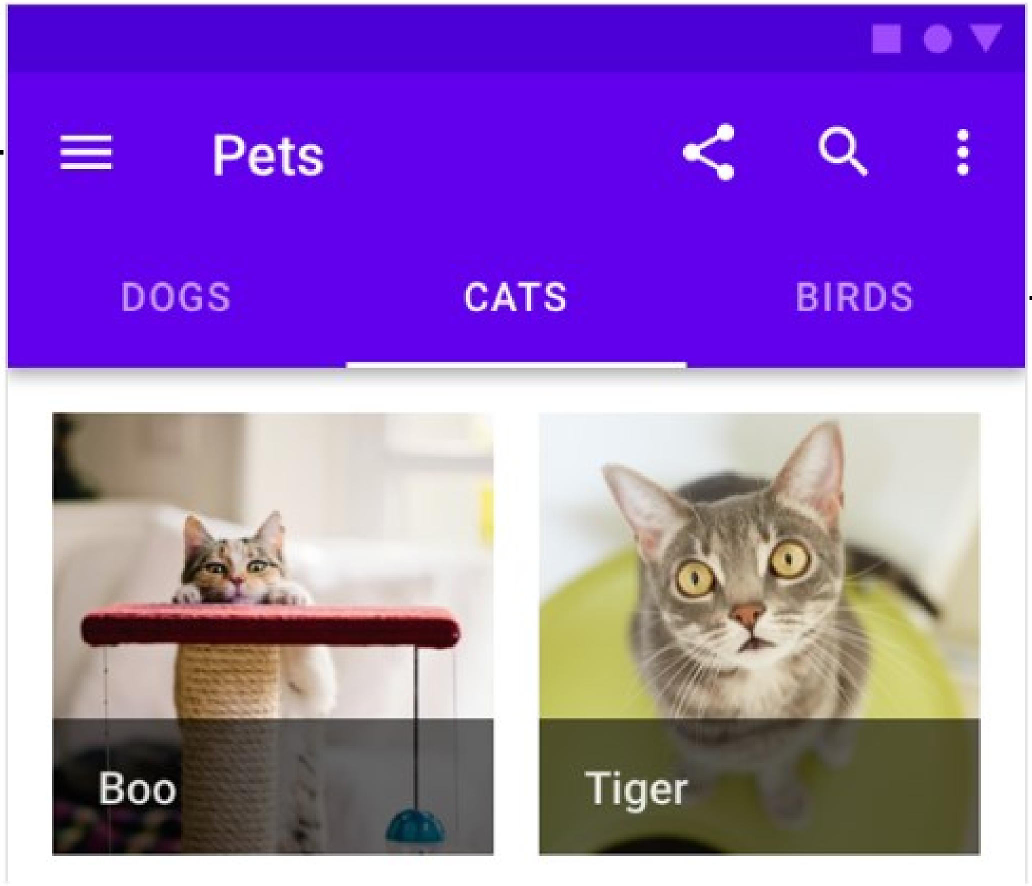
Are you sure you want to delete your account? If you delete your account ,you will permanently lose yout profile, messages and photos.

Delete Account?

