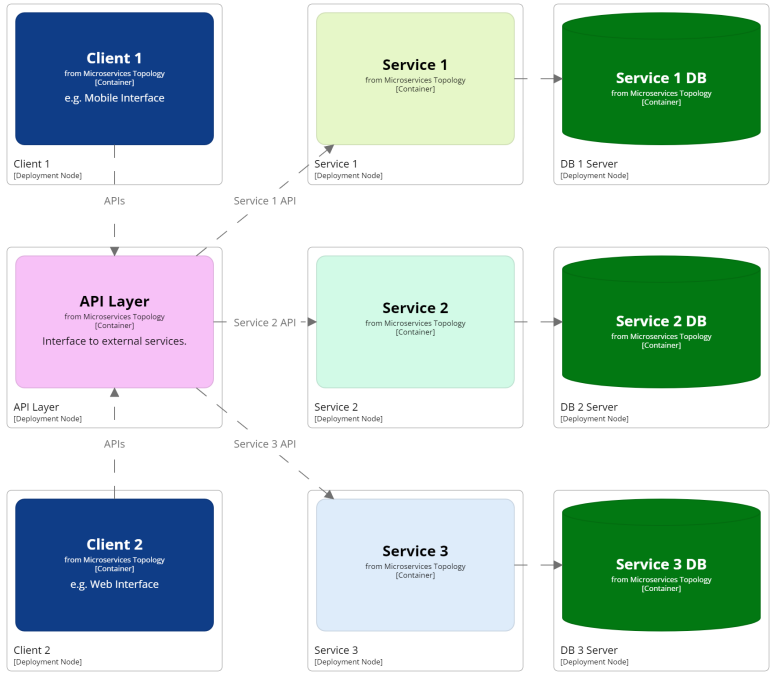
Microservices Architecture

Software Architecture

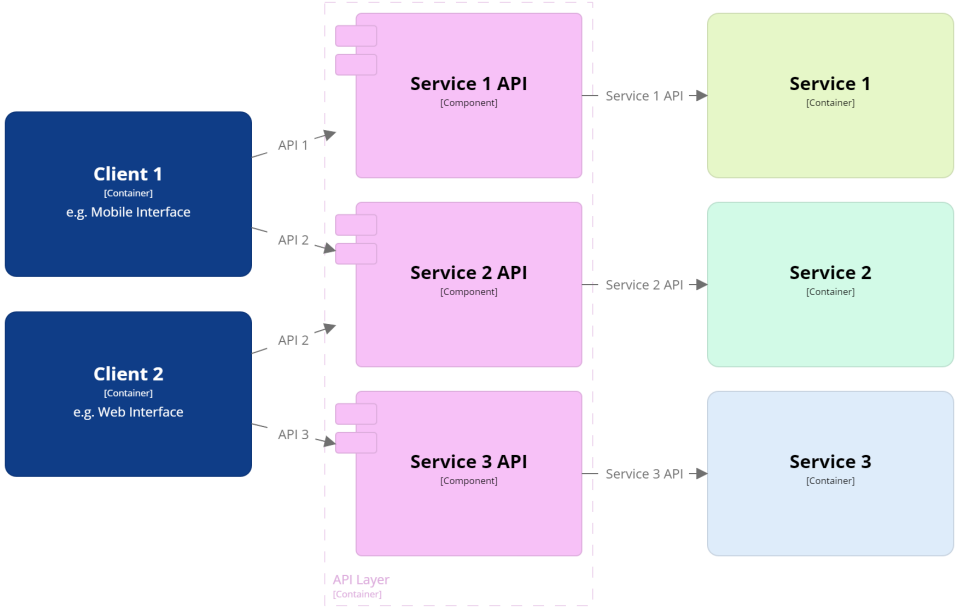
Richard Thomas

April 15, 2024

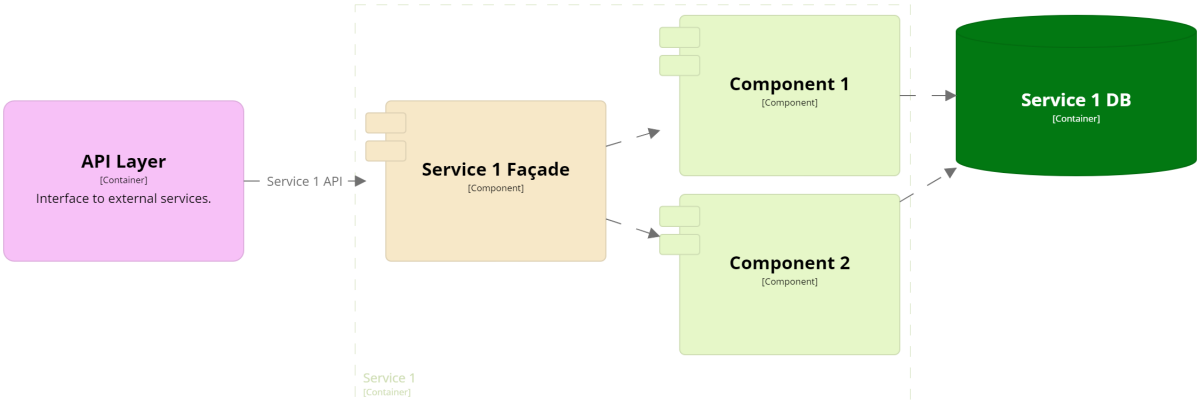
Microservices General Topology



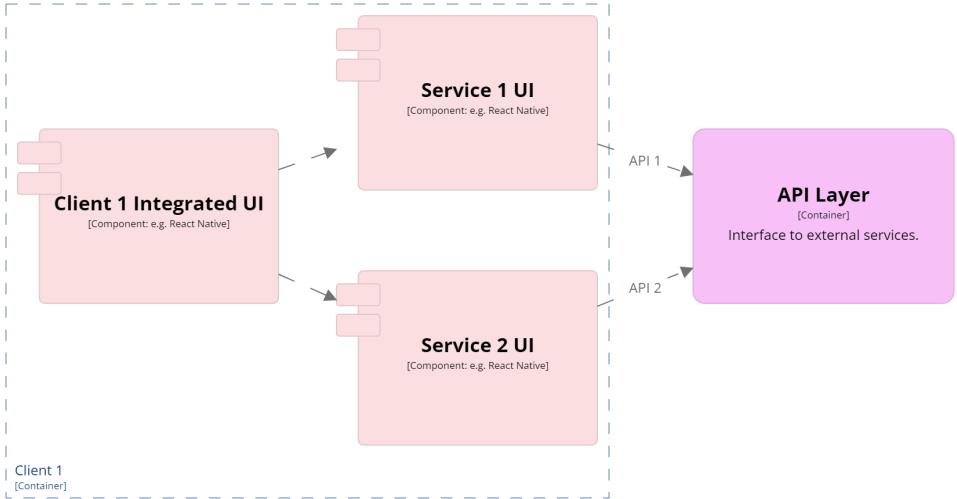
API Layer Components



Service 1 Components



Client with Monolithic UI



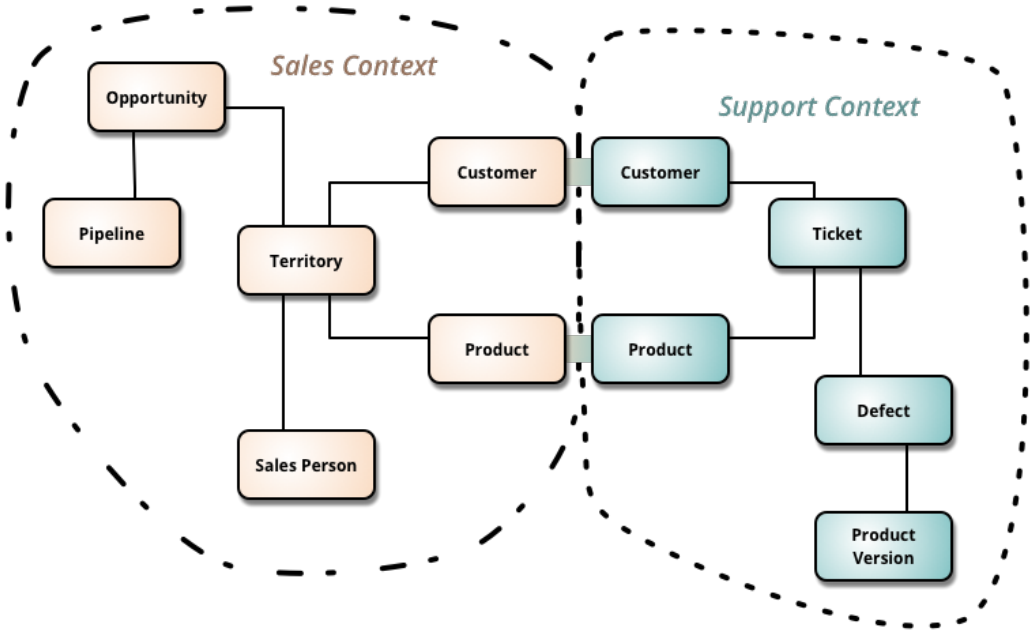
DDD Influence

Services are *bounded contexts*.

Bounded contexts are not necessarily *services*.

Definition 1. Bounded Context

Logical boundary of a domain where particular terms and rules apply consistently.



