MSA School

전자정부 표준 프레임워크 기반의

클라우드 네이티브 애플리케이션 설계

Part1. DDD 와 이벤트스토밍

UEngine

Tutors

장진영 유엔진솔루션즈 대표이사

MSA School 대표 강사
SAFe 공인 컨설턴트
(전) OMG / SEMAT 국제 표준 도구 분과 위원
現 디지털플랫폼정부 기술자문위원
유엔진BPMS / 오픈클라우드 엔진 파운더
클라우드 네이티브 (MSA, DDD) 강의
MSA와 SW모델링 Facebook 그룹 운영
(www.facebook.com/groups/cloudswmoding)

박용주 유엔진솔루션즈 이사

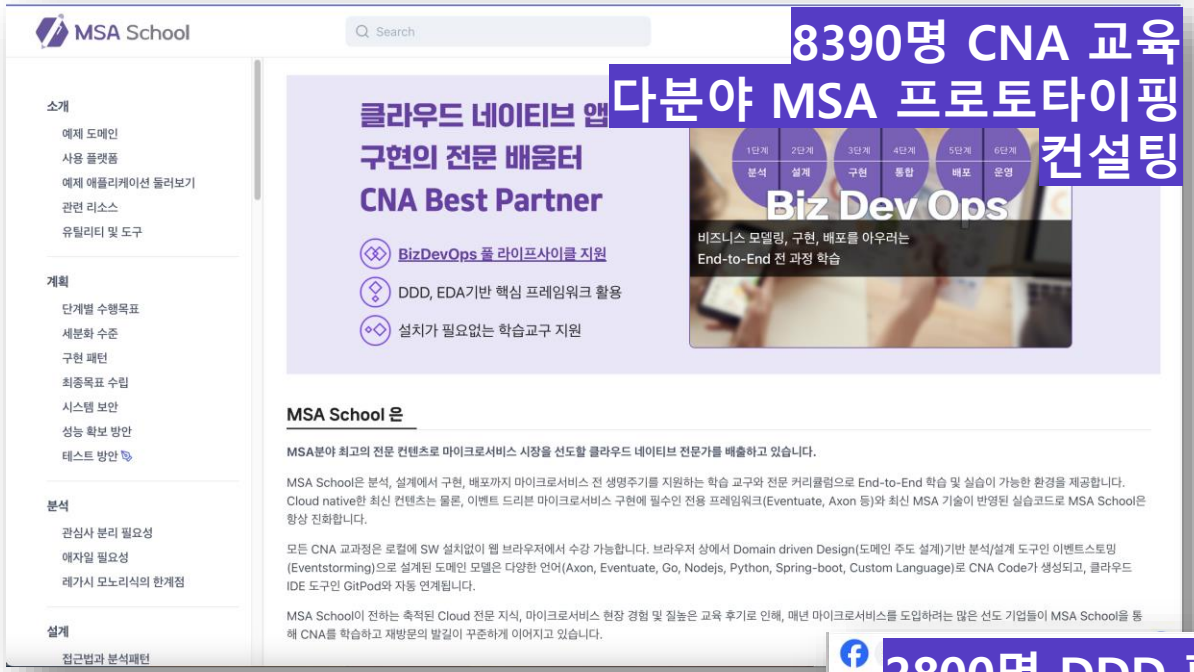
MSA School 대표 강사
• 現 MSA App. Engineering 기업과정
• 現 세종사이버대학교 컴퓨터/AI 공학과 겸임교수
• 한국소프트웨어기술진흥협회 전문강사
• '21 : SK MSA App. Engineering 과정
• '21. 06 : KT Microservice 직무전환과정
• '20. 09 : Doosan Microservices 교육
• '19. 09 : KOSTA Microservices 교육
• '19. 02 : LG CNS 이벤트스토밍 교육

윤성열 드림플로우 대표이사

MSA School 대표 강사
• 現 드림플로우 대표이사
• 現 한국소프트웨어기술진흥협회 BAPF 포럼, 교육과정위원회 및 전문강사
• '18. 10 : 원오원글로벌 디지털팀 팀장
• '18. 04 : TTA 사물인터넷 특별기술위원회 사물인터넷 융합서비스 프로젝트그룹 (SPG11) 부의장
• '17. 03 : 가천대학교 겸임교수

MSA School

About Us



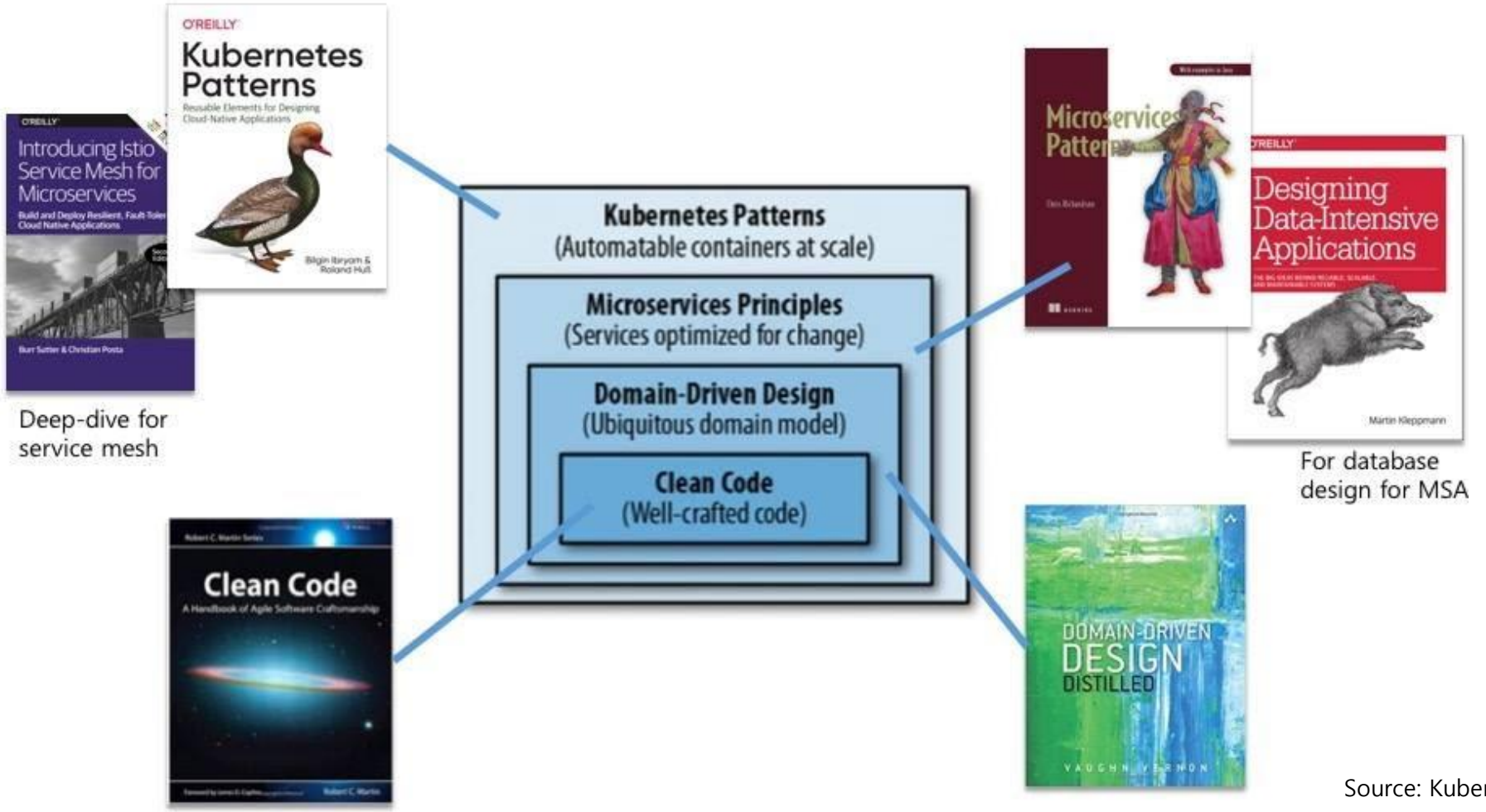
8390명 CNA 교육
다분야 MSA 프로토타이핑
컨설팅

MSA 분석/구현 전문 도구

2800명 DDD 페북그룹 운영

MSA School

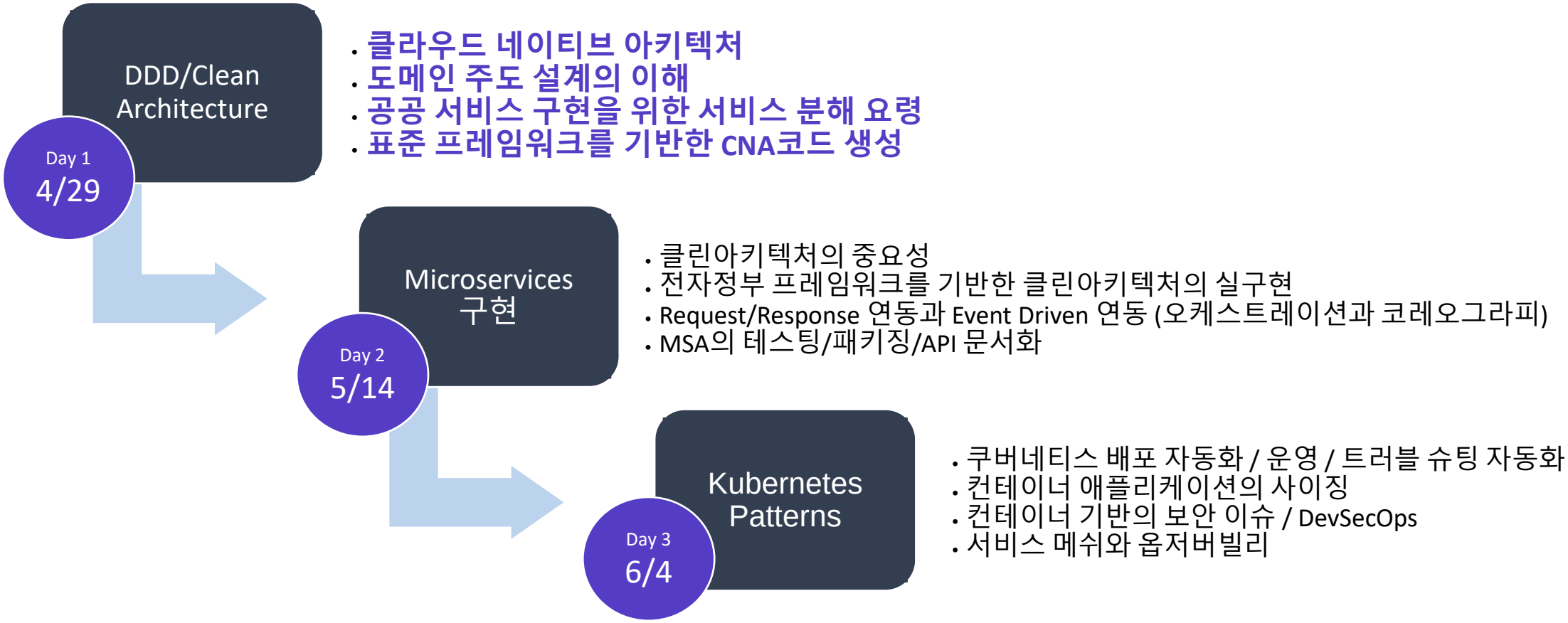
Learning Path to Cloud Native



Source: Kubernetes Patterns

MSA School

Seminar Schedule



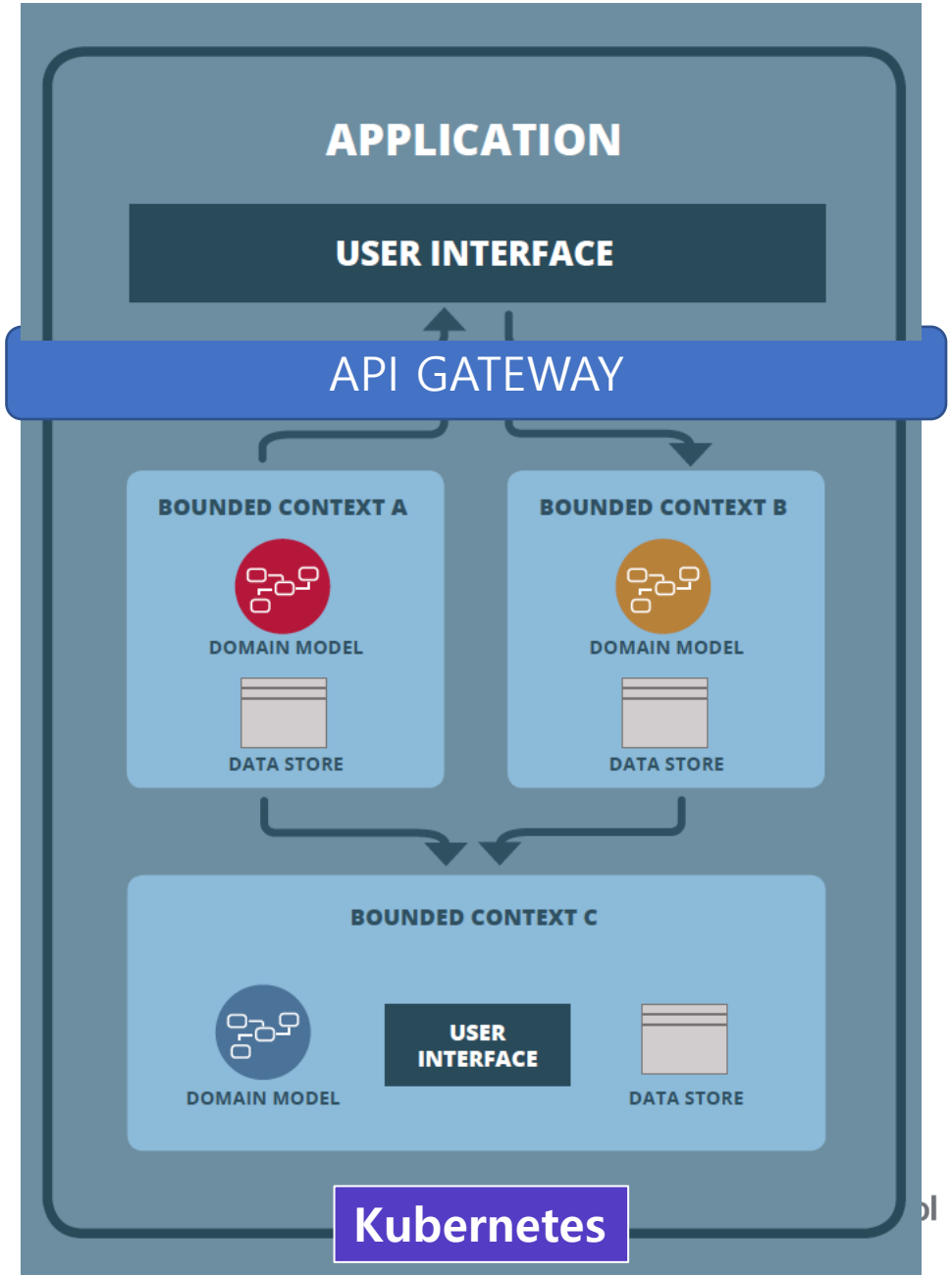
MSA School

Cloud Native Apps.

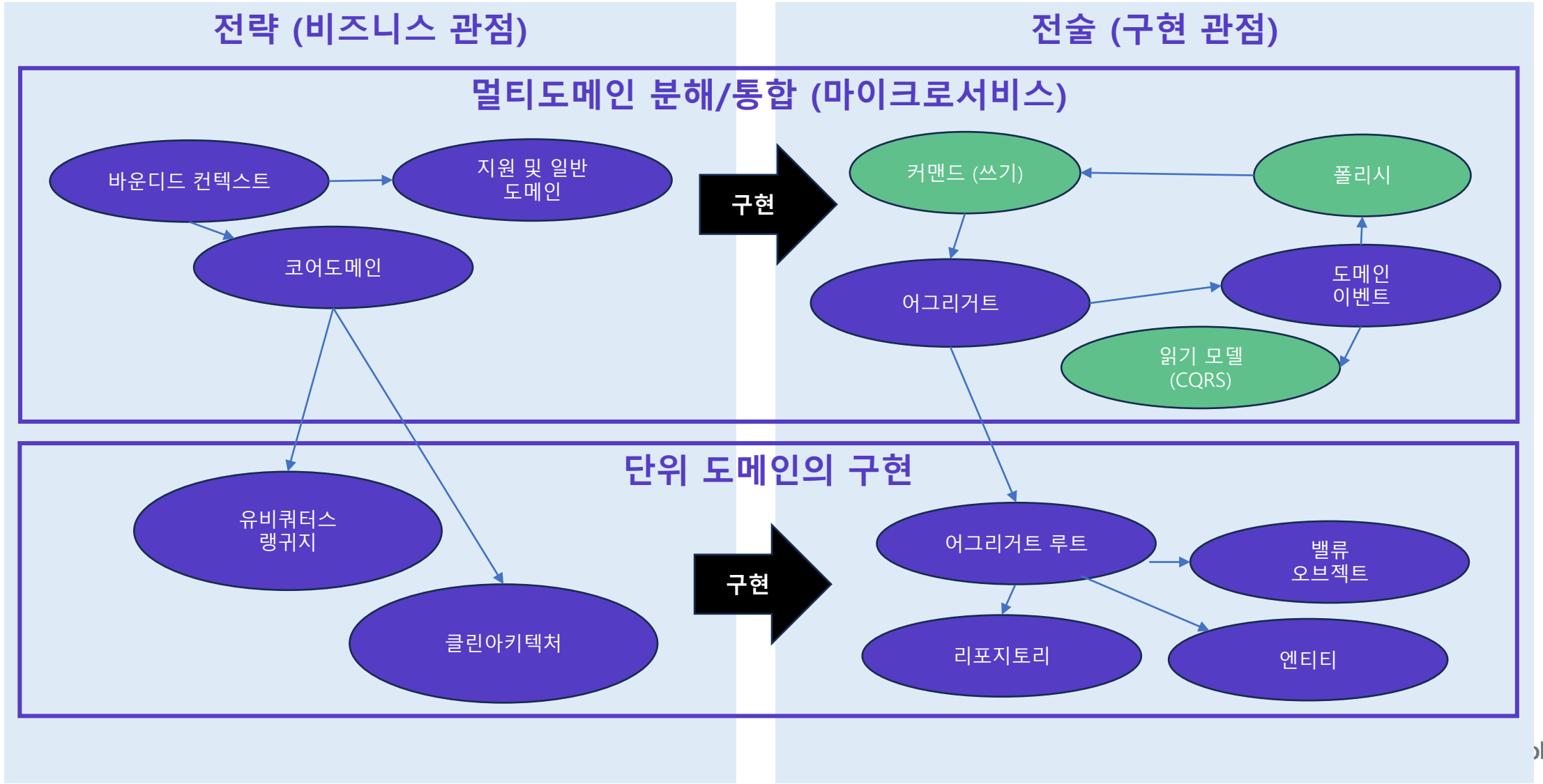
1. 경량의 컨테이너에 패키징한다.
2. 최신 언어와 프레임워크를 이용하여 구현한다.
3. 느슨한 결합구조의 마이크로 서비스로 구성한다.

4. API 를 통한 상호 연동을 한다.
5. 상태가 있는 서비스와 상태가 없는 서비를 엄격히 분리 관리한다.
6. 서버와 OS 의존성을 분리한다.
7. 셀프서비스, 확장성을 가진 클라우드 인프라에 배포한다.
8. 애자일 데브옵스 프로세스를 통하여 관리한다.
9. 용량확장의 자동화
10. 정의된 정책 기반의 리소스 할당

https://thenewstack.io/10-key-attributes-of-cloud-native-applications/

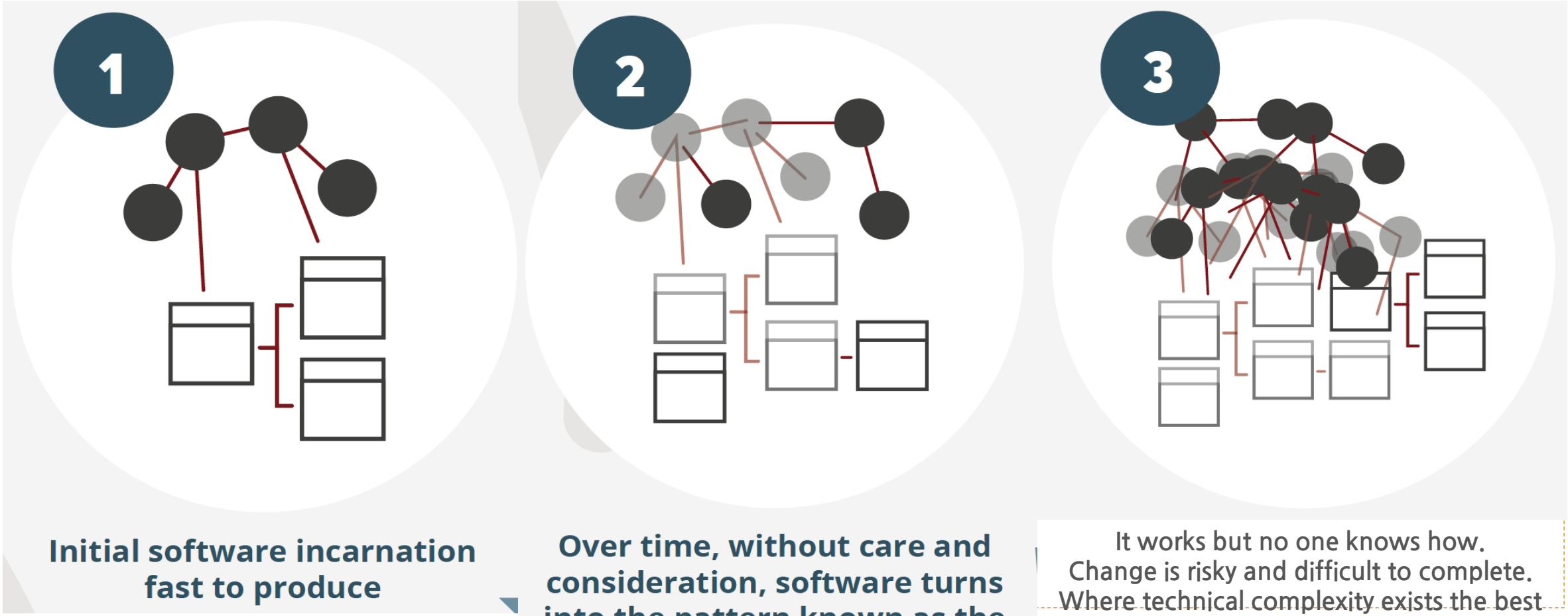


DDD Patterns



소프트웨어가 복잡해지고 관리하기 어려워지는 과정

Initial software incarnation fast to produce



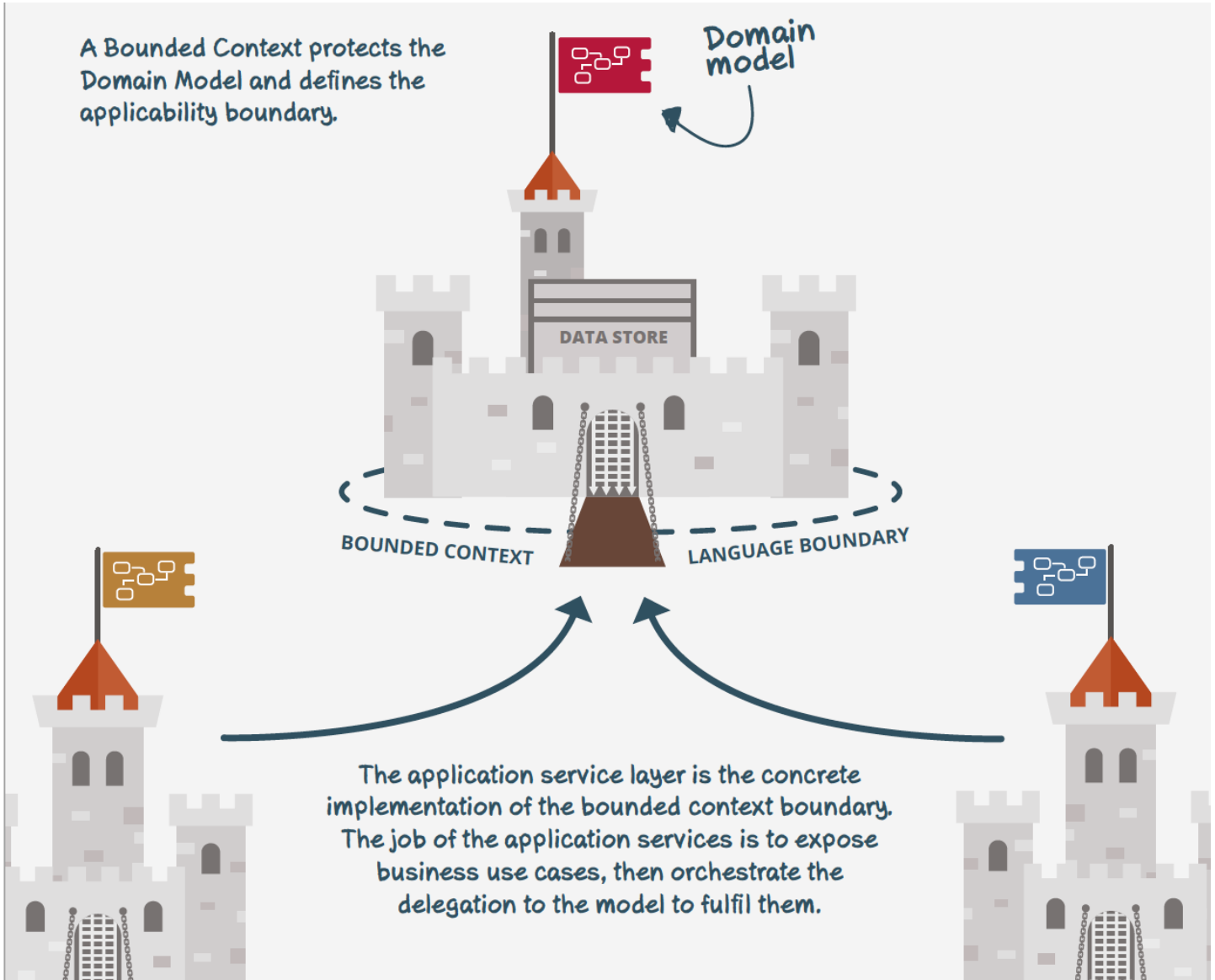
Source: Anatomy of DDD, Scott Millette

DDD Part 1 – Multiple Domain Approach

IMPLEMENT A BOUNDED CONTEXT TO PROTECT A DOMAIN MODEL

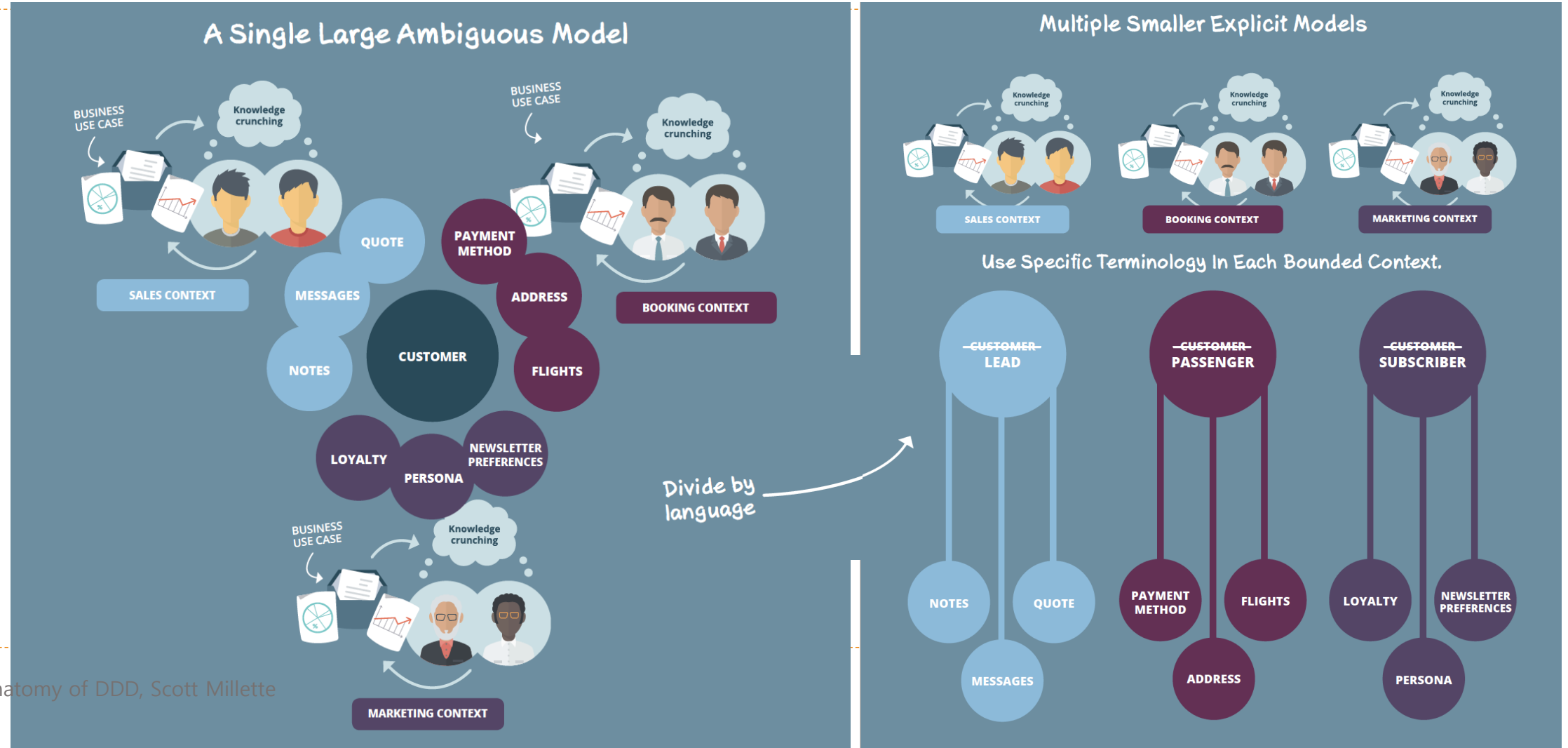
- 도메인 모델의 일관성을 효율적으로 유지하기 위해서는 다른 바운디드 컨텍스트와 클라이언트들에 서비스를 노출하여 연동하면서도 바운디드 컨텍스트가 인프라와 데이터스토어 그리고 어떤 경우 UI 까지도 캡슐화할 수 있도록 하는 것이 좋다.
- 이는 다른 영역의 복잡도를 한정시키면서도 상호연동되는 중요한 역할을 한다.