You'll permanently lose your:
- Profile
- Messages
- Photo

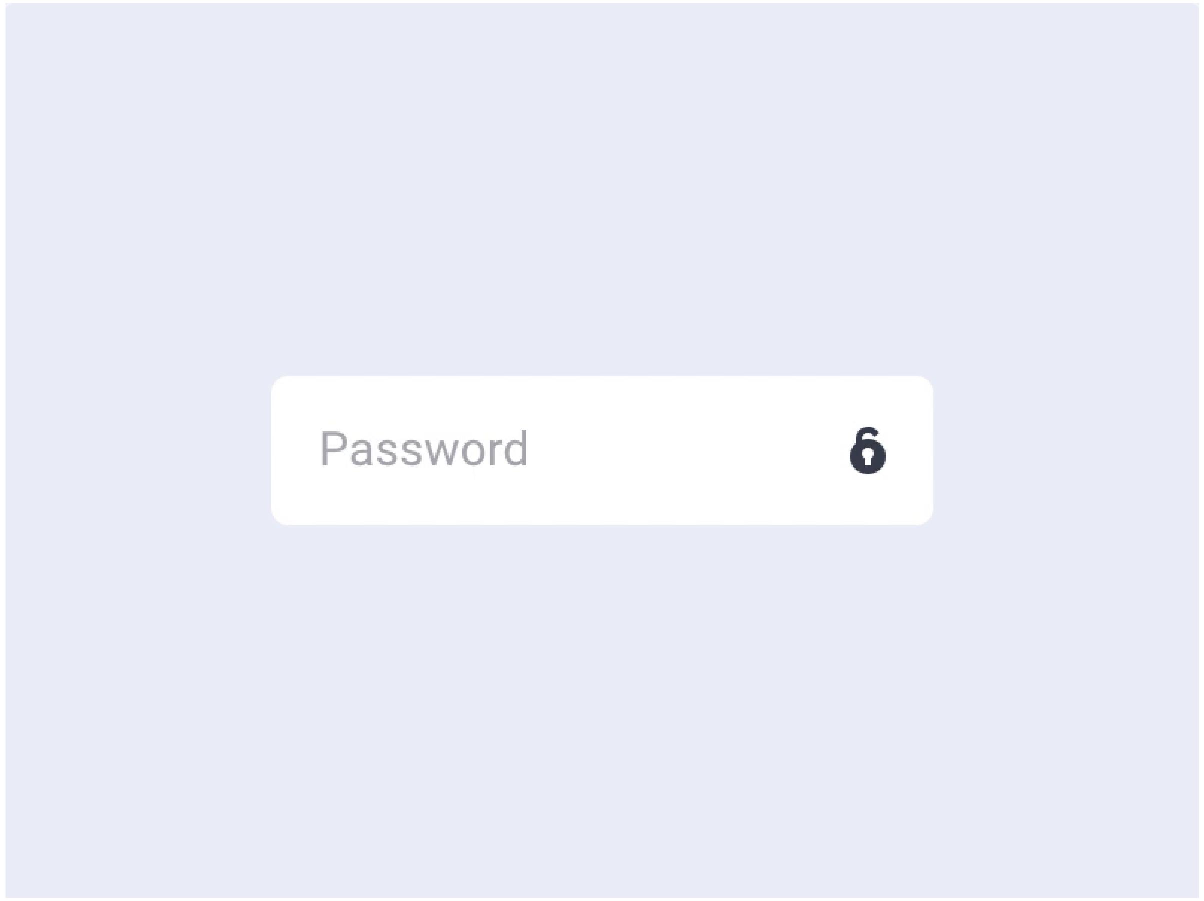
ACCESSIBILITY

汉堡菜单



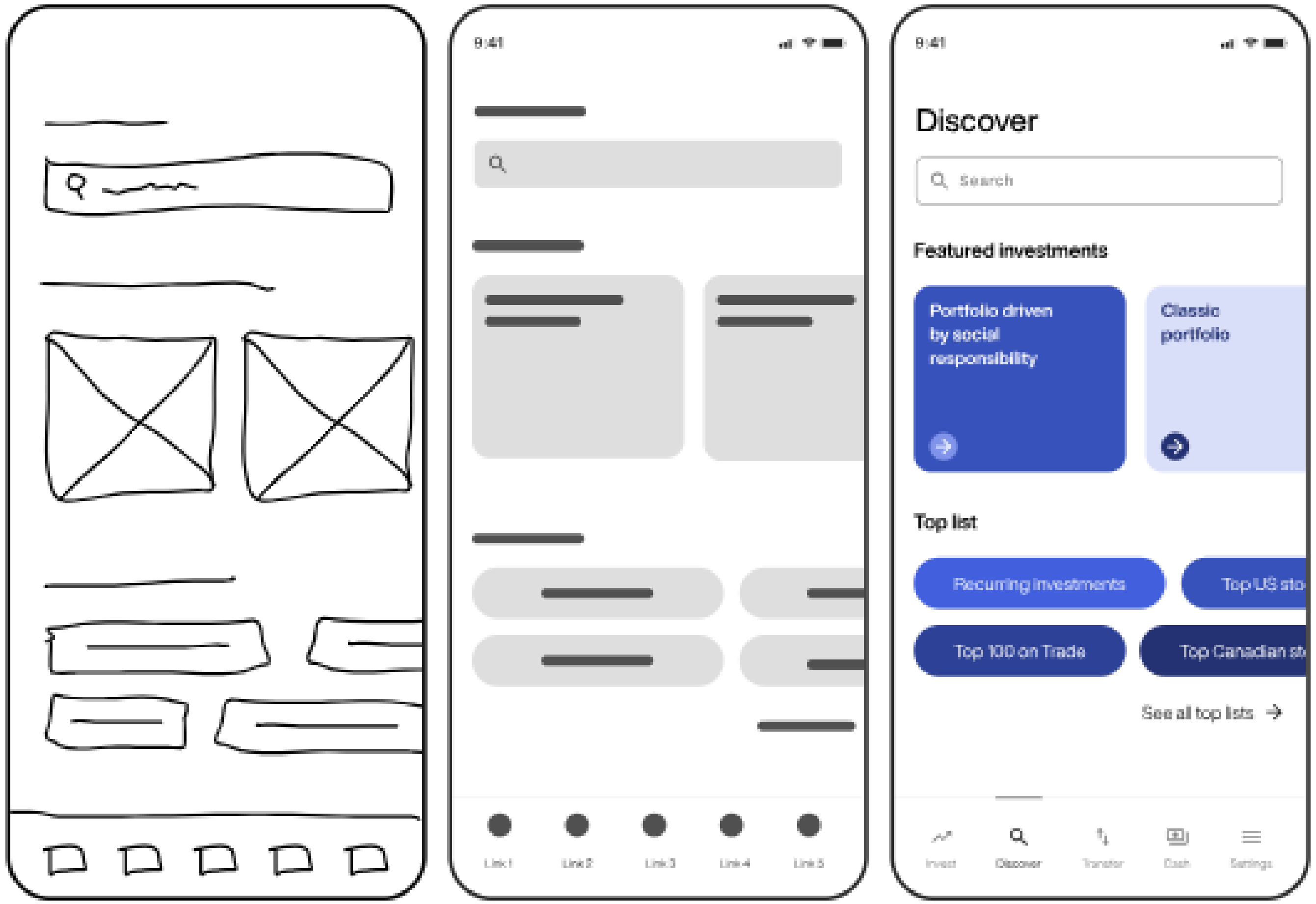
标签菜单

FEEDBACK



PROTOTYPING

- A prototype is an early sample or model of a user interface used to test concepts and gather feedback.
- Helps visualize design, test usability, and refine interactions before development.

