AGENDA - Day 3 - 4

- Context Application: Meetup
- Domain Driven Design
- Event Storming
- Microservices
- Architecture styles overview
- DDD Tactical Patterns
  - Application Services
  - Aggregates
  - Value Objects

Context Product: Meetup

https://www.meetup.com/

Part of **.NET Foundation – 382 groups**

Barcelona .NET Core

Barcelona, Spain
1,651 members · Public group
Organized by **Pau Lopez** and **2 others**

Share:

About | Events | Members | Photos | Discussions | More

Manage group | Create event

What we're about

This is a group created for .NET developers with open source mind and willing to explore open source cross platform modern web development with .NET...

Read more

Past events (27)

See all

THU, NOV 26, 6:30 PM GMT+1

Go Pro on .NET with F# by Isaac Abraham

Online event

76 attendees | Manage | Went

THU, OCT 29, 6:00 PM GMT+1

Event Sourcing/CQRS and Q&A with Alexey Zimarev

Online event

138 attendees | Manage | Went

Organizers

Pau Lopez and **2 others**
Message

Members (1,651)

See all

Use Case/User Story Driven Development

Use Case #1:

As a signed up user I can create a profile

Use Case #2:

As a meetup user with a subscription I can start organizing a meetup group

Use Case #3:

As a meetup user I can join and become a member of a group

Use Case #4:

As a meetup group member I can leave a group.

Use Case #5:

As a meetup group organizer I can create and publish an event.

Use Case #6:

As a meetup host I can cancel an event.

Use Case #7:

As a meetup group member I can attend to an published event

Use Case #8:

As a meetup group member I can cancel my attendance to an event.

Use Case/User Story Driven Development

Meetup

Use Case #1:

As a signed up user I can create a profile

Use Case #2:

As a meetup user with a subscription I can start organizing a meetup group

Use Case #3:

As a meetup user I can join and become a member of a group

Use Case #4:

As a meetup group member I can leave a group.

Use Case #5:

As a meetup group organizer I can create and publish an event.

Use Case #6:

As a meetup host I can cancel an event.

Use Case #7:

As a meetup group member I can attend to a published event

Use Case #8:

As a meetup event attendant I can cancel my invitation.

Use Case/User Story Driven Development

Use Case #1:

As a signed up user I can create a profile

Use Case #2:

As a meetup user with a subscription I can start organizing a meetup group

Use Case #3:

As a meetup user I can join and become a member of a group

Use Case #4:

As a meetup group member I can leave a group.

Use Case #5:

As a meetup group organizer I can create and publish an event.

Use Case #6:

As a meetup host I can cancel an event.

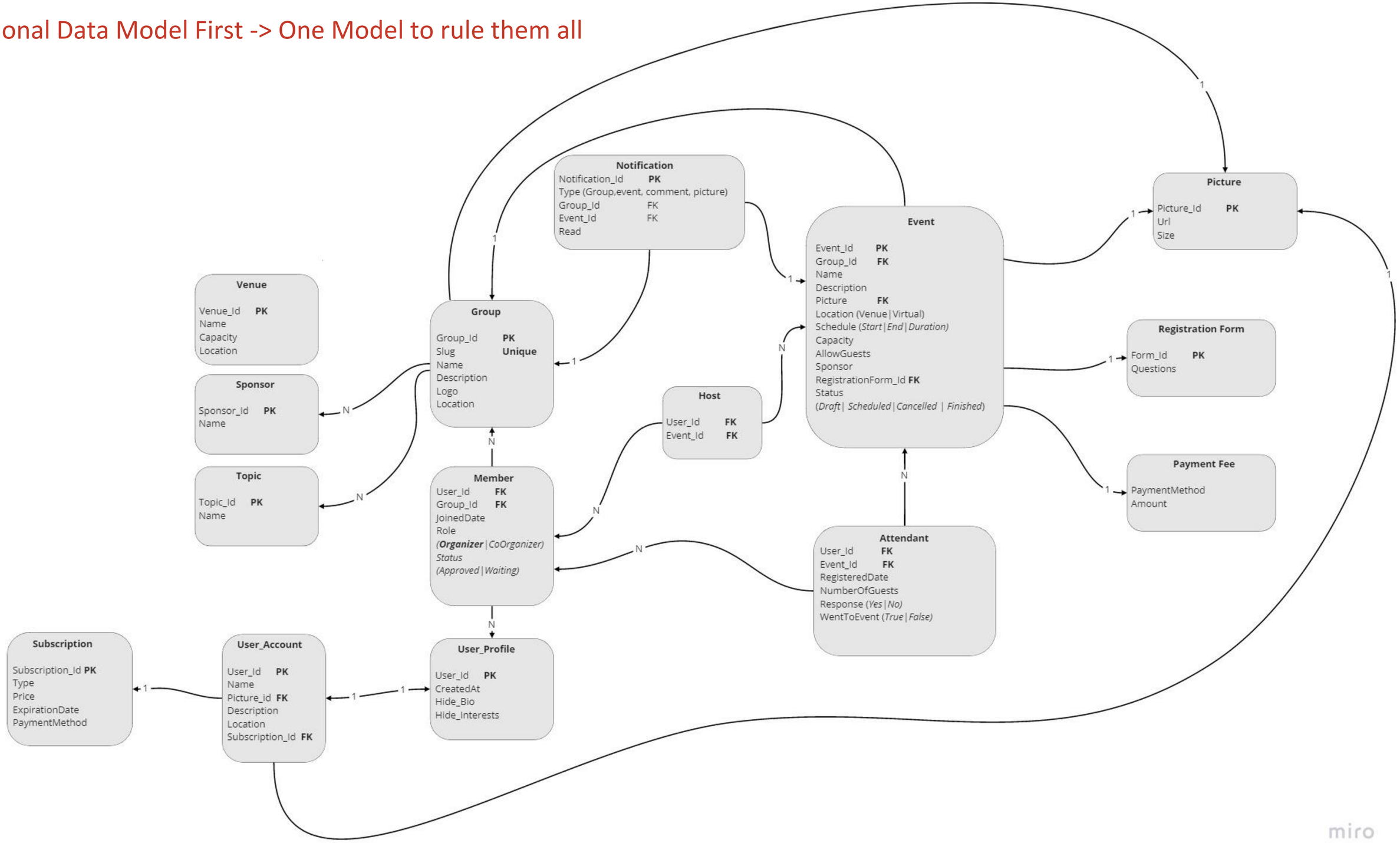
Use Case #7:

As a meetup group member I can attend to an published event

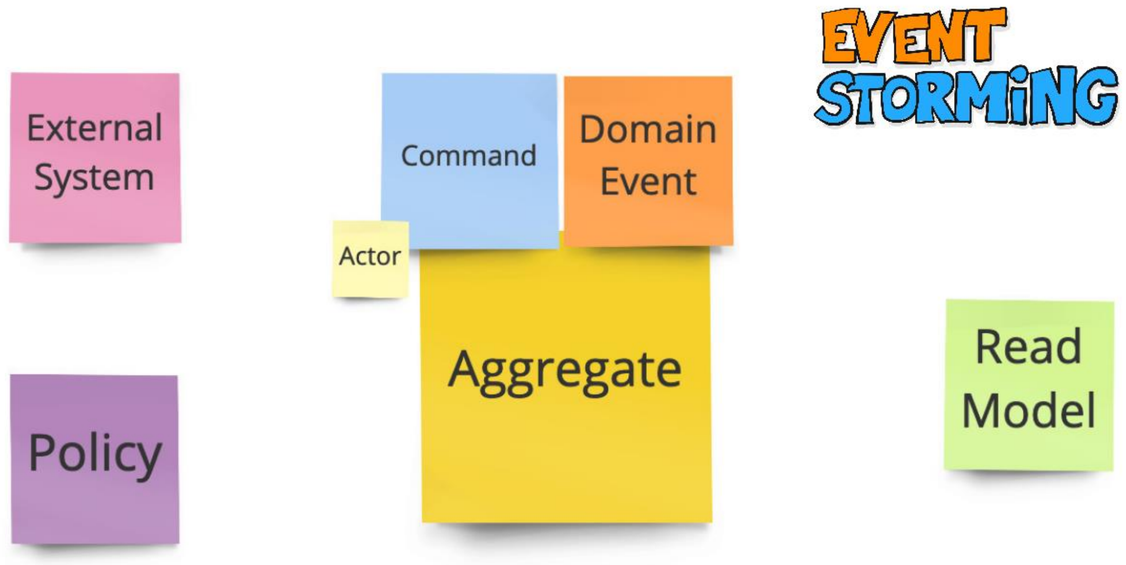
Use Case #8:

As a meetup group member I can cancel my attendance to an event.

Relational Data Model First -> One Model to rule them all



Event Storming



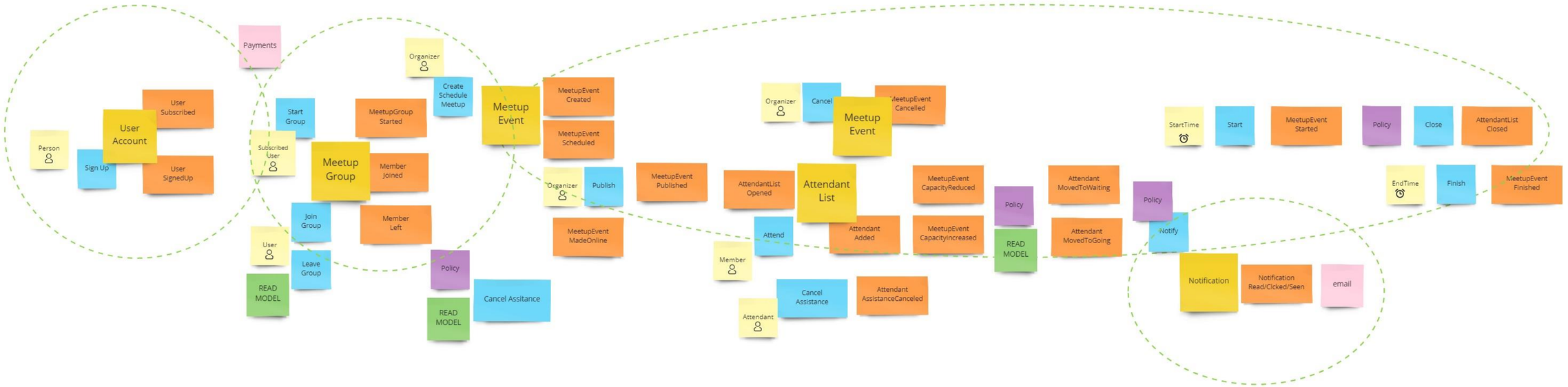
*“When you start **modeling events**, it forces you to think about the behavior of the system, as opposed to thinking about structure inside the system.*

Modeling events forces you to have a temporal focus on what’s going on in the system. Time becomes a crucial factor of the system.”

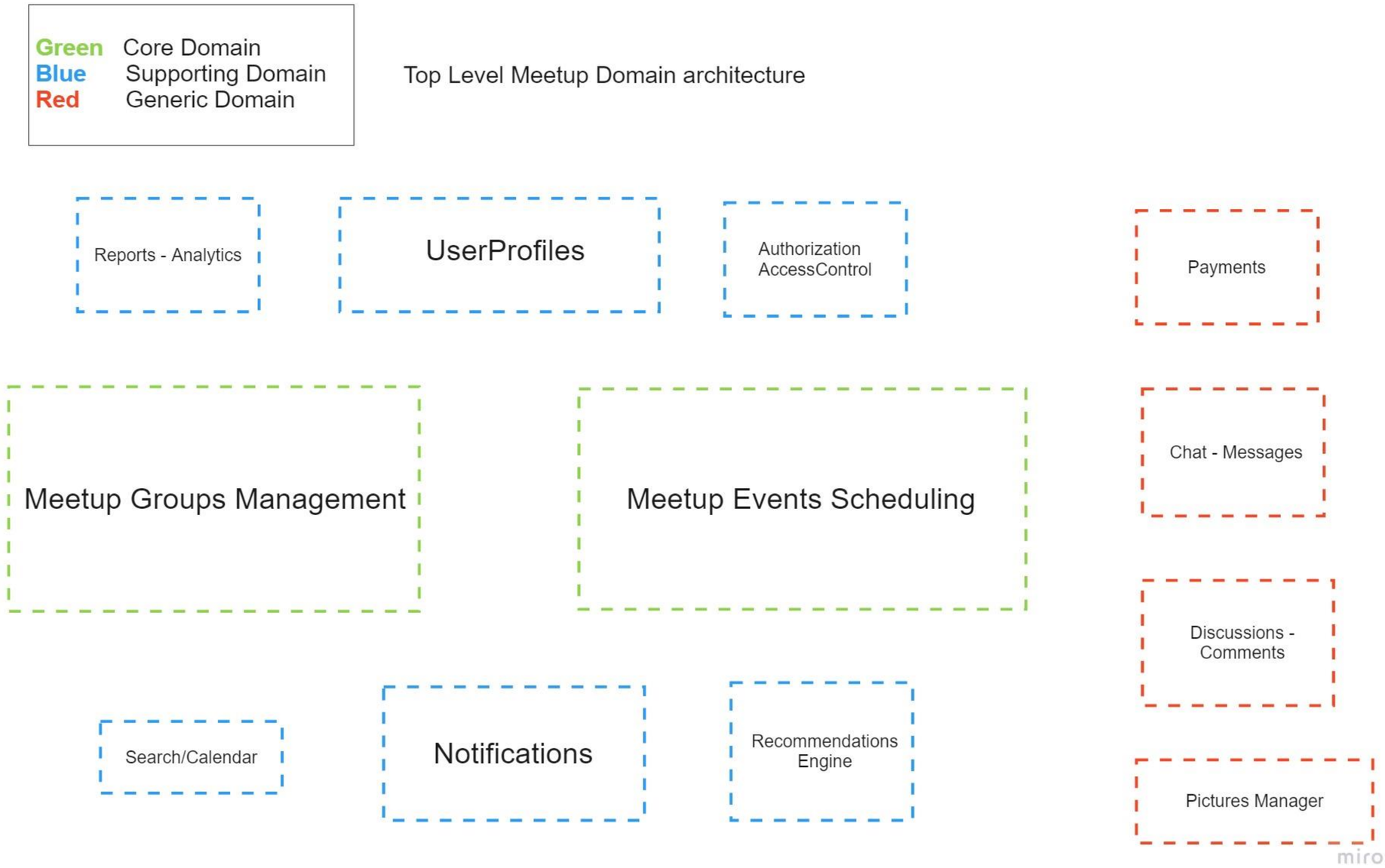
– Greg Young.

https://leanpub.com/introducing_eventstorming
https://medium.com/capital-one-tech/event-storming-decomposing-the-monolith-to-kick-start-your-microservice-architecture-acb8695a6e61

Event Storming – Meetup Domain



Domain Driven Design: Bounded Contexts



Hardest part of microservices: Finding The Right Boundaries

“Microservices without Domain Driven Design is a risky business”

- Trond Hjorteland

Domain Driven Design **Strategic Design**

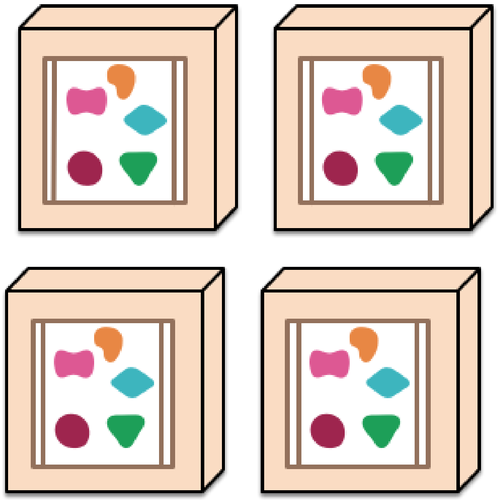
- **Ubiquitous** Language: “**Context** is King”
- **Bounded Contexts**: Core, Supporting and Generic Domains
- Model around use cases and behavior not around entities

Microservices and Domain Driven Design

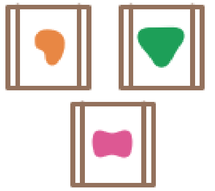
Bounded Contexts match the idea of a Microservice.

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

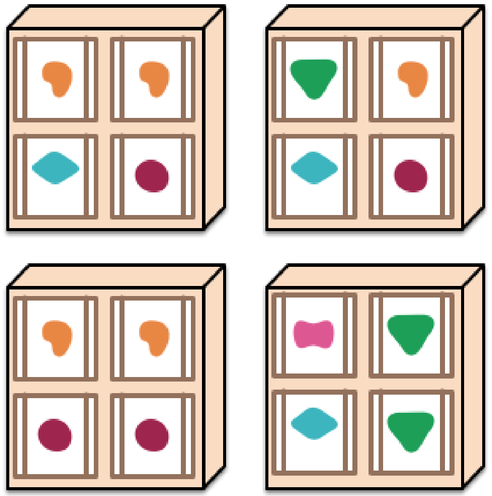
... and scales by replicating the monolith on multiple servers



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



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

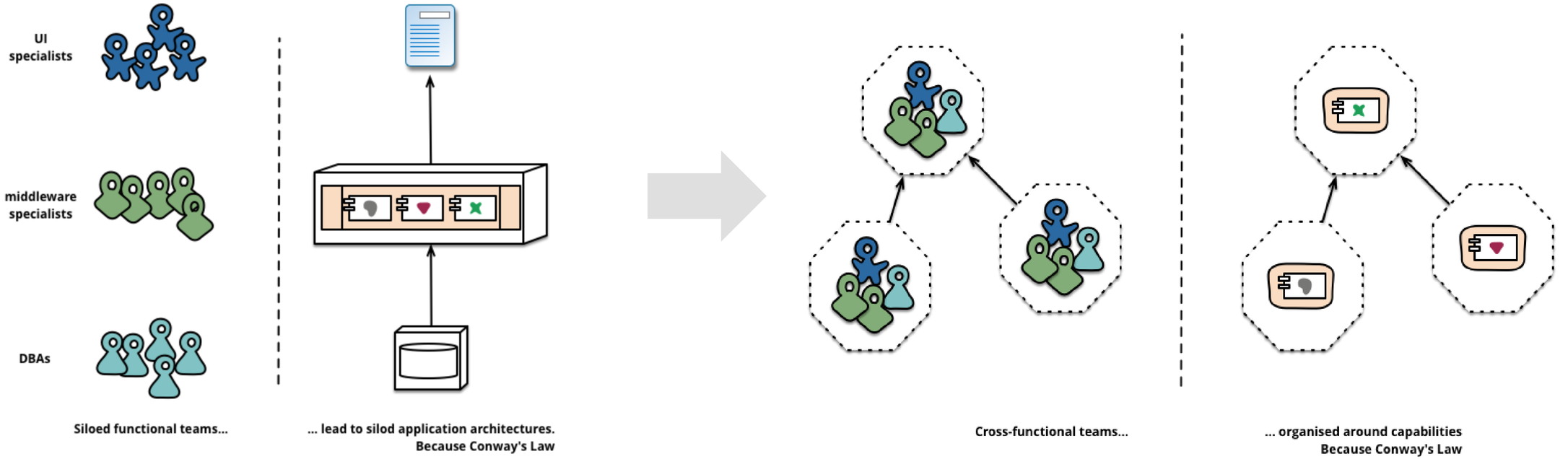


https://martinfowler.com/articles/microservices.html
https://docs.microsoft.com/en-us/azure/architecture/microservices/

Microservices and Domain Driven Design

Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.