From https://martinfowler.com/bliki/BoundedContext.html

Definition 2. Service Cohesion Principle

Services are cohesive business processes.
They are a bounded context.

Large Bounded Contexts

A bounded context may be too large to be a single service.

Split it into services that are *independent* sub-processes.

Definition 3. Service Independence Principle

Services should not depend on the implementation of other services.

Corollary 1. Low Coupling

There should be minimal coupling between services.

Corollary 2. No Reuse

Avoid dependencies between services.

Do not reuse components between services.

Bounded Domains Implications

- Duplication
  - Entities specialised for domain
    - Requires mapping of entity data between domains

Bounded Domains Implications

- Duplication
  - Entities specialised for domain
    - Requires mapping of entity data between domains
  - Should everything be duplicated?

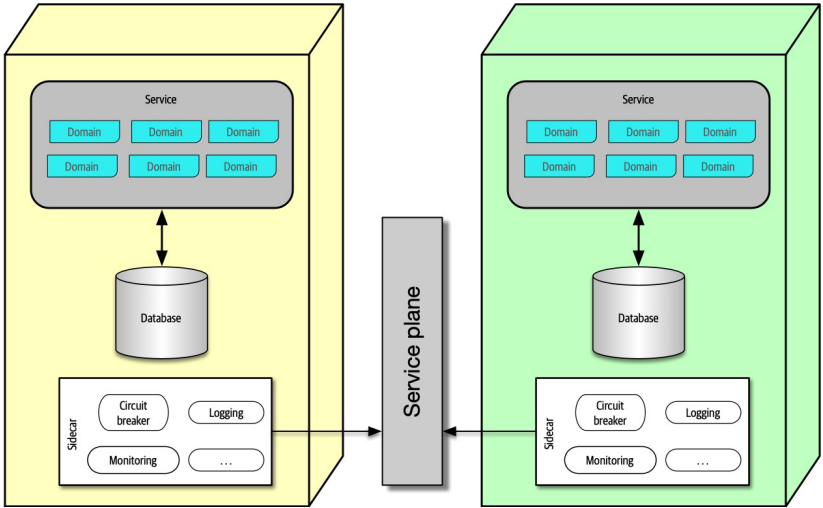
Bounded Domains Implications

- Duplication
  - Entities specialised for domain
    - Requires mapping of entity data between domains
  - Should everything be duplicated?
    - What about common services (e.g. logging, ...)?