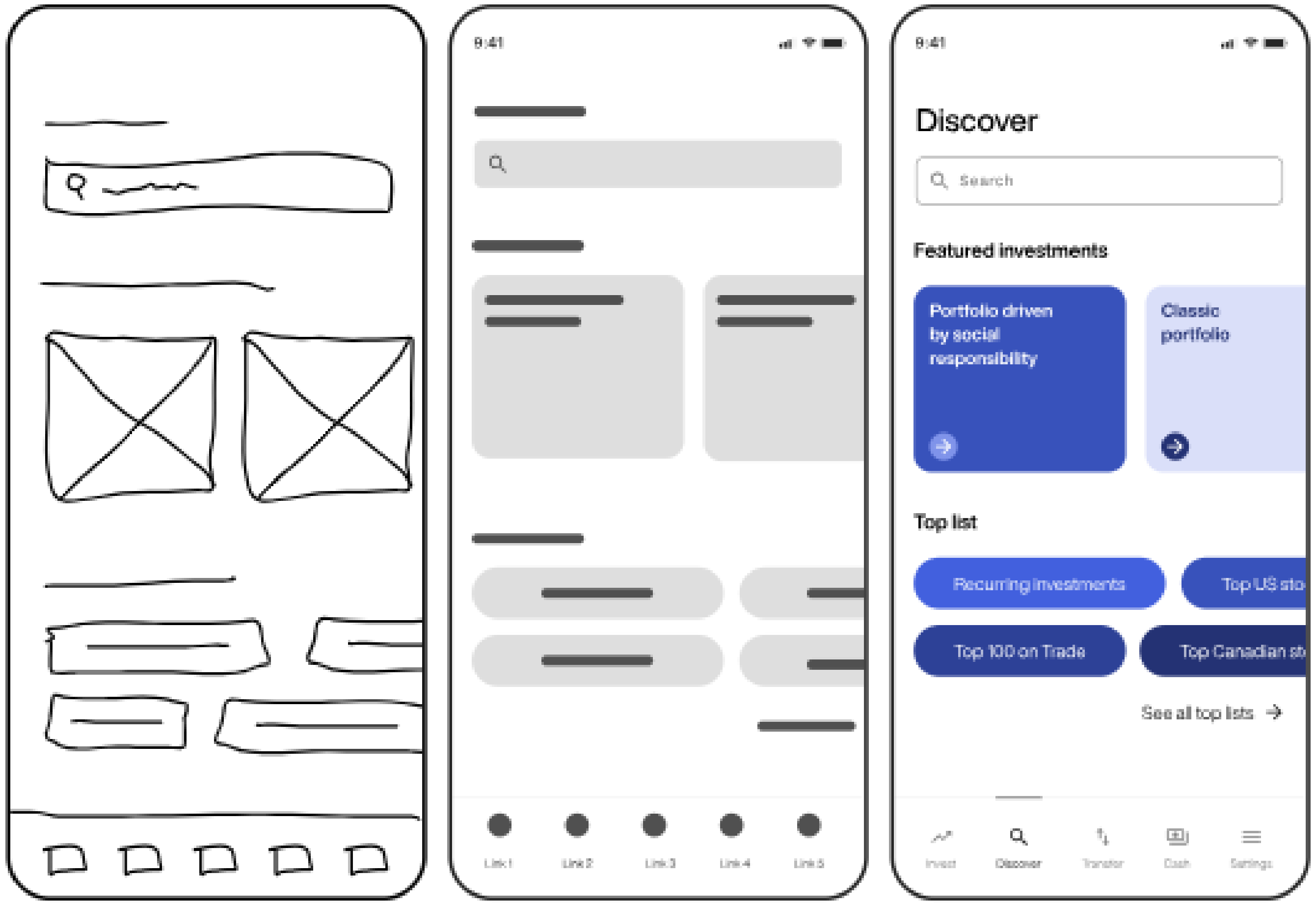


PROTOTYPING

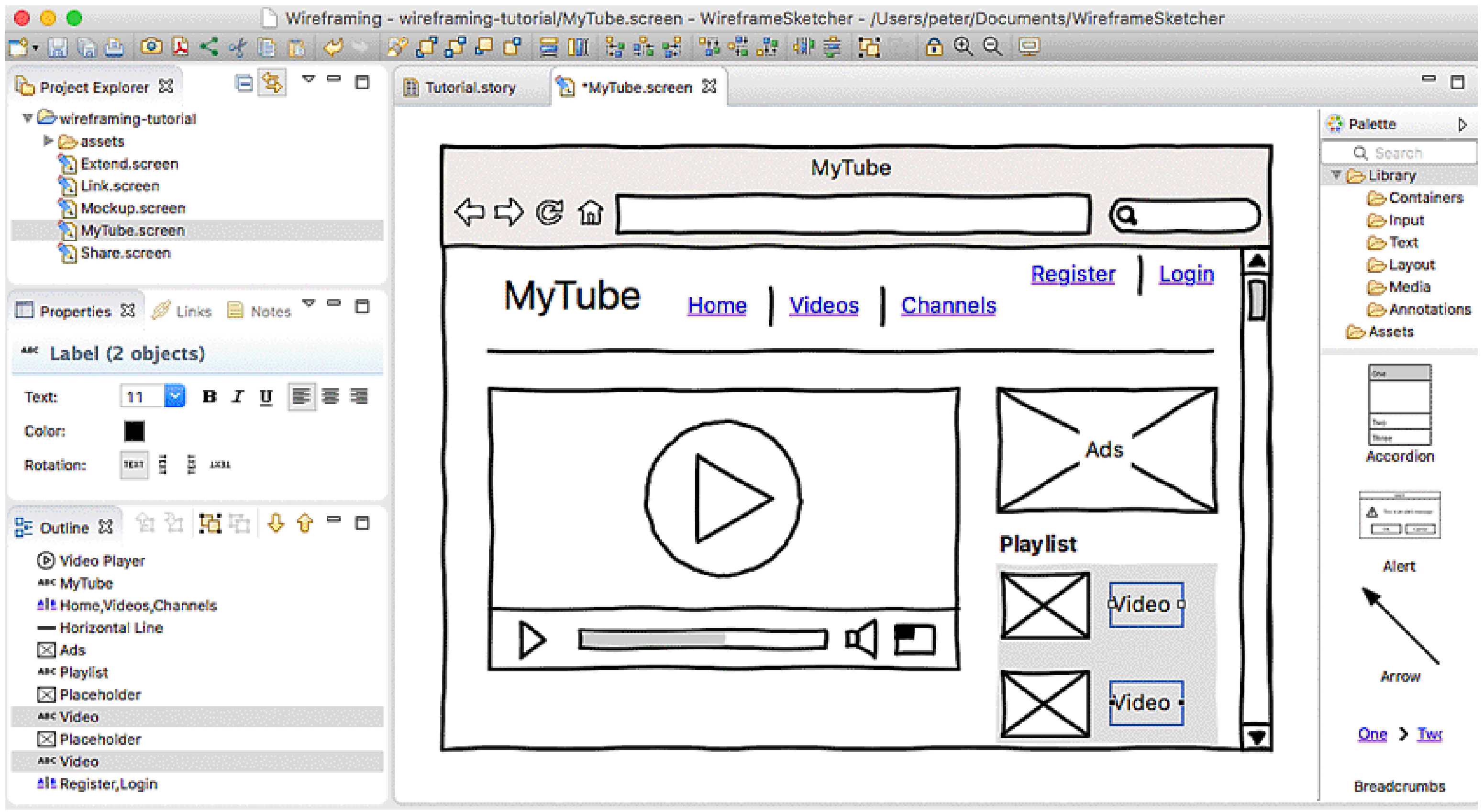
Low-Fidelity (Lo-Fi) Prototypes (低保真)
- Hand-drawn sketches or wireframes.
- Quick and cost-effective.

Mid-Fidelity Prototypes (中保真)