- E.g. 전자기기의 규격화된 반도체, 자동차의 교체 가능한 부품



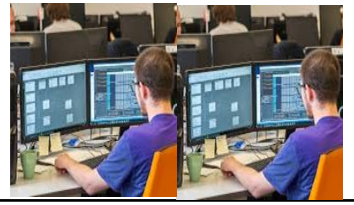
복잡하고 큰 모델을 바운디드 컨텍스트(한정된 맥락)들로 쪼개라

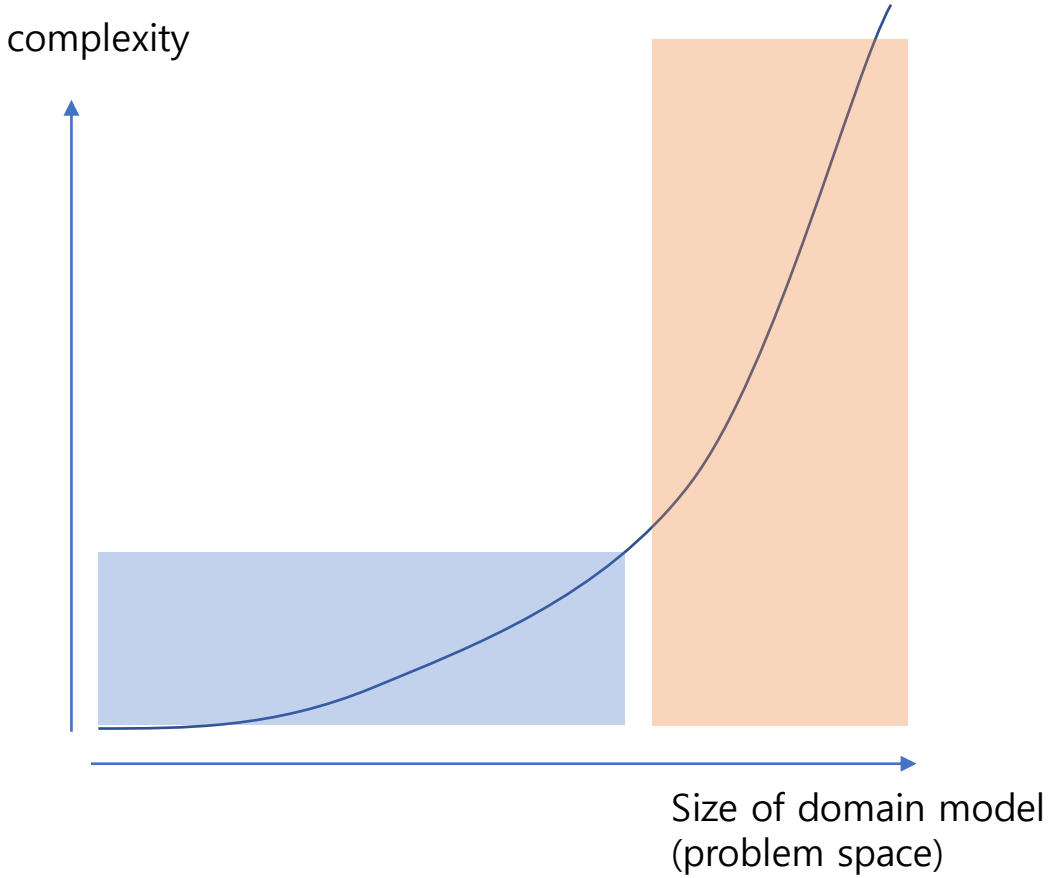
시간이 갈수록 모델은 일관성과 명확성을 잃어가고 복잡도가 증가한다.
다수의 팀이 일을 하면서 언어는 모호해진다. 이러한 경우라면, 크고 복잡한 모델은 특정한 맥락에서 분명하게 이해될 수 있는 명확한 모델인 Bounded Context 로 분리할 필요가 있다.
이러한 바운디드 컨텍스트로 모델들을 격리하고 절연하지 못한 소프트웨어는 " 큰 진흙 덩어리(Big Ball Of Mud)"의 패턴이 되어버리고 만다.



Source: Anatomy of DDD, Scott Millette

언어의 모호성과 개발자 뇌부화의 관계

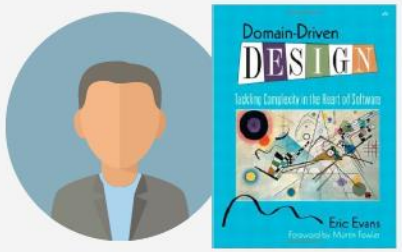
	SW	CONSTRUCTION
Project		
Architecture		
Developer		



MSA School

도메인 주도 설계는 소프트웨어의 복잡성을 관리하는 전략적인 도구입니다

- Domain-Driven Design 은 복잡한 문제 영역에 대한 언어이자 도메인 중심의 소프트웨어 디자인 접근법.
- 에릭에반스의 저서인 "Domain-driven Design: 소프트웨어의 복잡성 관리하기"에서 언급된 용어.
- 팀이 소프트웨어를 저작함에 있어 기술적, 비즈니스적 복잡성을 관리하여 비즈니스를 성공시키기 위한 핵심적인 패턴과 원칙, 그리고 실천법으로 구성됨.



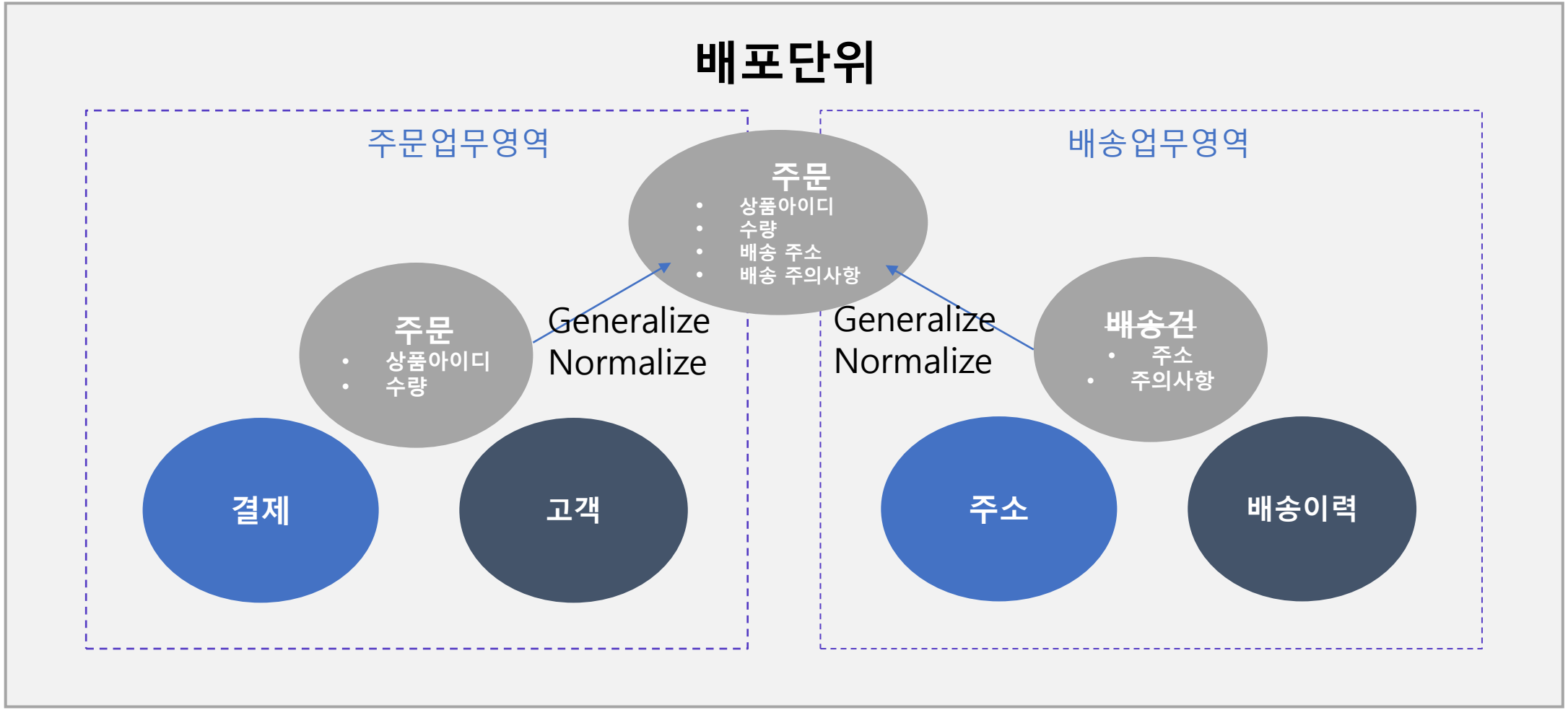
Eric Evans
Domain-Driven Design:
Tackling Complexity in the Heart of Software

Source: Anatomy of DDD, Scott Millette

Bounded Context
(한정된 맥락)

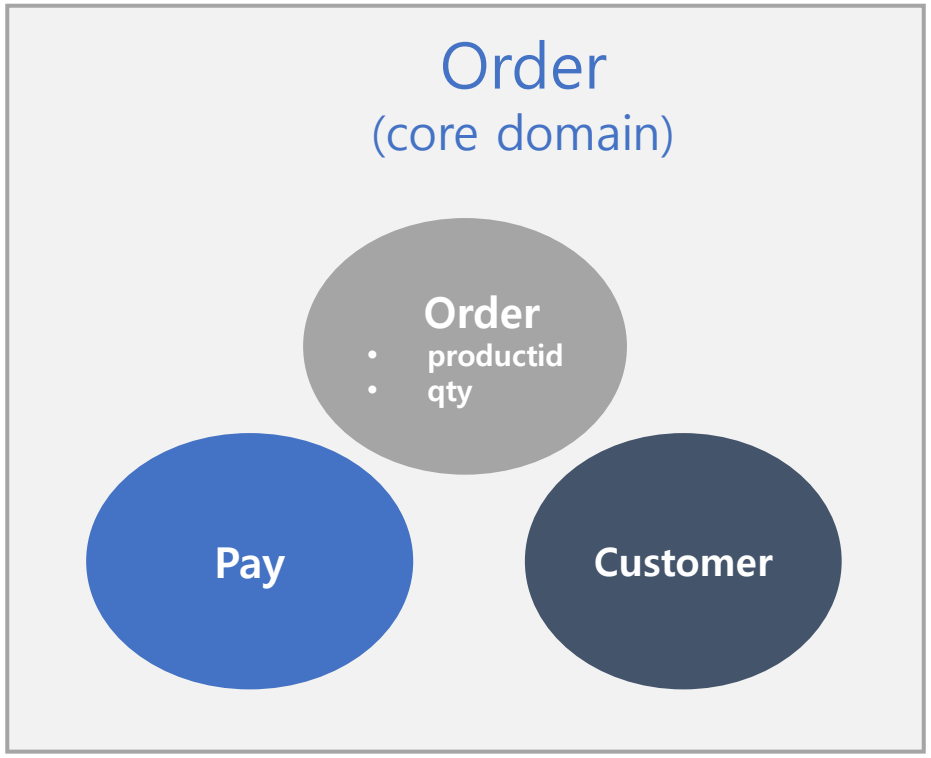
Ubiquitous Language
(도메인 언어)

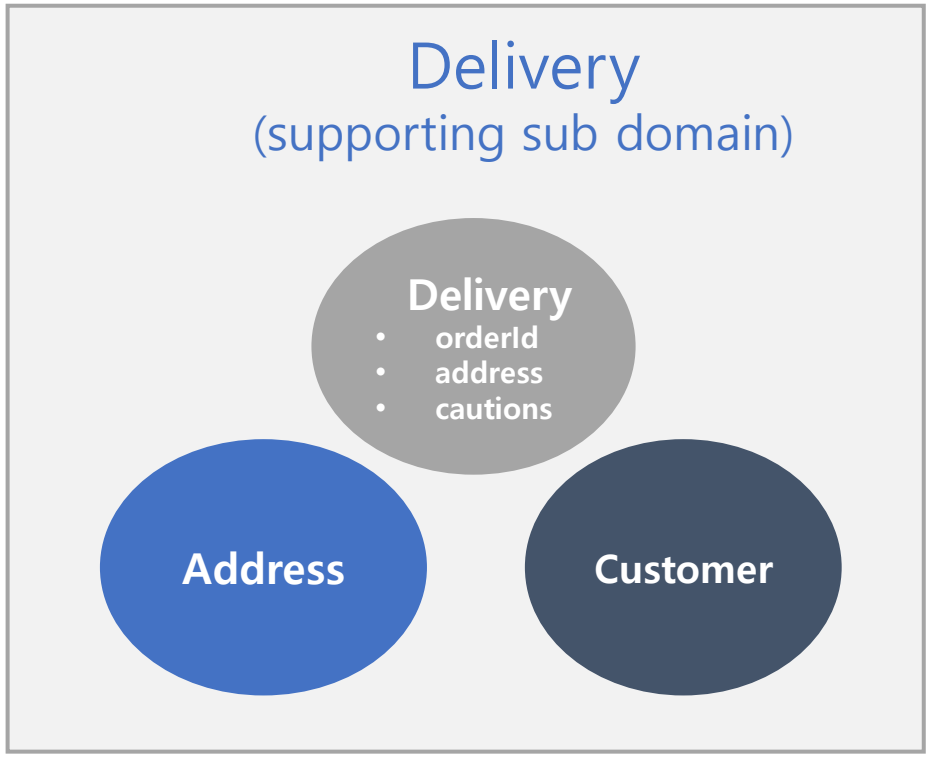
모놀리스: One Size Fits All



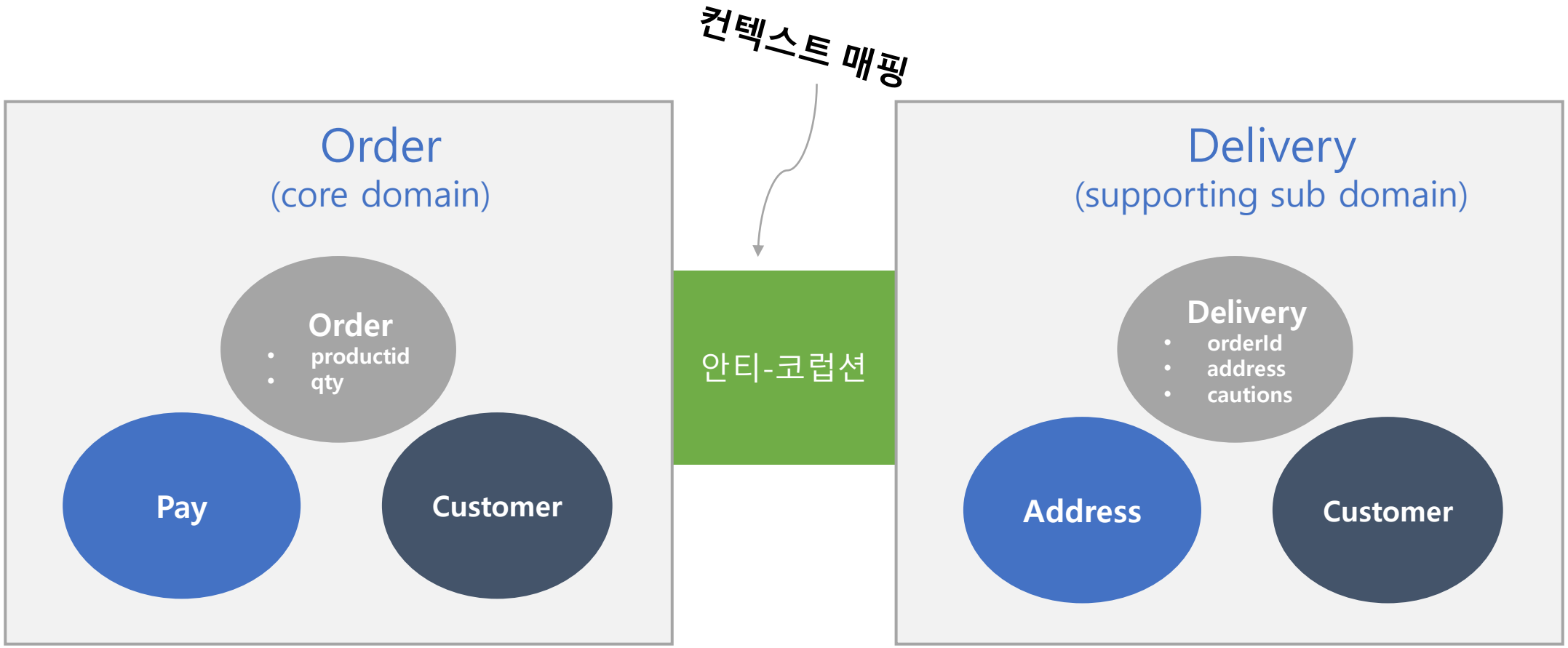
DDD/MSA: 멀티플 모델

다른 서브도메인을 넘어 일반화 하지 마라!





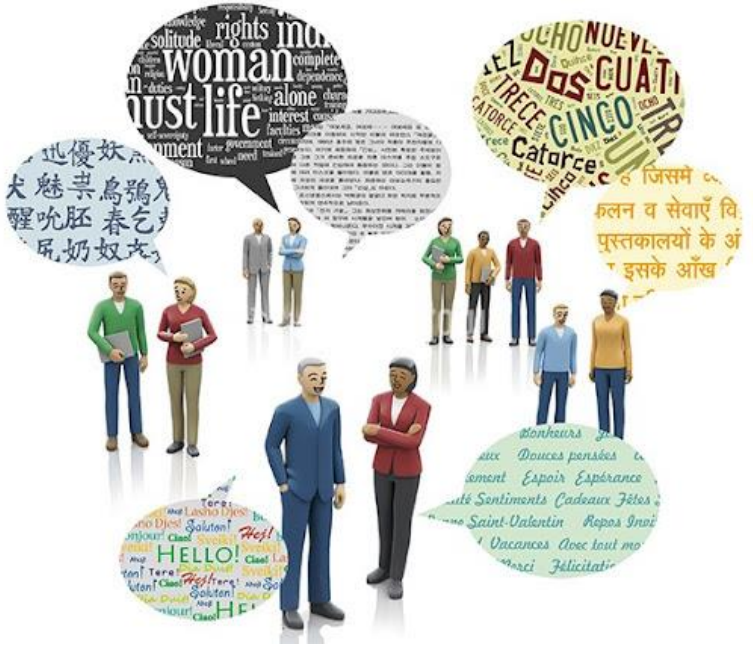
DDD/MSA: 상이한 모델간의 커뮤니케이션



Comparison of Total Communication Cost

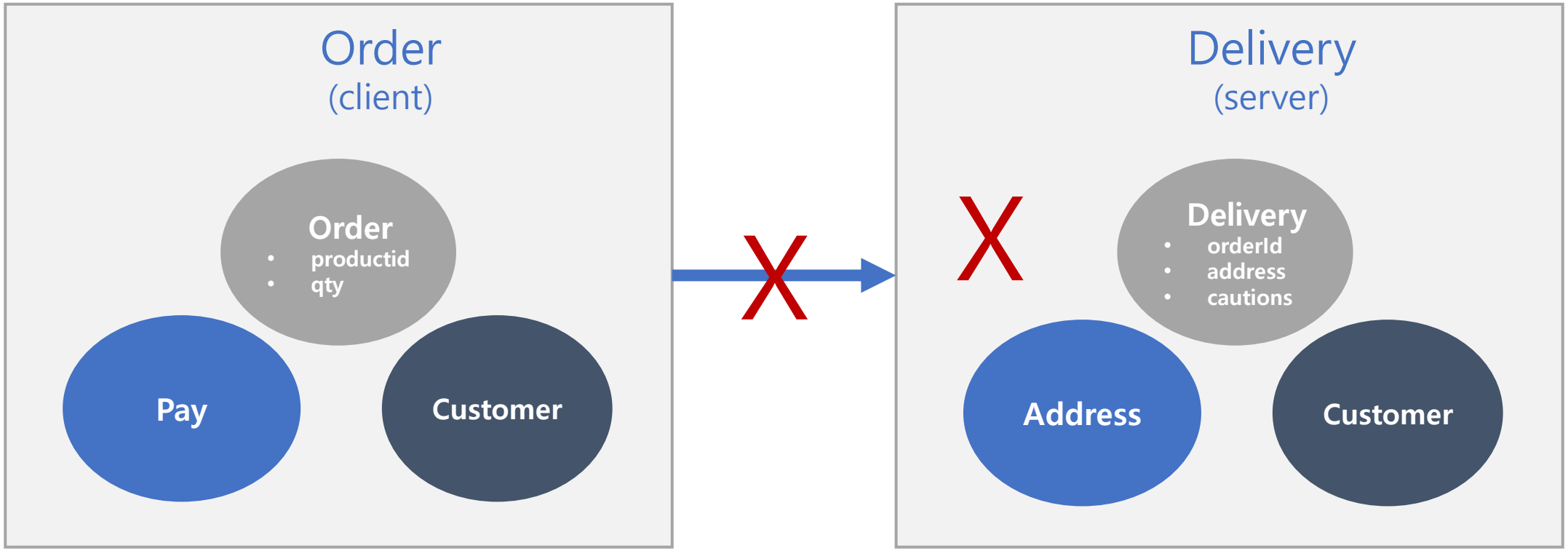
- Generalization and Normalization of single language

- Translators between multiple language
- Anti-corruption between Multiple languages