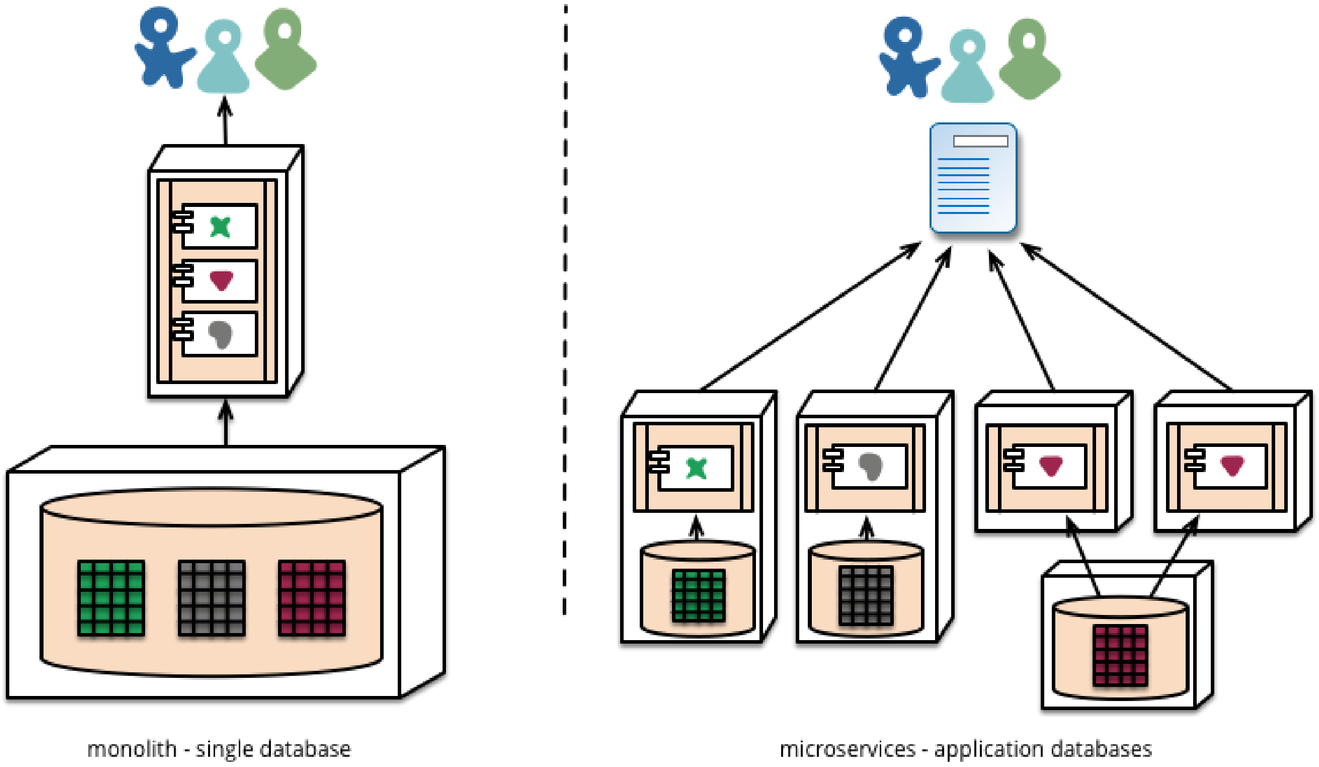
-- Melvin Conway, 1968



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

https://docs.microsoft.com/en-us/azure/architecture/microservices/

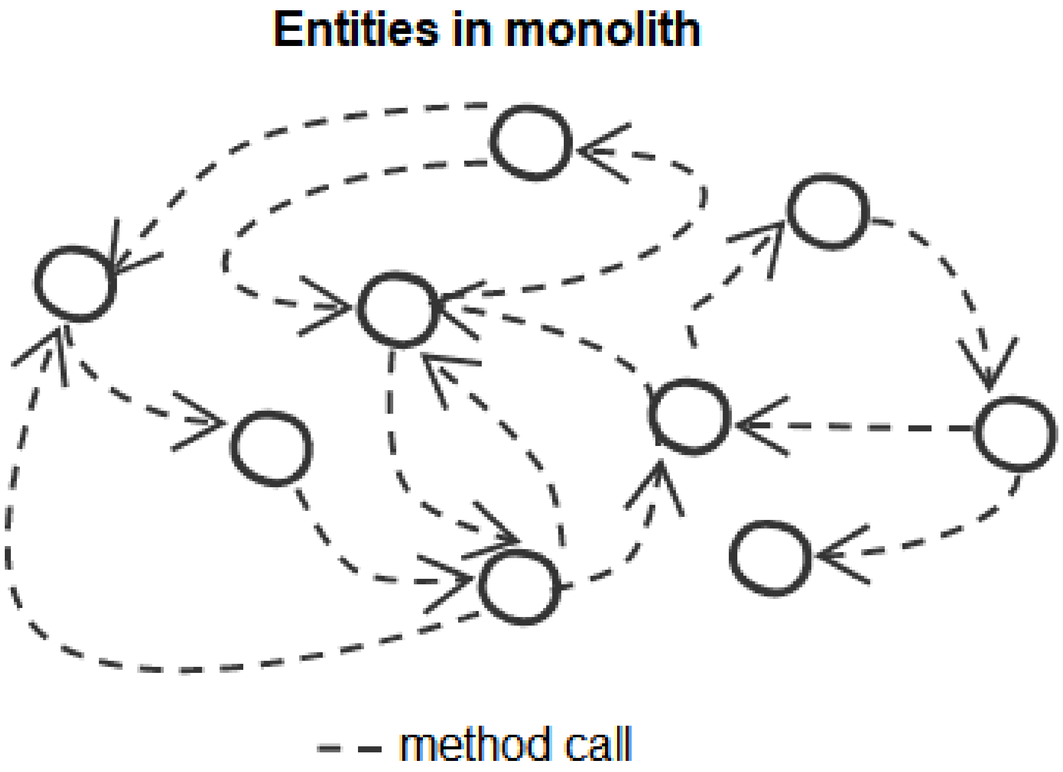
The Hardest Part About Microservices: Your Data



monolith - single database

microservices - application databases

Distributed monolith



https://medium.com/transferwise-engineering/from-monolith-to-distributed-monolith-fd53d8dbbeba

Microservices Drawbacks

- First Law of Distributed Object Design:

  “Don't distribute your objects“
  -- Martin Fowler

- Distributed Systems are really hard
- Fallacies of distributed computing
- Microservices is just “SOA done right”

https://en.wikipedia.org/wiki/Fallacies_of_distributed_computing

Microservices – Resources

- Microservices • Martin Fowler
- Principles Of Microservices by Sam Newman
- The Many Meanings of Event-Driven Architecture • Martin Fowler
- Vertical Slice Architecture - Jimmy Bogard
- Avoiding Microservice Megadisasters - Jimmy Bogard
- Managing Data in Microservices
- Designing Events-First Microservices
- Six Little Lines of Fail - Jimmy Bogard
- Applying the Saga Pattern • Caitie McCaffrey
- https://blog.christianposta.com/microservices/the-hardest-part-about-microservices-data/
- Microservice Architecture pattern (microservices.io)
- Autonomous microservices don't share data. Period - Dennis van der Stelt
- Finding your service boundaries - a practical guide - Adam Ralph
- Reliable Messaging Without Distributed Transactions

ARCHITECTURAL STYLES: 3 Layers

Presentation tier

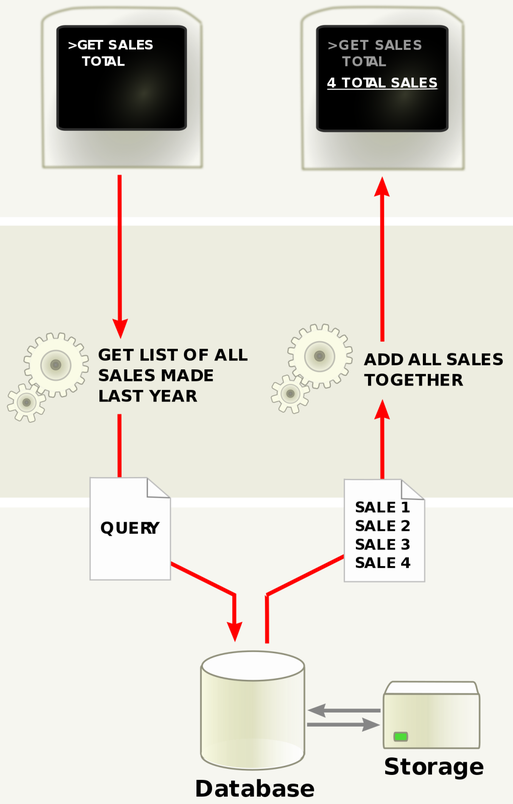
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

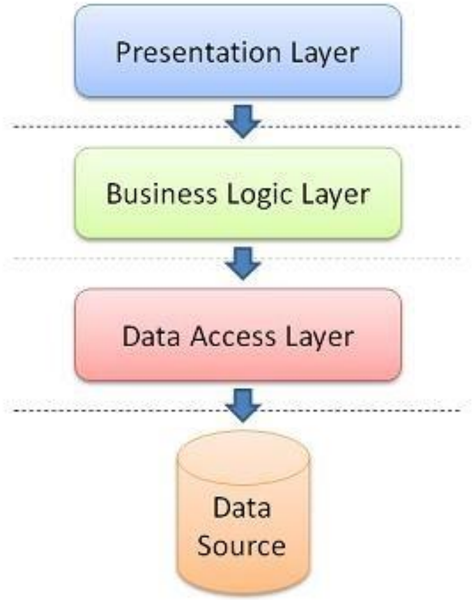
Logic tier

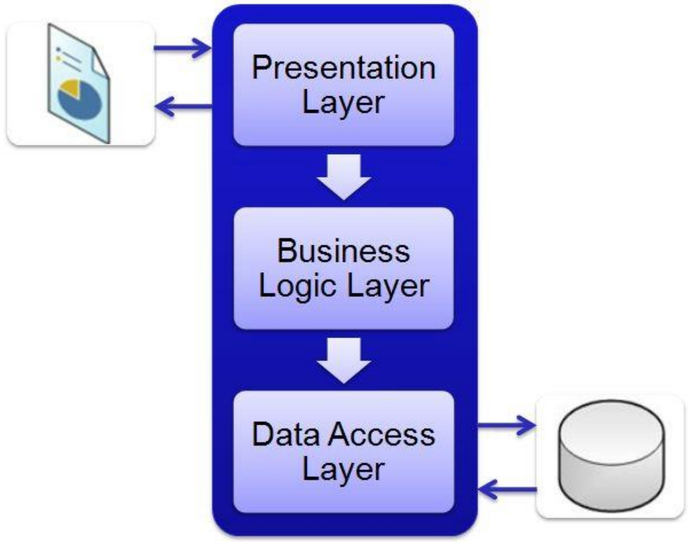
This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

Data tier

Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user

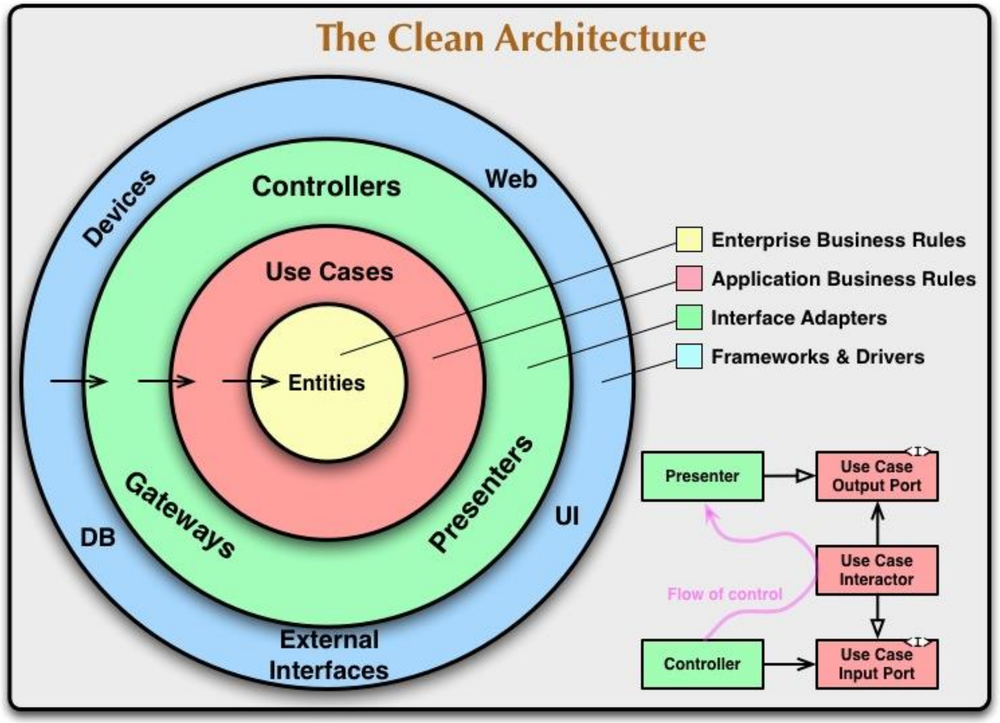




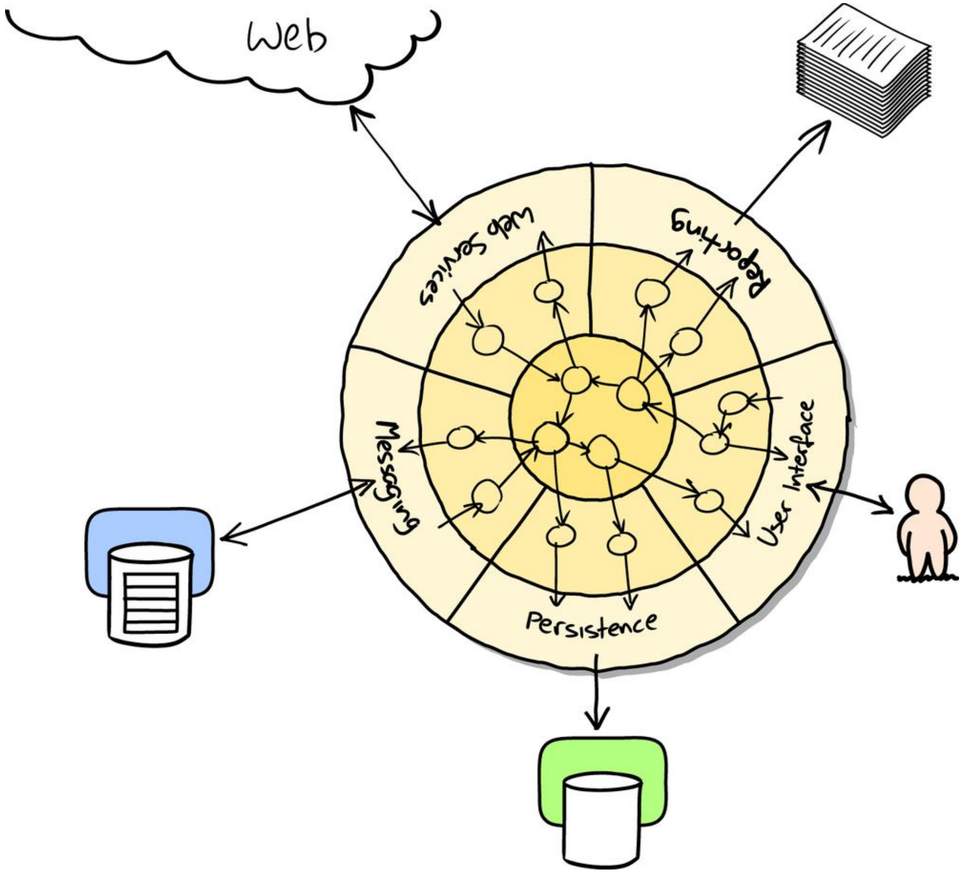


https://en.wikipedia.org/wiki/Multitier_architecture

ARCHITECTURAL STYLES: Clean



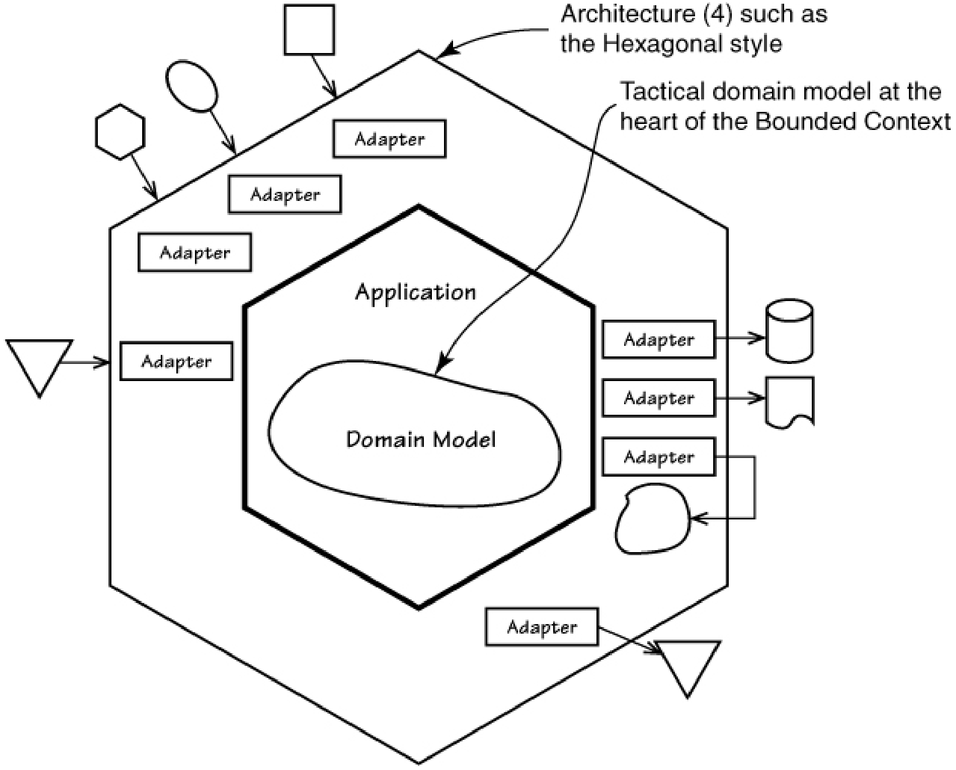
https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html