Bounded Domains Implications

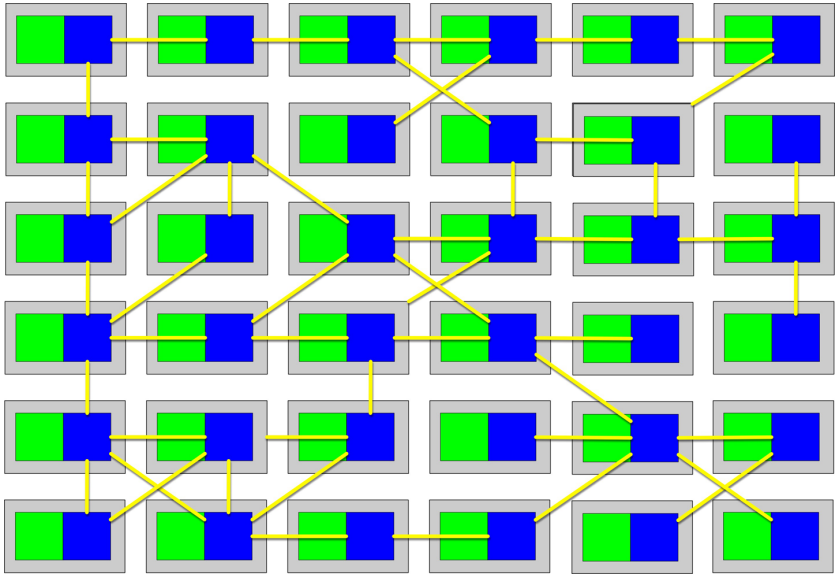
- Duplication
  - Entities specialised for domain
    - Requires mapping of entity data between domains
  - Should everything be duplicated?
    - What about common services (e.g. logging, ...)?
- Heterogeneity
  - Services can use different implementation technologies

Service Plane



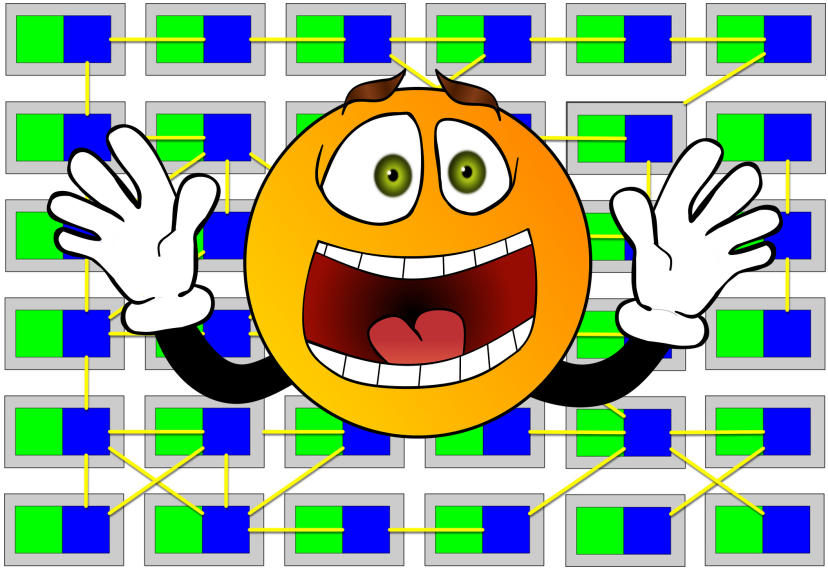
From *Fundamentals of Software Architecture*