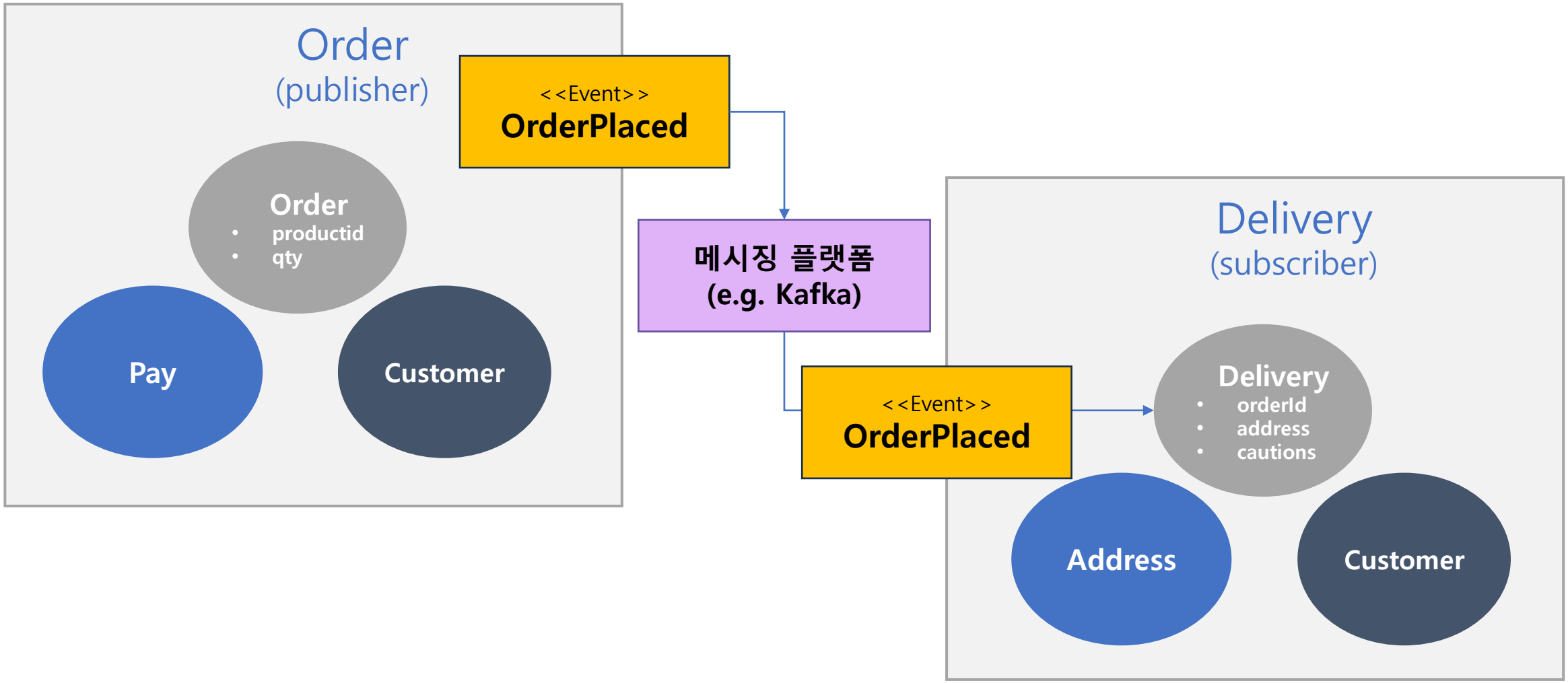


MSA School

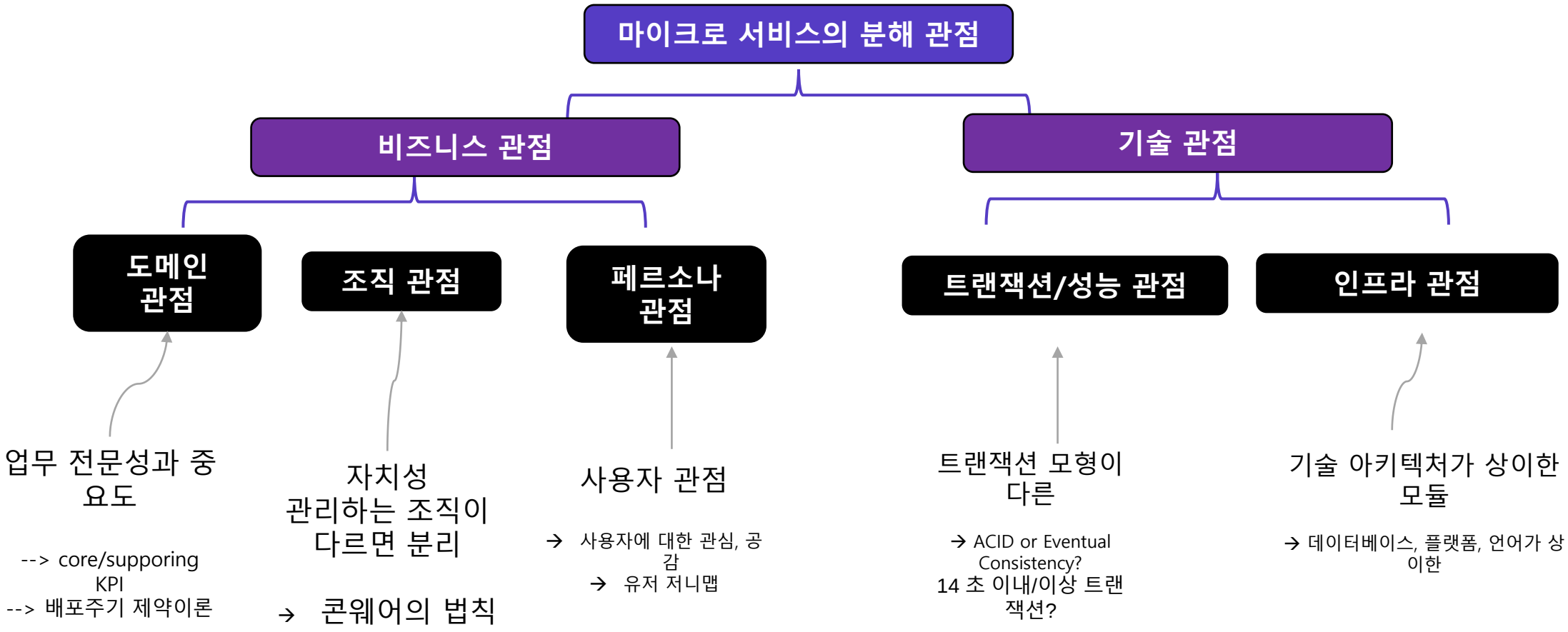
Request / Response 기반 연동 (RPC)



Pub / Sub 기반 통신 → Eventual Consistency

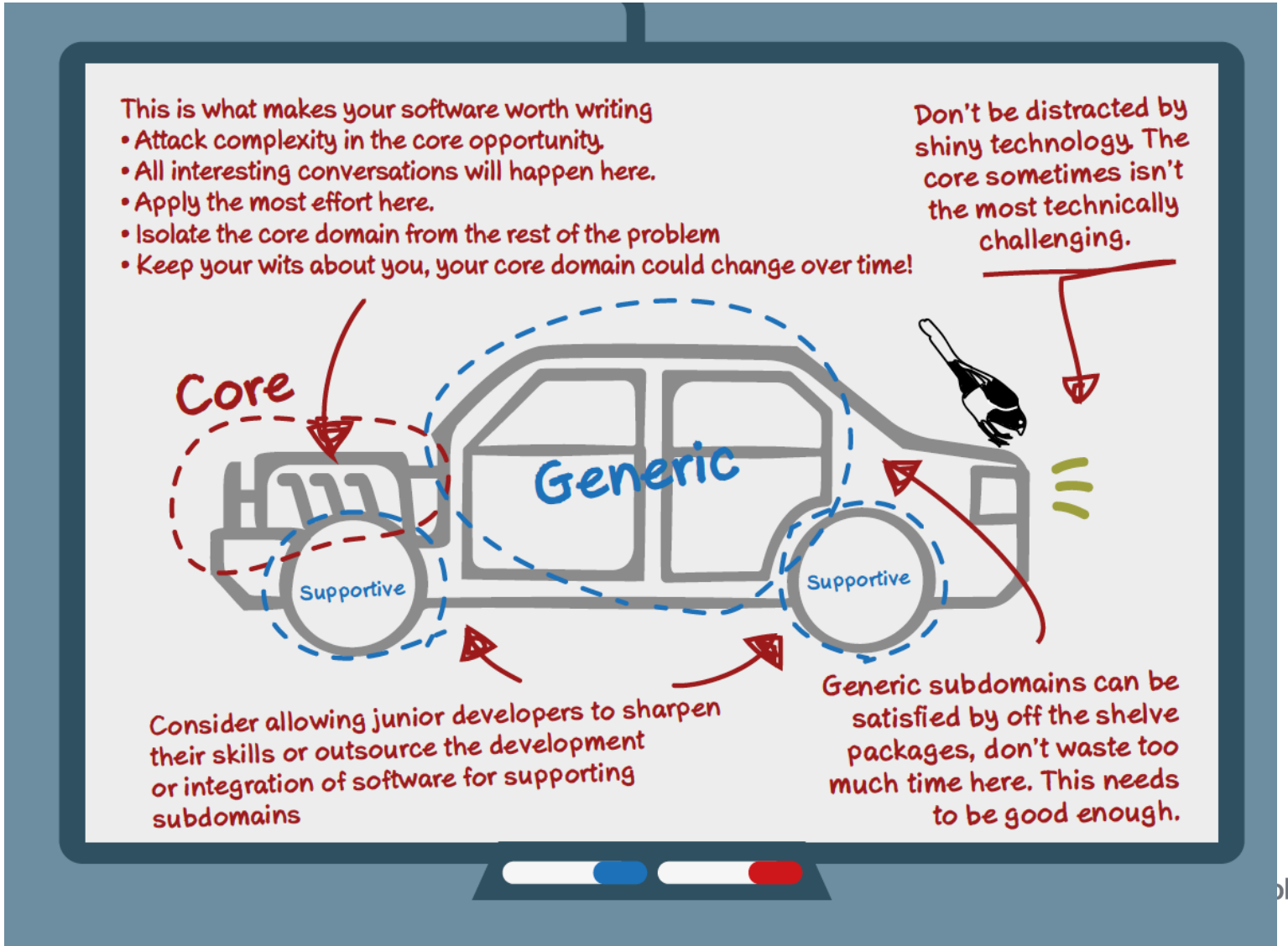


어떤 관점으로 BC 를 쪼갤 것인가?



Class of sub-domains

- Core
- Supportive
- Generic



Source: Anatomy of DDD, Scott Millette

마이크로 서비스간의 서열과 역학관계

" 무엇을 우선적으로 챙길 것인가? "

1순위 : Core Domain

- 버릴 수 없는. 이 기능이 제공되지 않으면 회사가 망하는

예) 쇼핑몰 시스템에서 주문, 카탈로그 서비스 등

2순위 : Supporting Domain

- 기업의 핵심 경쟁력이 아닌, 직접 운영해도 좋지만 상황에 따라 아웃소싱 가능한
- 시스템 리소스가 모자라면 외부서비스를 빌려쓰는 것을 고려할만한

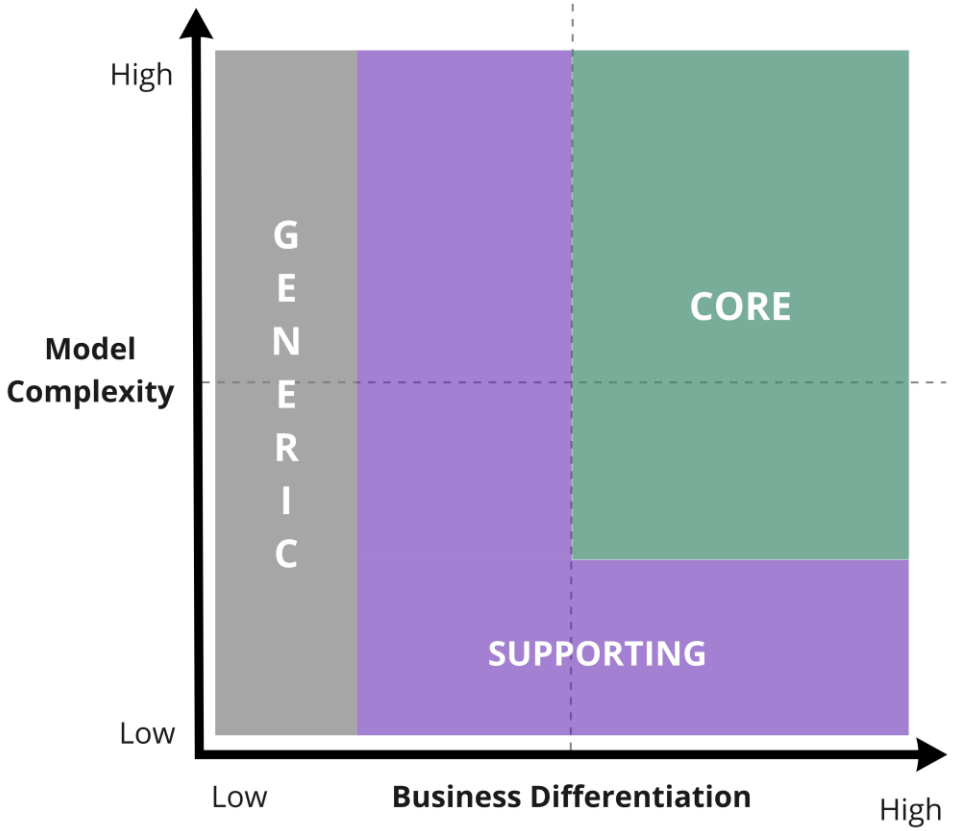
예) 재고관리, 배송, 회원관리 서비스 등

3순위 : General Domain

- SaaS 등을 사용하는게 더 저렴한, 기업 경쟁력과는 완전 무관한

예) 결재, 빌링 서비스 등

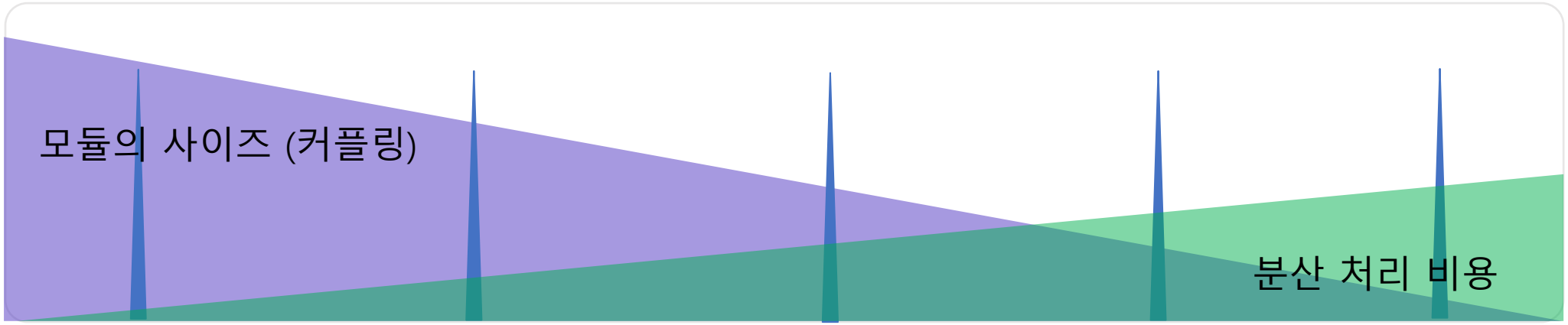
DDD의 경제적 투자 배분 관점



	경쟁적 우위	복잡도	변화성	구현
Core	Yes	High	High	**In-house**
Generic	No	High	Low	**구매/적용**
Supporting	No	Low	Low	**In-house/ 아웃소싱**

Source: https://github.com/ddd-crew/ddd-starter-modelling-process

분해수준 – 몇개까지 쪼개는게 좋은걸까?



Monolith

Note: 서비스 분해 개수는 조직 규모와 문제 영역의 크기마다 다르므로 설명을 위한 예시로만 보시기 바랍니다

Core/Supporting

2~3개

간섭이 가장 높은 부분을 경계로 한. 시급한.

구현이 용이한 단위
- 점진적 MSA 전환이 용이한 단위

Business Capability

3~10개

비즈니스적 목표에 따라 분리.

– 팀단위 KPI와 연결하기 좋은
- 자치적 동기부여.

Sub Domain

10~20개

커뮤니케이션이 원활해지는.

– Ubiquitous Language 유지

Aggregate

~100개

더이상 쪼갤 수 없는.

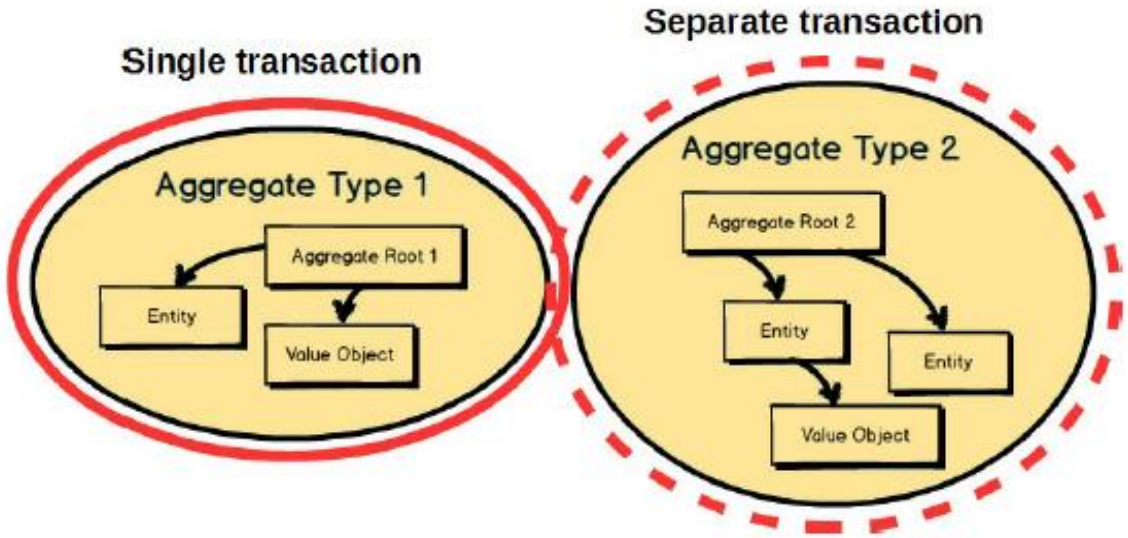
– ACID 가 준수되어야 하는 원자적 단위

MSA School

데이터 일관성 (트랜잭션) 문제!

솔루션: 어그리거트 (Aggregate) 패턴과 결국의 일치성 (Eventual Consistency)

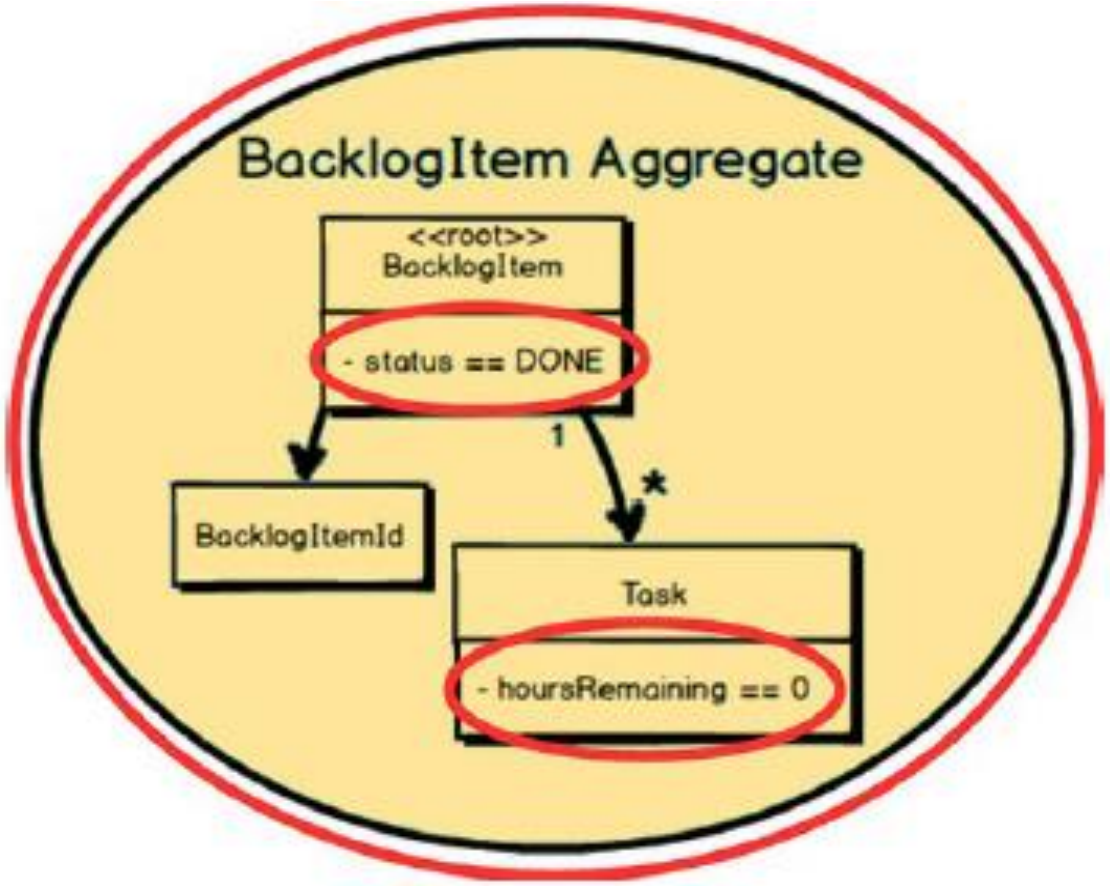
- 마이크로 서비스가 될 수 있는 가장 작은 원자적 단위의 엔티티 묶음 (더 이상 못 쪼개)
- = ACID 모델(All or Nothing) 일관성을 양보할 수 없는 최후의 사이즈
- = 어그리거트를 벗어난 데이터 일치성은 좀 천천히 맞춰줘도 된다는 비즈니스 적 허용
- = 어그리거트를 벗어난 단위의 연동은 비동기로 처리하여 성능을 높히겠다는 의도



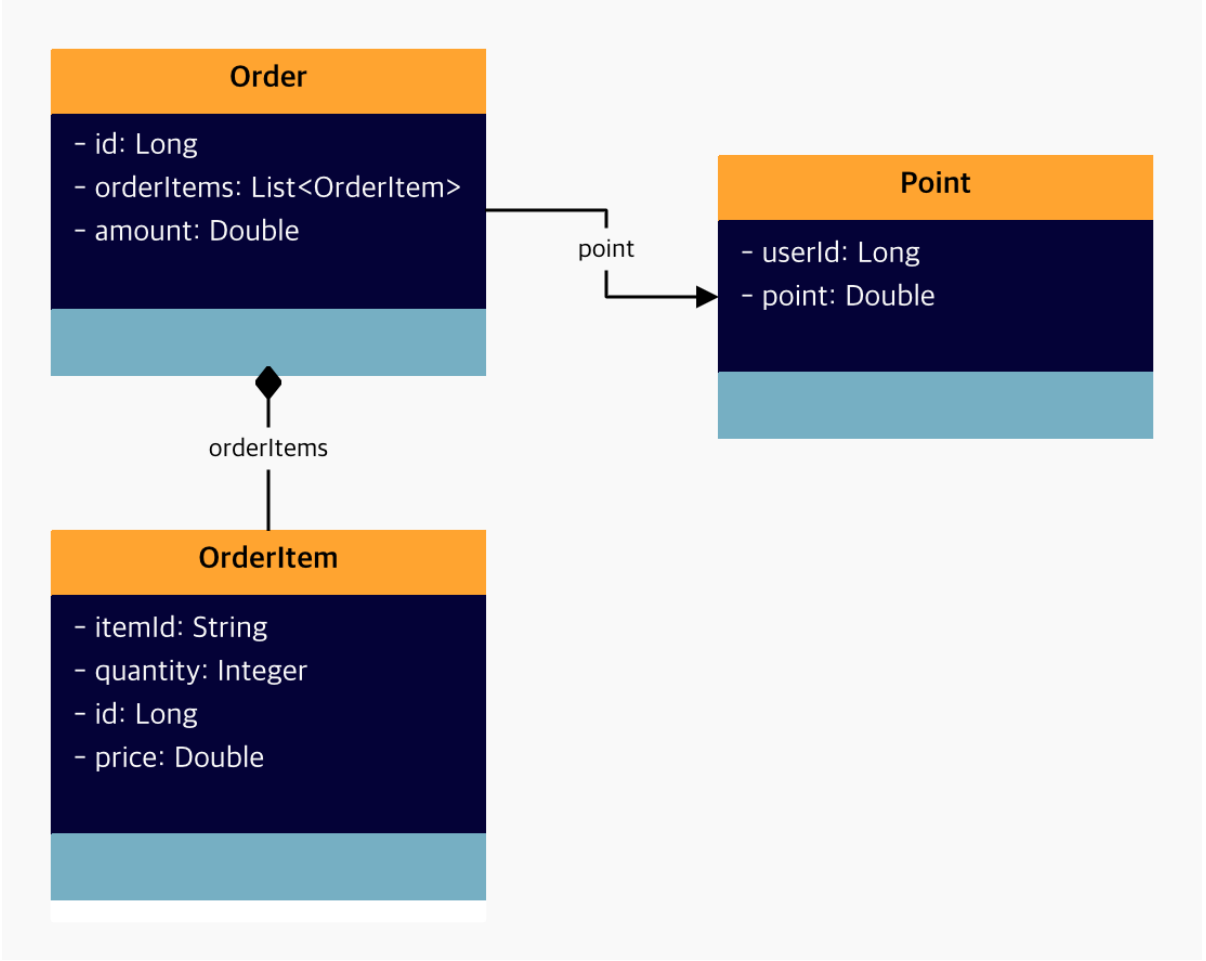
어그리거트 도출 규칙: 불변식

그 순간에 일치해야 하는 정보의 일관성

예) 백로그와 완료되었다면 남은 태스크는 없어야 한다.