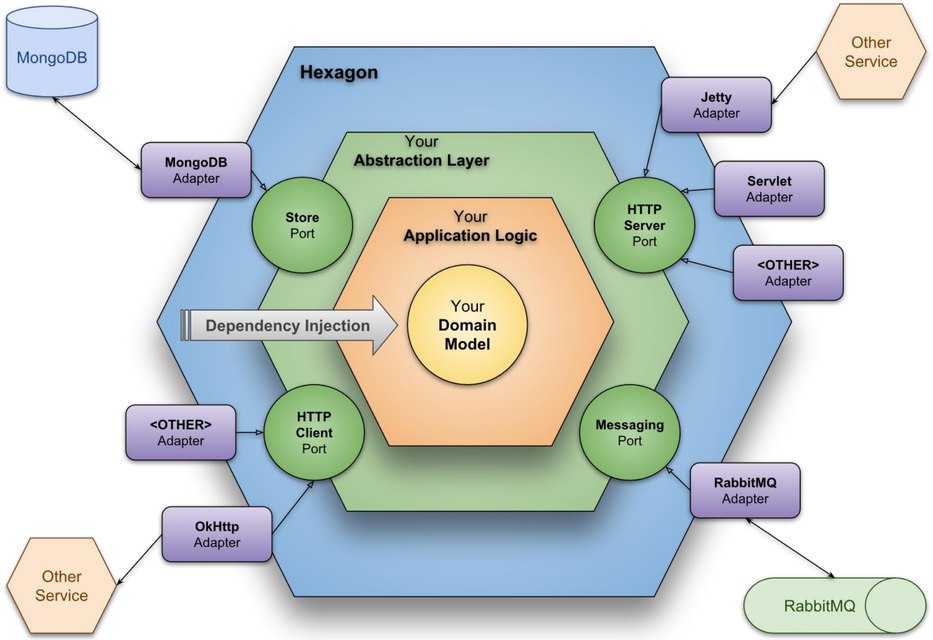
https://www.oreilly.com/library/view/test-driven-development-with/9781449365141/ch22.html