Service Mesh



From *Fundamentals of Software Architecture*

Service Mesh



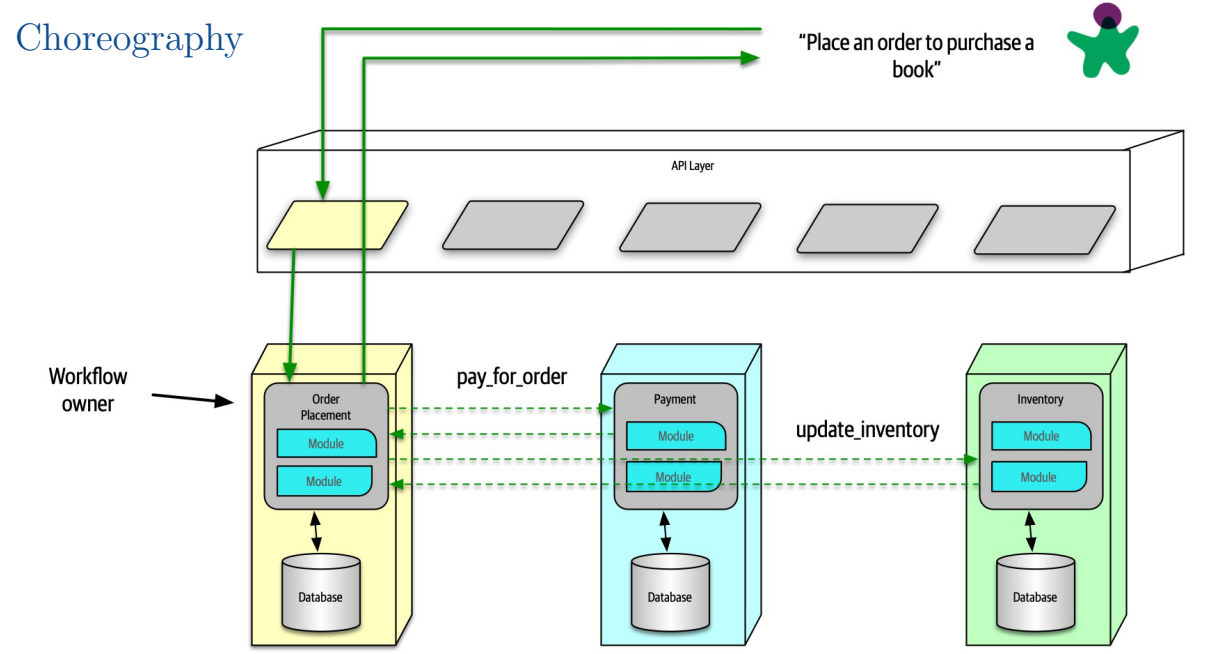
From *Fundamentals of Software Architecture*

Choreography & Orchestration

Choreography Similar to event-driven *broker*

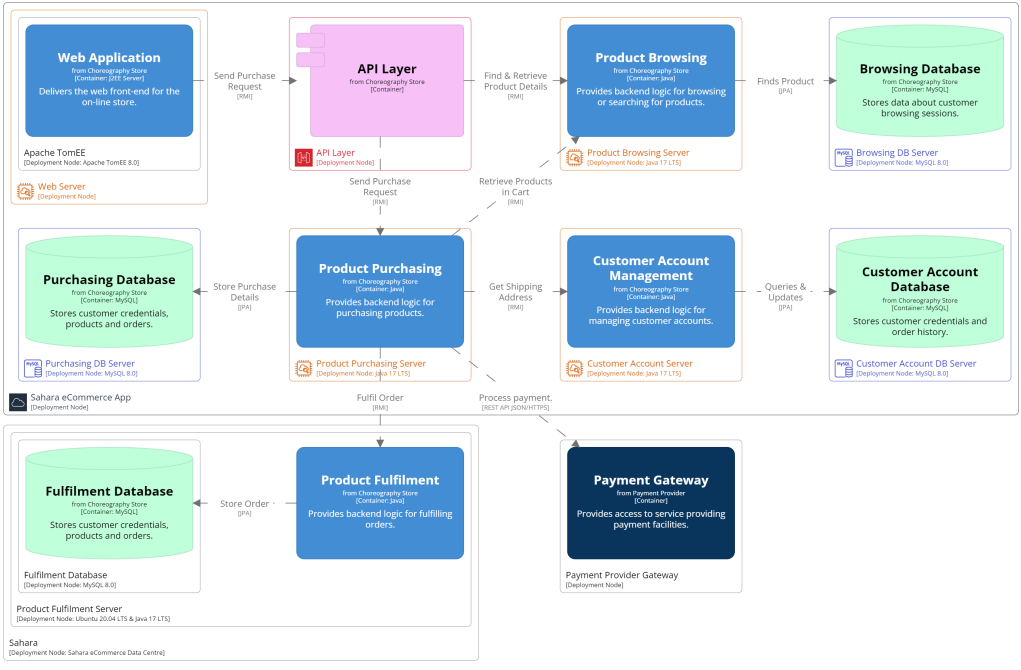
Orchestration Similar to event-driven *mediator*