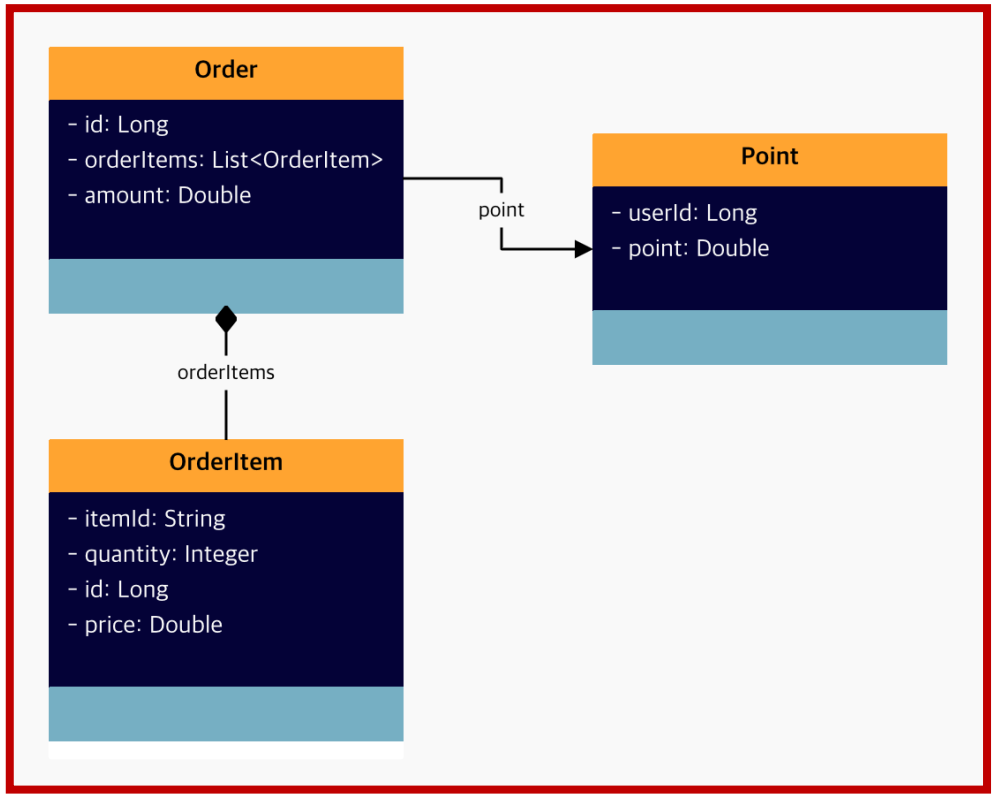


Aggregate 도출 예시 – 주문/포인트 시스템



Aggregate 도출 예시 – 주문/포인트 적립 시스템

유즈케이스 단위에 관여된 모든 테이블을 Aggregate 로 묶는 경우

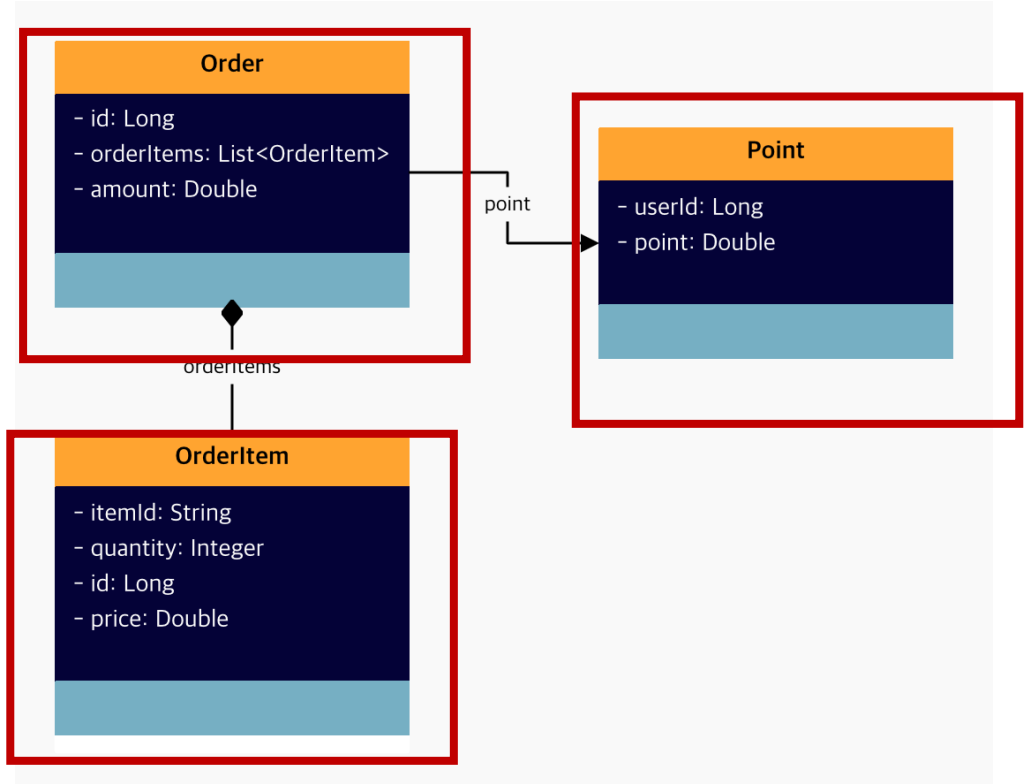


포인트 적립 시스템의 로직에 오류가 있을 경우 주문 자체가 이루어지지 못함

고객입장에서 생각해보라, 배가 고픈데, 포인트 적립에 문제가 생겨 주문이 롤백된다. 얼마나 슬픈 일인가!

Aggregate 도출 예시 – 주문/포인트 적립 시스템

모든 테이블을 하나의 어그리거트로 정의한 경우

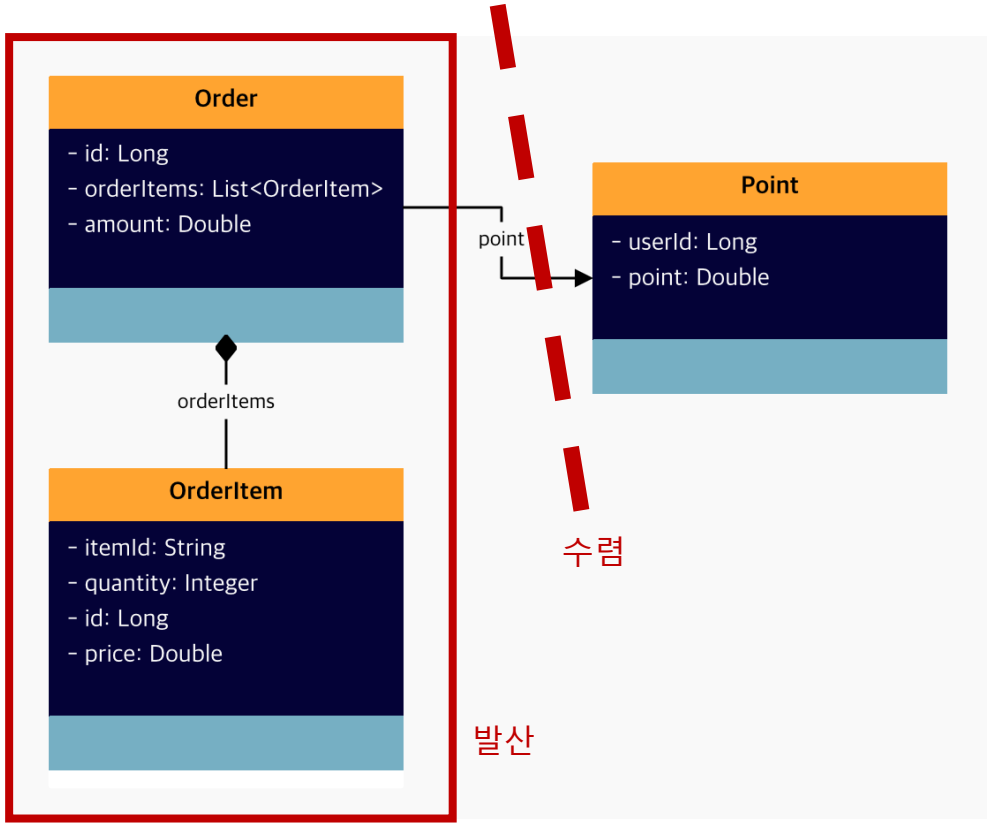


전체 주문금액과 주문 내역이 일치하지 않음
주문 내역 일부가 누락 혹은 중복 입력가능
(1 Submit = 1 Aggregate Operation)

빵과 우유를 먹고 싶었는데, 우유가 누락됐다면 목이 메인다. 얼마나 슬픈 일인가!

Aggregate 도출 예시 – 주문/포인트 적립 시스템

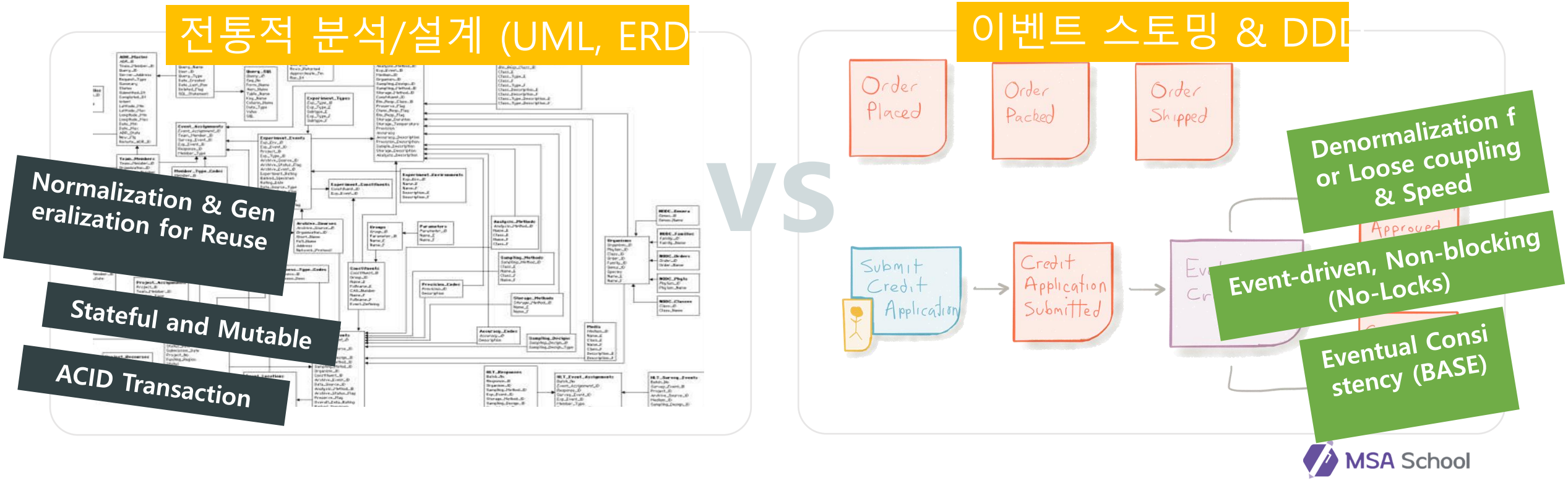
불변식 (주문총량과 주문내역 합산은 같아야 한다) 의 범위만을 어그리거트로 잡은 경우



ACID TX가 필요한 최소한의 Entity (테이블) 묶음 – Point 는 천천히 처리 되어도 괜찮아 (Eventual Consistency)

Event Storming : DDD 설계를 쉽게 하는 방법

- **이벤트스토밍은** 시스템에서 발생하는 **이벤트를 중심 (Event-First) 으로** 분석하는 기법으로 특히 Non-blocking, Event-driven 한 **MSA 기반 시스템을 분석에서 개발까지 필요한 도메인에 대한 탁월하게 빠른 이해를 도모**하는데 유리하다.
- 기존의 유즈케이스나 클래스 다이어그래밍 방식과 다르게 별다른 **사전 훈련된 지식과 도구 없이 진행할 수 있다**.
- 진행과정은 참여자 워크숍 방식의 방법론으로 결과는 **스티키 노트를 벽에 붙힌 것으로 결과가 남으며, 오랜지색 스티키 노트들의 연결로 비즈니스 프로세스가 도출**되며 이들을 이후 BPMN과 UML 등으로 정재하여 전환할 수 있다.



MSA School

Levels of Event Storming

BIG PICTURE	EVENTS	HOT SPOTS, SYSTEMS, PEOPLE	CONFLICTS, GOALS, BLOCKERS, BOUNDARIES
PROCESS MODELLING	EVENTS	+ POLICIES, COMMANDS, READ MODELS	VALUE PROPOSITION, POLICIES, PERSONAS, INDIVIDUAL GOALS
SOFTWARE DESIGN	EVENTS	+ AGGREGATES	AGGREGATES, POLICIES, READ MODELS, IDS

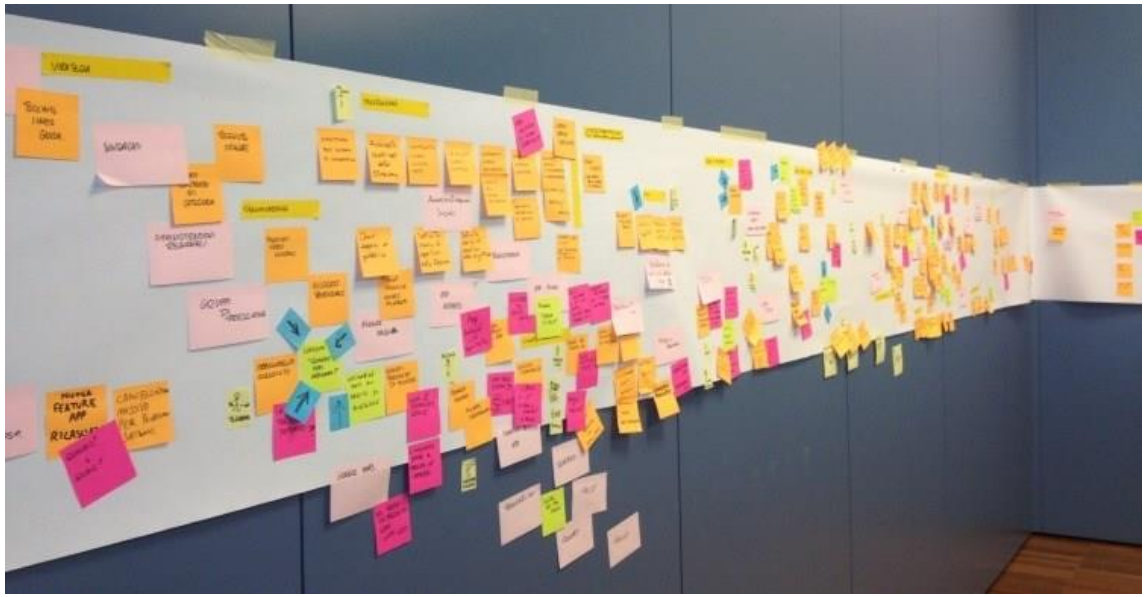
Event Storming : Prepare

People

- 모든 팀 및 프로세스, UI, DB, 아키텍처를 책임질 팀당 2명 이상 구성
- 이벤트스토밍 전문가 퍼실리테이터가 초기에는 필요할 수 있음

Tools

- 큰 종이 시트 및 종이 시트를 여러 장 붙일 수 있는 충분한 벽이 있는 넓은 공간
- 여러 종류의 색깔 스티커, 검은색 펜, 검은색 or 파란색 테이프
- 서서 하는 방식으로 의자 필요 없음.



Type of stickers

Domain Event (Orange)

발생한 사실, 결과, Output

도메인 전문가가 정의
이벤트 퍼블리싱

사용자, 페르소나, 스테이크 홀더

유저 인터페이스를 통해 데이터를 소비하고 명령을 실행하여 시스템과 상호 작용

Definition

정의

도메인에 대한 용어 등의 설명,기술

Command (Sky Blue)

의사결정, Input, API, UI 버튼

현재형으로 작성,
행동, 결정 등의 값들에 대한
UI 혹은 API

Aggregate (Yellow)

구현체 덩어리, 시스템, 모듈

비즈니스 로직 처리의 도메인 객체 덩어리. 서로 연결된 하나 이상의 엔터티 및 value objects의 집합체

Read Model (Green)

의사결정에 필요한 자료, View, UI

행위와 결정을 하기 위하여
유저가 참고하는 데이터, 데이터 프로젝션이 필요 : CQRS 등으로 수집

External System (Pink)

외부 시스템

시스템 호출을 암시 (REST)

Policy (Lilac)

정책, 반응(Reaction)

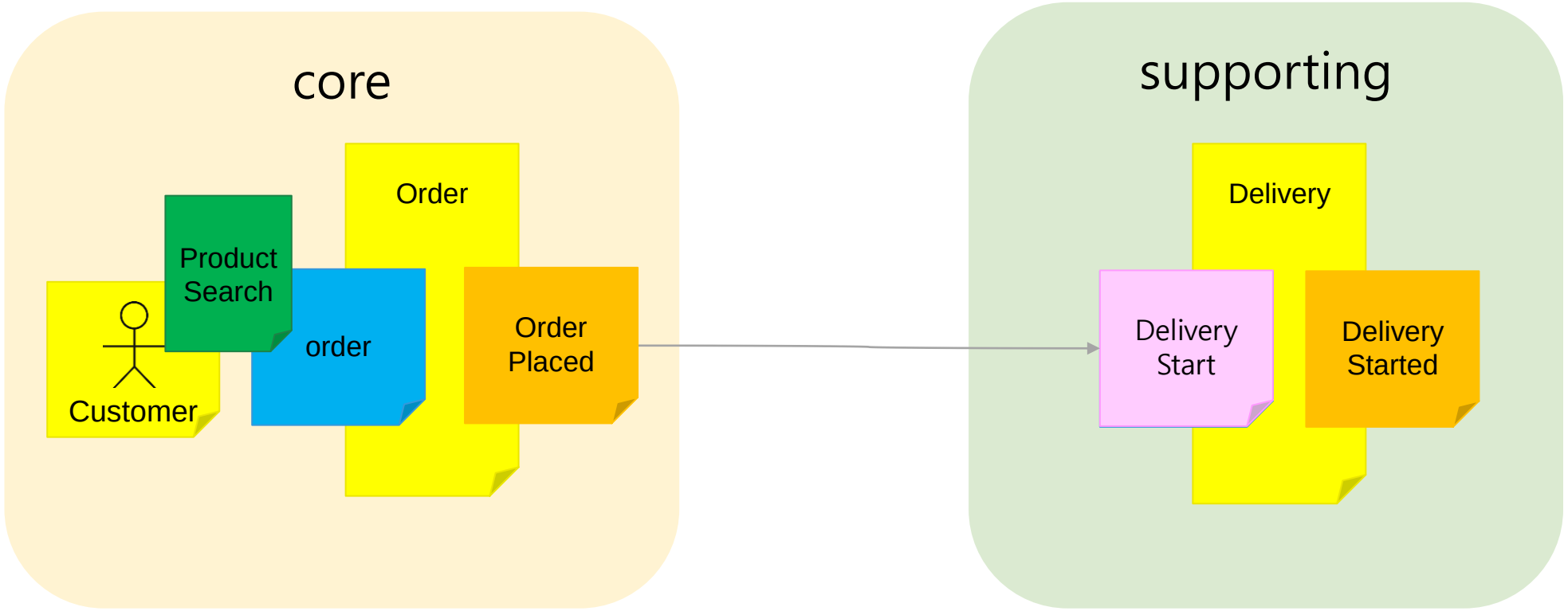
이벤트에 대한 반응
(서브스크라이브)
비즈니스 룰 엔진 등

Comment Or Question (Purple)

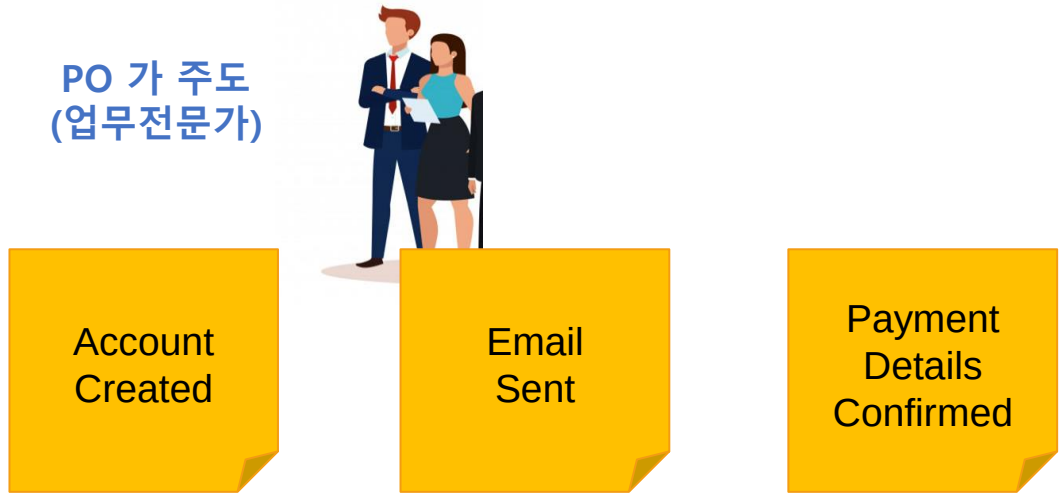
의견 또는 질문

추기적인 네용 입력,
예측되는 Risk

Examples


core
Order
Product Search
order
Order Placed
Customer
supporting
Delivery
Delivery Start
Delivery Started

Firstly, Event Discovery

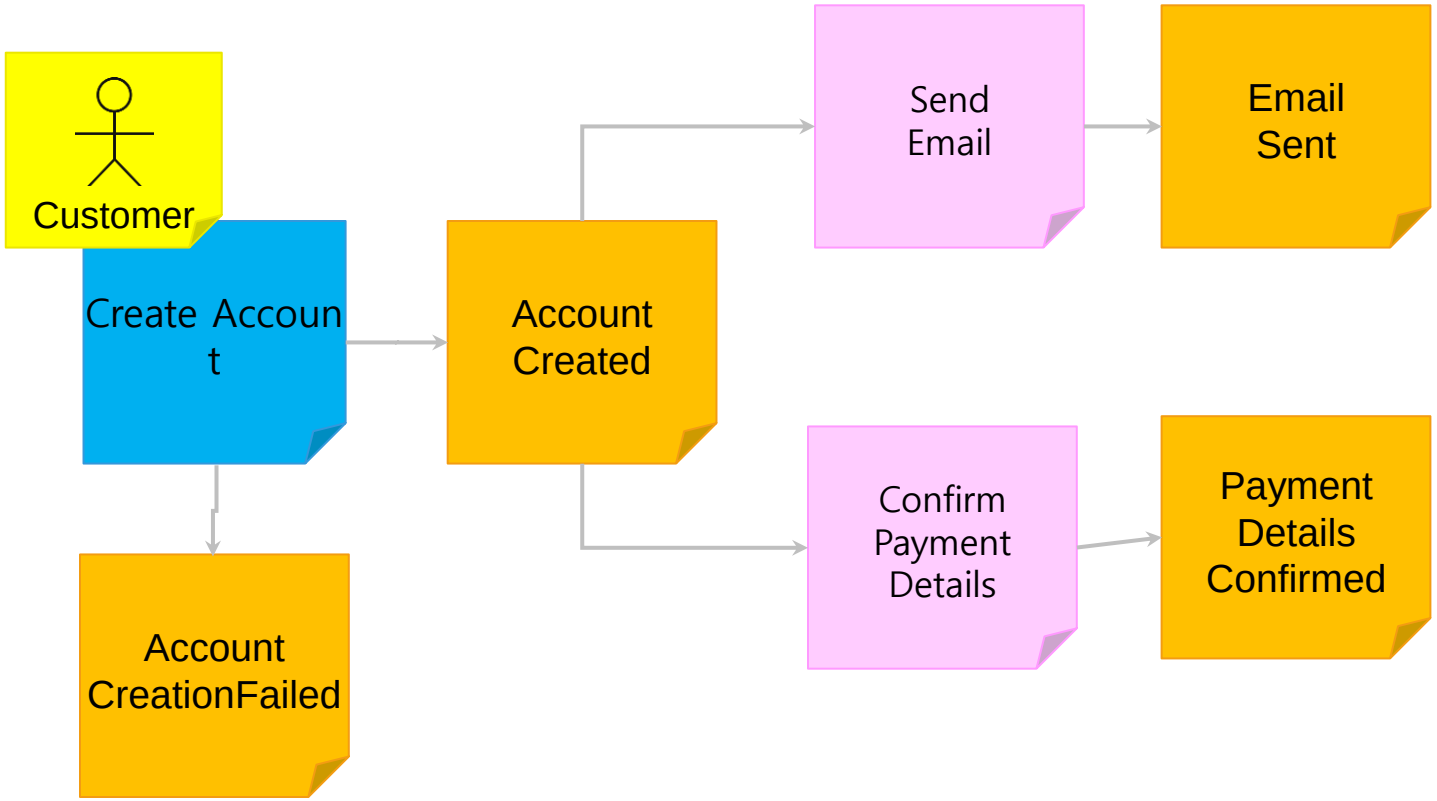


- 오렌지(주황색) 스티커 사용
- 각 도메인 전문가들이 개별 도메인 이벤트 목록 작성
- 이벤트는 도메인 전문가와 비즈니스 관계자 서로 이해할 수 있는 의미 있는 방식(유비쿼터스 랭귀지)으로 표현하고 동사의 과거형 (p.p.)으로 표현한다.
- 시작 및 종료 이벤트를 식별하고 스티커를 붙일 벽의 시작과 끝의 타임라인에 배치
- 이벤트를 페르소나와 관련시키는 방법에 대해 논의
- 중복된 이벤트를 발견하면 중복된 이벤트를 벽에서 제거
- 불분명한 경우 다른 색상의 스티커 메모를 사용하여 질문이나 의견을 추가(빨간색 스티커)
- 이벤트에 대해서 동사를 과거 시제로 입력하고 다른 이벤트와 명확하게 구분되는 용어를 사용

Command, Policy, Actor 도출

PO + UI/UX 가 주도
(업무전문가)

- 하늘색 or 라일락 색의 스티커 사용
- Command 는 사용자의 의사결정에 의하여 (UI) 결과이벤트에 이르게 하는 Input
- Command 는 어떠한 상태의 변화를 일으키는 서비스*
- Policy는 이벤트가 발생한 후 발생하는 반응형 논리 (Input)
- Policy는 결과로서 Event 를 트리거 할 수 있고, 다른 Command를 호출 할 수도 있음
- Policy는 시스템에 의하여 자동화 될 수 있다.



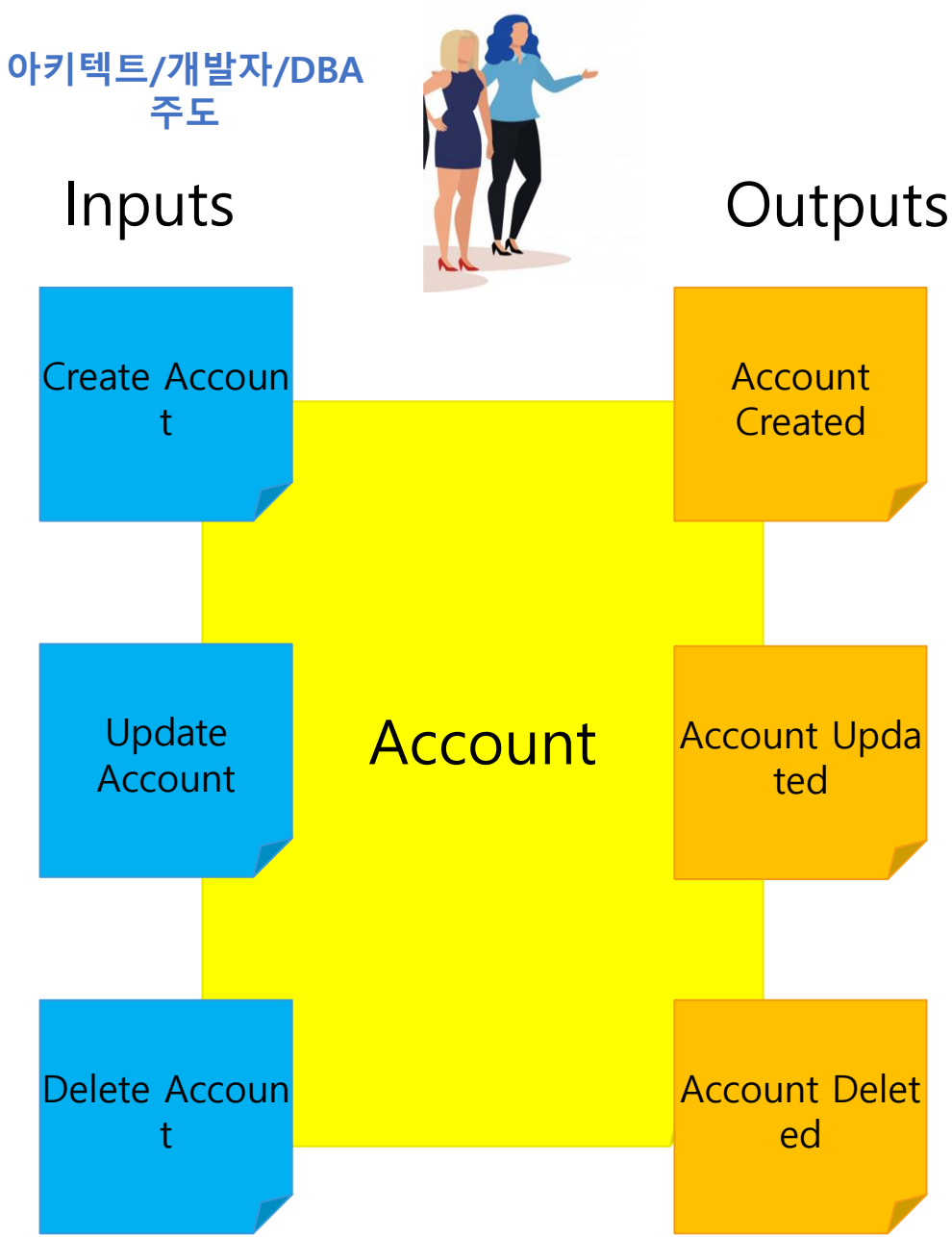
*"Whenever a **new user account is created** we will **send her an acknowledgement by email.**"*

* Command 라는 용어는 CQRS 에서 유래했으며, 쓰기서비스인 Command 와 읽기행위인 Query는 구분됨.

