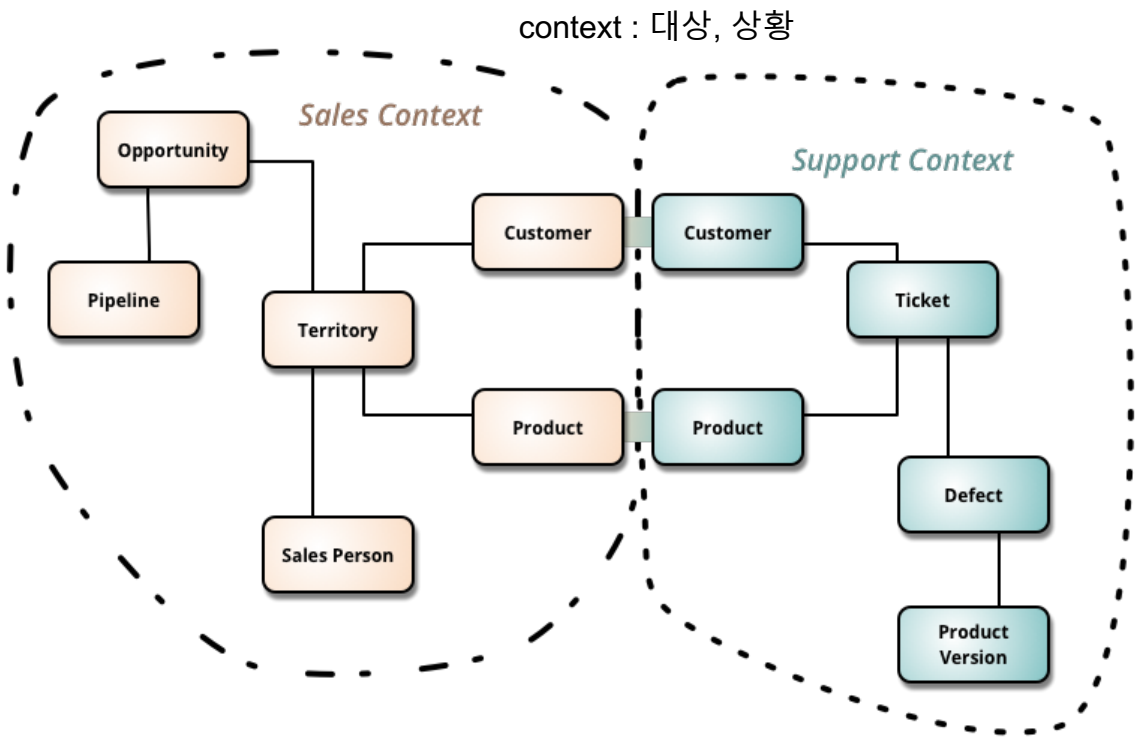
DDD

https://youtu.be/Plm9YYmCmU4

Domain

- 영역, 집합
- DDD에서의 Domain == 비즈니스 Domain
- 유사한 업무의 집합 (마케팅, 구매, 연구, 영업 등)
- 어플리케이션을 Domain 별로 나누어 설계 및 개발



DDD

- Domain Driven Design
- Domain 별로 나누어 설계하는 방식
- 비즈니스 논리를 캡슐화 하고, 현실과 코드 간의 격차를 완화

DDD

- Loosely coupling
  - 모듈간 의존성 최소화

- High cohesion
  - 응집성 최대화

DDD

- Strategic Design
  - 개념 설계

- Tactical Design
  - 구체적 설계

Strategic Design

- Business Domain의 상황(context)에 맞게 설계하자는 컨셉
- context를 event storming으로 공유하고, 비즈니스 목적별로 서비스들을 그룹핑
- bounded context : business domain의 사용자, 프로세스, 정책, 규정 등을 목적별로 그룹핑한 것
- domain model : business domain의 서비스를 추상화한 설계도