Choreography

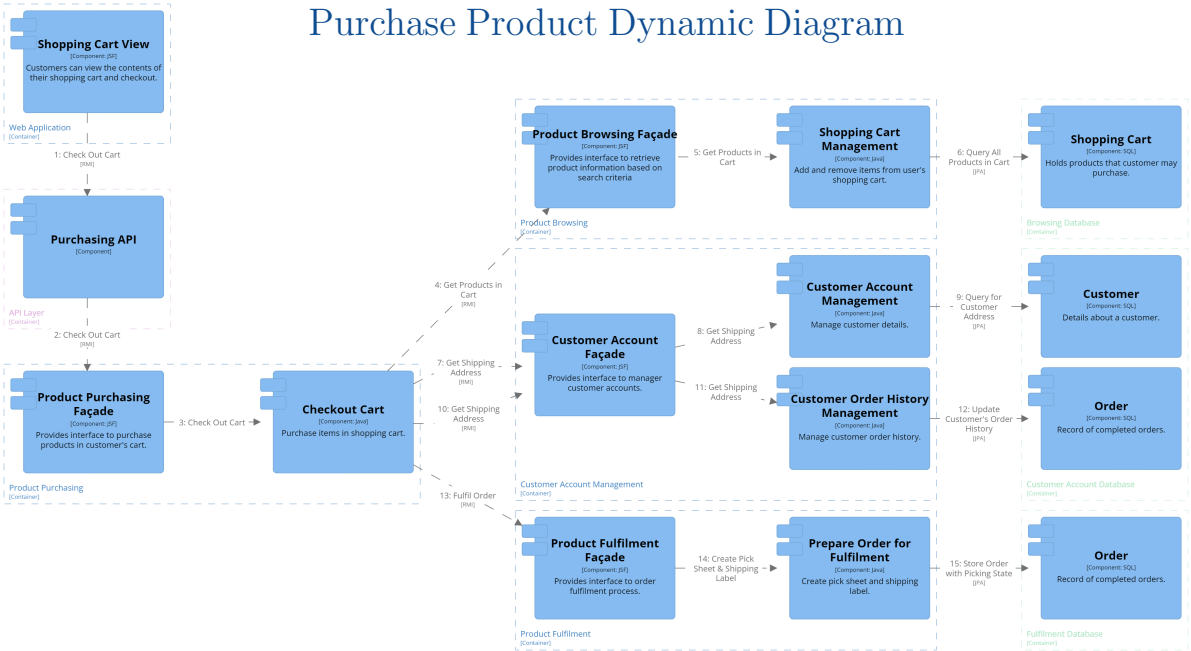


From *Fundamentals of Software Architecture*

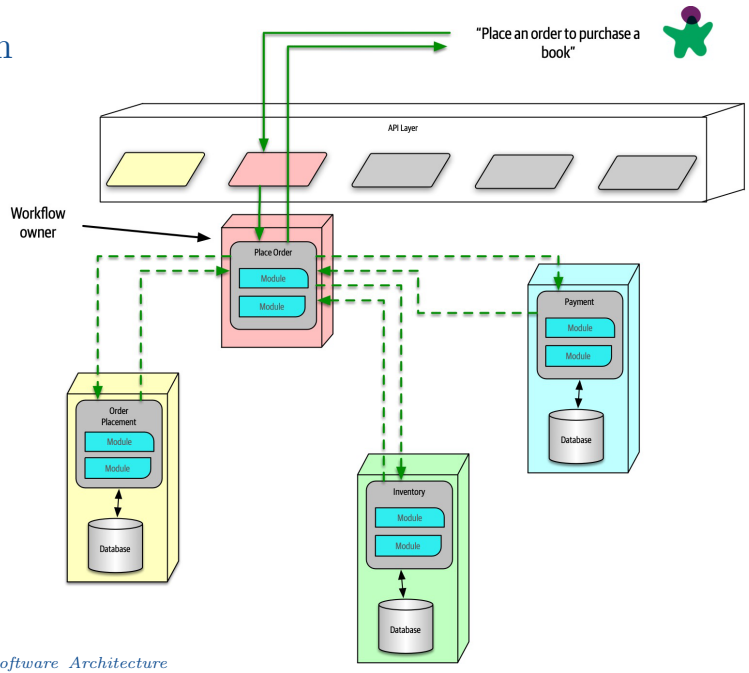
Sahara using Choreography



Purchase Product Dynamic Diagram



Orchestration