Aggregate 도출

- 노란색 스티커 사용
- 같은 Entity를 사용하는 연관 있는 도메인 이벤트들의 집합
- 관련 데이터 (Entity 및 value objects)뿐만 아니라 해당 Aggregates의 Life Cycle에 의해 연결된 작업(Command)으로 구성

도메인 이벤트가 발생하는 데는, 어떠한 도메인 객체의 변화가 발생했기 때문이다.
하나의 ACID 한 트랜잭션에 묶여 변화되어야 할 객체의 묶음을 도출하고, 그것들을 커맨드, 이벤트와 함께 묶는다.



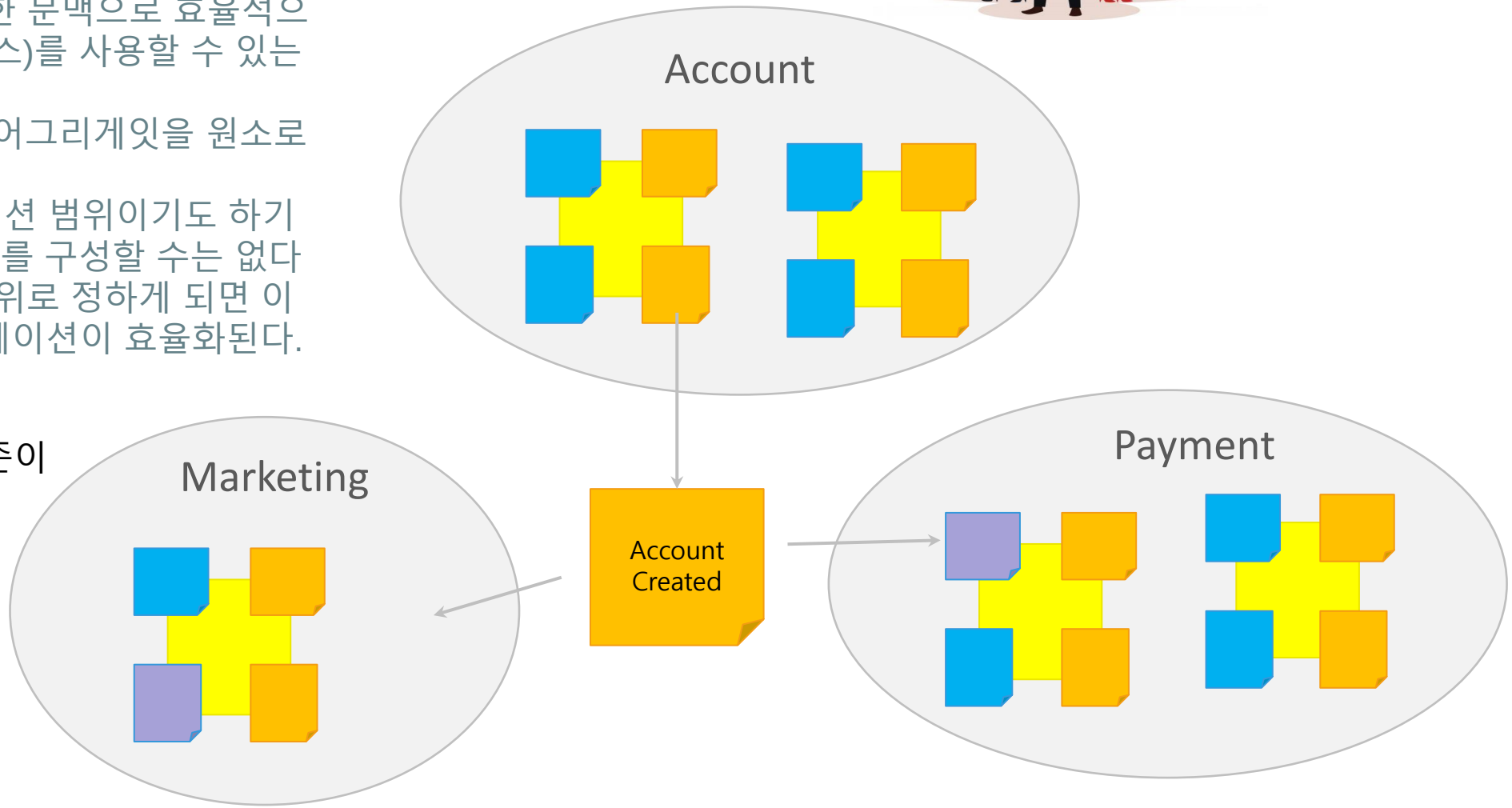
Bounded Contexts 도출

- Bounded Context 는 동일한 문맥으로 효율적으로 업무 용어 (도메인 클래스)를 사용할 수 있는 범위를 뜻한다.
- 하나의 BC 는 하나이상의 어그리게잇을 원소로 구성될 수 있다.
- 어그리게잇은 ACID 트랜잭션 범위이기도 하기 때문에 이를 더 쪼개서 BC 를 구성할 수는 없다
- BC를 Microservice 구성단위로 정하게 되면 이를 담당한 팀 내의 커뮤니케이션이 효율화된다.

찾는 방식은 여러가지 기준이 있다

- Domain Boundary
- Transaction Boundary
- Technical Stack

...

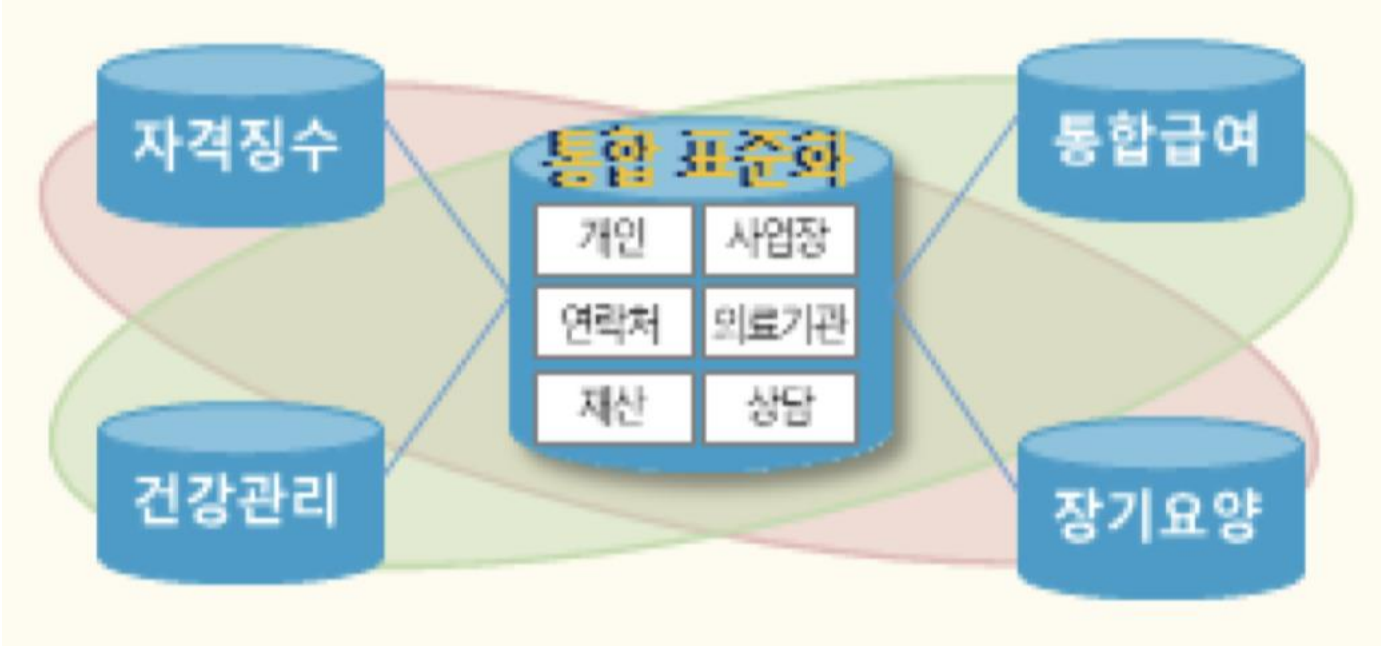


Source: https://blog.redelastic.com/corporate-arts-crafts-modelling-reactive-systems-with-event-storming-73c6236f5dd7, Kevin webber

건강보험공단

- (현황 및 문제점) 그간 건강보험, 장기요양보험 등 통합된 프로세스 없이 제도도입에만 몰두하여, 단위업무 중심으로 시스템을 구축·운영함에 따라 **관리의 복잡성 증가 및 현행 시스템 운영의 한계** 발생

◈ 주요 문제점
- ► 비효율 프로세스 증가, 연계 미흡, 관리 복잡, 데이터 중복·비표준화
- ► 사업별 인프라 구축으로 시스템 복잡도 증가 및 유지보수 한계 발생



Single, Canonical Model

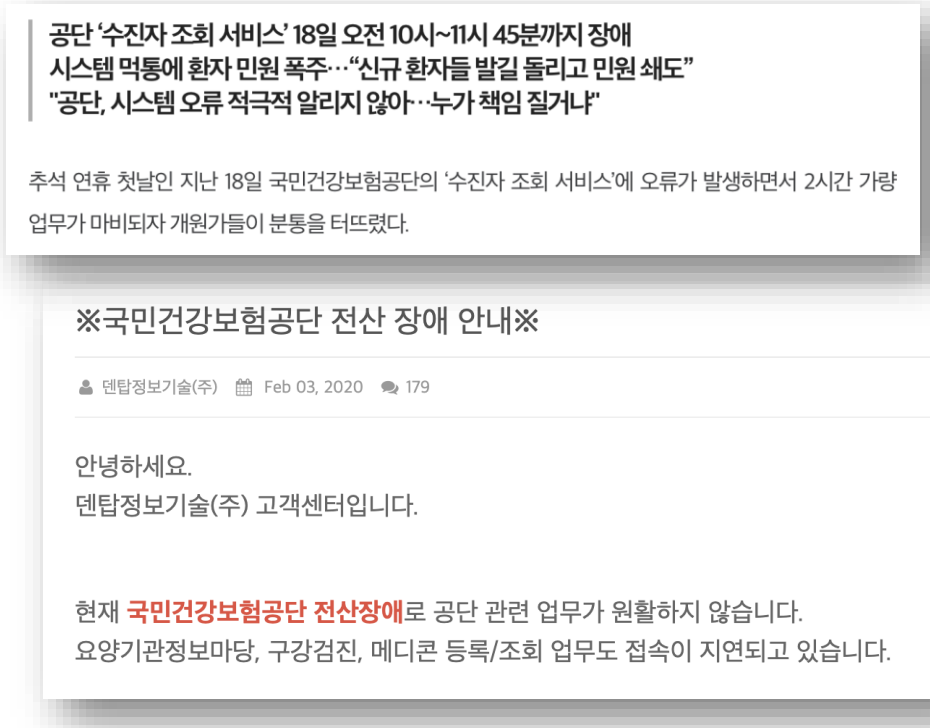
에 대한 노력

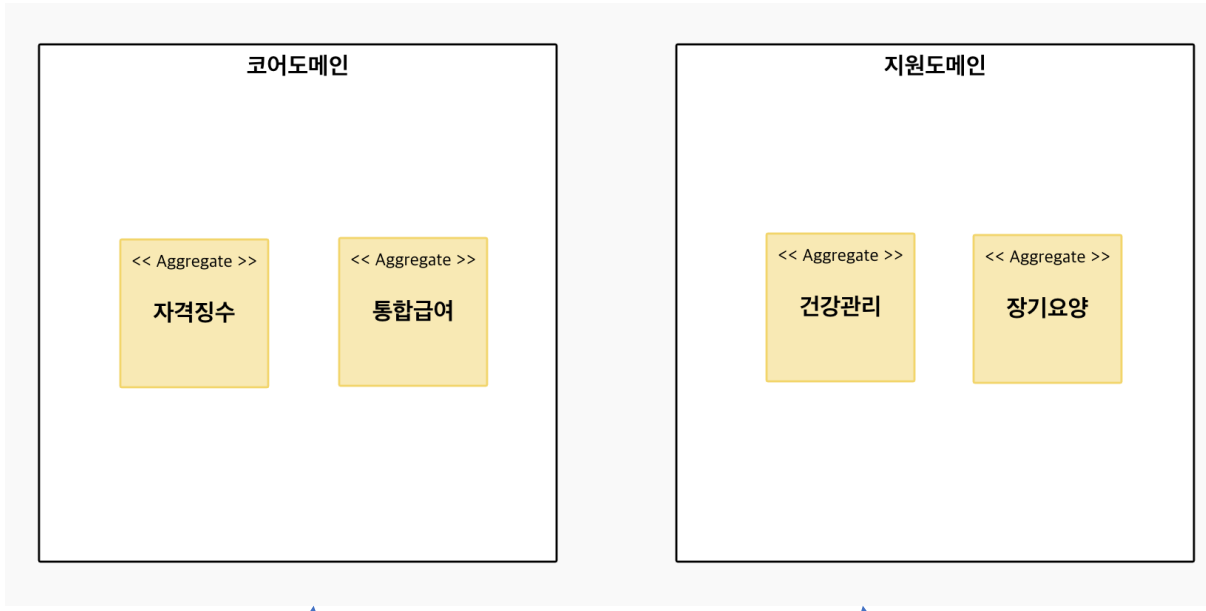
*** 변화의 속도 저하**

분해 전략: 도메인 중요도로 분리 → 2~3개 미니서비스

코어도메인

Note: 본 예시는 DDD 적용방법을 설명하기 위한 예시일 뿐이며, 상세한 분해 전략은 기관의 전략이나 상황에 따라 상세한 분석 후 적용되어야 합니다



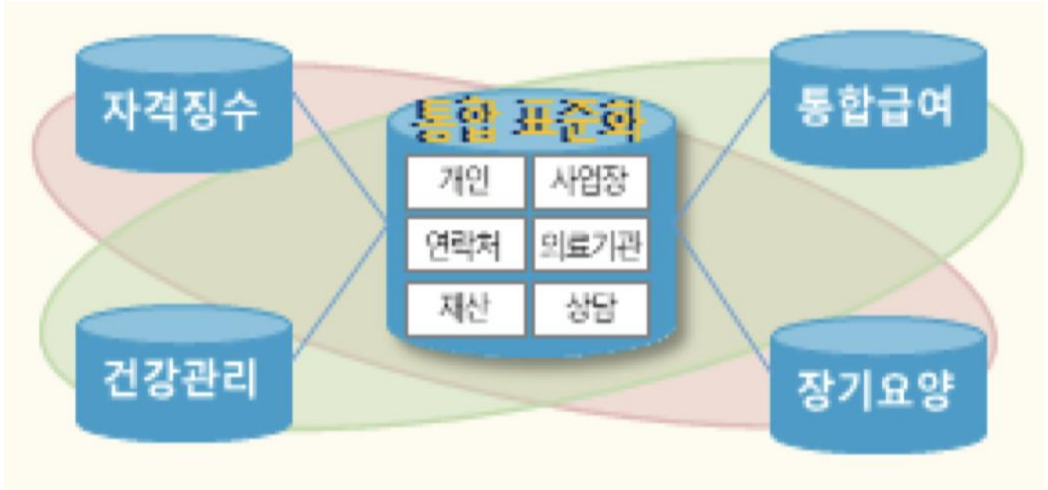


- 핵심적이고 장애에 크리티컬한 영역
- 대고객 접점의 업무 영역
- 잦은 변경이나 배포가 필요한 영역

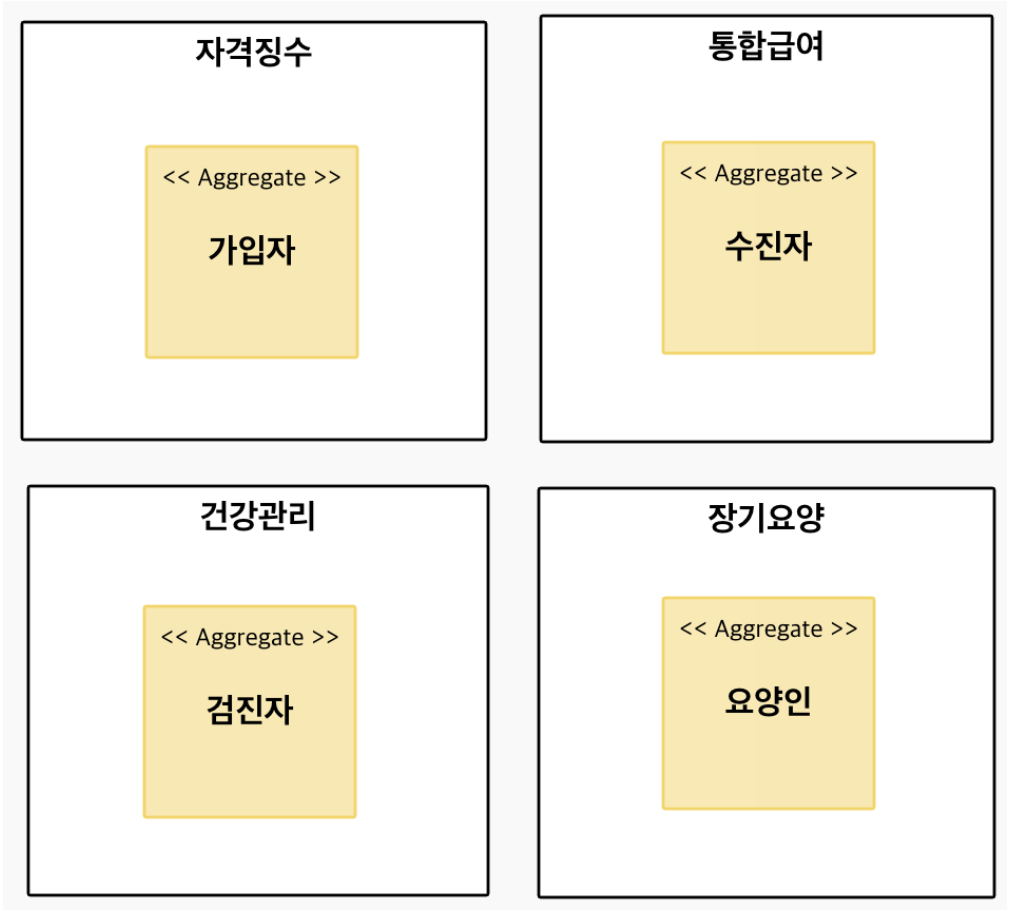
- 일부 사용자에 한정된
- 새롭게 추진되는 시범적인 영역
- 배포주기가 잦지 않은 영역

MSA School

분해 전략: 서브 도메인으로 분리



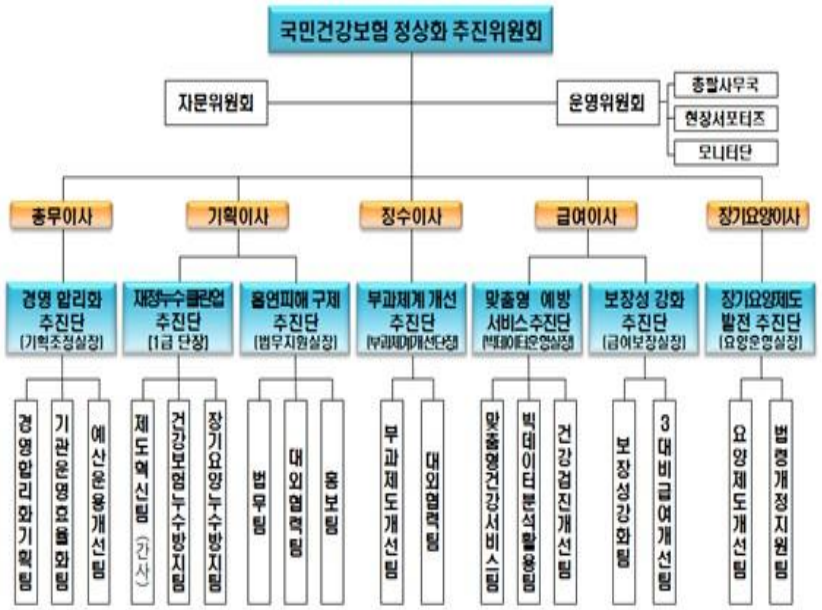
Canonical Model

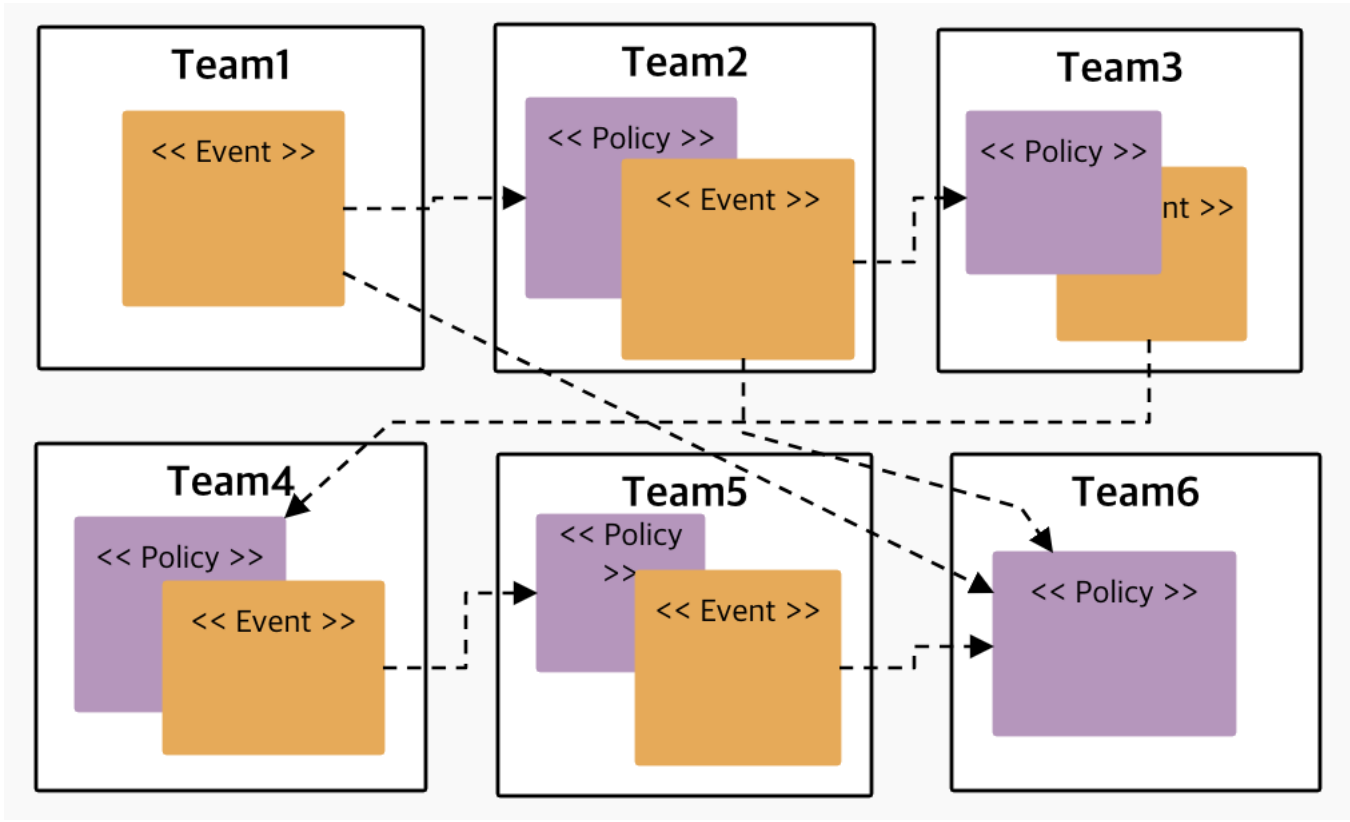


- 업무 영역의 전문성으로 분리
- 데이터 구조를 일반화/표준화 하지 않음
  (해당 영역의 도메인 모델을 존중)