Event storming

- business domain 내에서 일어나는 것들을 찾아 bounded context를 식별하는 방법론

- ‘주문’ 도메인의 ‘주문’ 서브도메인을 통해 설명

Event storming - step 1

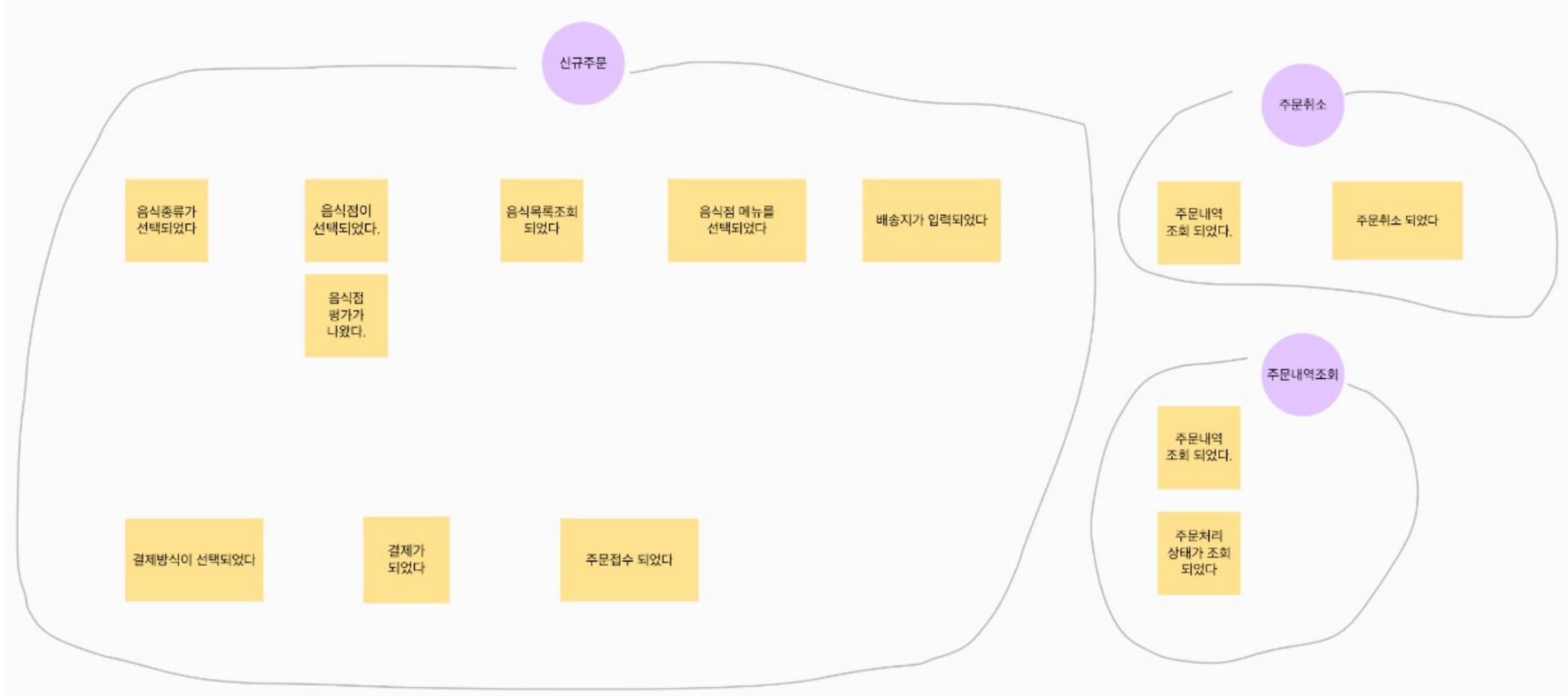
- domain event 정의
  - 발생하는 모든 이벤트를 과거형으로 기술