ARCHITECTURAL STYLES: Hexagonal, Onion, Ports and Adapters

https://alistair.cockburn.us/hexagonal-architecture/

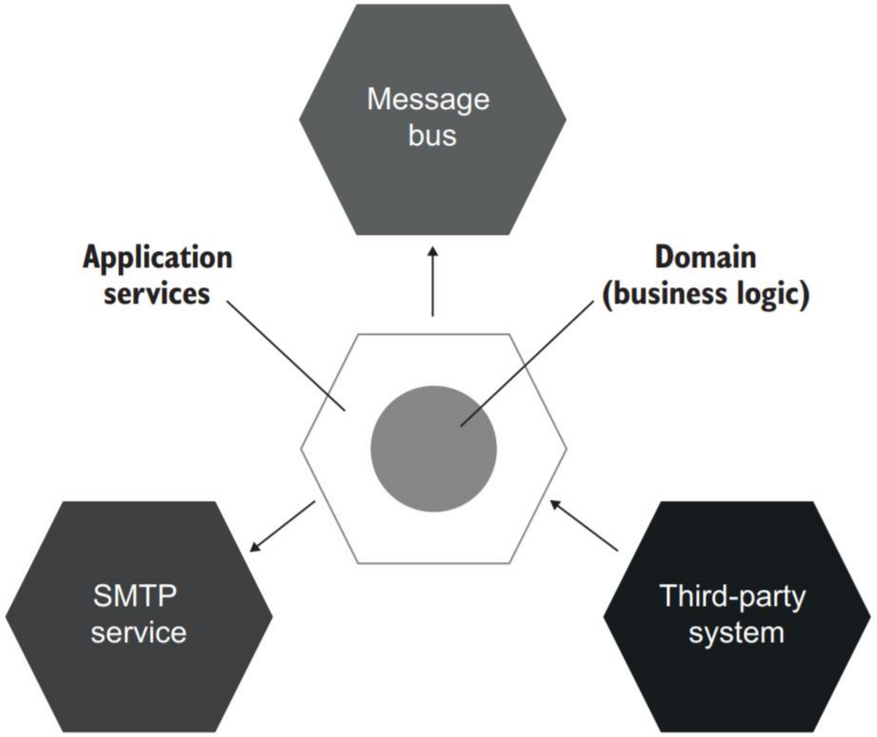


https://github.com/VaughnVernon/IDDD_Samples



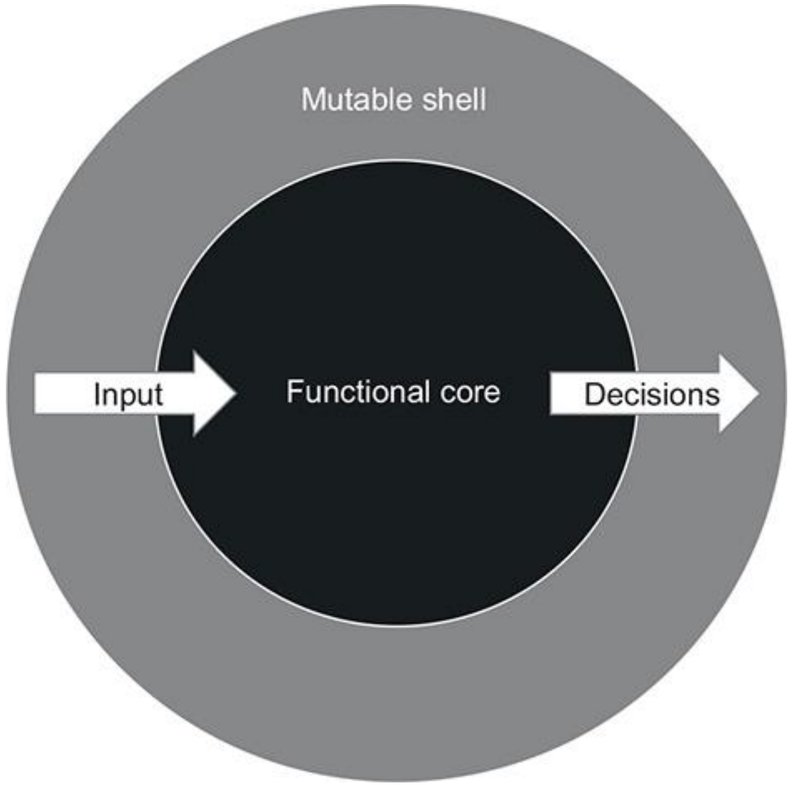
https://hexagonkt.com/

ARCHITECTURAL STYLES: Hexagonal, Onion, Ports and Adapters

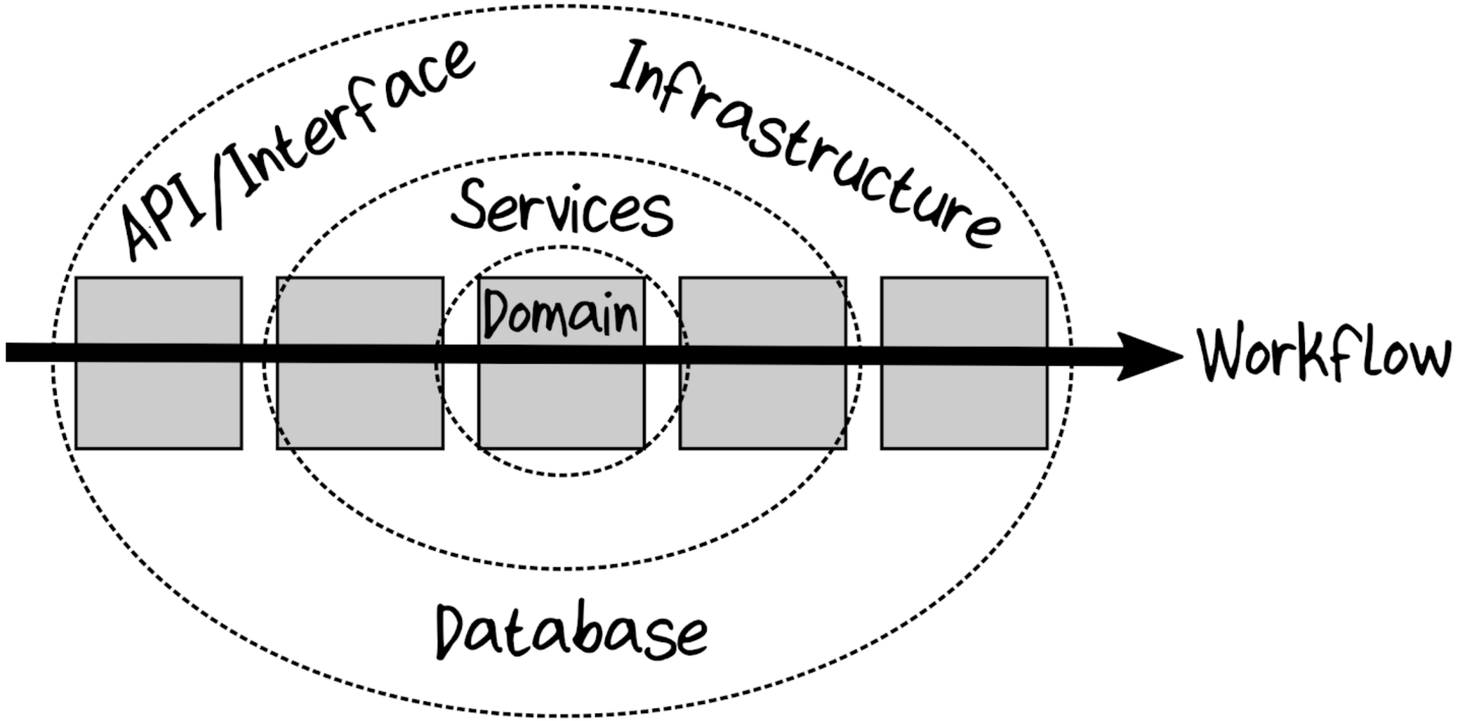


https://www.manning.com/books/unit-testing

ARCHITECTURAL STYLES: Functional core and Imperative shell

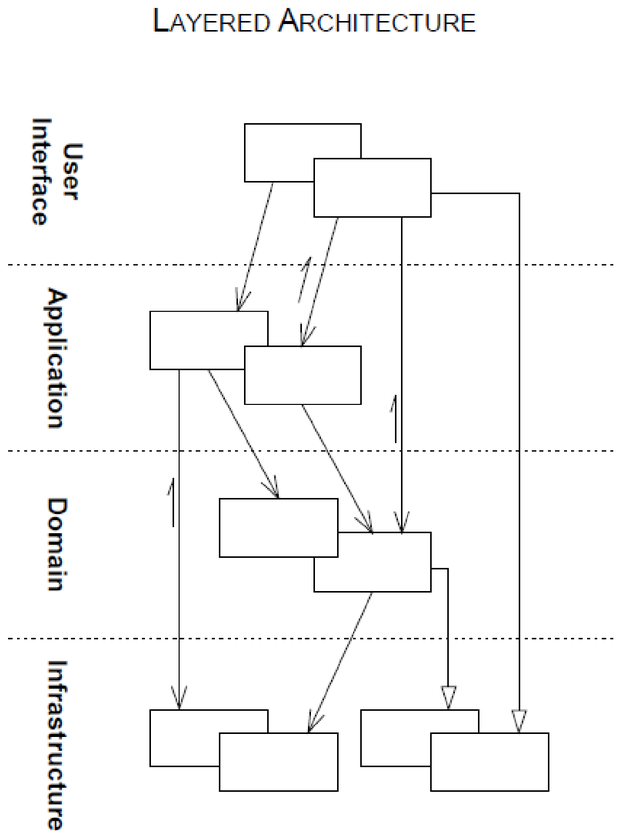


https://elixirforum.com/t/functional-architecture-in-elixir/29766

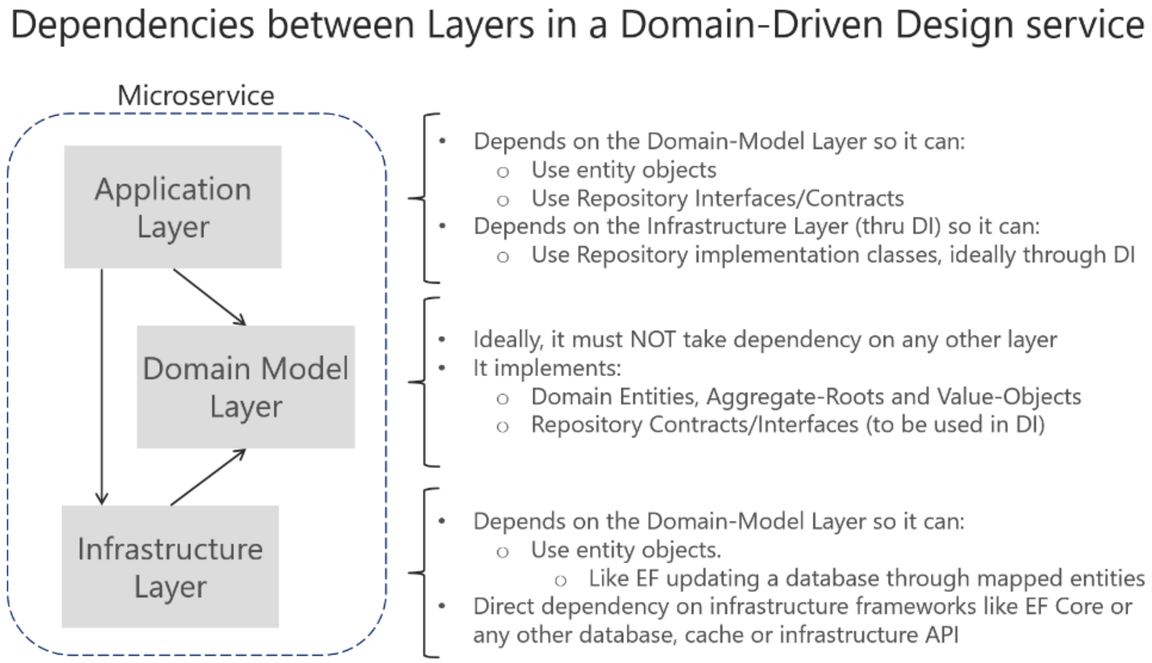


https://fsharpforfunandprofit.com/books/

ARCHITECTURAL STYLES: Domain Driven Design

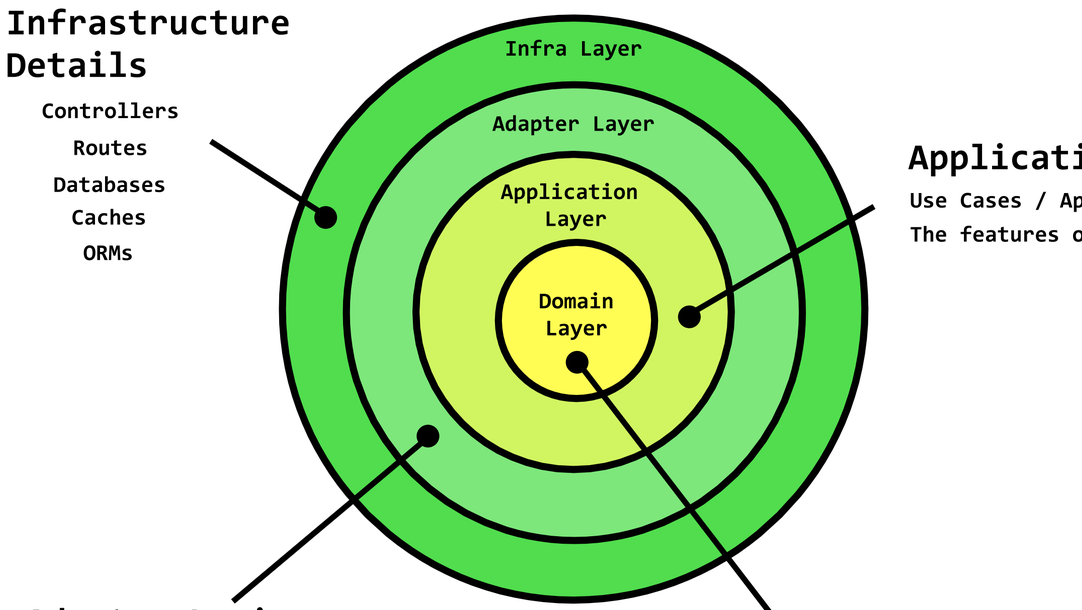


https://www.dddcommunity.org/book/evans_2003/



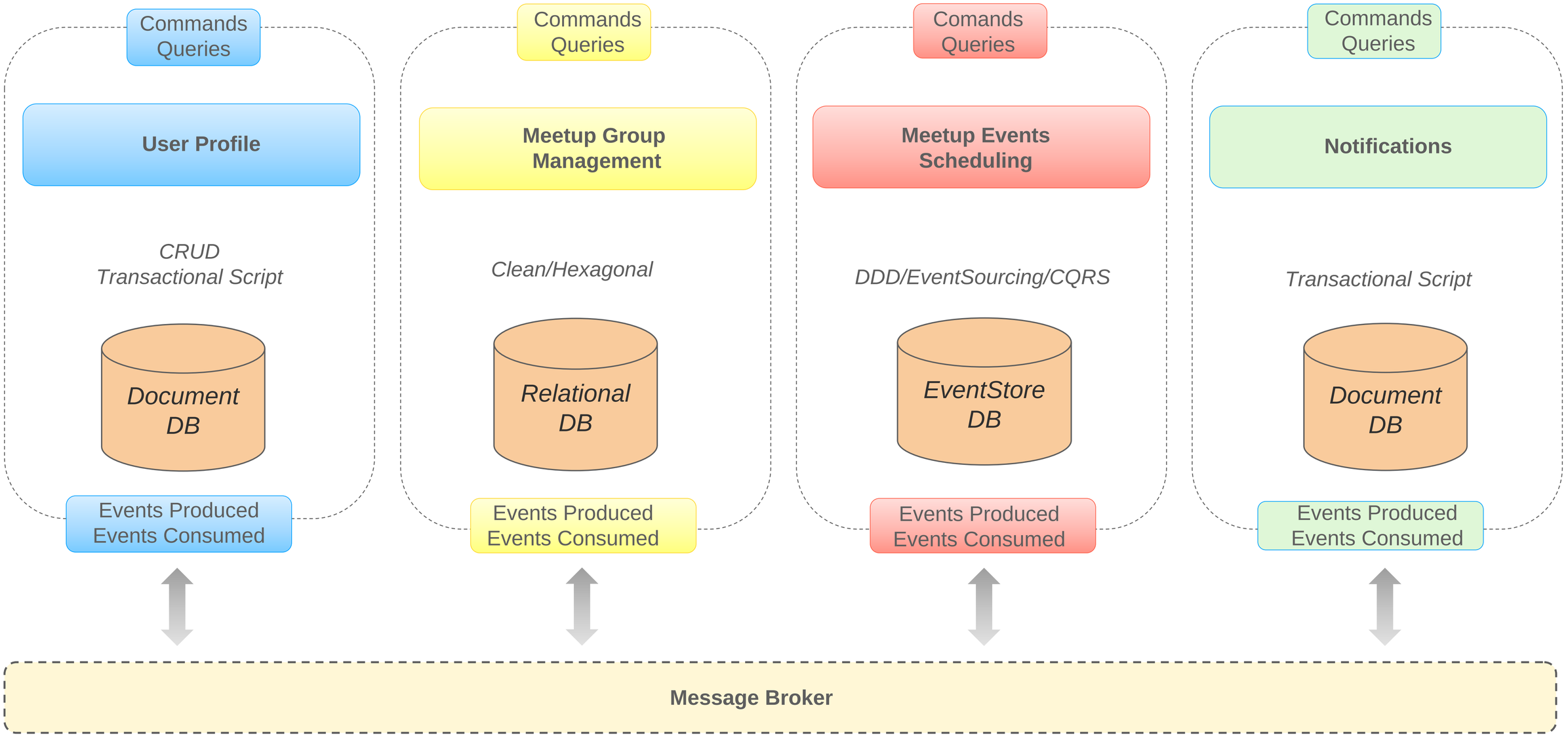
https://docs.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/ddd-oriented-microservice

ARCHITECTURAL STYLES: Domain Driven Design

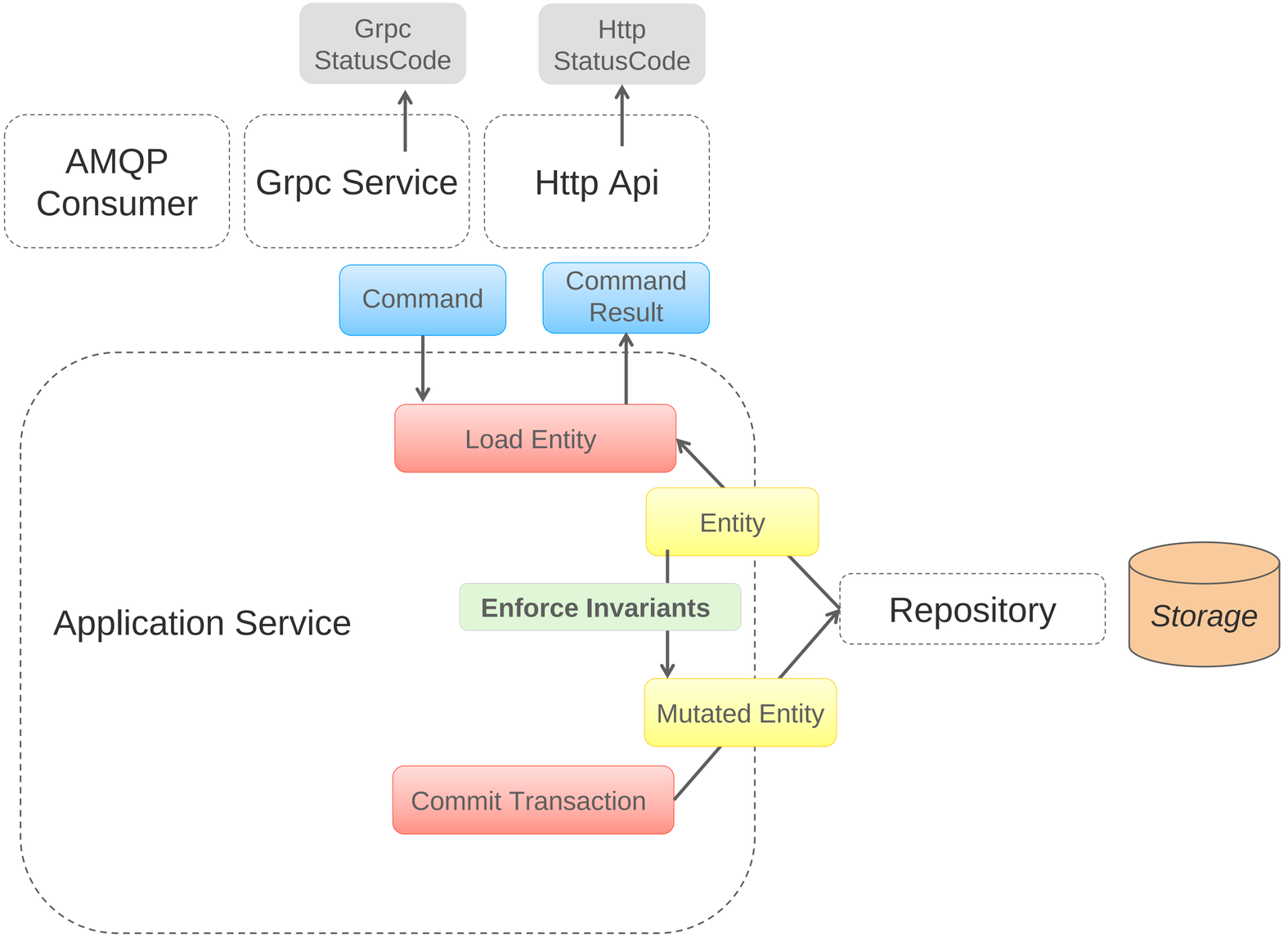


https://khalilstemmler.com/articles/software-design-architecture/organizing-app-logic/

Meetup Top Level Architecture



DDD Tactical Patterns: Application Services (use case orchestrator)



https://martinfowler.com/eaaCatalog/repository.html

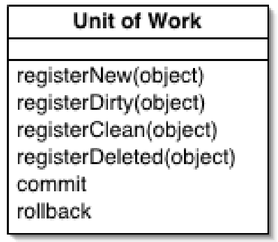
DDD Tactical Patterns: Repository and Unit of Work

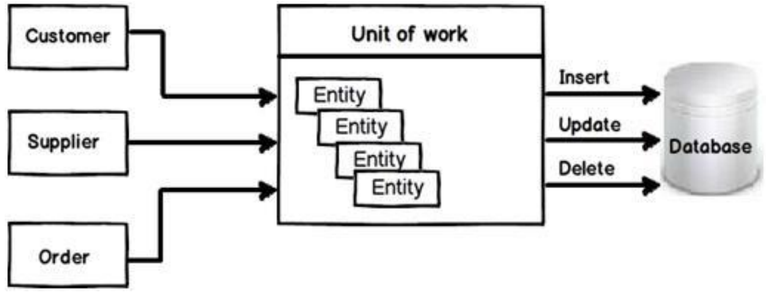
Repository pattern

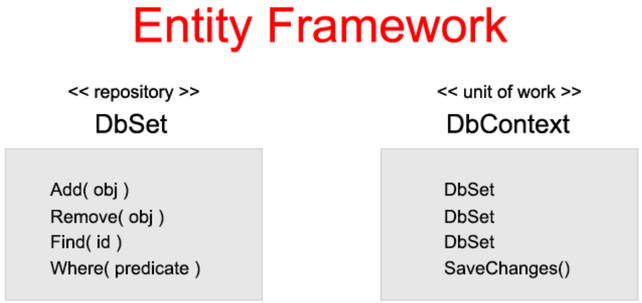
Mediates between the domain and data mapping layers using a collection-like interface for accessing domain objects.

Unit of work pattern

Maintains a list of objects affected by a business transaction and coordinates the writing out of changes and the resolution of concurrency problems.



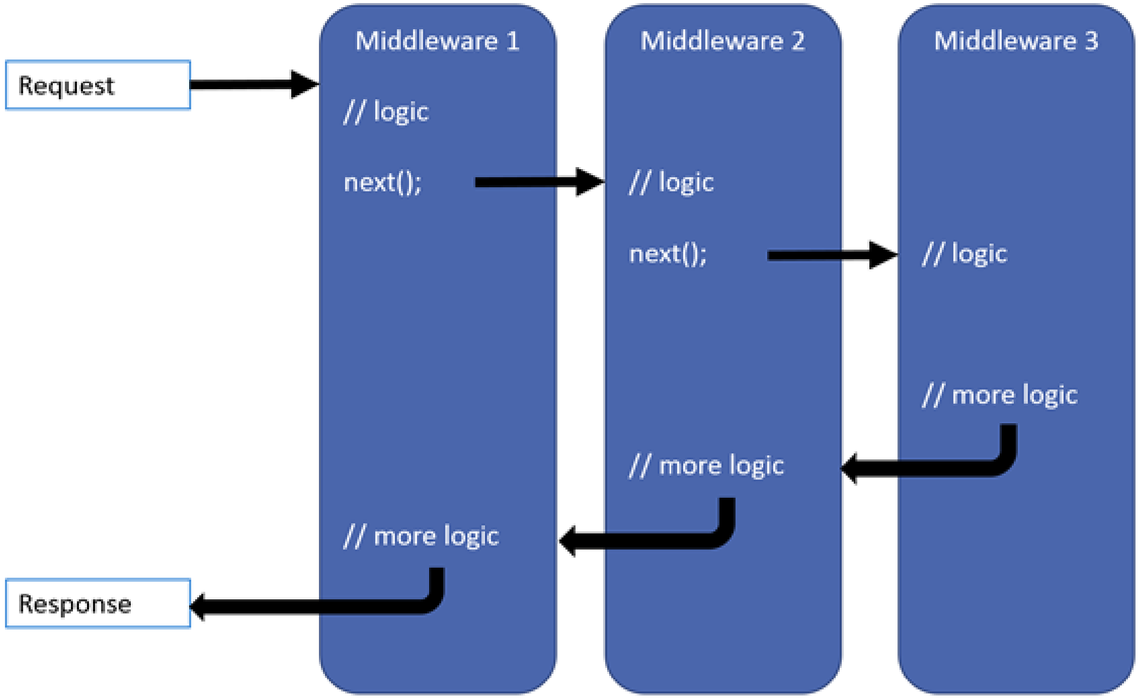




https://martinfowler.com/eaaCatalog/unitOfWork.html

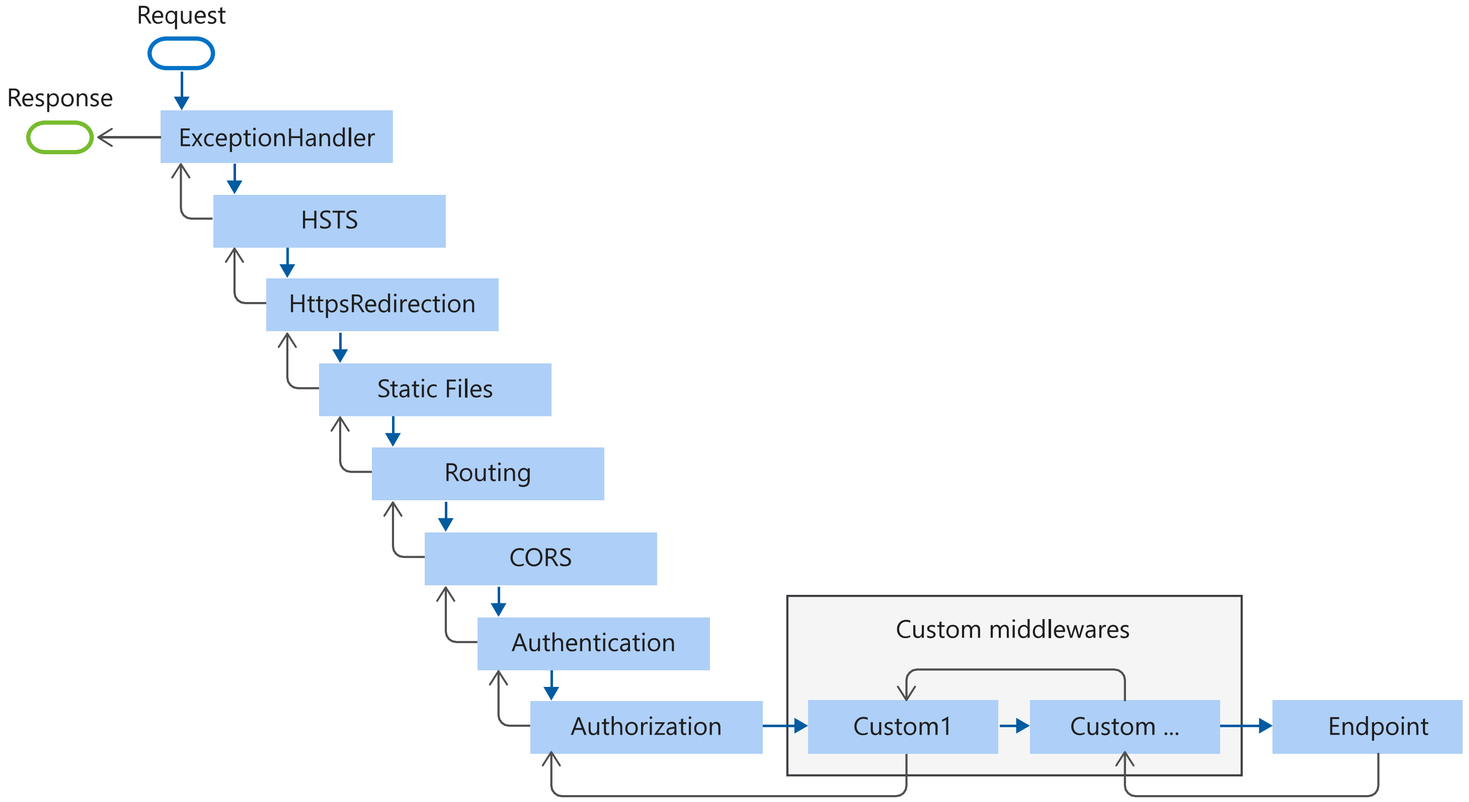
https://martinfowler.com/eaaCatalog/repository.html