MSA School

분해 전략: 조직구조로 분해

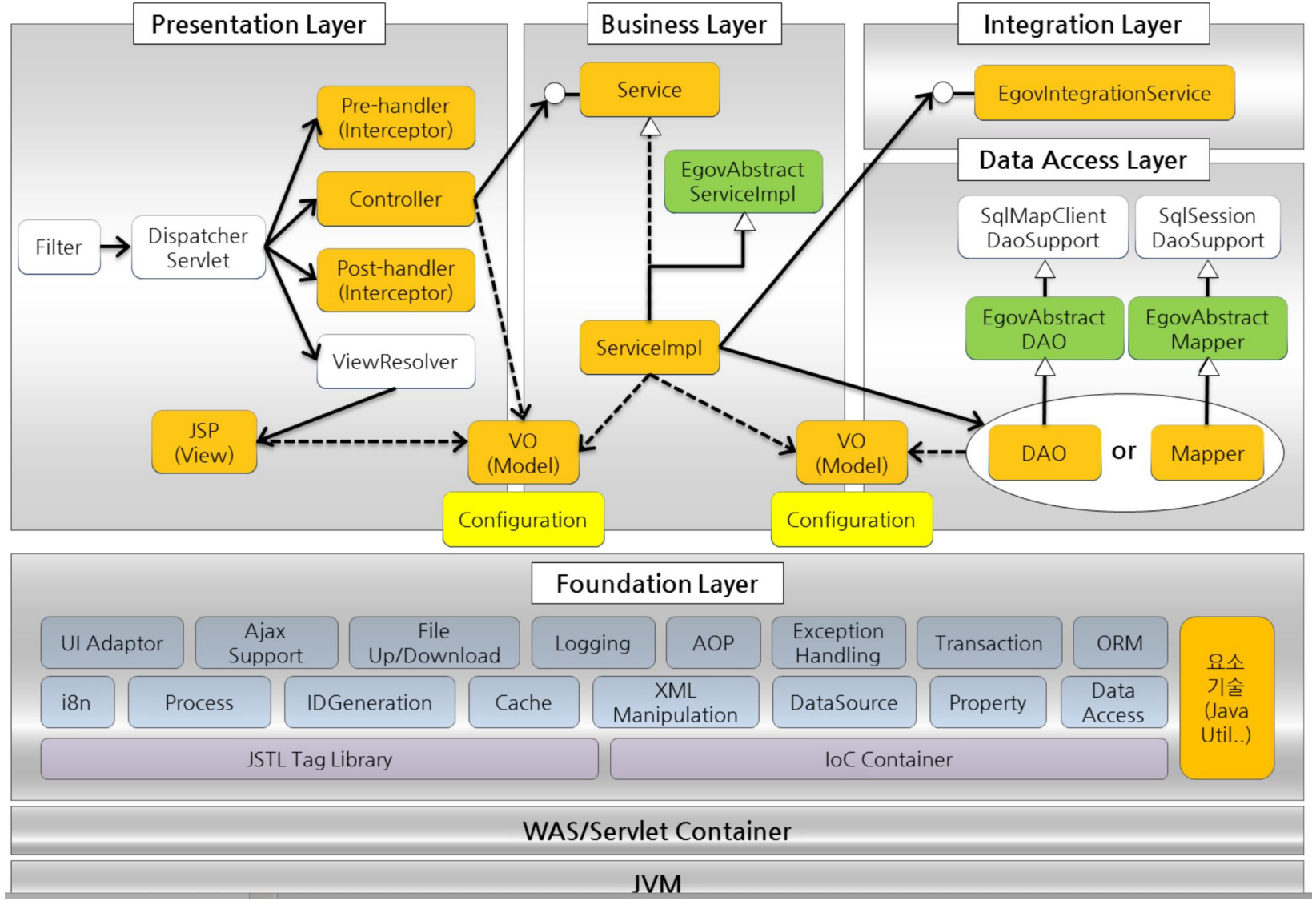




팀 = 도메인 전문성 = 서브 도메인 = 마이크로 서비스

- 팀의 자치성과 동기부여 차원
- 팀 KPI가 다른 경우
- 타팀의 관심있는 이벤트에 주목하여 서로 Pub/Sub으로 연동

CNA 구현 측면의 표준 프레임워크

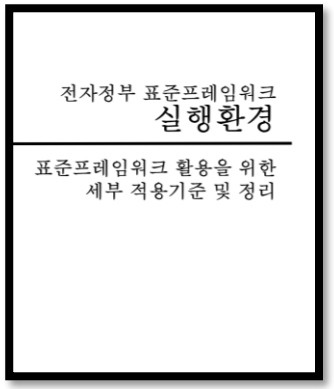


CNA 적용 활용 영역:

- 다양한 제네릭 도메인 서비스들의 활용 (로그인, 게시판, 자료실, 일정관리, 보안 등)
- Hexagonal / Clean Architecture 적용하기 좋은 구조
- RESTful API 지원 (Spring Data Rest 허용)
- 기존 JSP/JSTL → 경량 UI Frameworks (React/Vue)

MSA School

표준 프레임워크 적용 준수사항 요약



레이어드 아키텍처

- Clean Arch. 지향 (Active Record 이상)
- 레이어 구분을 위한 역할 마킹(Annotation 기반)

재사용

- 서비스 상위 클래스 적용 EgovAbstractServiceImpl
- iBatis 사용시 준수사항 (Repository 표시 및 기타)

기타

- 프레임워크 내부 코드의 임의 변경 금지
- 플랫폼 패키지명의 임의 사용 금지
- 경량 UI(MVVM,RIA) 인 경우 RESTful 권고

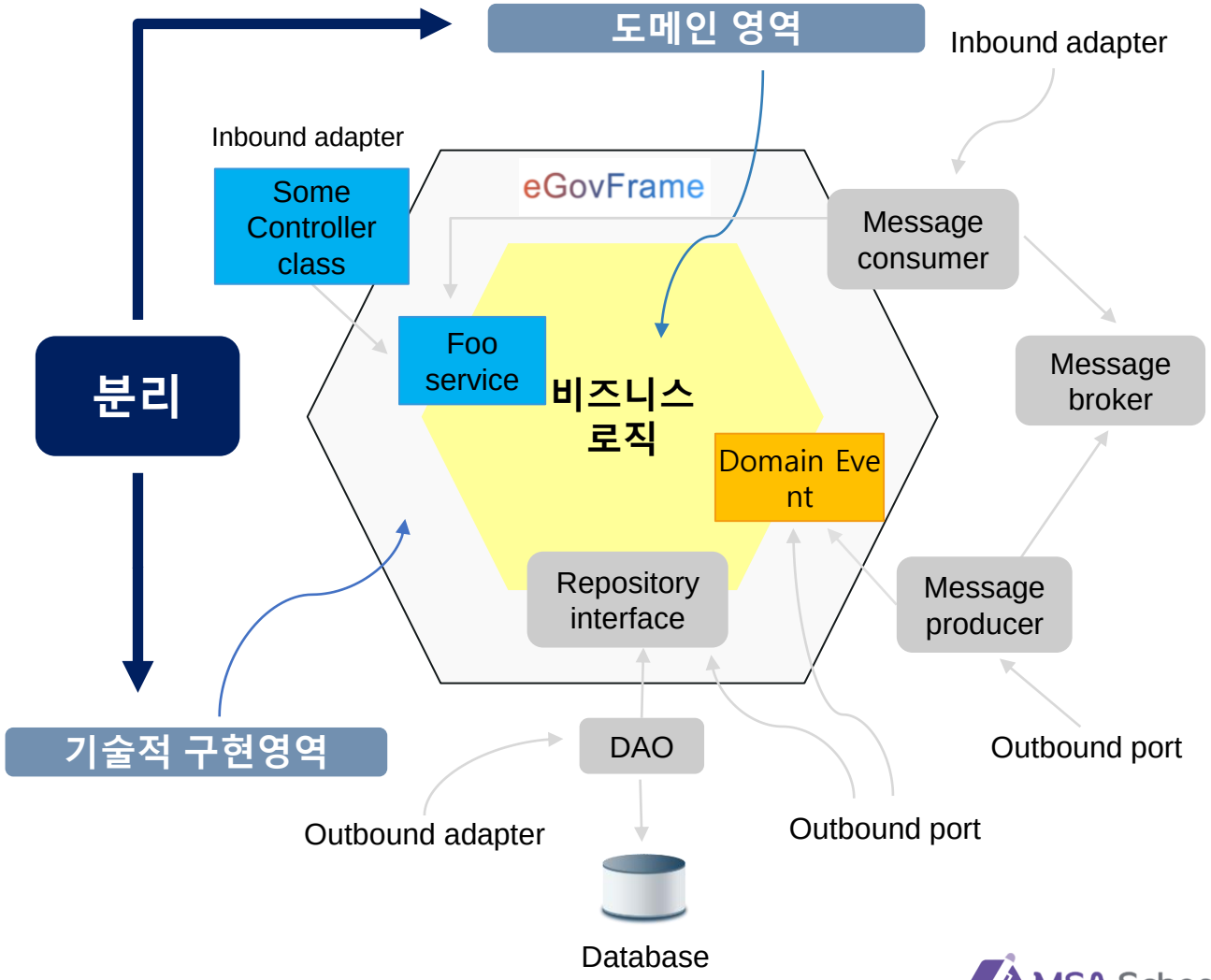
Microservice Implementation Pattern (1)

- **Hexagonal Architecture**

Create your service to be independent of either UI or database and to provide adapters for different input/output sources such as GUI, DB, test harness, RESTful resource, etc.

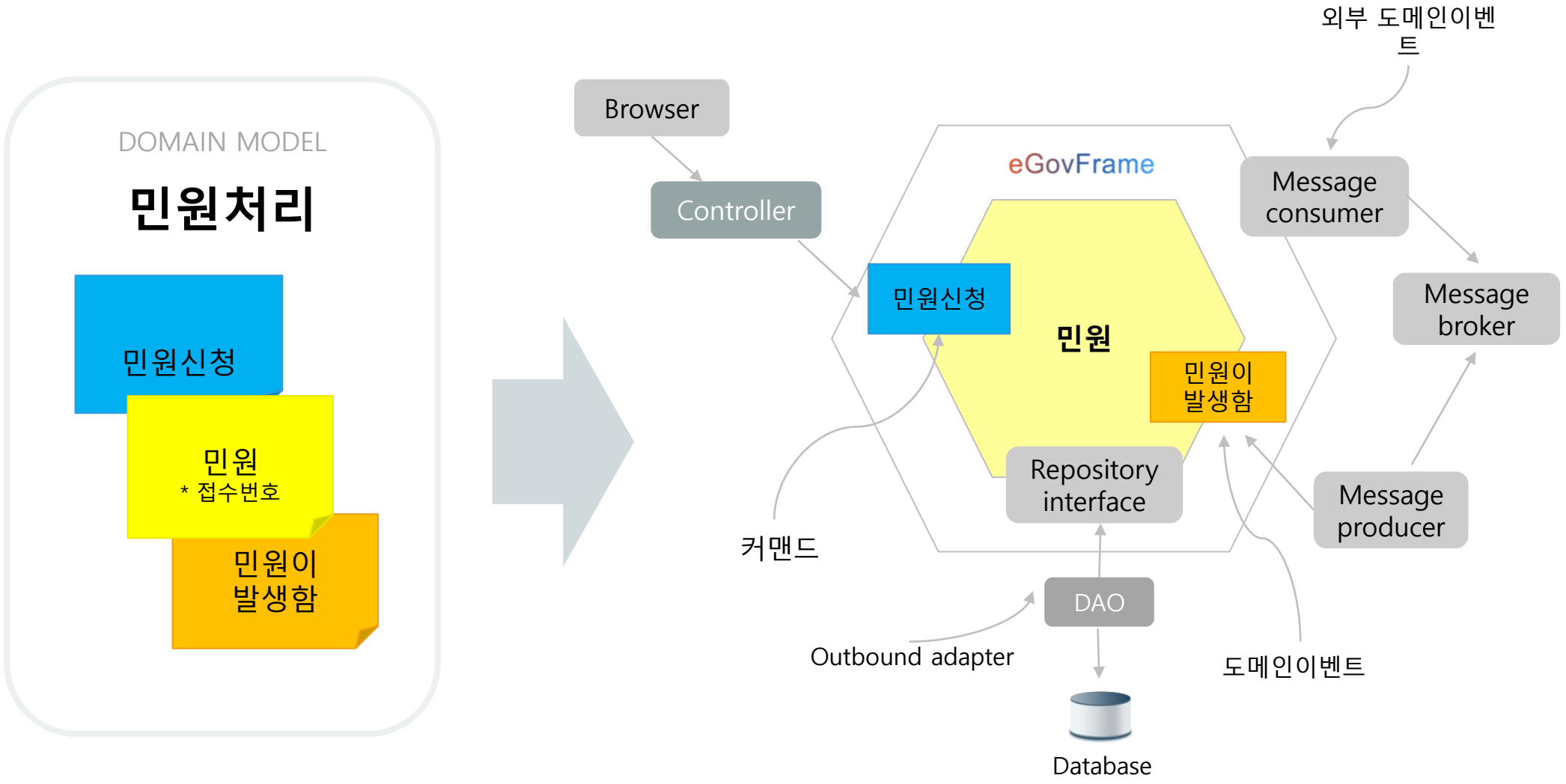
Implement the publish-and-subscribe messaging pattern: As events arrive at a port, an adapter (a.k.a. service agent) converts it into a procedure call or message and passes it to the application. When the application has something to publish, it does it through a port to an adapter, which creates the appropriate signals needed by the receiver.

https://alistair.cockburn.us/Hexagonal+architecture



MSA School

Applying Hexagonal Architecture



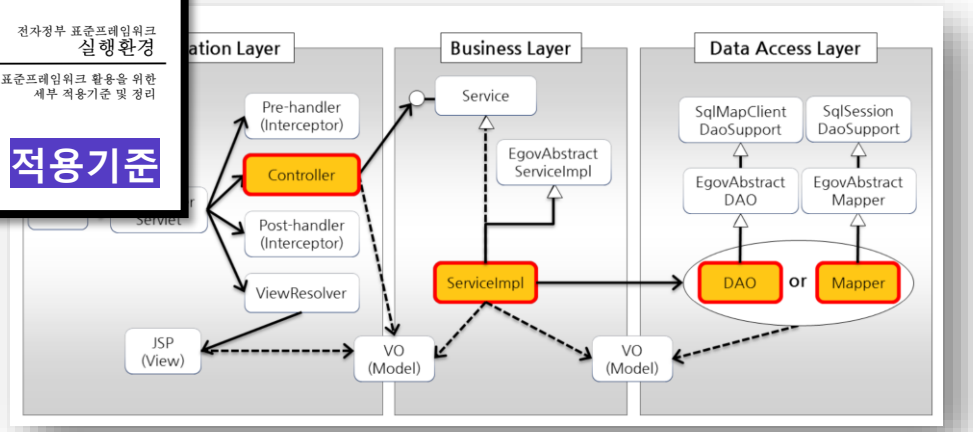
MSA School

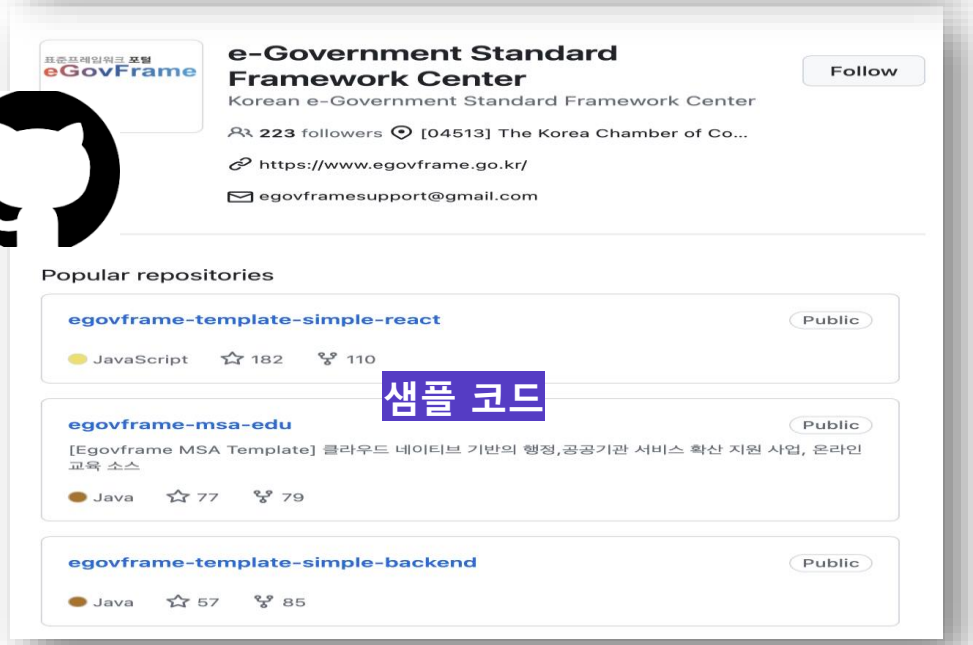
e-Gov Framework MSA Easy Template

전자정부 표준프레임워크
실행환경

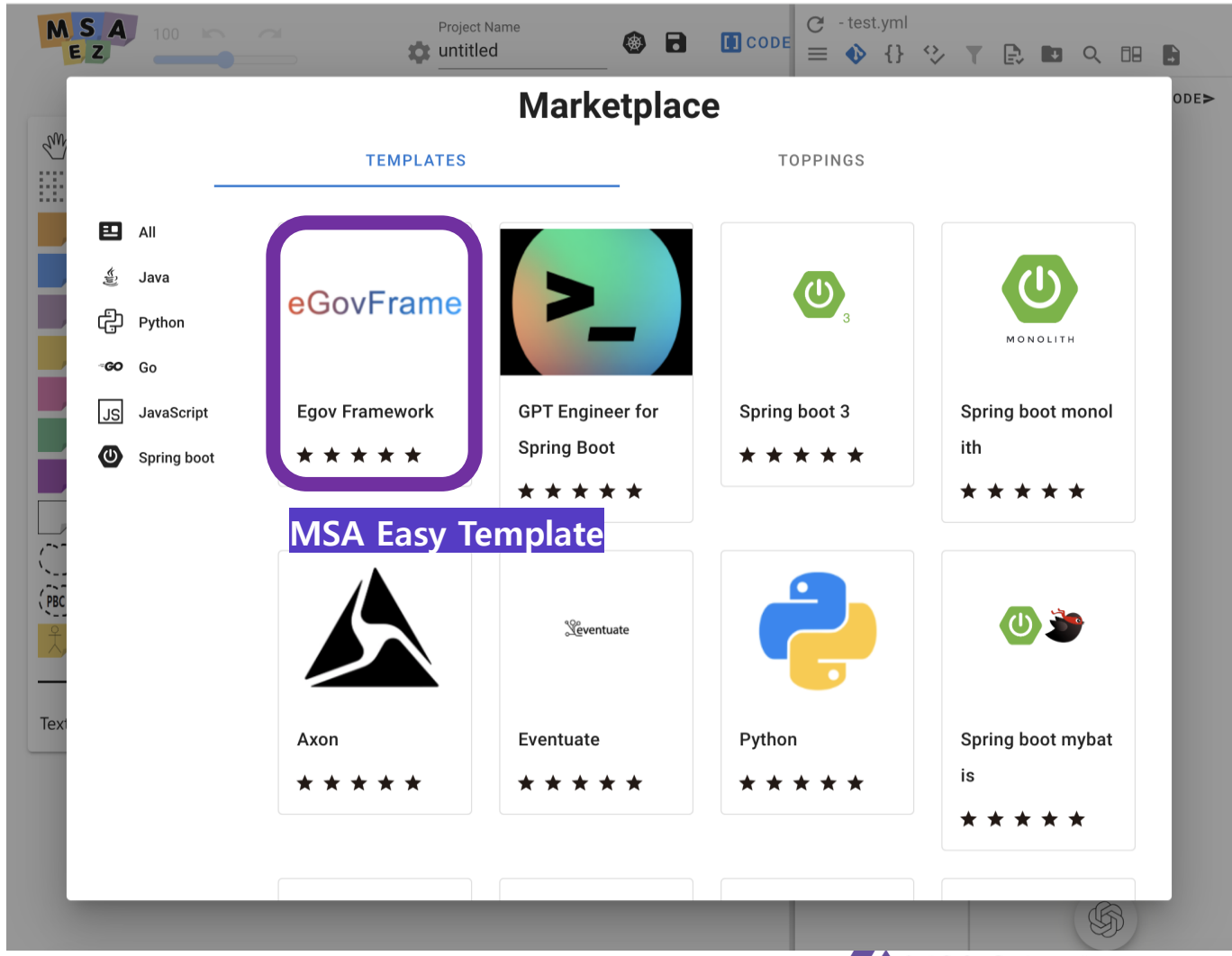
표준프레임워크 활용을 위한
세부 적용기준 및 정리

적용기준





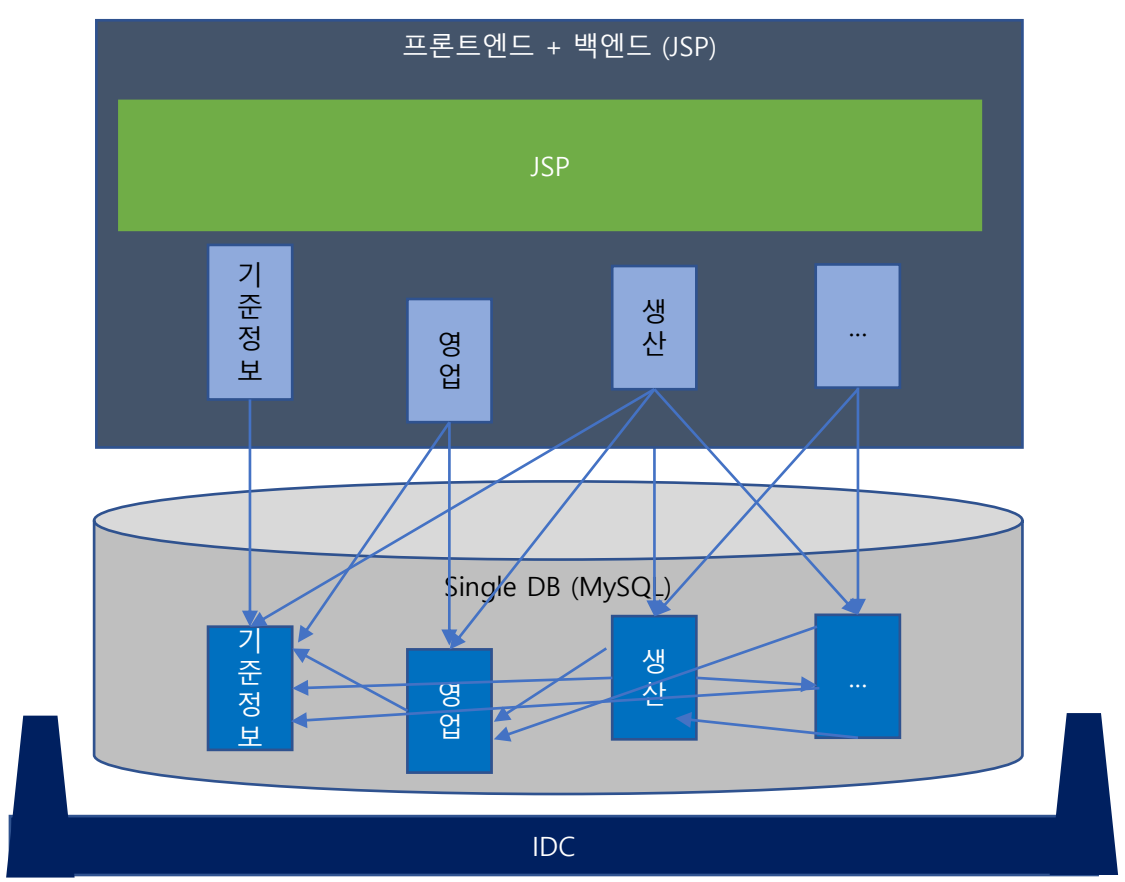
샘플 코드

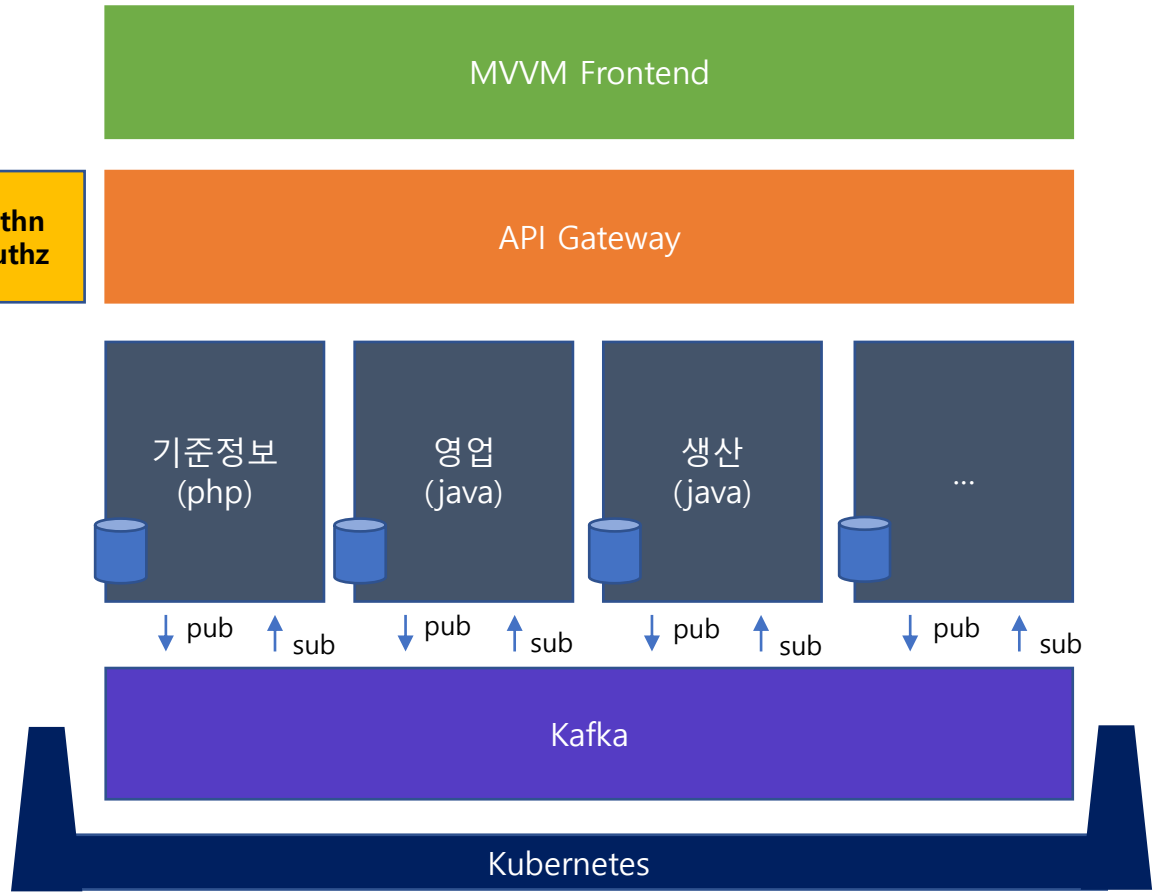


MSA School

Architecture

Separate db
Event driven
Run on Container / Kubernetes





MSA School

MSA 전문 설계-구현-운영 도구

www.msaez.io
github.com/msa-ez/platform

도메인 모델링
- 이벤트 스토밍
- 서브 도메인 모델 설계
- 템플릿 적용 코드 생성

단위 마이크로 서비스 구현
- 마이크로 서비스 패턴 적용
- 테스트 생성 / 로직 구현
- API 문서화 / Mock 생성

쿠버네티스 배포
- 객체 선언
- 파이프라인 구성
- 코드 배포 운영

실습 – MSAez 를 이용한 공공 서비스 앱 생성

기능적 요구사항

<Core Domain>
- 민원 접수
- 내 민원 진행상태 보기
- 민원 서류 보완 및 보정 요청
- 민원 유형별 처리
- 민원 유형별 사실 조사
- 보상 신청
- 보상 심의
- 보상 (지급) 처리
- 처리 결과 회신 또는 민원 취하