Event storming - step 2

- Tell the story
  - 도출된 이벤트를 통해 도메인의 업무 흐름을 이해하고 토론하여 보완

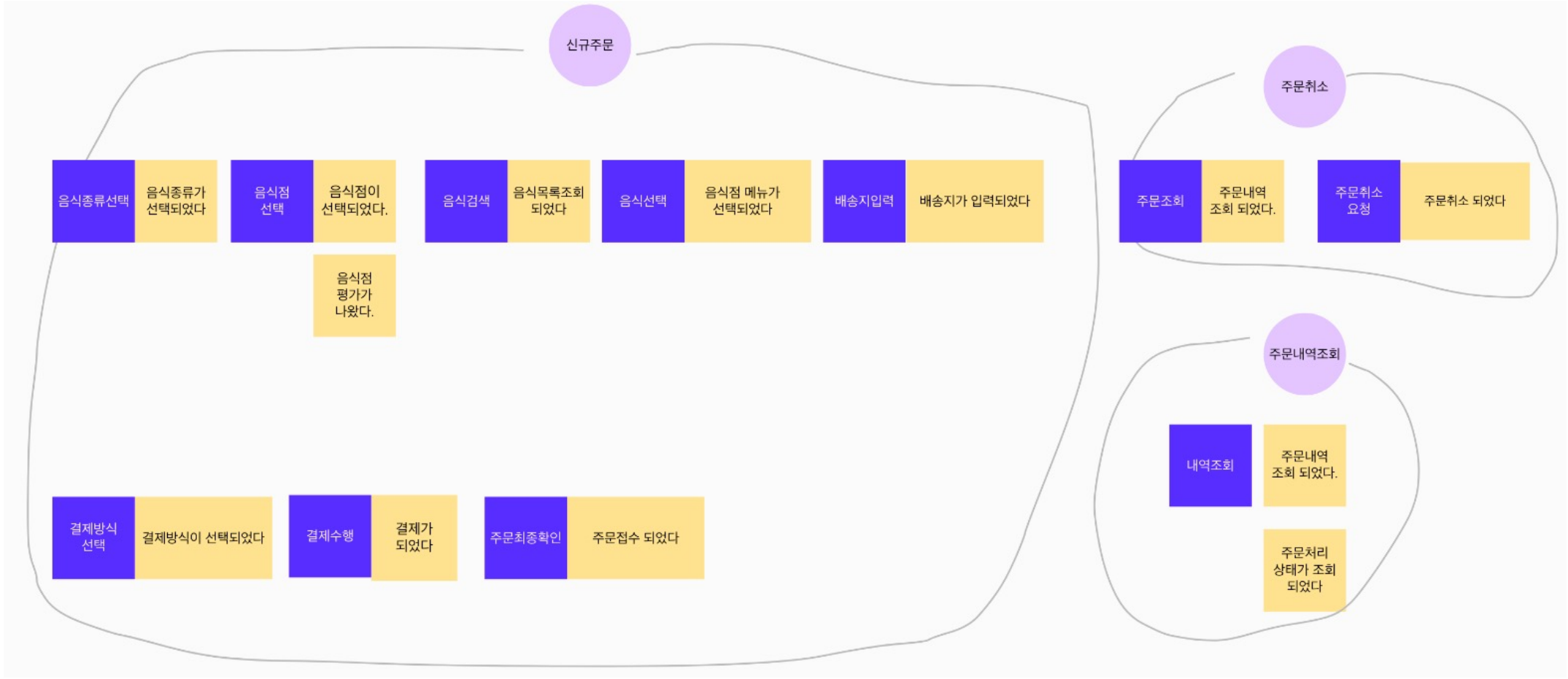
Event storming - step 3