From *Fundamentals of Software Architecture*

Question

How bad is the coupling with choreography or orchestration?

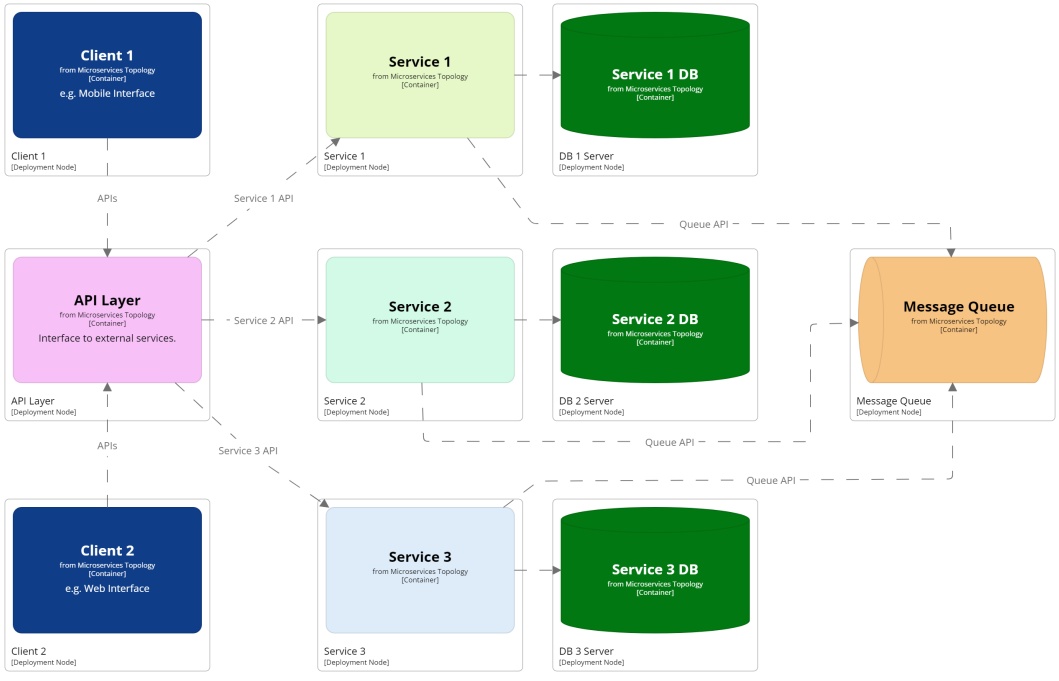
Question

How bad is the coupling with choreography or orchestration?