DDD Tactical Patterns: Application Services Middlewares

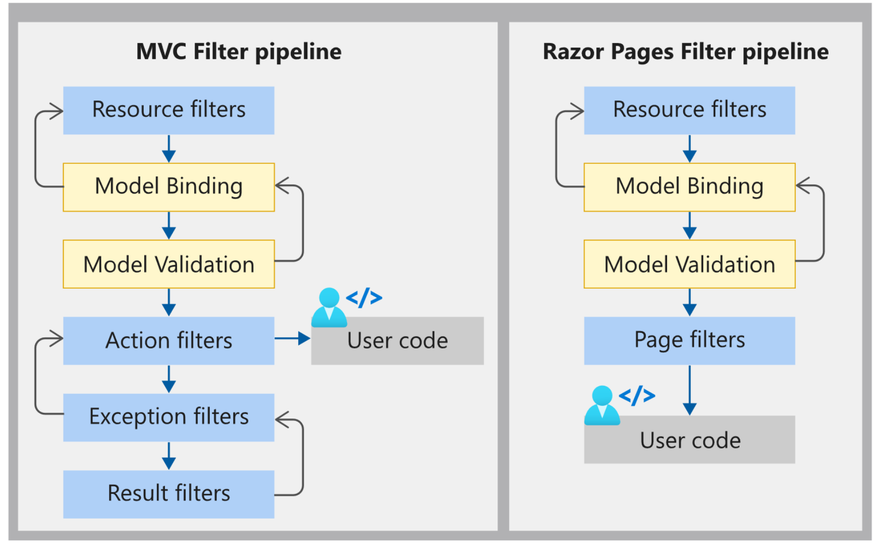


https://docs.microsoft.com/en-us/aspnet/core/fundamentals/middleware/

DDD Tactical Patterns: Application Services Middlewares

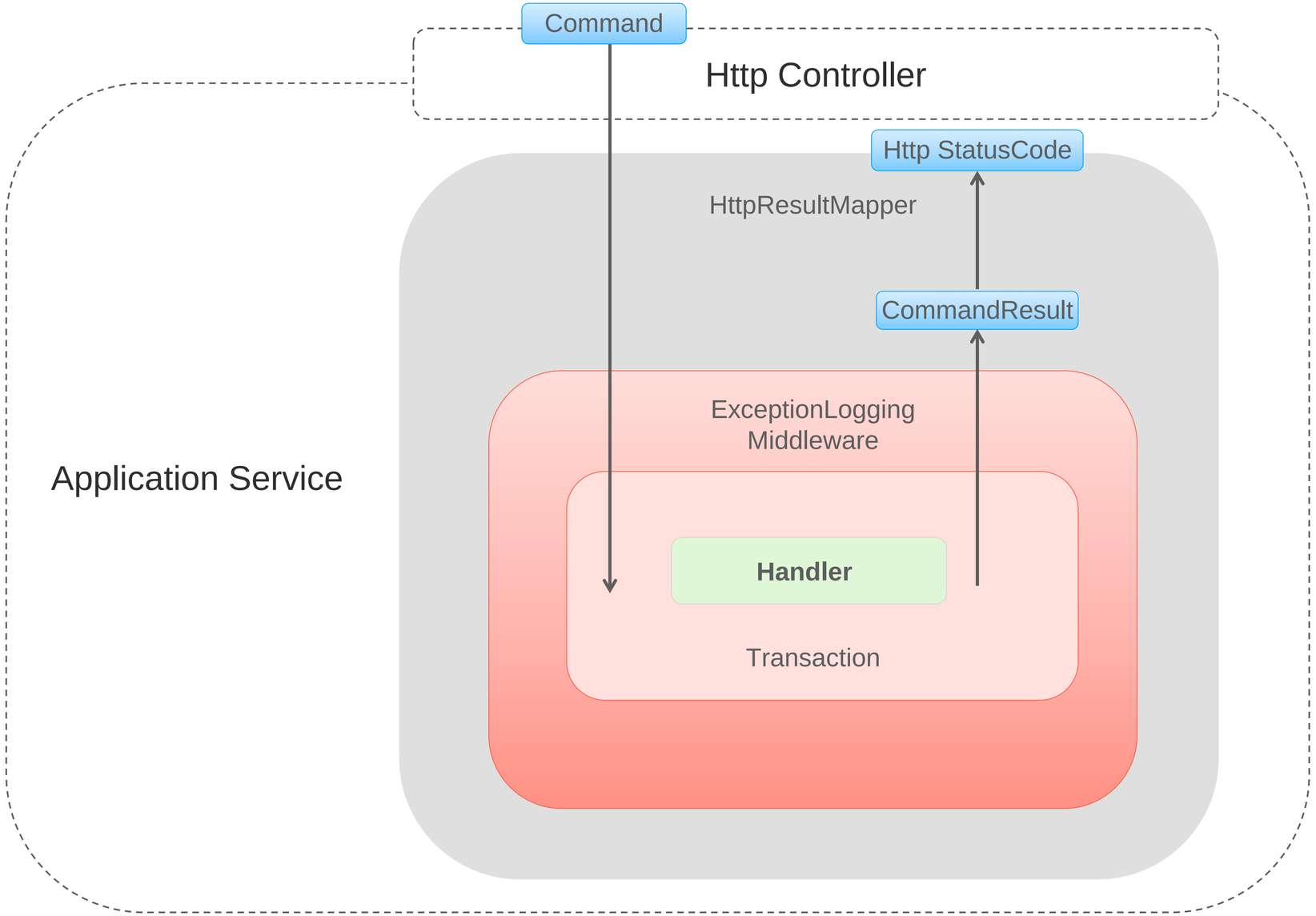


MVC Endpoint
(called by the Endpoint Middleware)

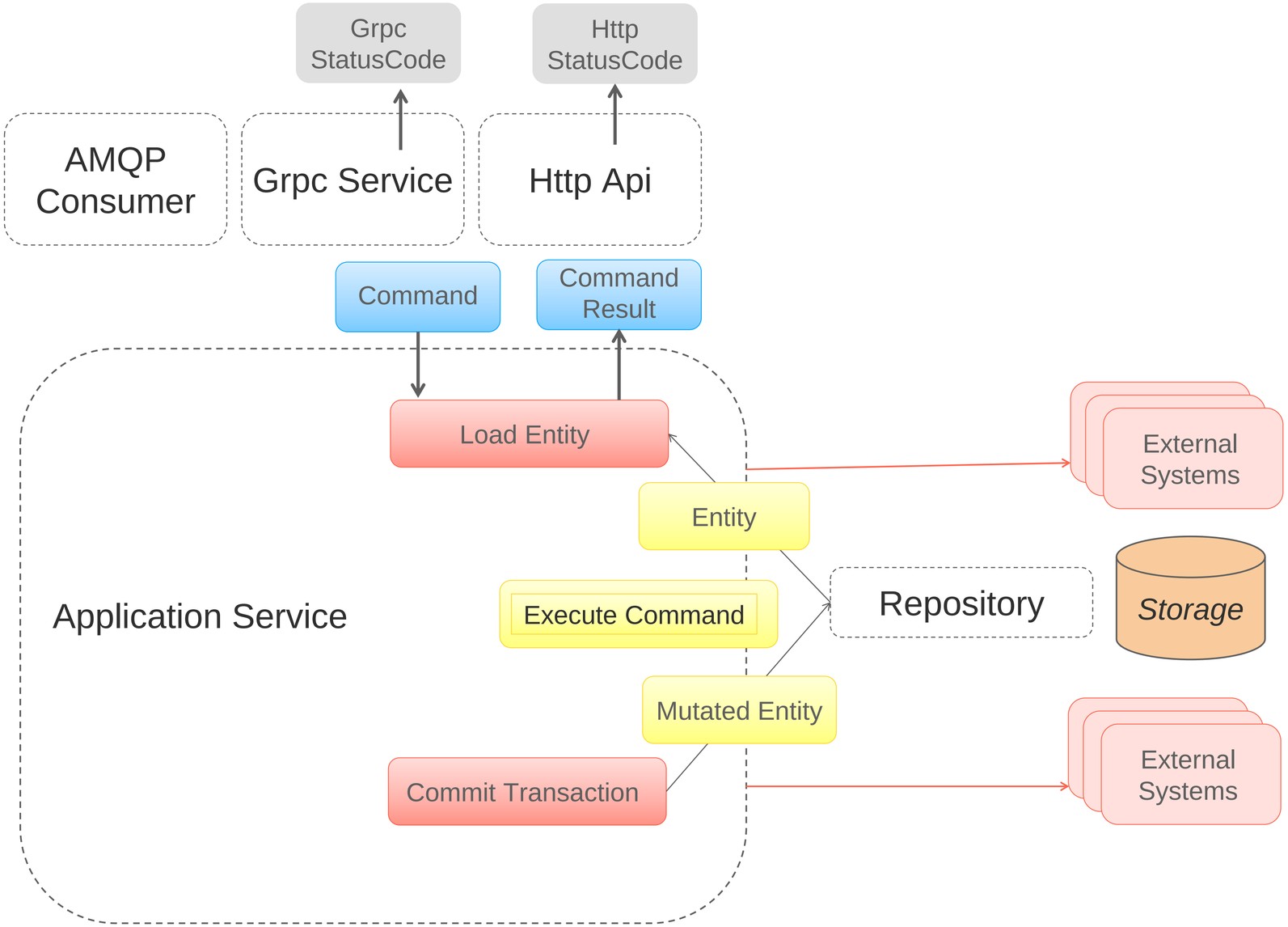


https://docs.microsoft.com/en-us/aspnet/core/fundamentals/middleware/
https://docs.particular.net/nservicebus/pipeline/
https://github.com/jbogard/MediatR/wiki/Behaviors
https://masstransit-project.com/advanced/middleware/
https://paramore.readthedocs.io/en/latest/BuildingAnAsyncPipeline.html#implementing-a-pipeline

DDD Tactical Patterns: Application Services Middlewares



DDD Tactical Patterns: Application Services and External Systems



DDD Tactical Patterns: Aggregates

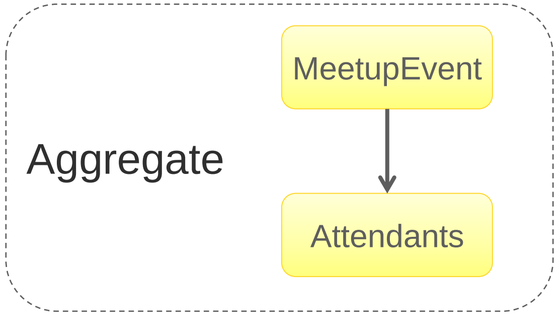
A DDD aggregate is a ***cluster of domain objects*** that can be treated as a ***single unit***.

An example may be an order and its line-items, these will be separate objects, but it's useful to treat the order (together with its line items) as a single aggregate.

Aggregates are the basic element of transfer of data storage - you request to load or save whole aggregates.
Transactions should not cross aggregate boundaries.

https://martinfowler.com/bliki/DDD_Aggregate.html

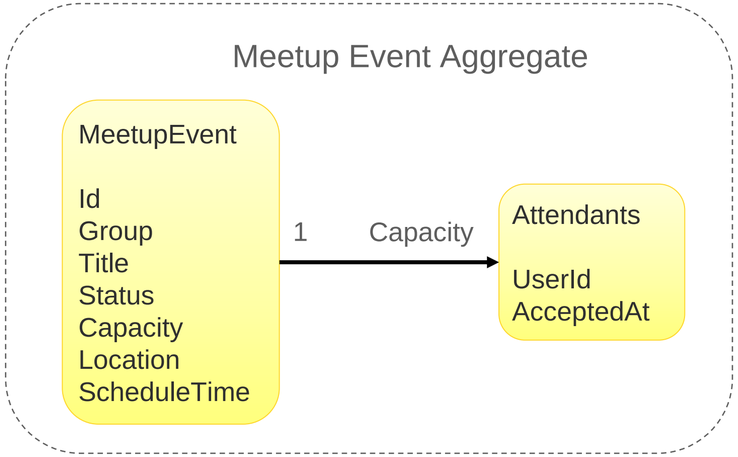
Consistent Local Transactional Boundary



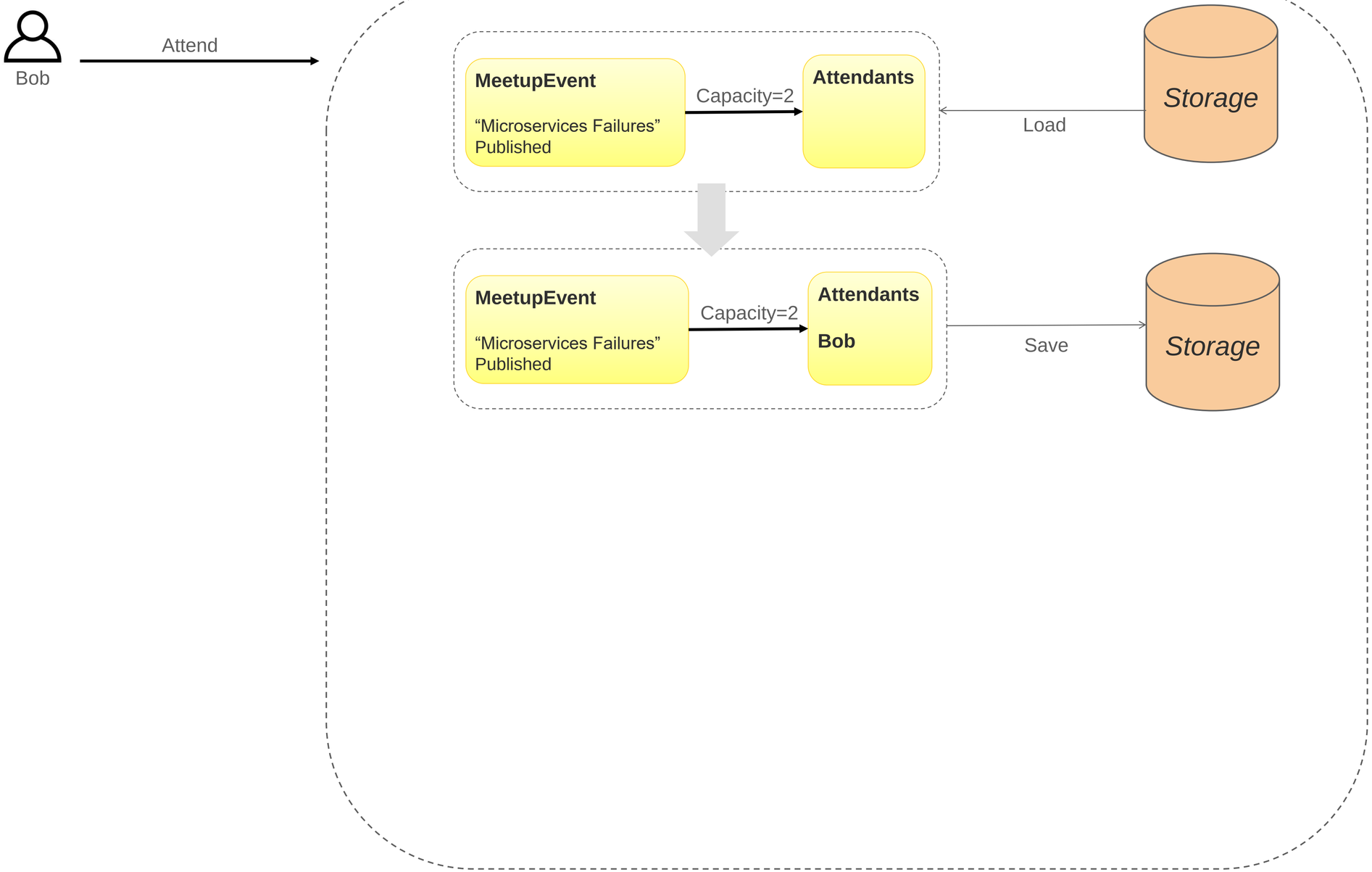
DDD Tactical Patterns: Aggregates

Aggregate enforces invariants using a local transaction

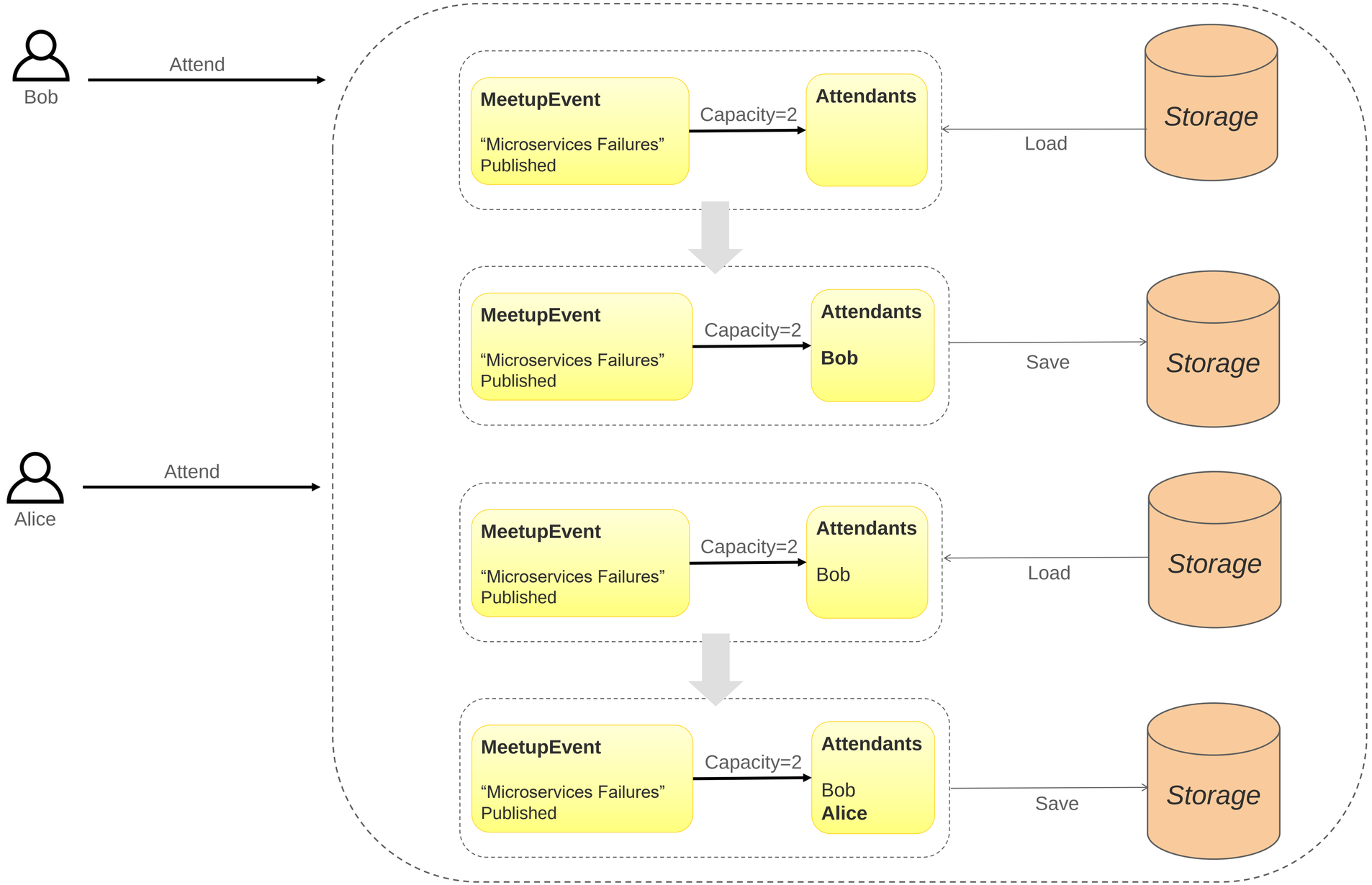
- Meetup event can not be created without Title and Capacity.
- Created Meetup without location or schedule time can not be published
- Cancelled meetup event can not published.
- Number of attendants can not be greater than the capacity.