- 이벤트들을 프로세스로 그룹핑



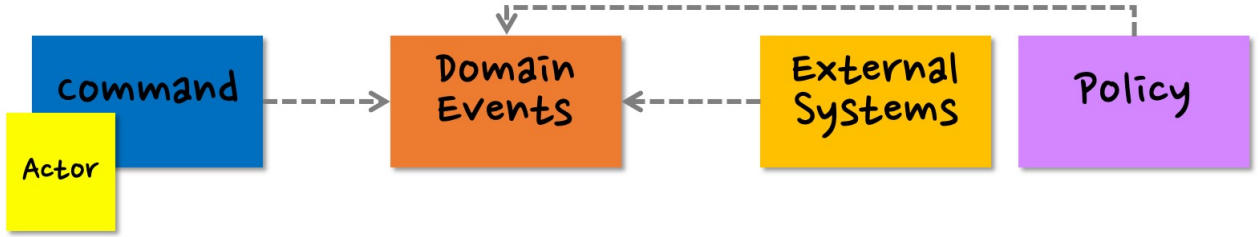
Event storming - step 4

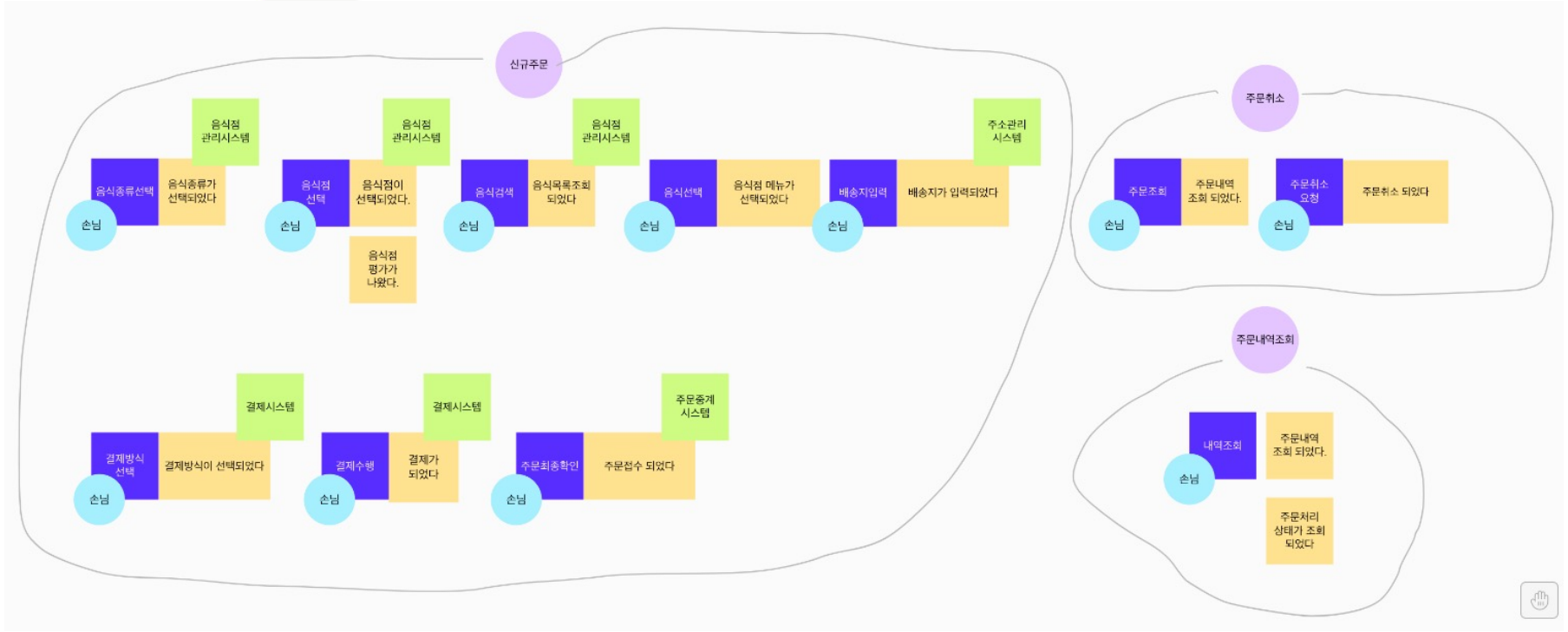
- command 정의 : 이벤트를 발생시키는 명령을 현재형으로 정의