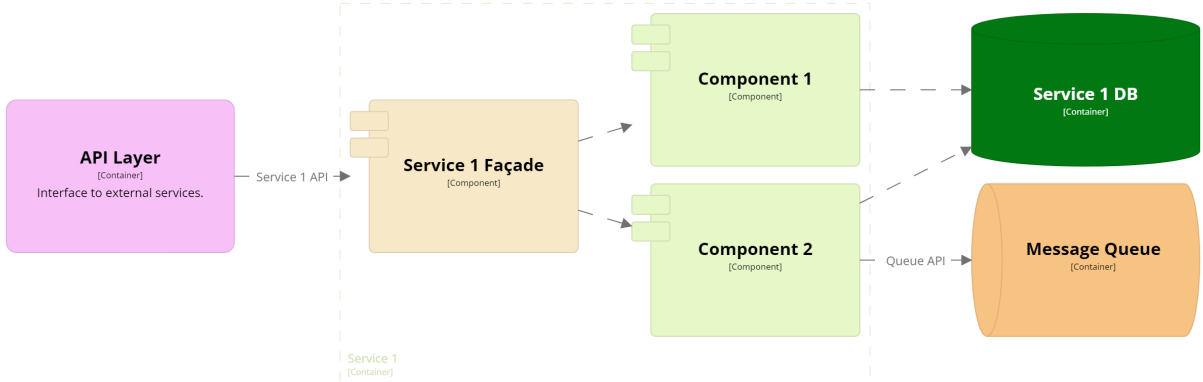
Answer

For a large system, *very bad*.

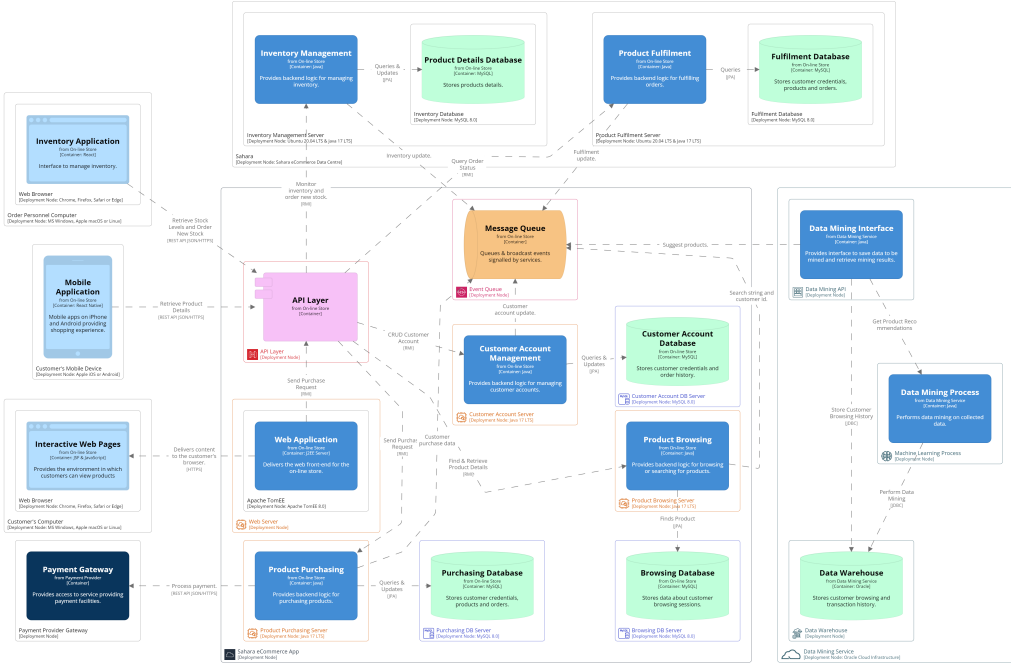
Microservices with Event Queue



Service 1 Components with Event Queue



Sahara using an Event Queue



Question

Are *browsing* and *purchasing* separate contexts?

Question

Are *browsing* and *purchasing* separate contexts?

Answer

- Are they a single business process or different processes?
- Do they share much or little data?

Question

- What about *inventory management* and *browse*?
- How do they maintain a consistent product database?

Pros & Cons

- Modularity
- Extensibility
- Reliability
- Interoperability
- Scalability
- Security
- Deployability
- Testability
- Simplicity