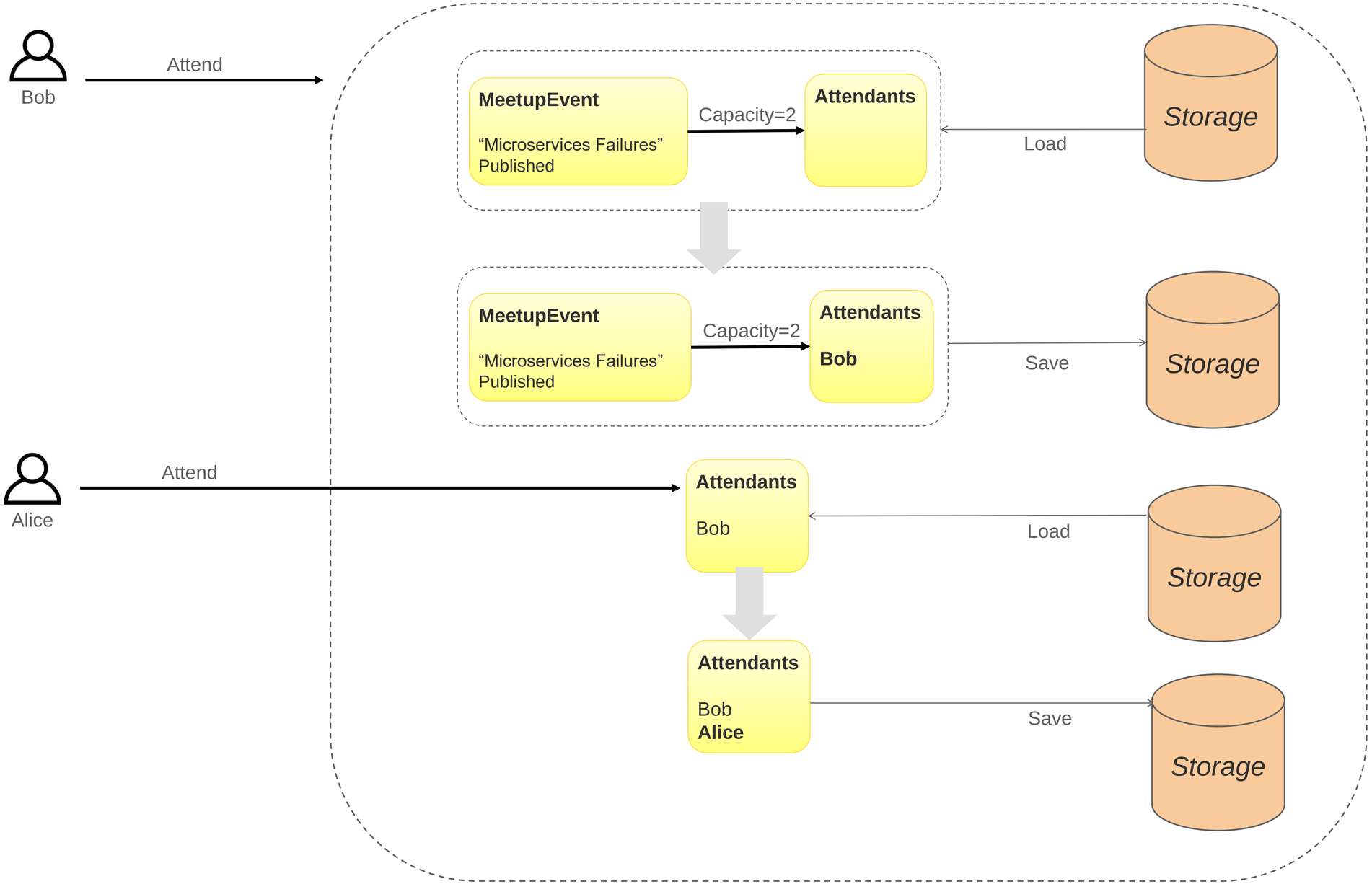
DDD Tactical Patterns: Aggregate versioning and Optimistic concurrency



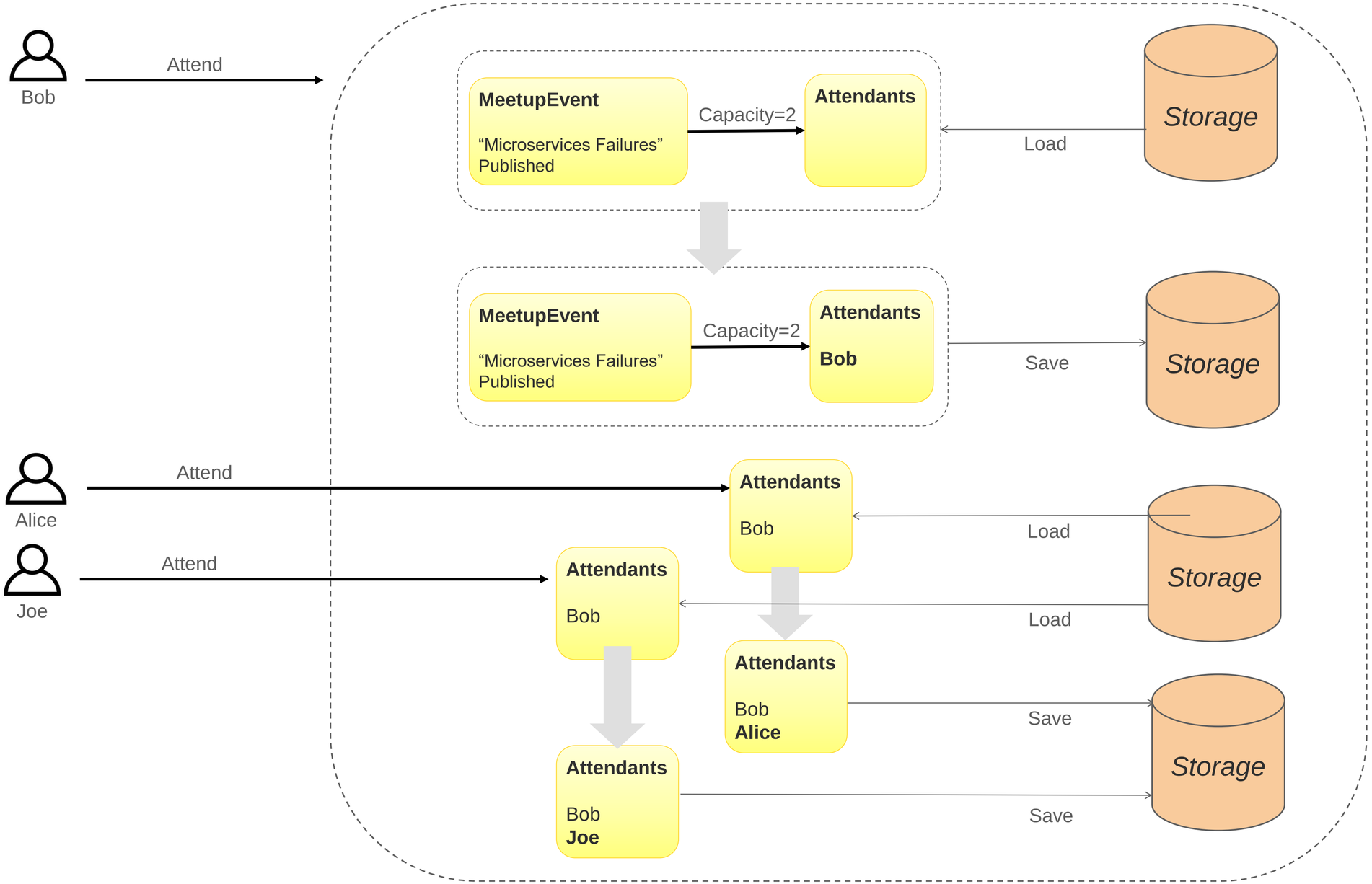
DDD Tactical Patterns: Aggregate versioning and Optimistic concurrency



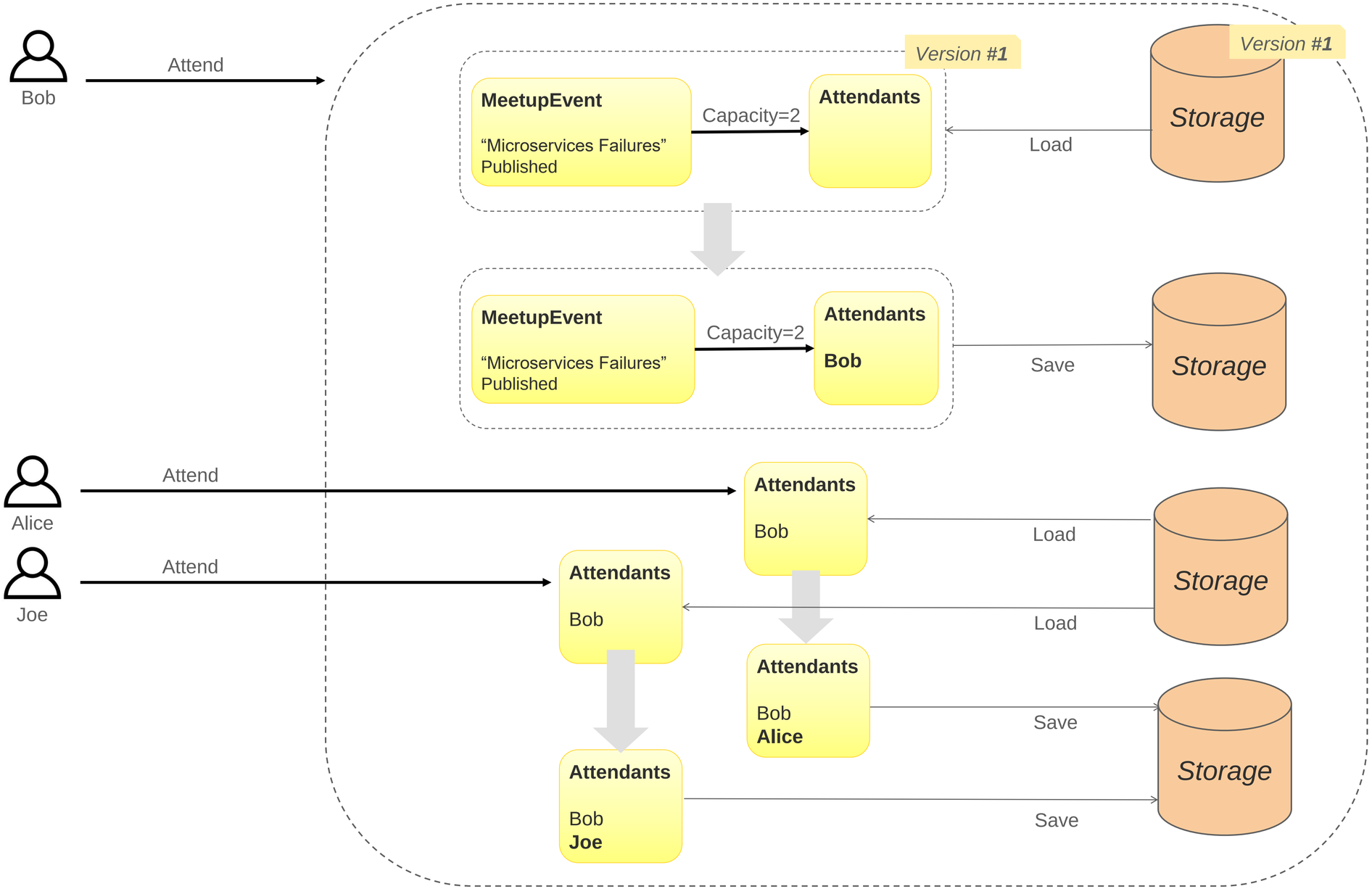
DDD Tactical Patterns: Aggregate versioning and Optimistic concurrency



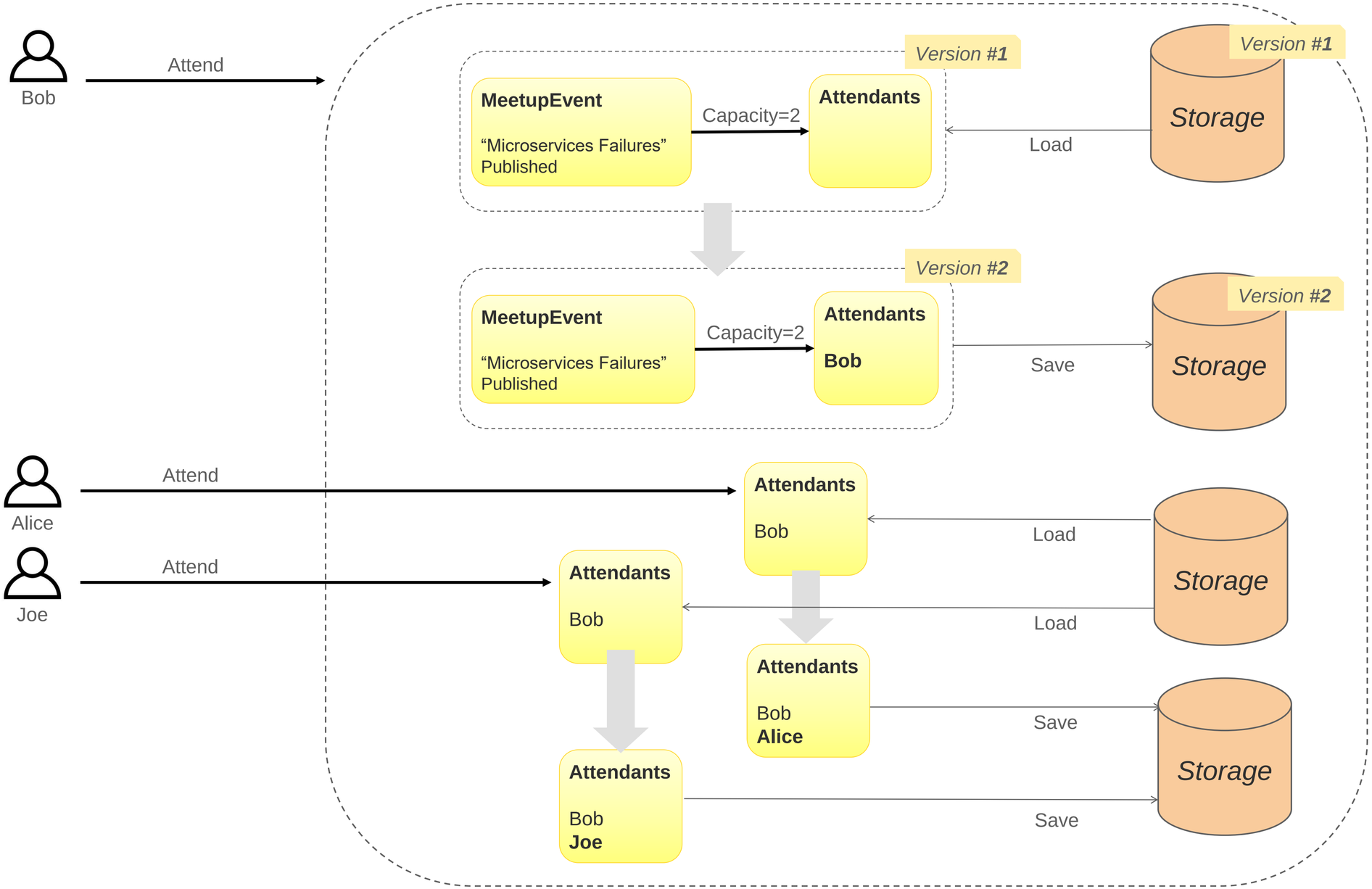
DDD Tactical Patterns: Aggregate versioning and Optimistic concurrency