- Digital wireframes with basic interactions.
- Used for layout and navigation testing.

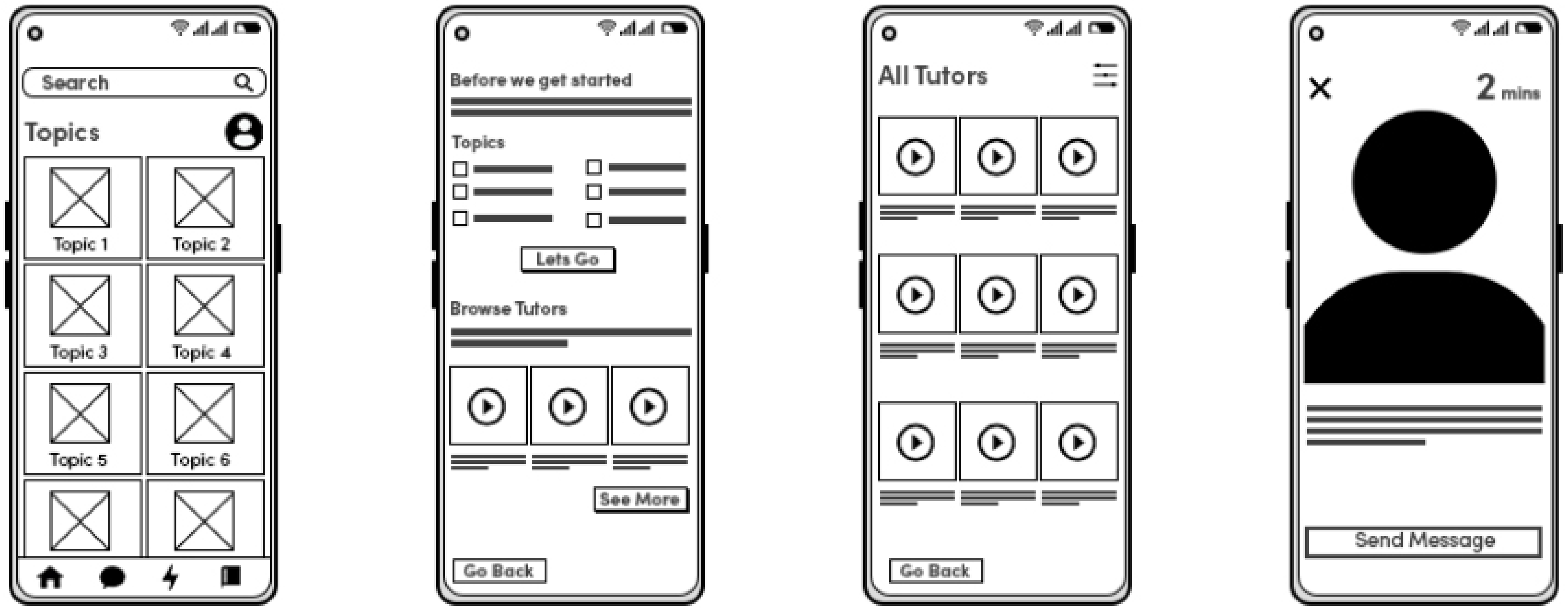
High-Fidelity (Hi-Fi) Prototypes (高保真)
- Interactive and visually detailed.
- Simulates final product experience.