Event storming - step 5

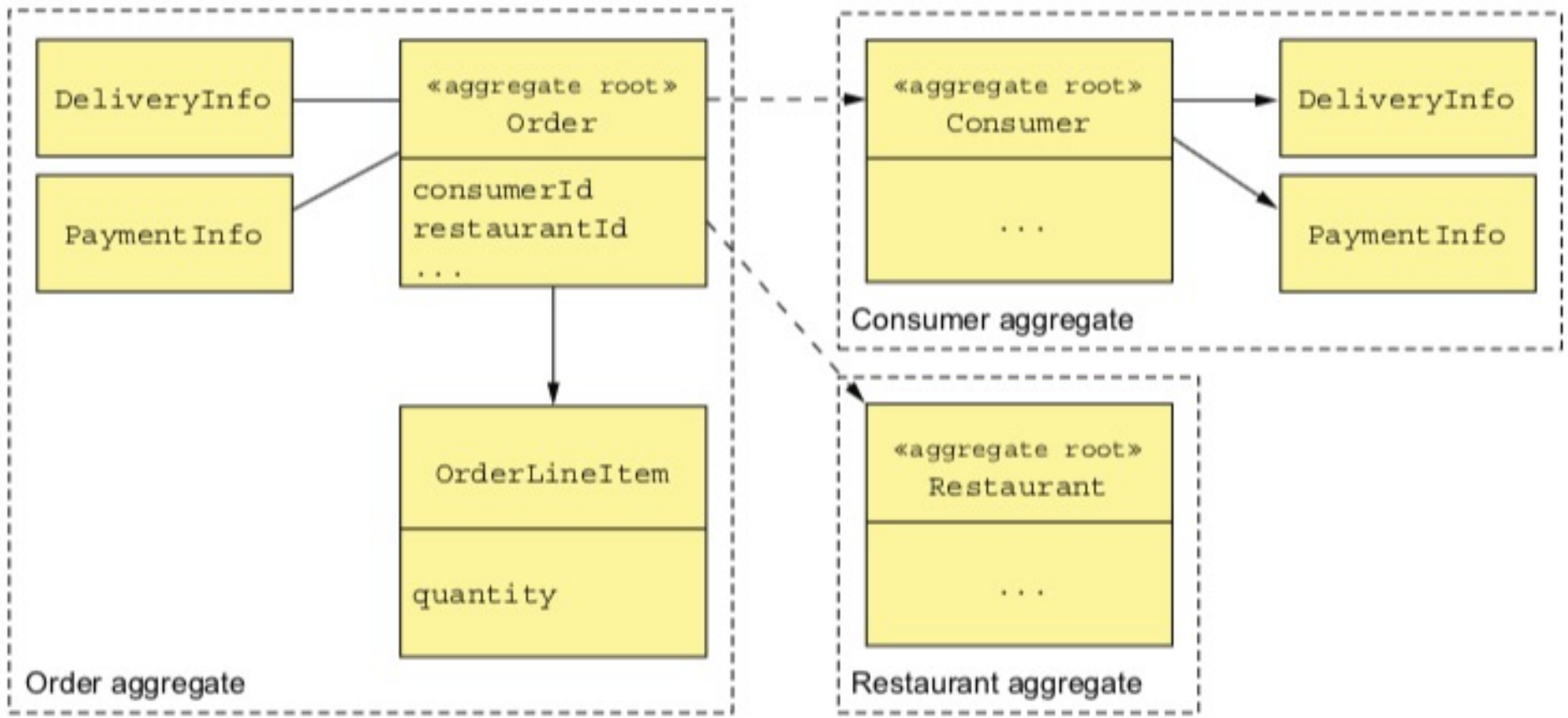
- trigger 정의 : command를 일으키는 actor, event를 일으키는 external system, policy/rule 정의