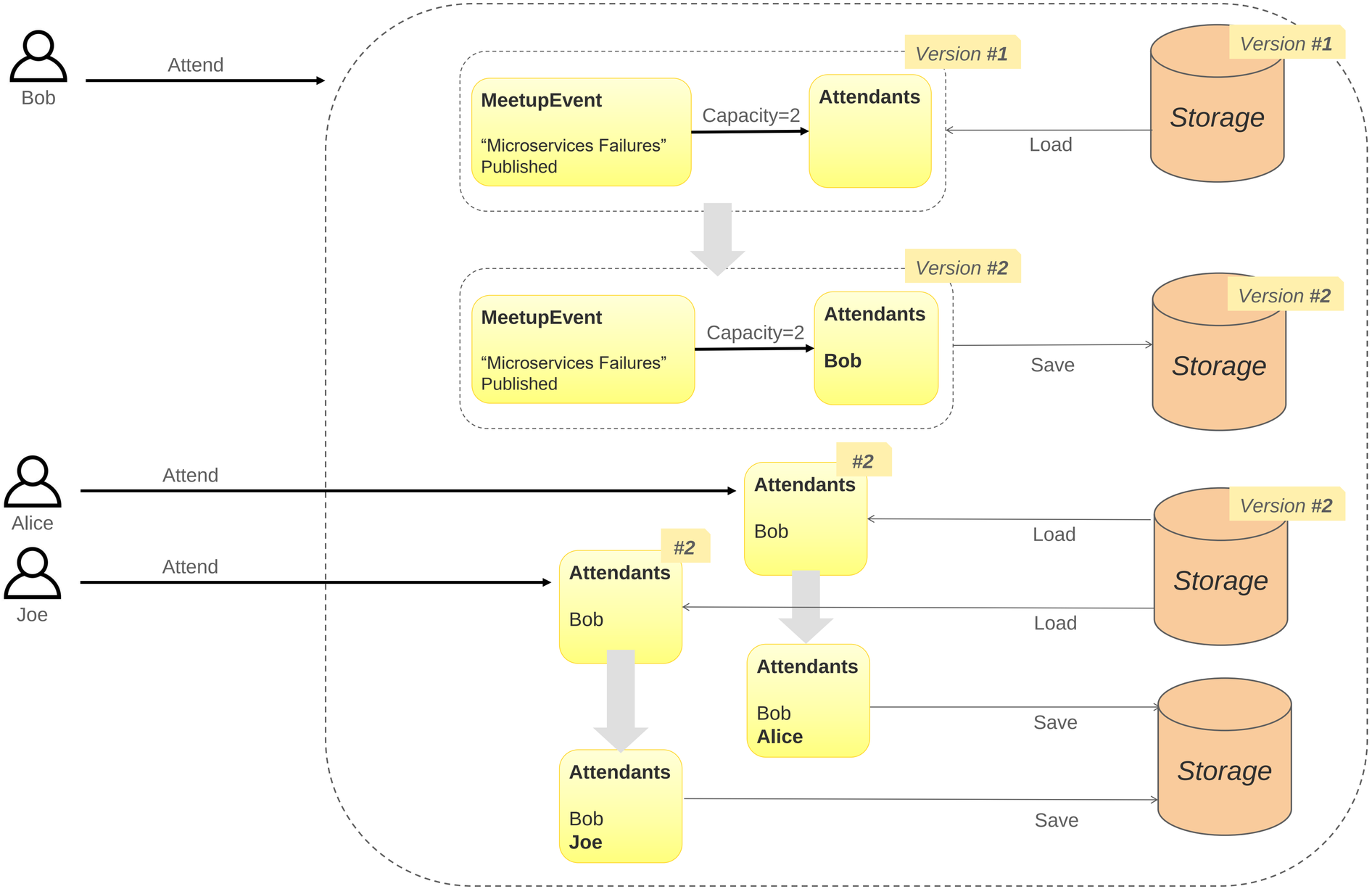
DDD Tactical Patterns: Aggregate versioning and Optimistic concurrency



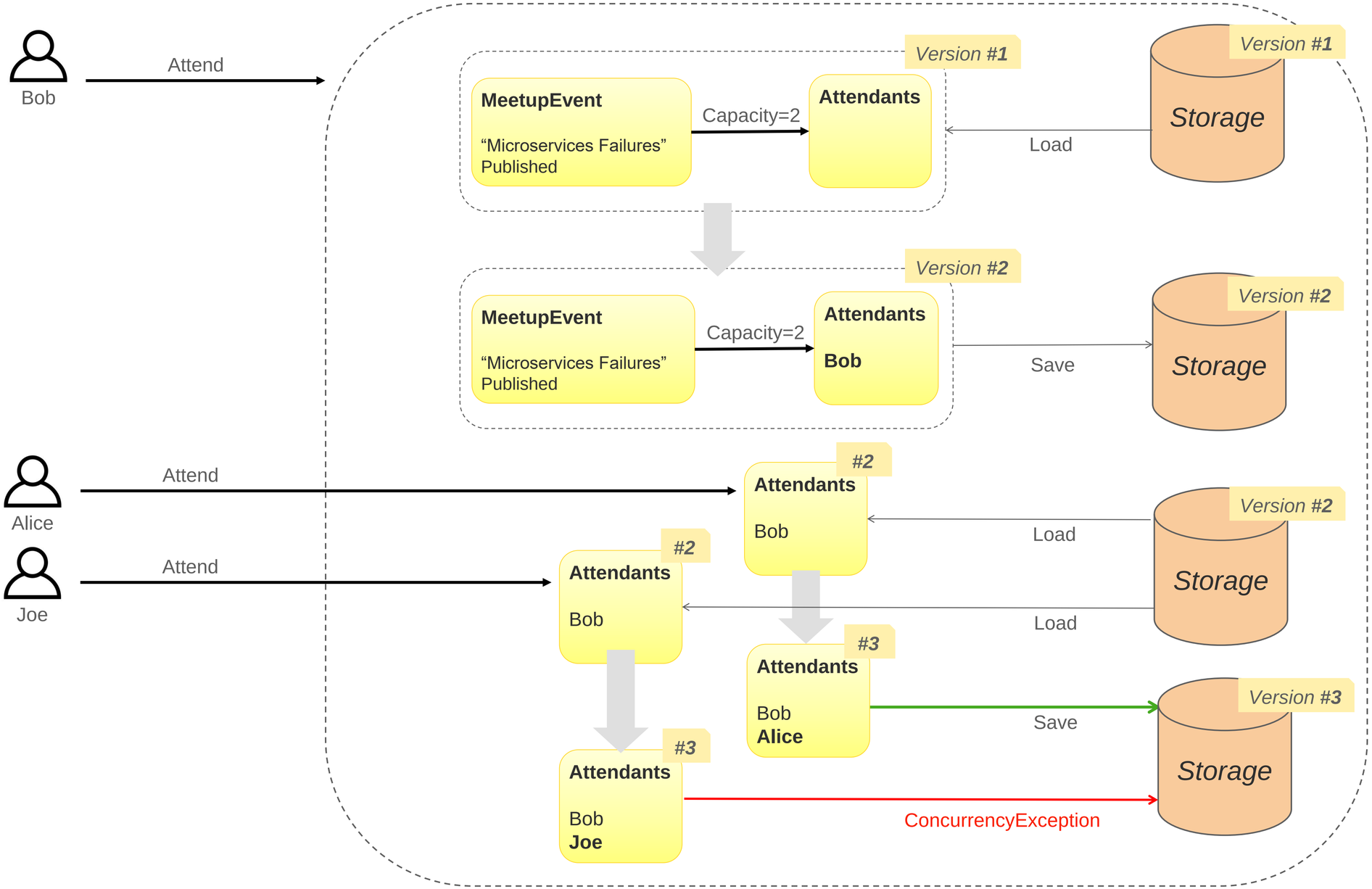
DDD Tactical Patterns: Aggregate versioning and Optimistic concurrency



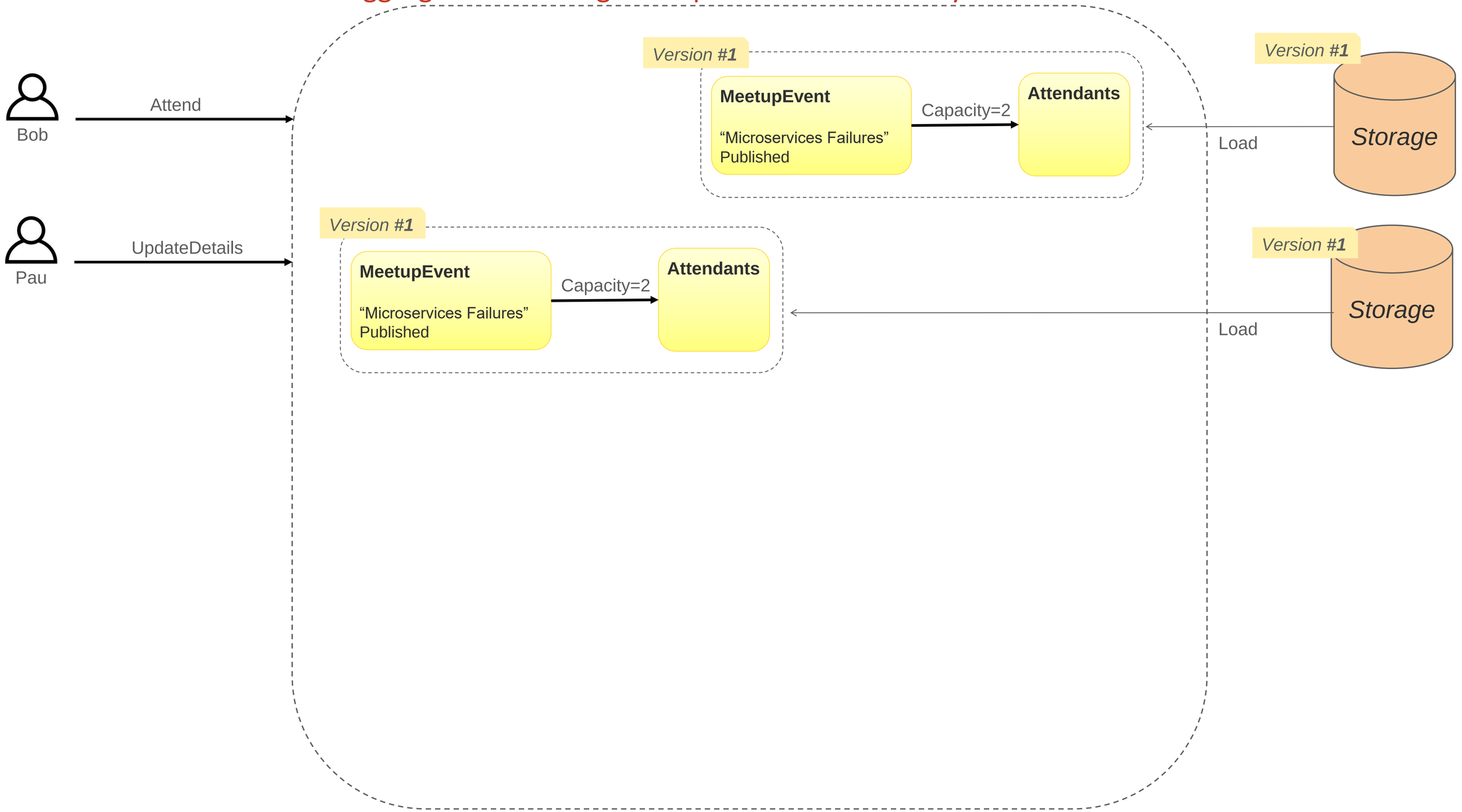
DDD Tactical Patterns: Aggregate versioning and Optimistic concurrency



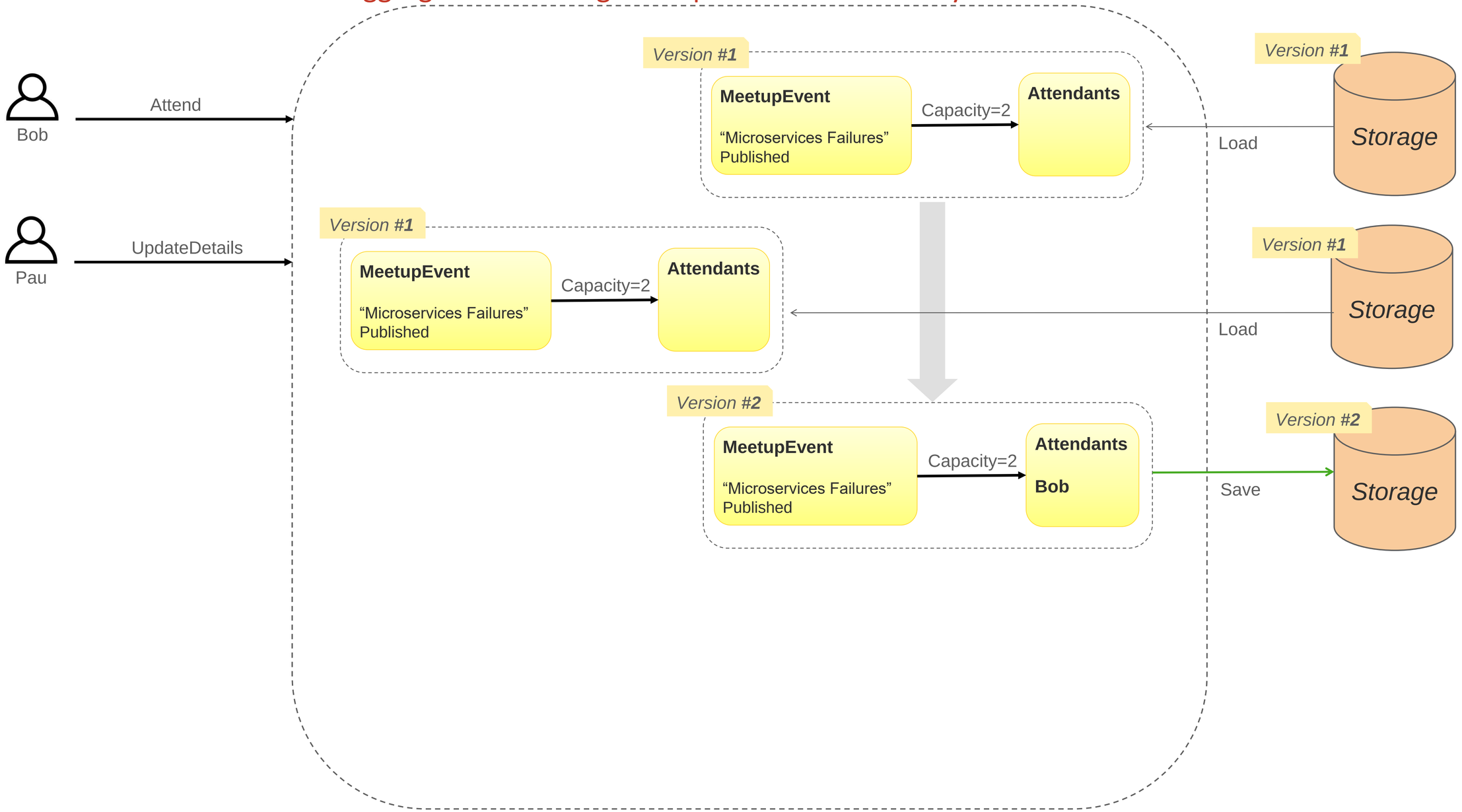
DDD Tactical Patterns: Aggregate versioning and Optimistic concurrency



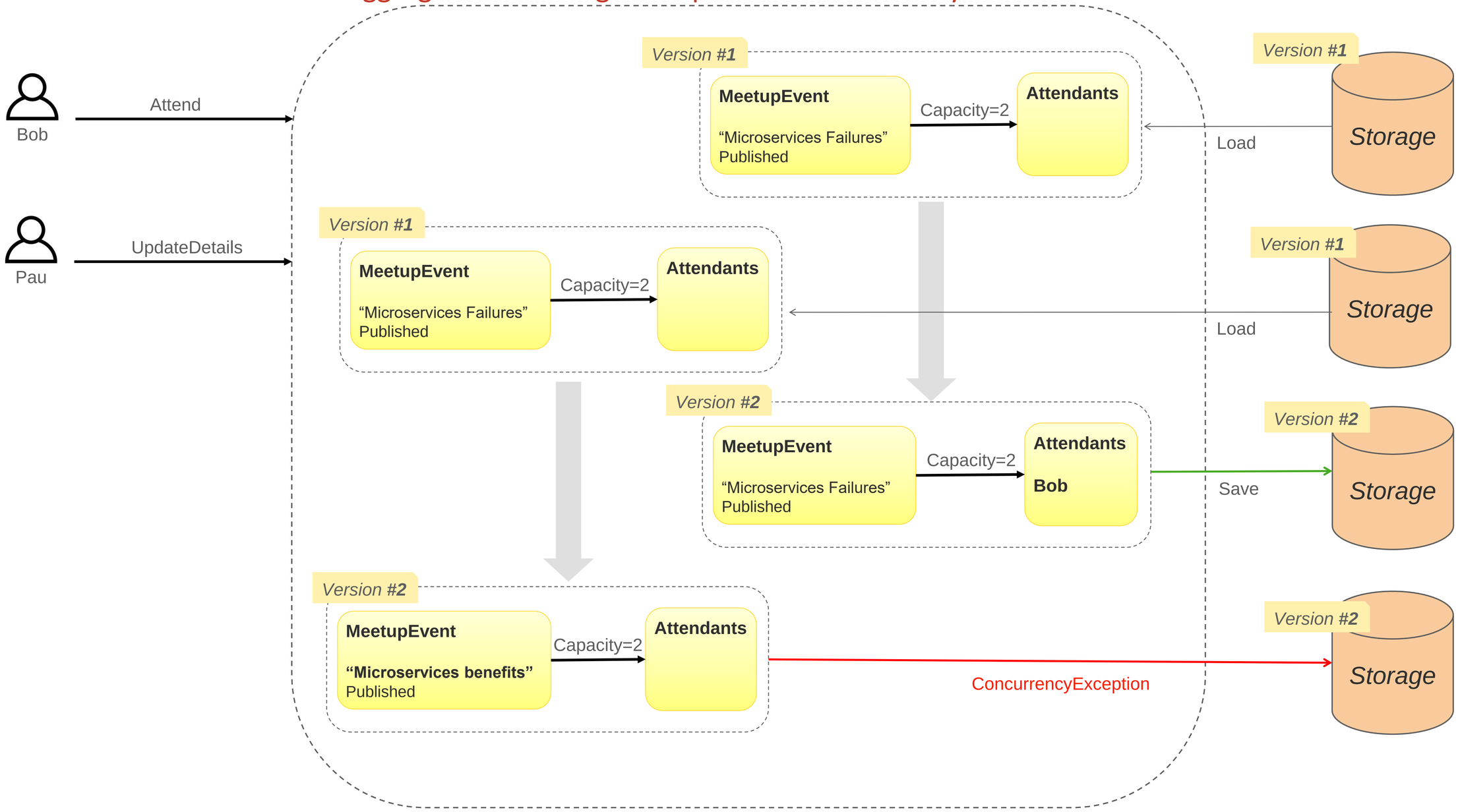
DDD Tactical Patterns: Aggregate versioning and Optimistic concurrency