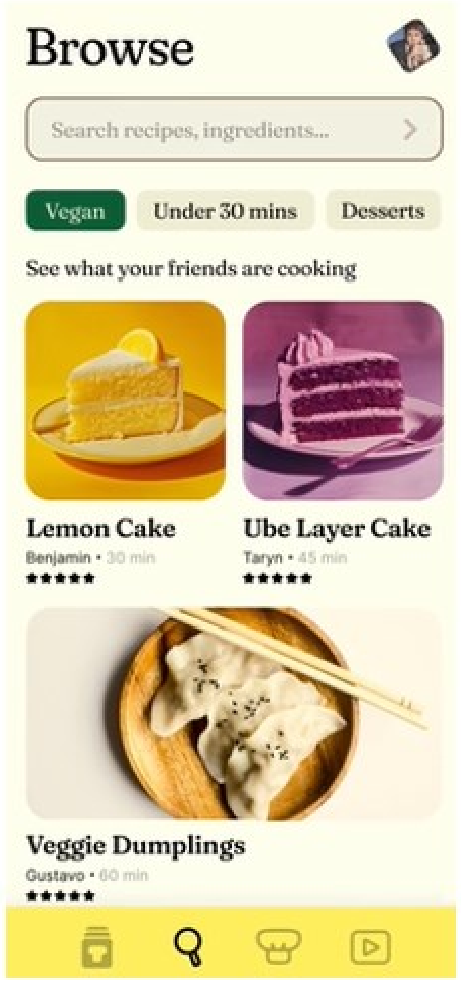
WIREFRAMING TOOLS



WIREFRAMING TOOLS



> Make a form with filters for a recipe app


Browse

Search recipes, ingredients...

Vegan

Under 30 mins

Desserts

See what your friends are cooking

Lemon Cake
Benjamin • 30 min
★★★★★

Ube Layer Cake
Taryn • 45 min
★★★★★

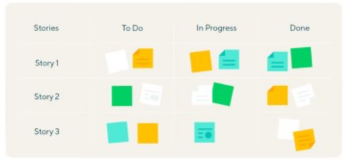
Veggie Dumplings
Gustavo • 60 min
★★★★★

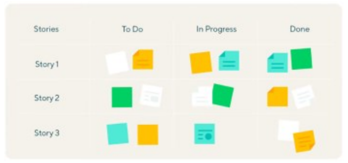
AI TOOLS FOR UI DESIGN

Image source: Figma

OUR TEAM PROJECT

1. Architectural design
2. UI design
3. Git collaboration
4. Demo
5. Scrum board for the next sprint





Week 1
Team up

Week 5
Proposal

Week 9
Sprint 1

Week 16
Sprint 2

NEXT

- Build Systems