*This is a summary of the****Introducing Domain-Oriented Microservice Architecture****whitepaper, which can be found at*[***https://eng.uber.com/microservice-architecture/***](https://eng.uber.com/microservice-architecture/)

What is a Microservice?

1. Why we do we adopt Microservice architecture? Organizations adopt Microservice for an operational benefit at the expense of performance.

Why was DOMA in Uber?

1. Availability Risks - A single regression within a monolithic code base can bring the whole system down.
2. Risky, expensive deployments- These are painful and time consuming to perform with the frequent need for rollbacks.
3. Poor separation of concerns- It was difficult to maintain good separations of concerns with a huge code base.
4. Inefficient execution- It is difficult for teams to execute independently.

Advantages of deploying DOMA

1. System reliability - A single service can go down (and be rolled back) without taking down the whole system.
2. Separation of concerns - It will more clearly defining the roles of different components.
3. Clear ownership - Services are typically owned at the individual, team, or org level enabling faster growth.
4. Developer velocity - Teams can deploy their code independently, which enables them to execute at their own pace.

DOMA – Principles and Terminologies

1. Domains - It is the collection of related microservices.
2. Layer design - The layer that the domain belongs to establish what dependencies the microservices within that domain are allowed to take on.
3. Gateways - Clean interfaces for domains treated as a single point of entry into the collection.
4. Extension Architecture - It supports well defined extension points within the domain.

Uber’s Implementation of DOMA

1. Domain design A. How big should a domain be? Some domains can include tens of services, some domains only a single service. We must think carefully about the logical role of each collection.
2. Layer design - It describes a mechanism for thinking about failure blast radius and product specificity across service dependencies.
3. Types of Layers
   1. Infrastructure layer-Provides functionality that any engineering organization could use.
   2. Business layer - Provide functionality that Uber as an organization could use, but that is not specific to a particular product category or line of business (LOB) such as Rides, Eats, or Freight.
   3. Product layer - Provides functionality that relates to a particular product category or LOB, but is agnostic to the mobile application.
   4. Presentation Layer- Provide functionality that directly relates to features that exist within a consumer-facing application (mobile/web).
   5. Edge layer- Safely exposes Uber services to the outside world.
4. Gateways - gateways exclusively as a single entry-point into a collection of underlying services, which we call a domain.
   1. Benefits
      1. Future Migrations
      2. Discoverability
      3. Reduction in system complexity
5. Extensions - It provides a mechanism for extending the functionality of an underlying service without changing the actual implementation of that service and without impacting its overall reliability.
   1. Logic Extensions-Provide a mechanism for extending the underlying logic of a service.
      1. Plugin Pattern- Extending teams can implement extension logic in an interface-driven way without modifying the core code of the underlying platform.
      2. Go-online Endpoint- Each extension to conform to with a predefined request type and a response.
   2. Data Extensions - Provide a mechanism for attaching arbitrary data to an interface to avoid bloat in core platform data models.
      1. Protobuf’s Any - We use this protocol so the teams can add arbitrary data to requests. For a simpler implementation, one could just as easily use a JSON string to represent arbitrary data.
6. Custom - Outside of logic and data extensions, many teams at Uber have introduced their own extension patterns that are appropriate for their domain.

Benefits

1. Products and Platforms - Platform support costs often dropped an order of magnitude. Product teams benefited from guard rails and accelerated development.
2. Reduced Complexity - By reducing the number of touchpoints to onboard a new feature, platforms were able to reduce onboarding time by 25-50%.
3. Future Migrations - Ever changing microservices constantly require upstream migrations
4. New Lines of Business and Products - Platforms designed using DOMA have proven to be much more extensible and easier to maintain.

Practical Advice - Adopting DOMA

1. Startups - Microservice architectures often require dedicated engineering resources to support which may be out of budget for an early stage company or else suboptimal from a prioritization perspective.
2. Midsized Companies - Microservice architectures become more obviously useful in midsized companies. It might make sense to adopt extensions at this point to accomplish that goal.
3. Large Engineering Organizations - DOMA is fully useful at this point because several clusters of micro services easily grouped together into domains with a gateway.