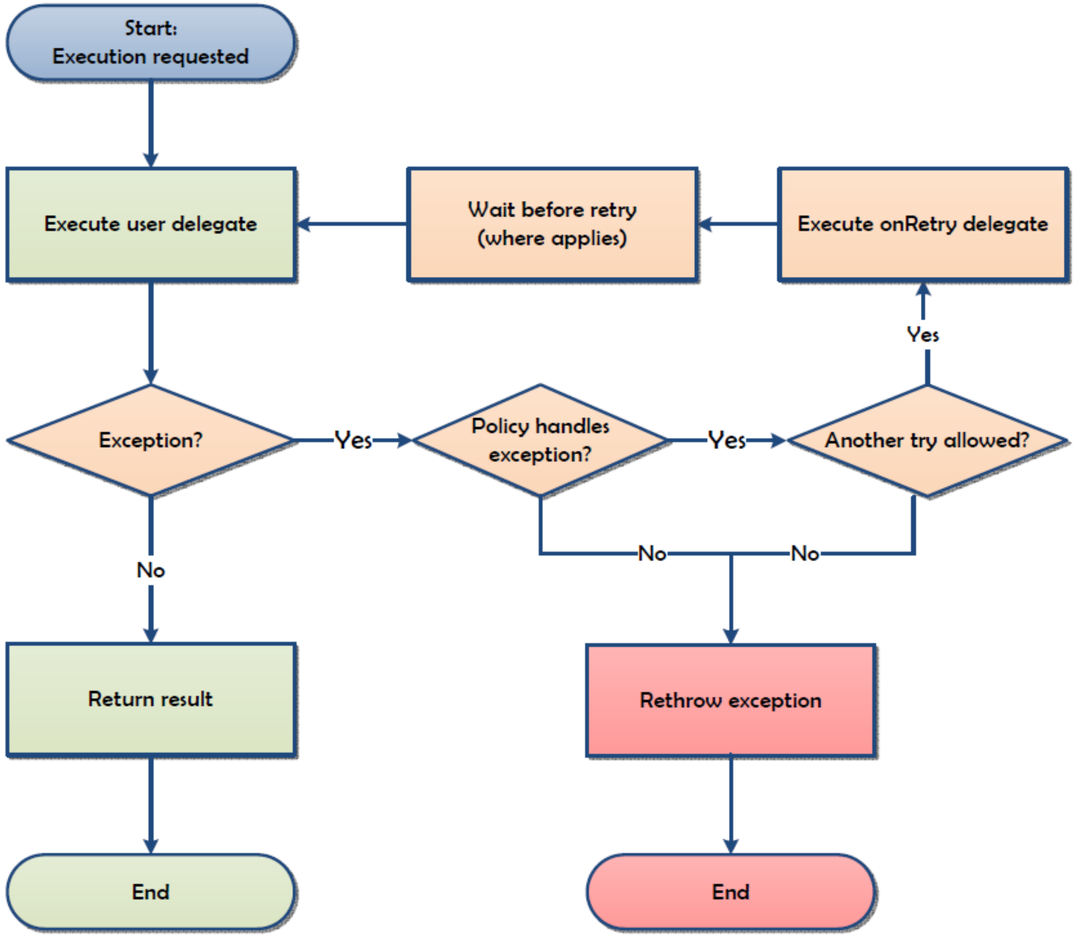
DDD Tactical Patterns: Aggregate versioning and Optimistic concurrency



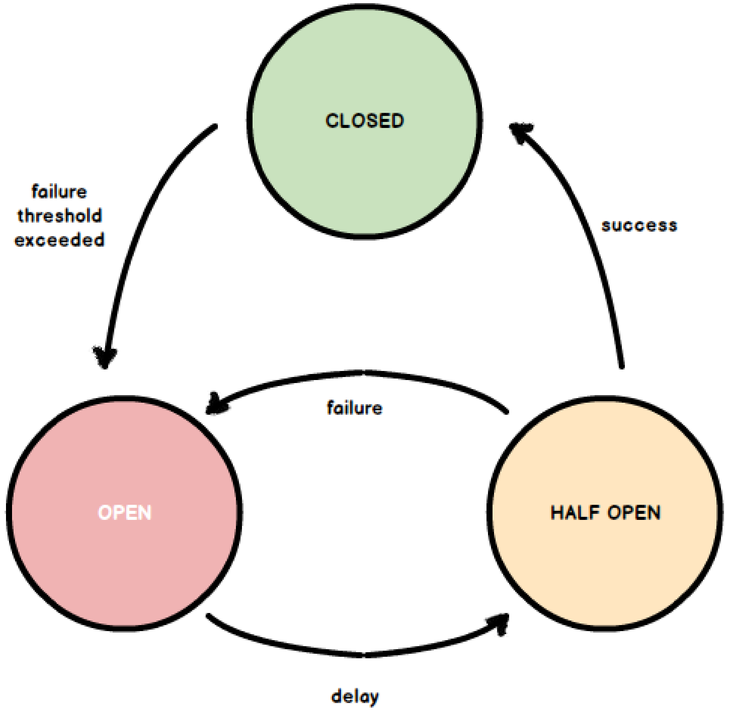
DDD Tactical Patterns: Aggregate versioning and Optimistic concurrency



Polly: Retry and Circuit Breaker

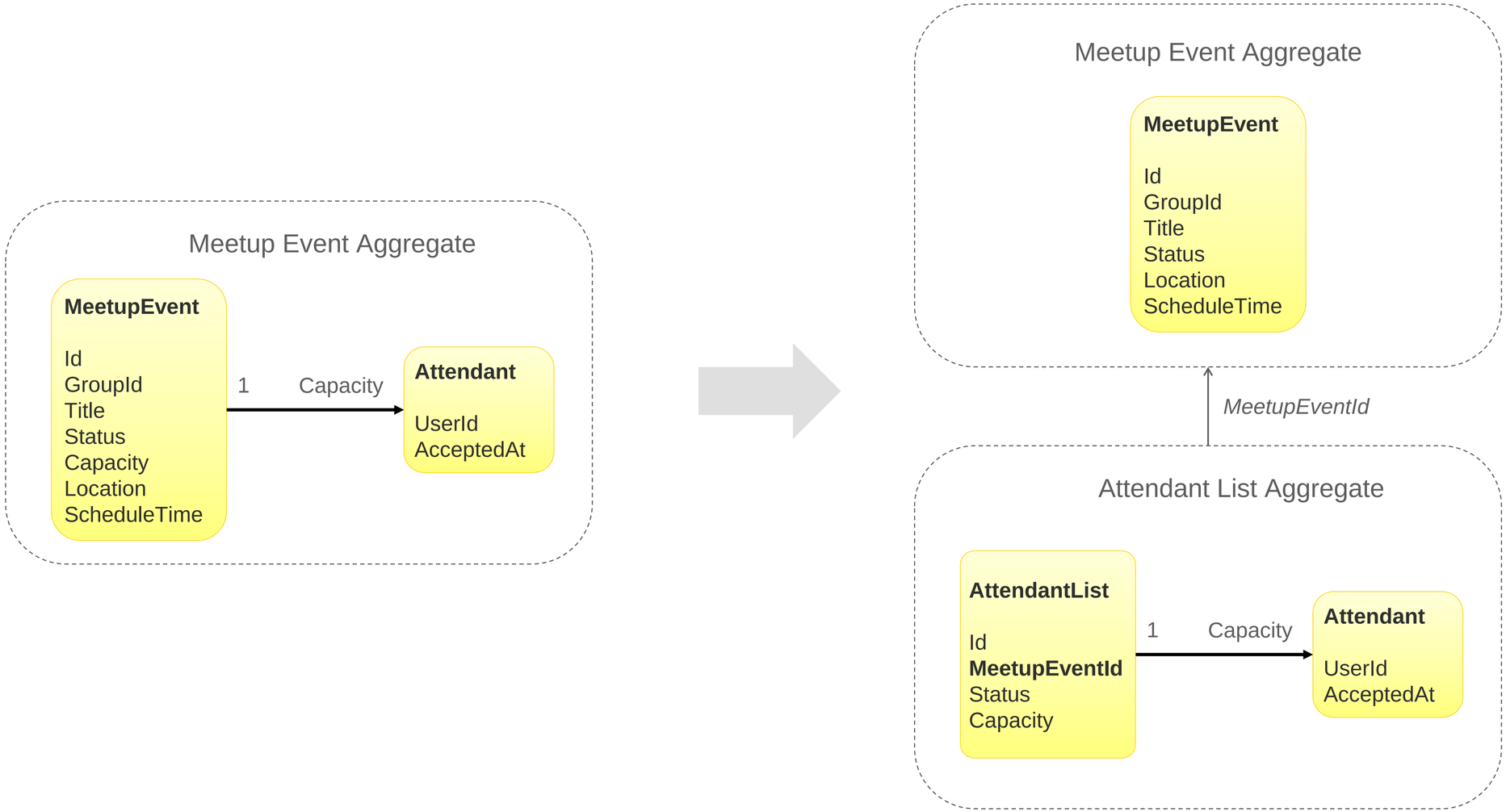


https://github.com/App-vNext/Polly/wiki/Retry



https://github.com/App-vNext/Polly/wiki/Circuit-Breaker

DDD Tactical Patterns: Aggregate Redesign



DDD Tactical Patterns: Aggregate Resources

Developing microservices with aggregates - Chris Richardson

http://www.kamilgrzybek.com/design/handling-concurrency-aggregate-pattern-and-ef-core/

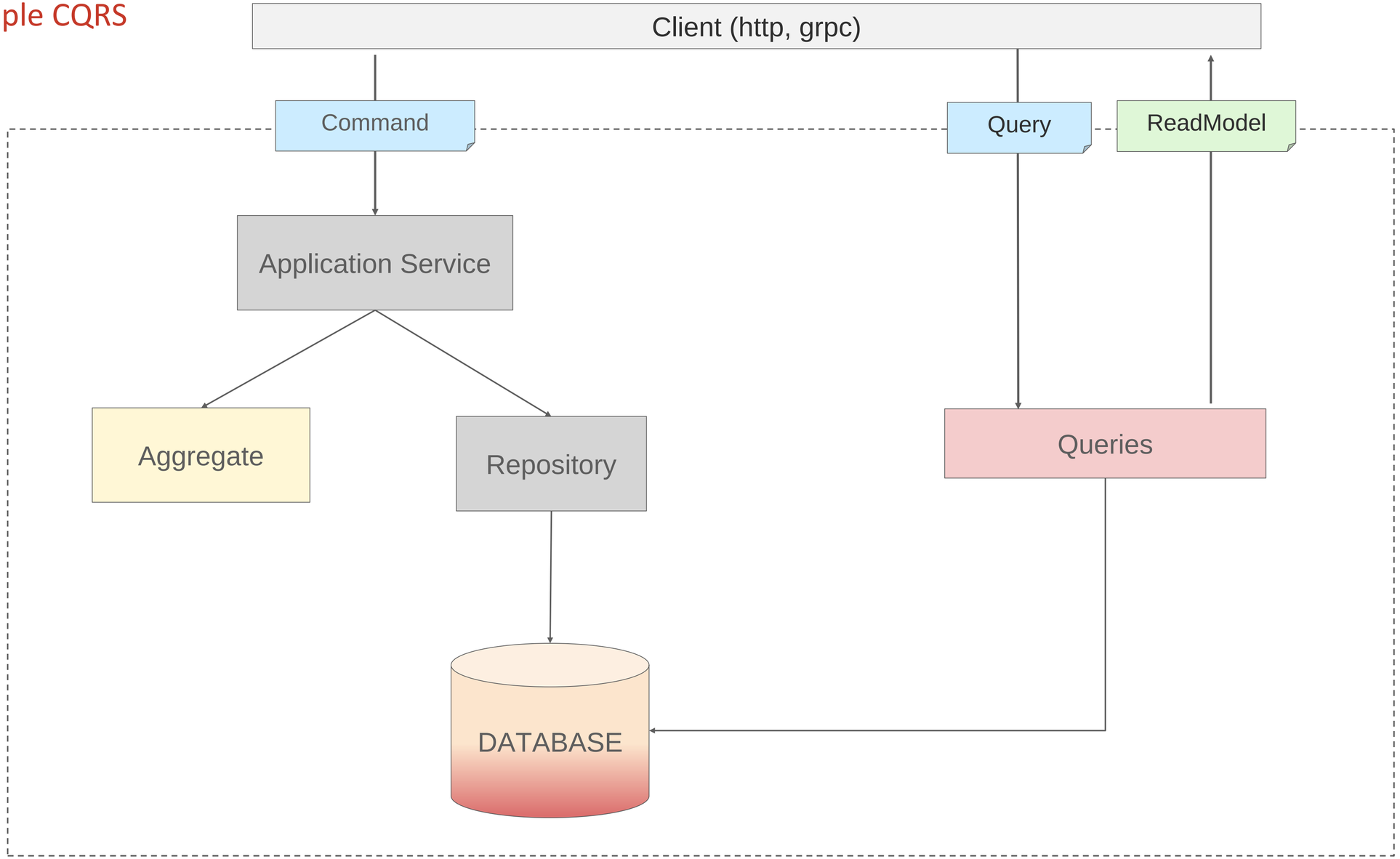
https://www.dddcommunity.org/wp-content/uploads/files/pdf_articles/Vernon_2011_1.pdf

https://www.dddcommunity.org/wp-content/uploads/files/pdf_articles/Vernon_2011_2.pdf

https://www.dddcommunity.org/wp-content/uploads/files/pdf_articles/Vernon_2011_3.pdf

https://docs.particular.net/persistence/mongodb/document-version

Simple CQRS


Client (http, grpc)
Command
Query
ReadModel
Application Service
Aggregate
Repository
Queries
DATABASE

DDD Tactical Patterns: Value objects

- **Primitive Obsession Problem**

- When a domain object (aggregate) begins to contain ***too many*** **responsibilities**, we start to break out those extra responsibilities into things like value objects.

- **Value Objects** are a fundamental building block of Domain-Driven Design, and they're used to **model concepts** of your Ubiquitous Language in code.

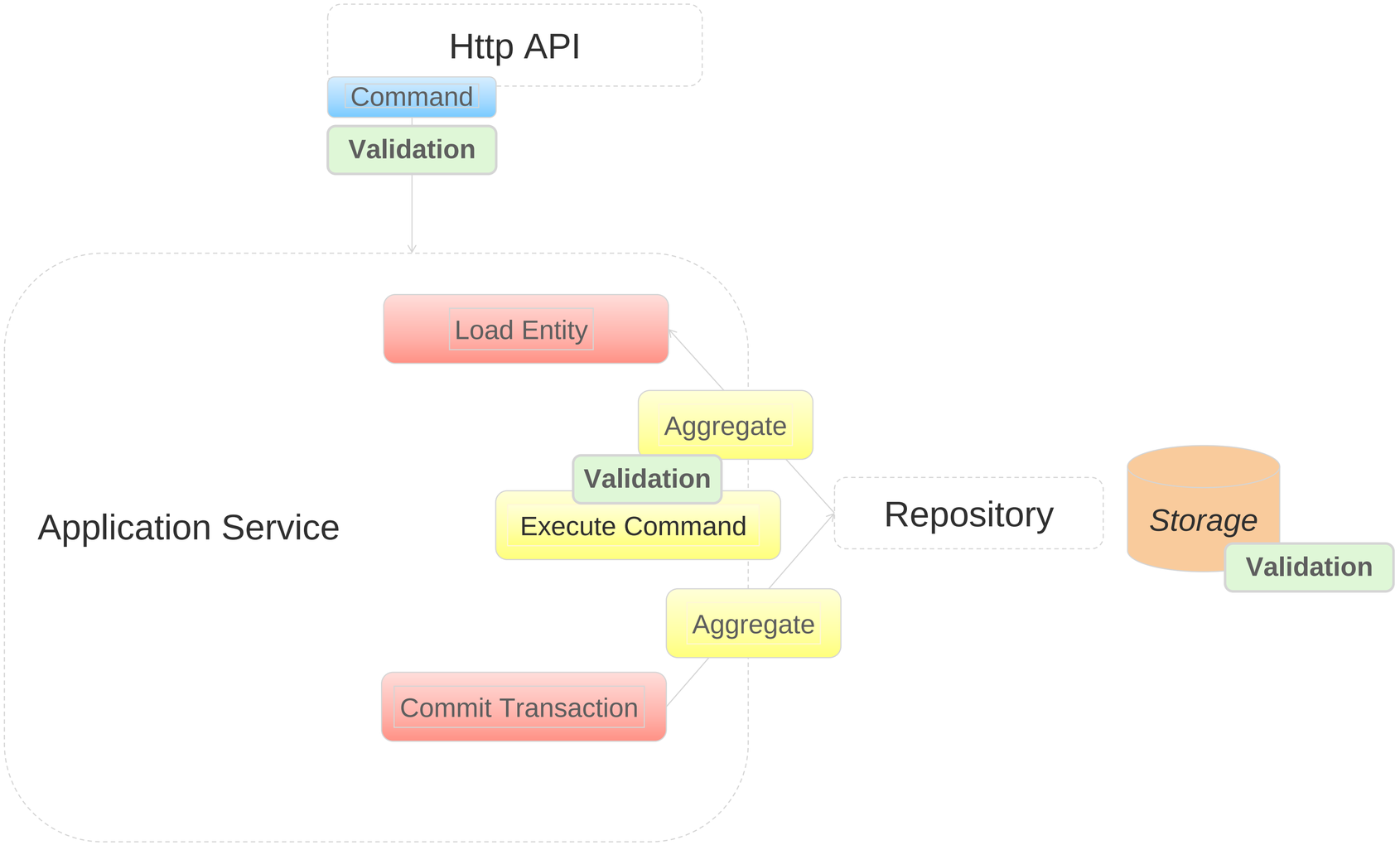
- Value objects **doesn't have identity**.

- **Immutability** and Make illegal states irrepresentable

- Examples: Address, Money, DateTime, 3DPoint, Guid

https://martinfowler.com/bliki/ValueObject.html

DDD Tactical Patterns: Value objects vs Validation



DDD Tactical Patterns: Value objects vs Validation