<Supporting Domain>
- 민원 분석

<Generic Domain>
- 접수 사실 통지
- 공지사항
- 자료실
- 인증서 로그인

비기능적 요구사항

장애격리

- 민원 처리 서비스가 장애 시 라도 민원 접수는 문제가 없어야 한다 → Event-driven, Eventual Consistency
- 일부 시스템이 과중되면 적절히 확장이 이루어져야 한다 → Horizontal Auto Scale

프로세스 준수

- 각 마이크로 서비스를 넘나드는 프로세스는 일관성이 있게 준수되어야 한다. (예: 민원 처리 기한 준수)

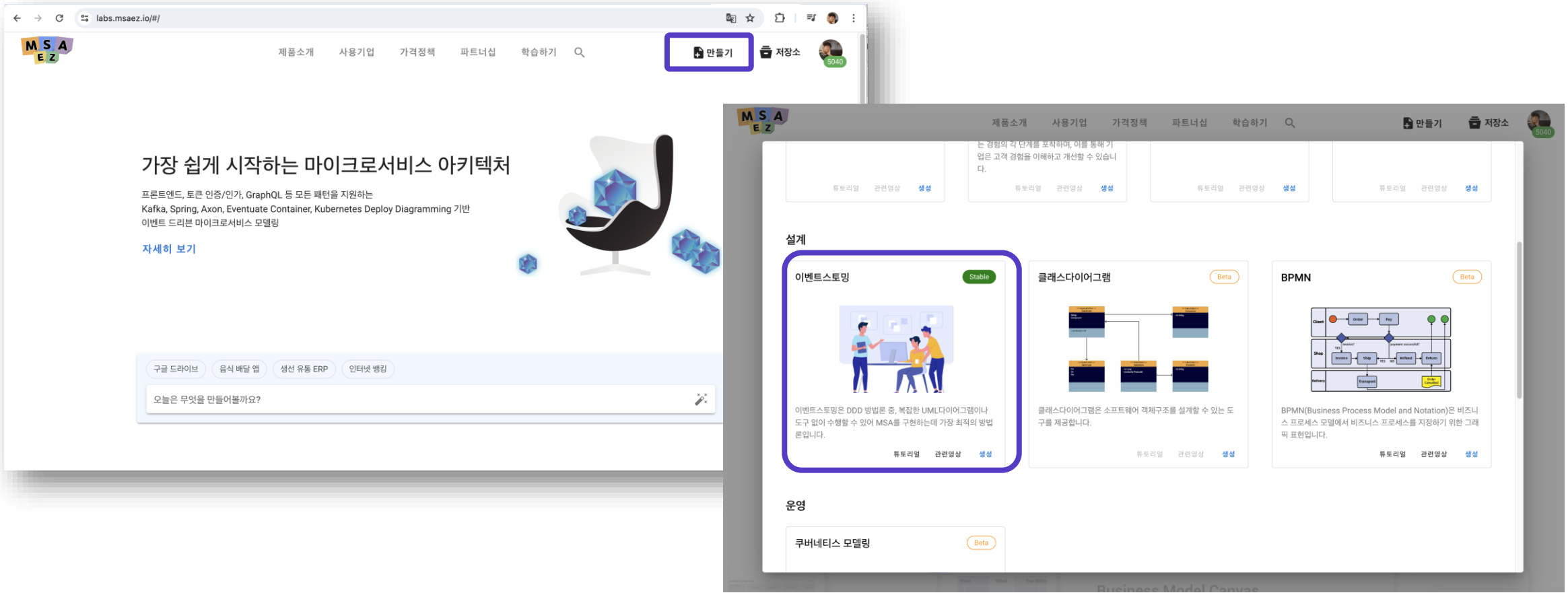
성능

- 민원인이 자주 민원 진행 상태를 조회하더라도 성능저하가 없어야 한다. → CQRS

바운디드 컨텍스트

1. 민원 접수
2. 민원유형 1
3. 민원유형 n
4. 보상처리
5. 민원분석
6. 기타

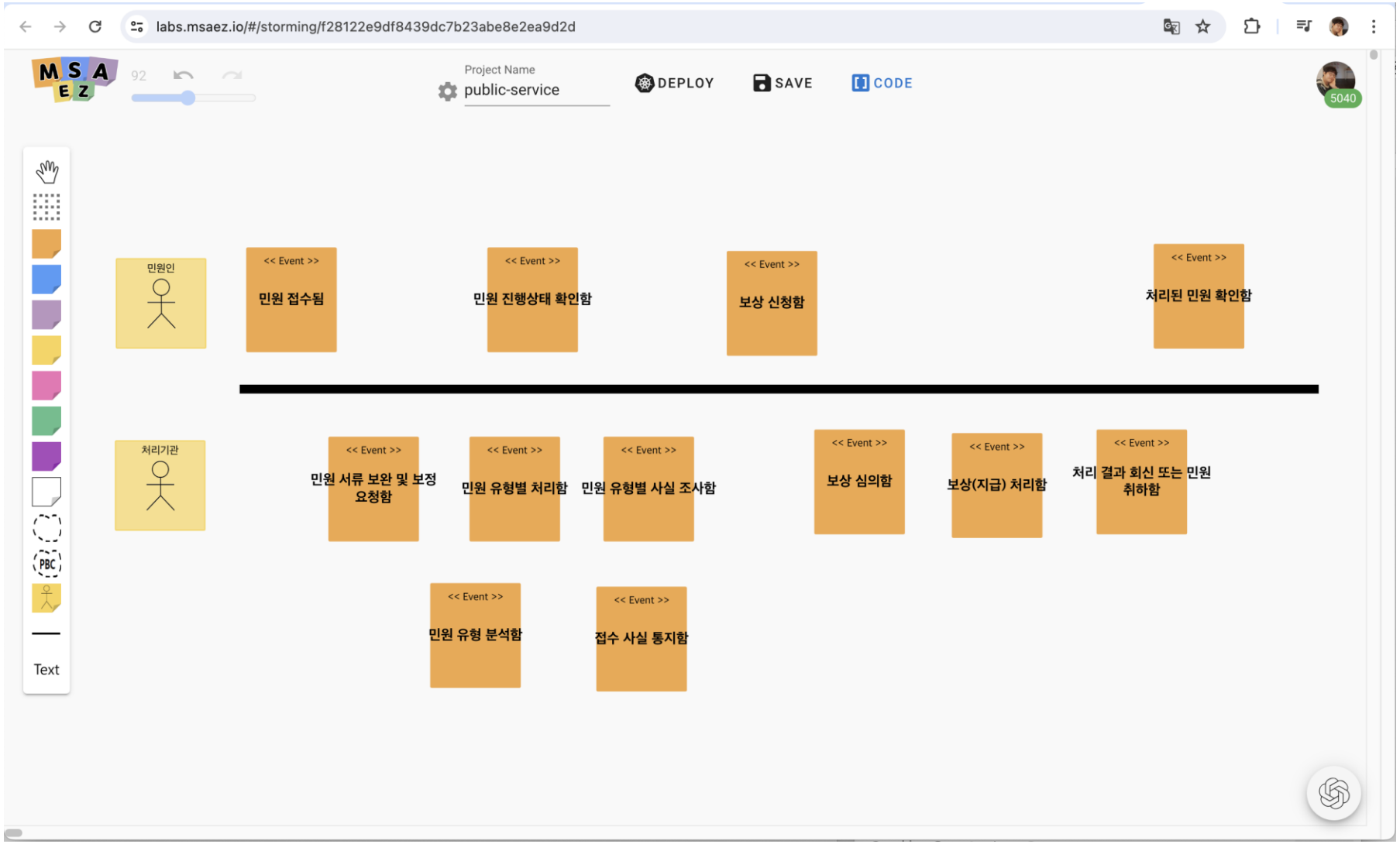
www.msaez.io 접속 > 만들기 > 이벤트스토밍



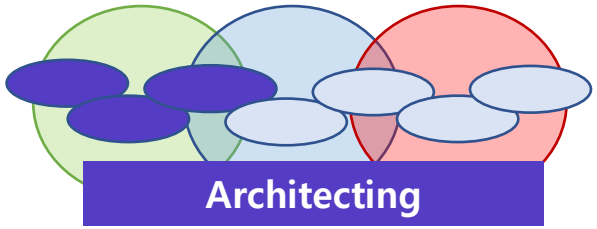
* 깃헙 계정으로 로그인 필요

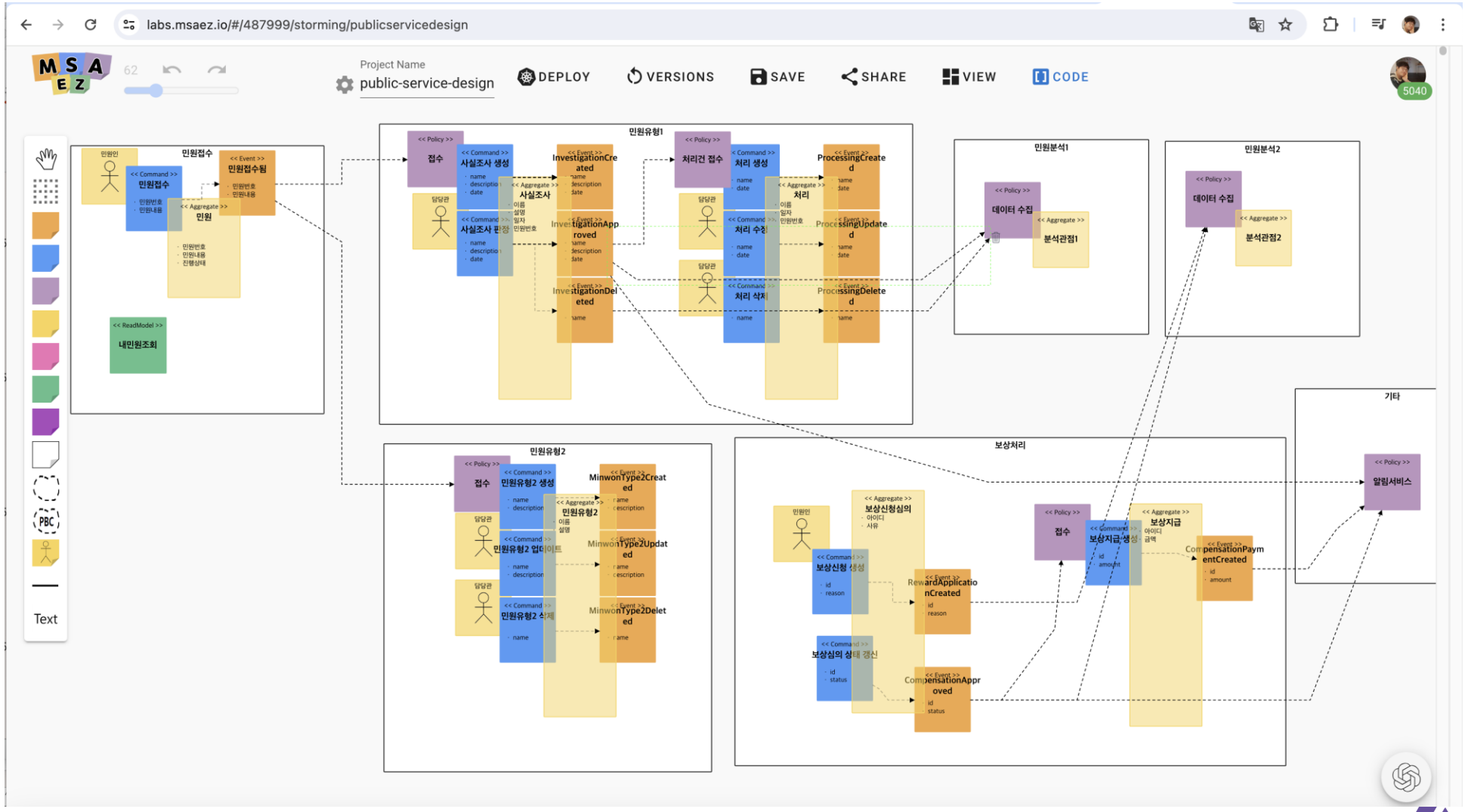
MSA School

이벤트 도출 – 액터, 프로세스 식별

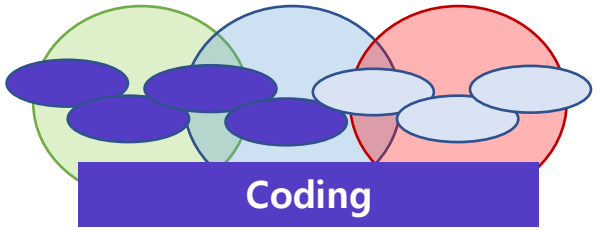


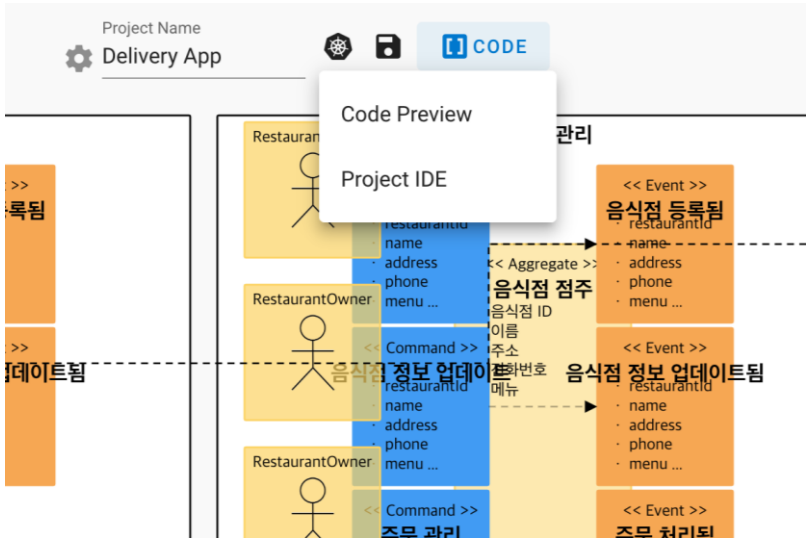
이벤트스토밍 모델 생성

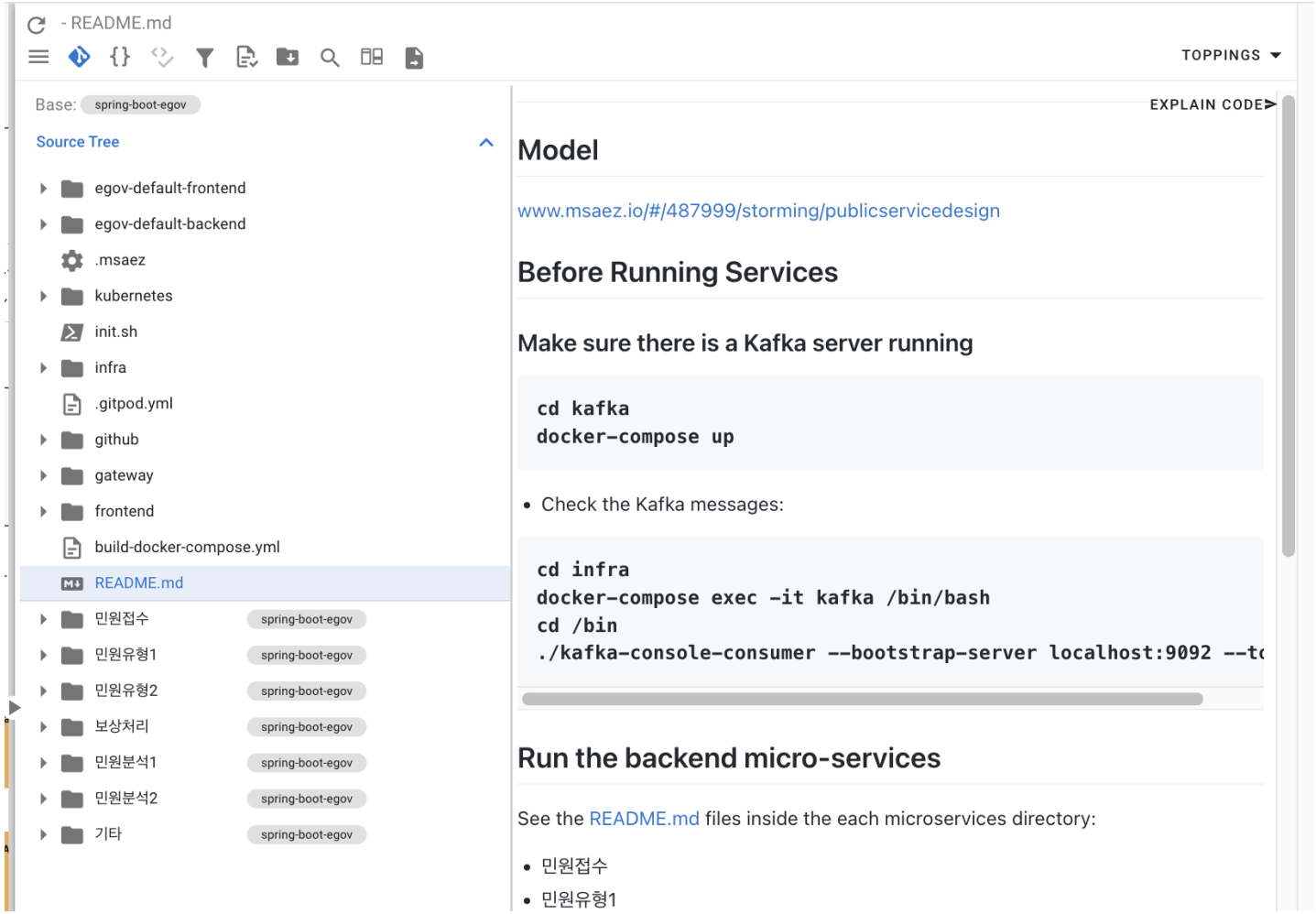

Architecting


labs.msaez.io/#/487999/storming/publicservicedesign
Project Name
public-service-design
DEPLOY
VERSIONS
SAVE
SHARE
VIEW
CODE
5040
민원접수
민원유형1
민원분석1
민원분석2
민원유형2
보상처리
기타
Text

코드의 생성

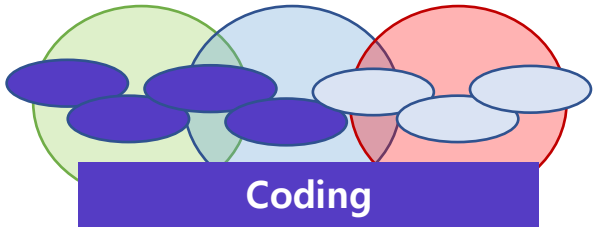


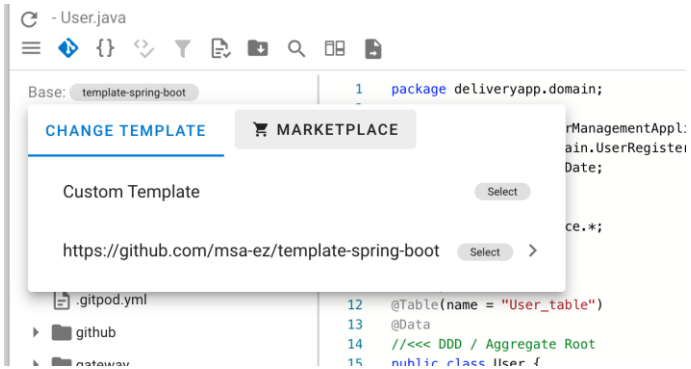


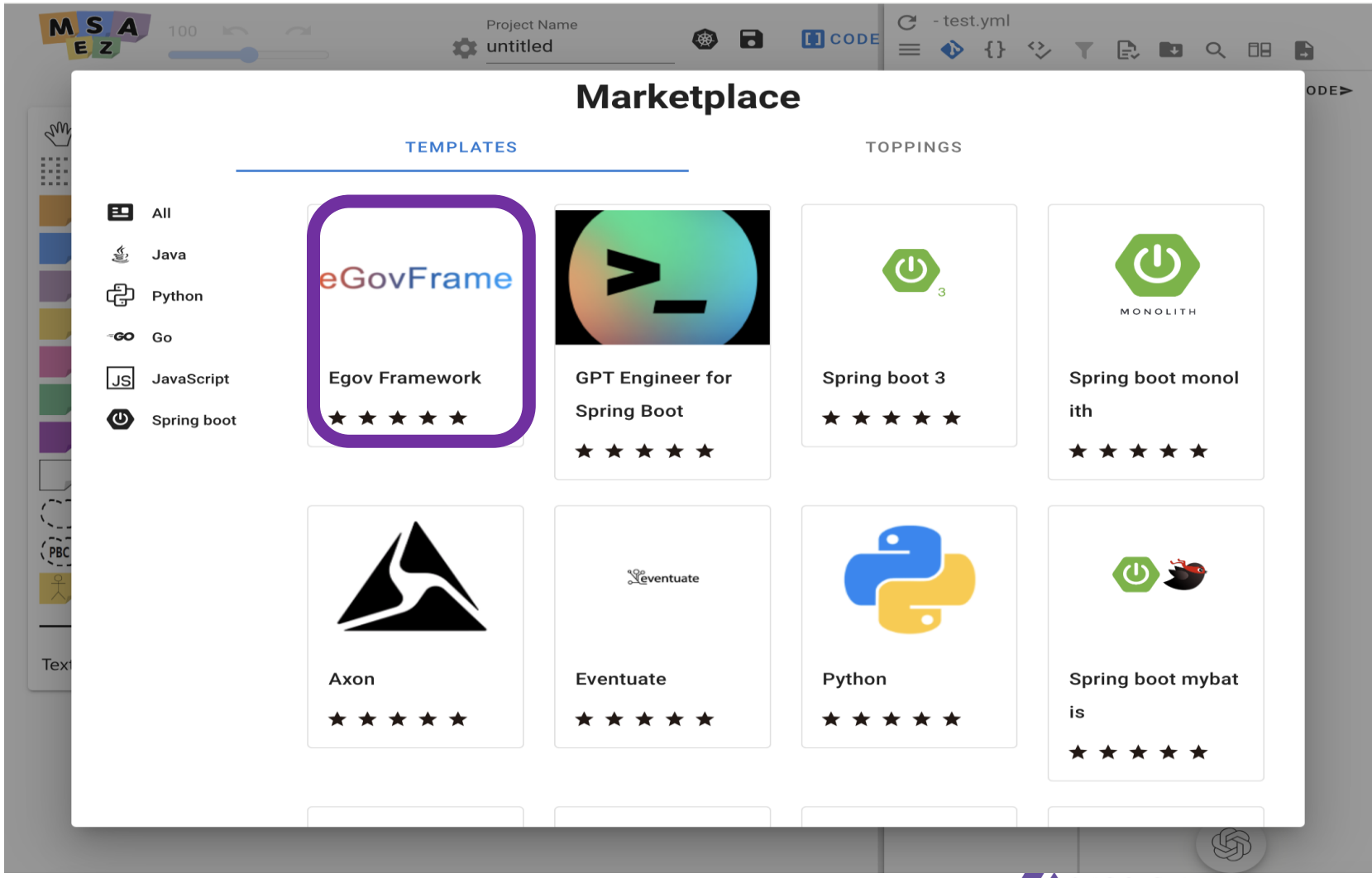


MSA School

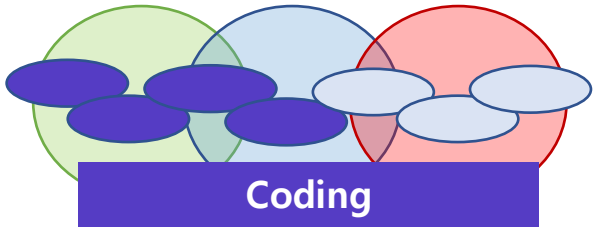
코드생성 > 템플릿


Coding


- User.java
Base: template-spring-boot
CHANGE TEMPLATE
MARKETPLACE
Custom Template
Select
https://github.com/msa-ez/template-spring-boot
Select
.gitpod.yml
github
package deliveryapp.domain;
@Table(name = "User_table")
@Data
//<<< DDD / Aggregate Root


MSA EZ
Project Name
untitled
CODE
- test.yml
Marketplace
TEMPLATES
TOPPINGS
All
Java
Python
Go
JavaScript
Spring boot
eGovFrame
Egov Framework
GPT Engineer for Spring Boot
Spring boot 3
Spring boot monolith
MONOLITH
Axon
Eventuate
Python
Spring boot mybatis

토핑추가



제네릭 도메인:
게시판/일정관리/로그인

Marketplace

TEMPLATES

TOPPINGS

- All
- Multitenancy
- Kubernetes
- Security
- Service Mesh
- DevOps
- Data Projection
- Frontend

eGov Default App ★★★★★

GPT Engineer for Spring Boot ★★★★★

Materio ★★★★★

Micro Wijmo ★★★★★

Unit Test for Microservices ★★★★★

Microfrontend with Single-SPA + VueJS ★★★★★

Local Microservice Development Dependencies ★★★★★

Spring boot multitenancy schema ★★★★★

Spring boot multitenancy partitioned ★★★★★

Github action ★★★★★

Wijmo ★★★★★

IsVanillaK8s ★★★★★

Istio ★★★★★

Ingress ★★★★★

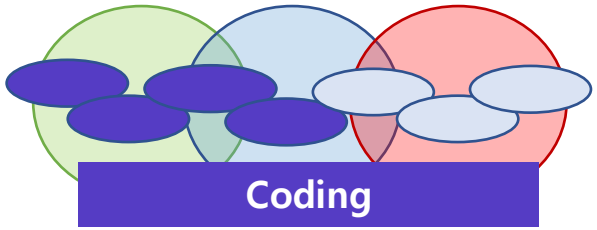
Argo ★★★★★

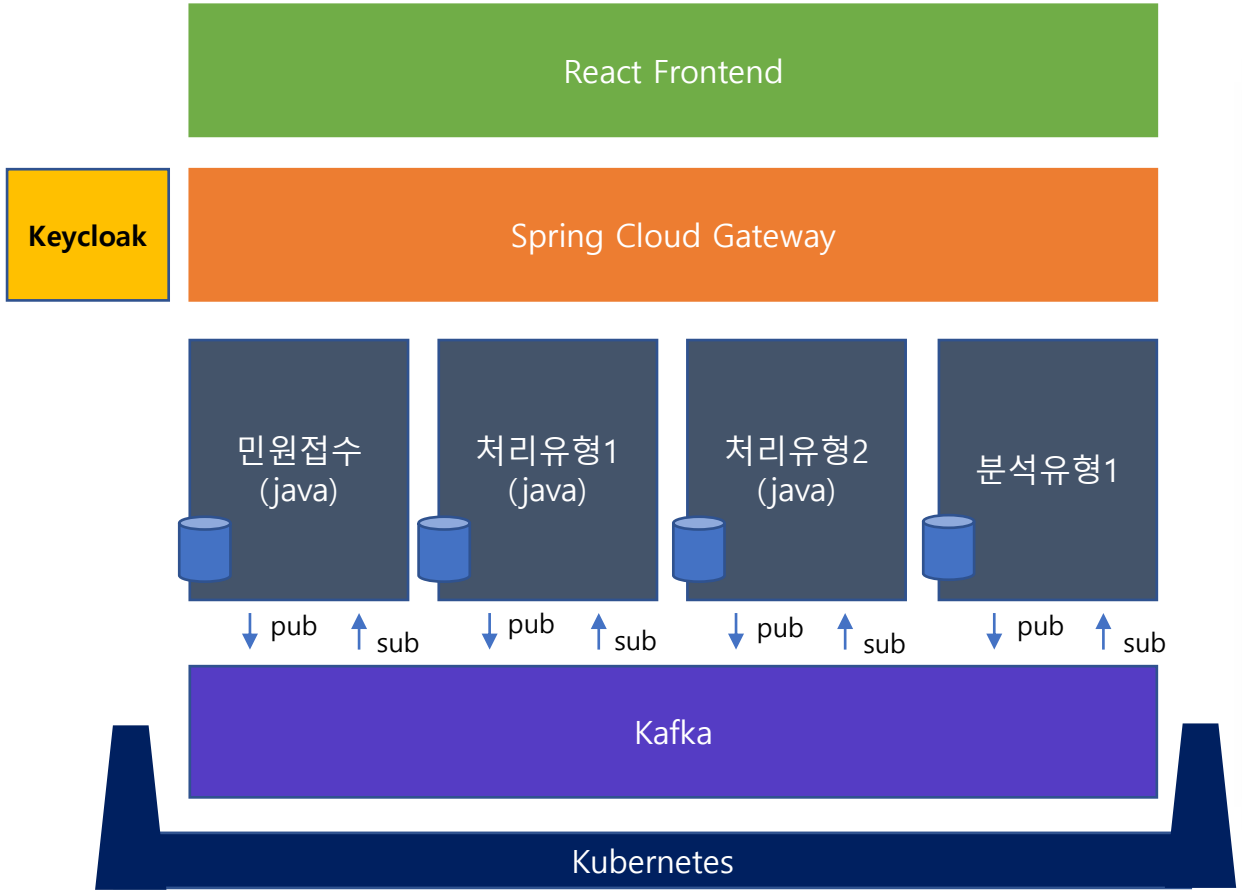
Spring security ★★★★★

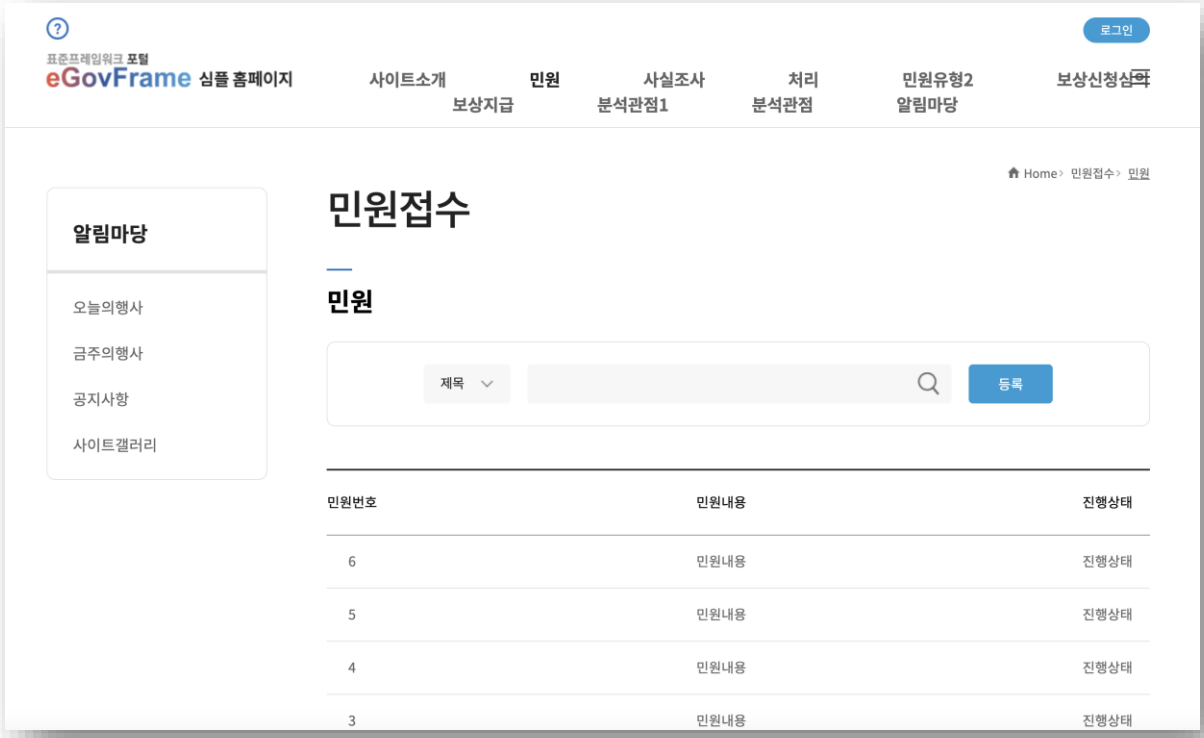
Apollo graphql ★★★★★

Keycloak security ★★★★★

완성된 애플리케이션

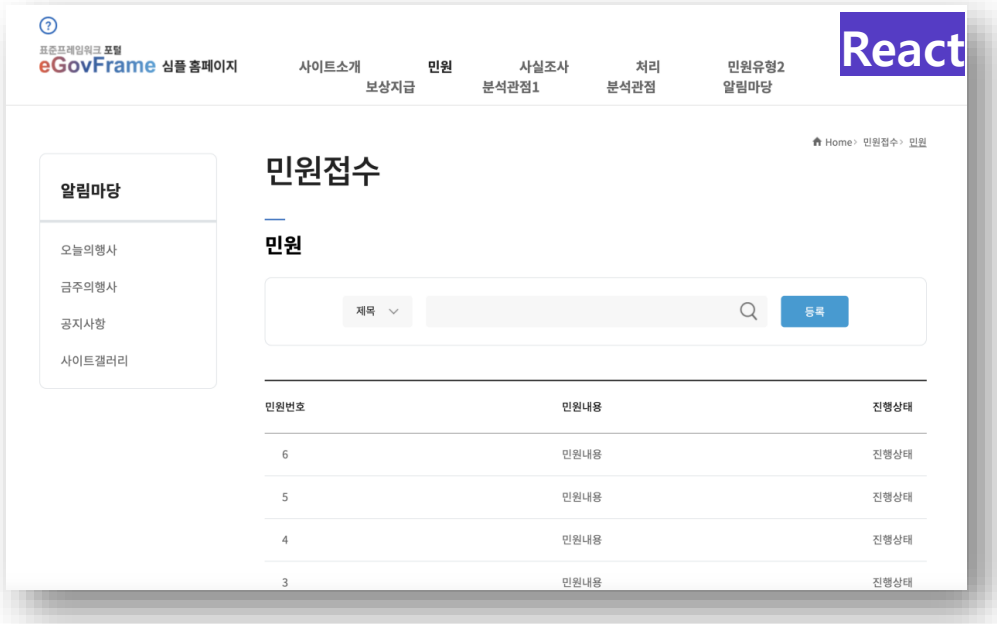


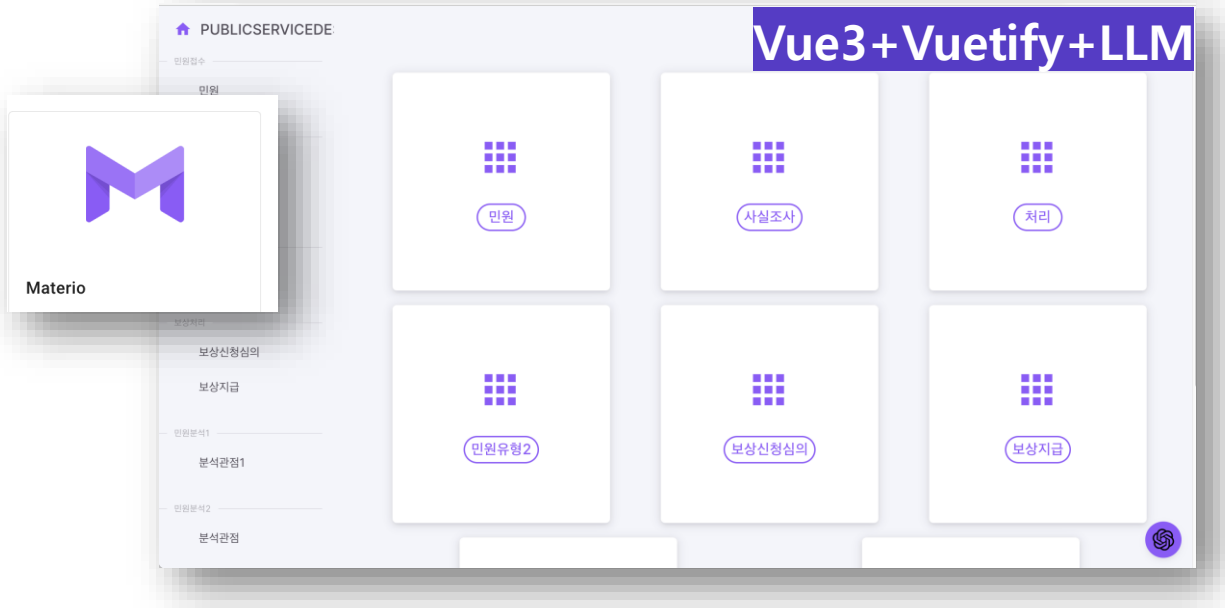


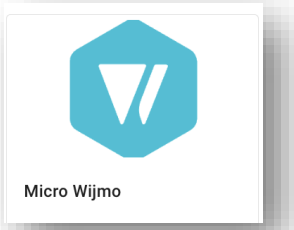


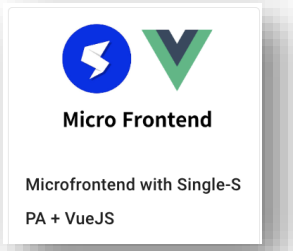
MSA School

토핑: 다양한 프론트엔드 지원









MSA School

토핑: DevOps 도구들

테스팅

Local Microservice Develop
ment Dependencies

★ ★ ★ ★ ★

API 문서화

Unit Test for Microservices

★ ★ ★ ★ ★

쿠버네티스

IsVanillaK8s

★ ★ ★ ★ ★

Ingress

★ ★ ★ ★ ★

서비스 매시

Istio

★ ★ ★ ★ ★

깃 옵스

Argo

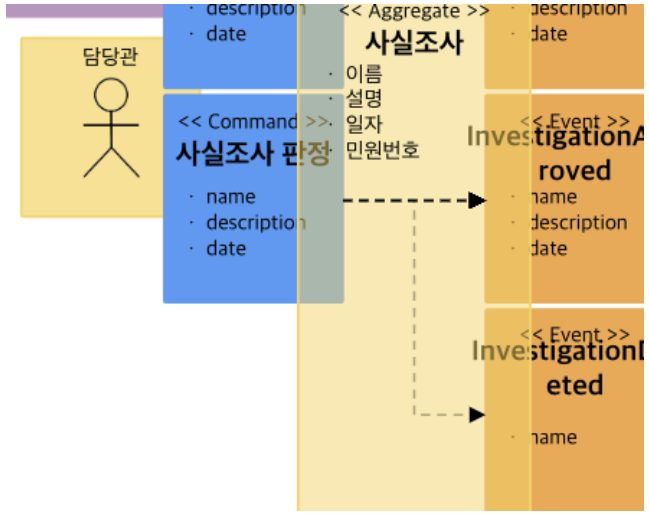
★ ★ ★ ★ ★

Github action

★ ★ ★ ★ ★

MSA School

토핑: 보안관련

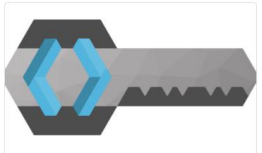


스프링 시큐리티

Spring security

★ ★ ★ ★ ★

키클락



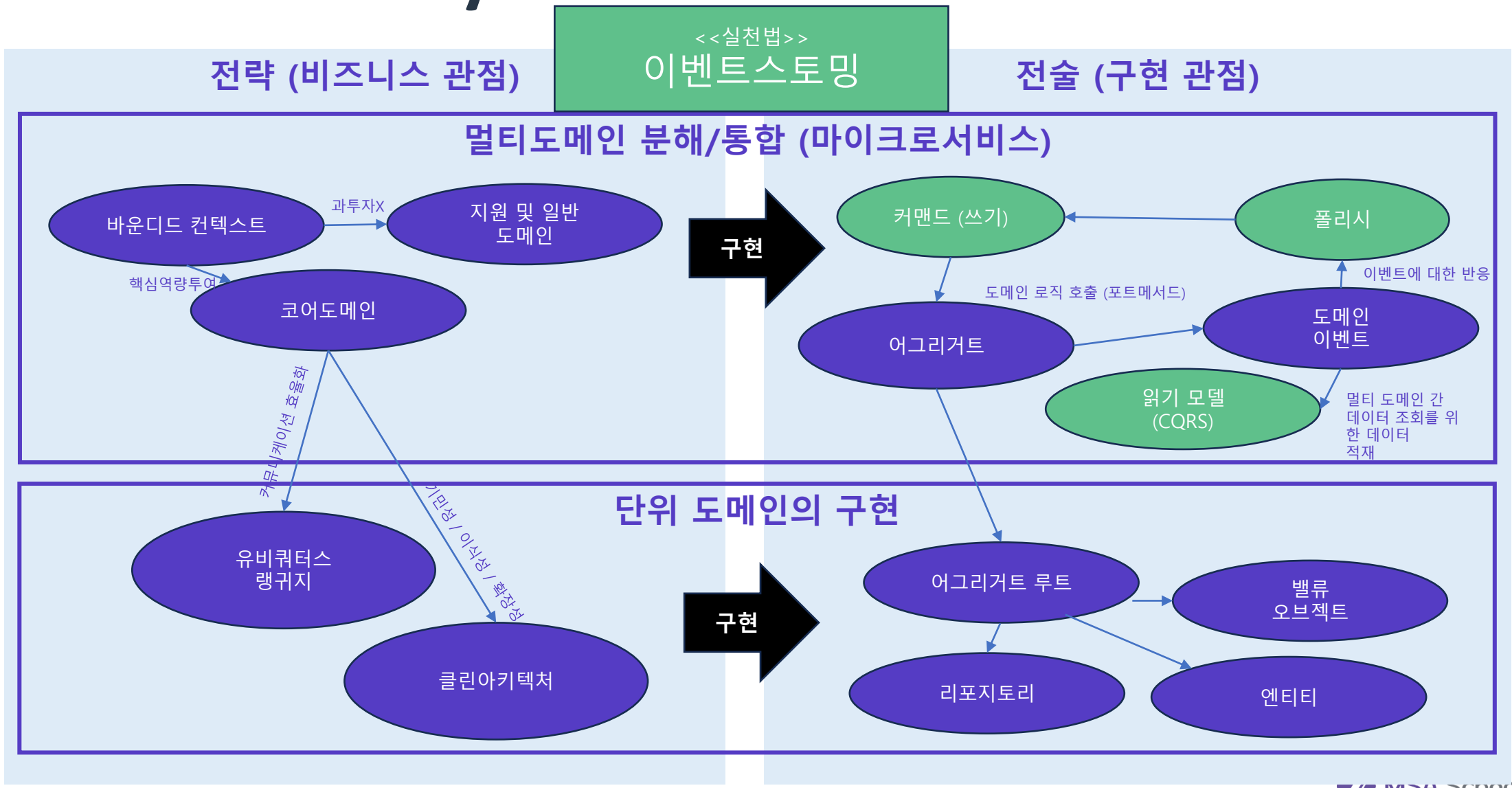
Keycloak security

★ ★ ★ ★ ★

각 Command에 설정한 액터
→ 권한

Frontend와 Backend의 보안설정 (@AllowedScopes)으로 코드 생성

DDD Summary

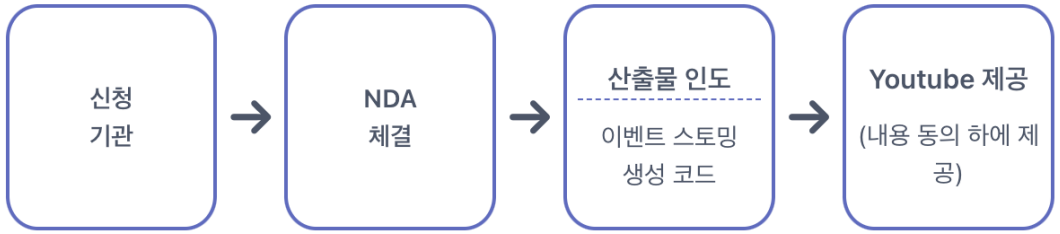

<<실천법>>
이벤트스토밍
전략 (비즈니스 관점)
전술 (구현 관점)
멀티도메인 분해/통합 (마이크로서비스)
바운디드 컨텍스트
과투자X
지원 및 일반 도메인
핵심역량투여
코어도메인
구현
커맨드 (쓰기)
폴리시
이벤트에 대한 반응
도메인 로직 호출 (포트메서드)
어그리거트
도메인 이벤트
읽기 모델 (CQRS)
멀티 도메인 간 데이터 조회를 위한 데이터 적재
커뮤니케이션 효율화
민성 / 이식성 / 확장성
단위 도메인의 구현
유비쿼터스 랭귀지
클린아키텍처
구현
어그리거트 루트
밸류 오브젝트
리포지토리
엔티티

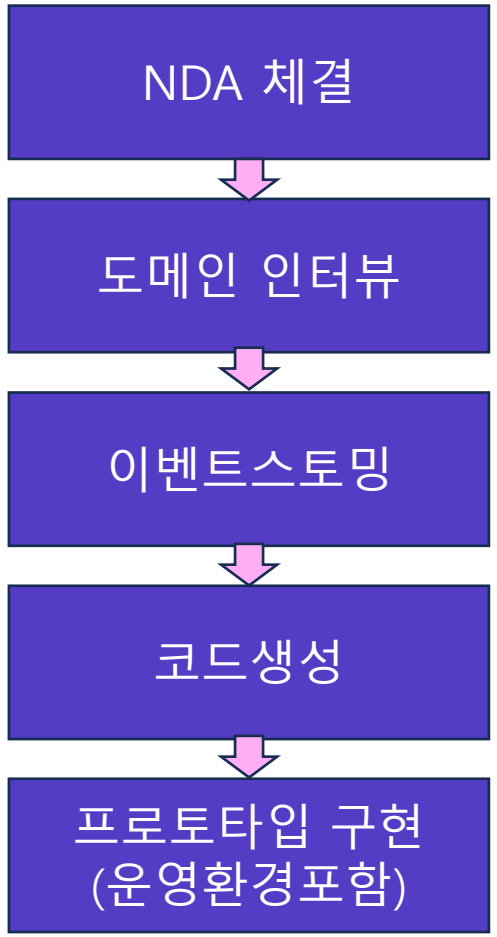
보다 자세한 사항은 컨설팅 서비스를 신청하세요

무상 컨설팅 Test Drive 행사 (선착순 3개 기업)

1.5 Hour ver. **(컨설팅 과정 비공개)**

1 Day ver. **(컨설팅 과정 유튜브 공개)**





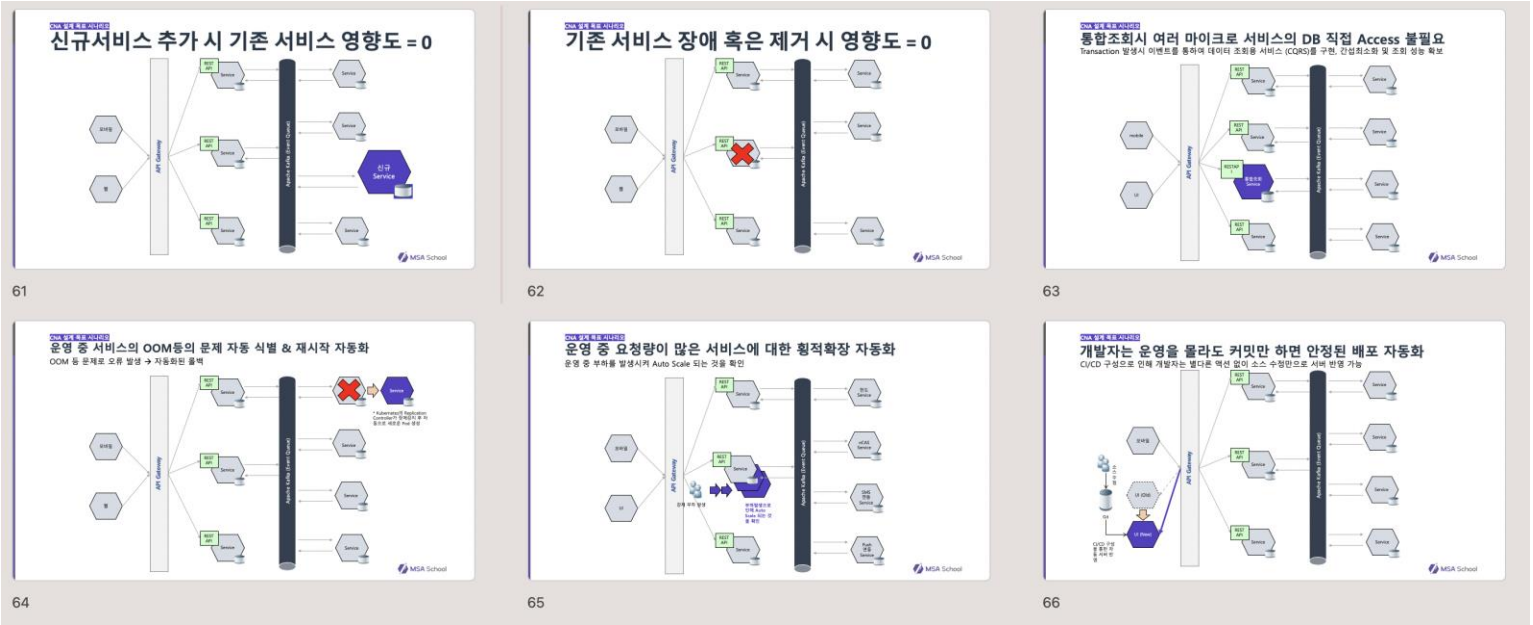
설문 및
무상 컨설팅 신청

MSA School

다음 편 예고 (5/14) – MSA 의 구현

- CNA 설계 목표 시나리오
- 마이크로 서비스 구현 패턴
- RPC와 서킷브레이킹
- EDA (오케스트레이션과 코레오그라피)와 카프카
- 데이터 프로젝션: GraphQL / CQRS
- 마이크로 프론트엔드
- MSA 테스팅과 목업 생성 (컨트랙트 테스트/단위테스트)
- 보안처리
- 컨테이너 패키징

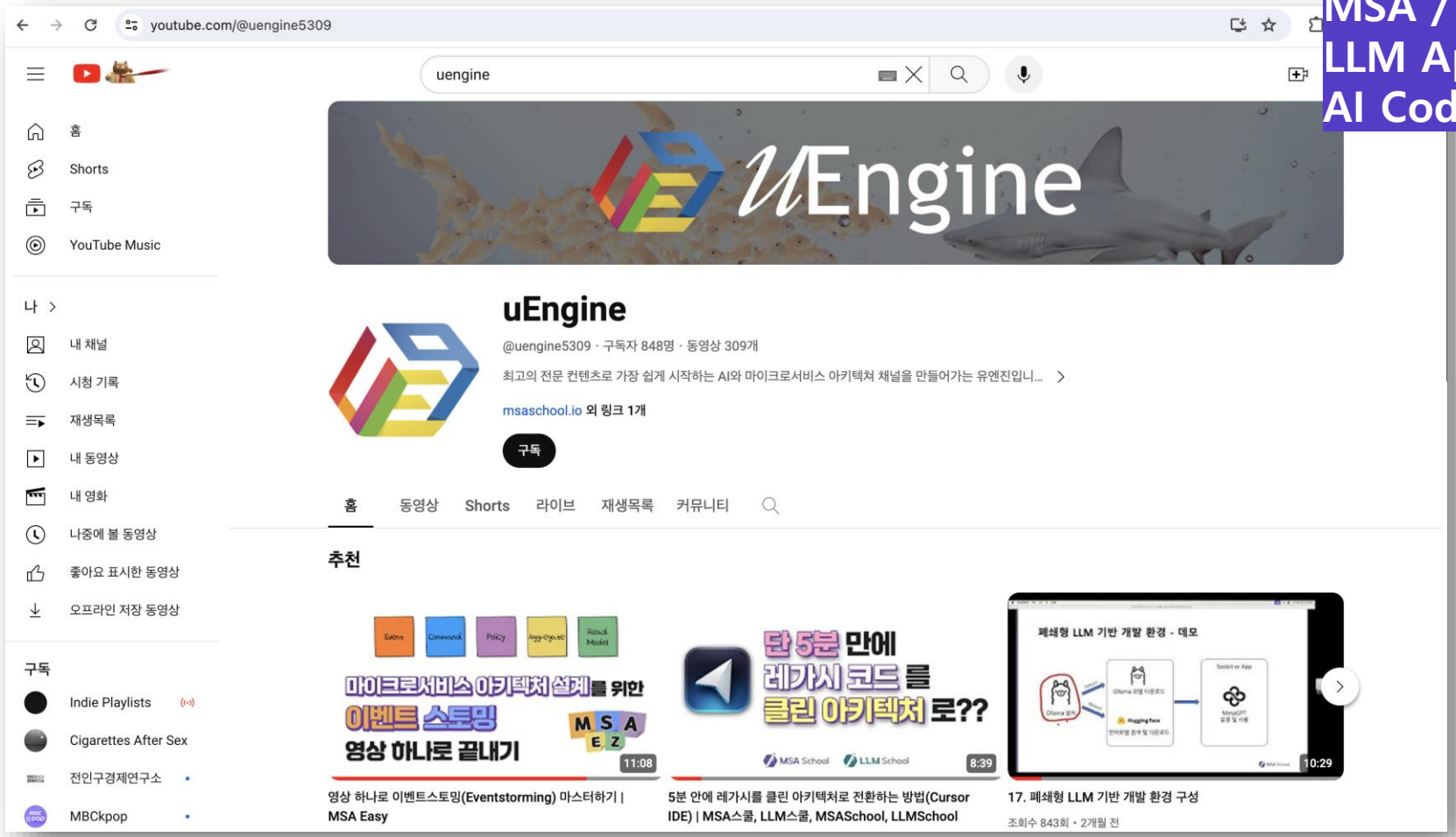
등을 MSA Easy 를 통하여 세부적으로 설계하고 구현하는 과정을 살펴봅니다



MSA School

유엔진 콘텐츠를 계속 시청하시려면 유튜브를 구독하세요

주제:
MSA / CNA / DDD
LLM Application 개발방법
AI Coding / Legacy Modernization



MSA School