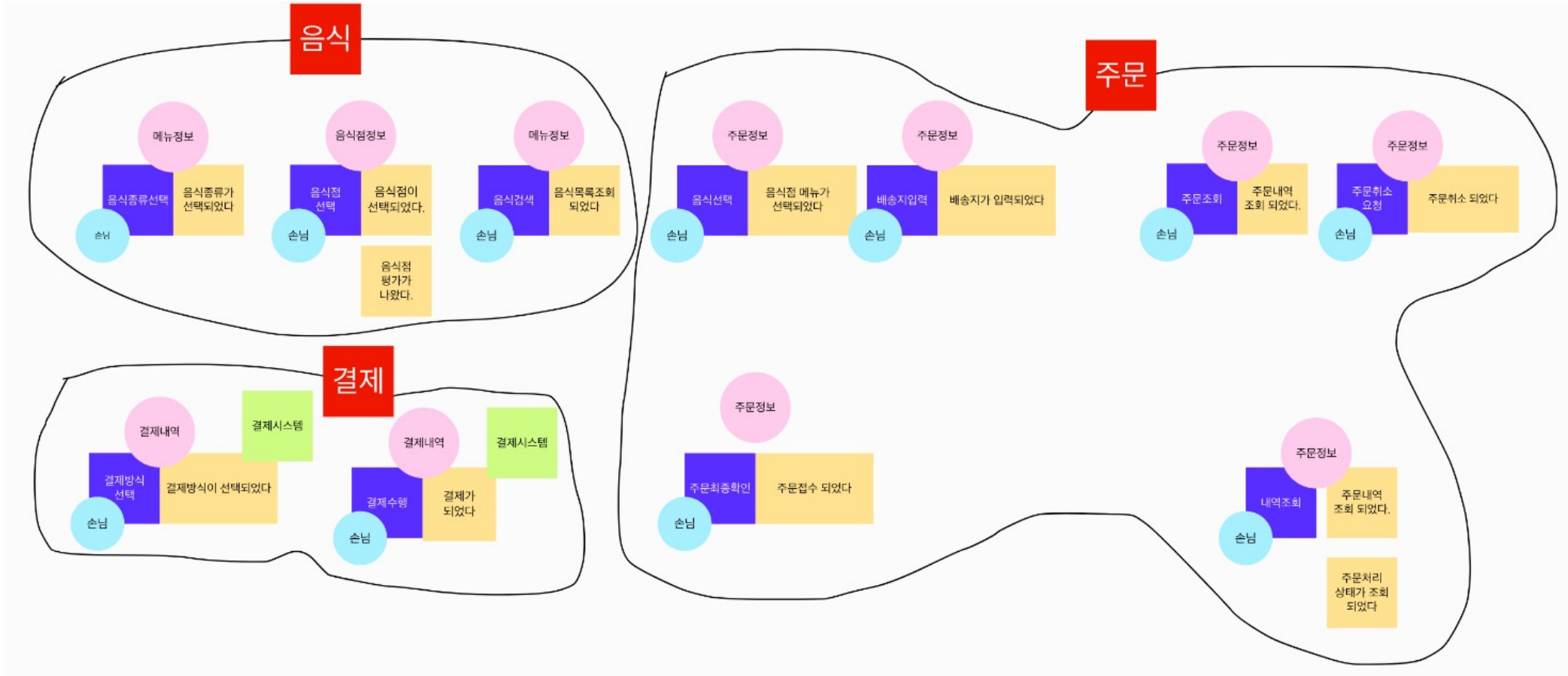
Event storming - step 6

- command 수행을 위해 CRUD 해야 하는 데이터 객체 정의



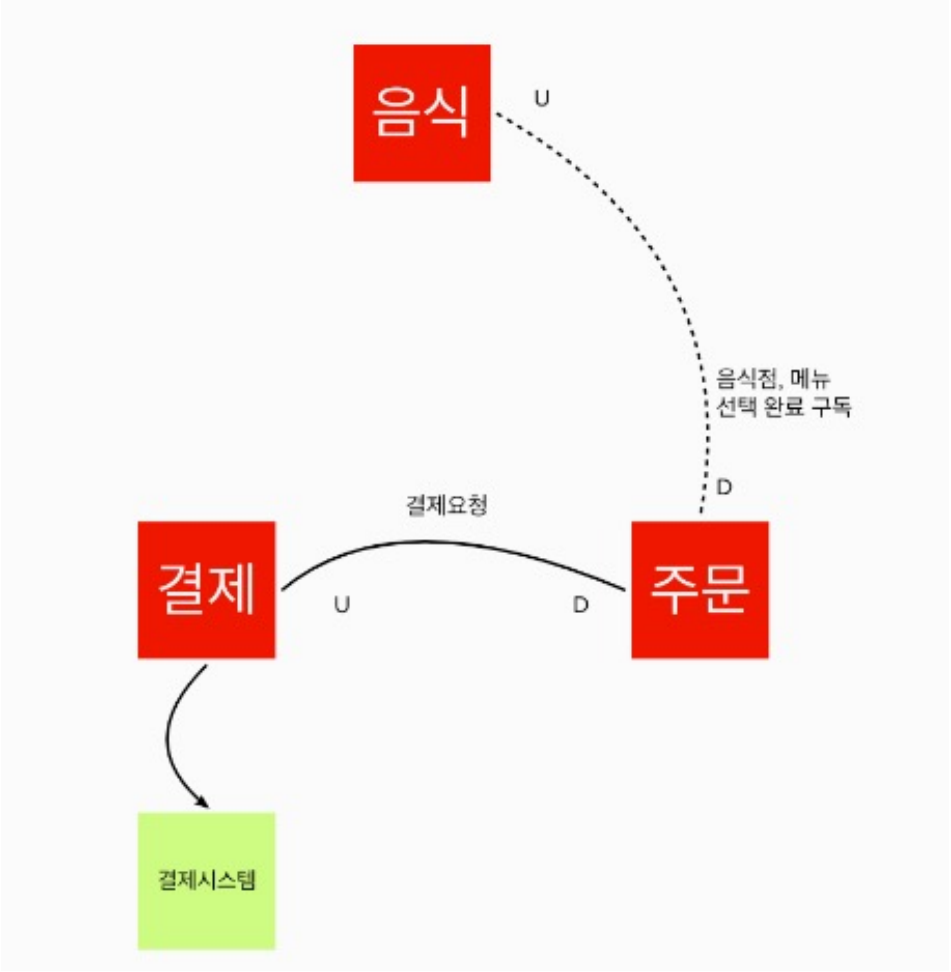
Event storming - step 7

- bounded context 정의



Event storming - step 8

- context map 작성
- 외부 시스템이 어떤 bounded context와 관계 되는지를 표현
- 옆 그림에서는 외부 시스템 : 결제 시스템, bounded context : 음식-주문-결제



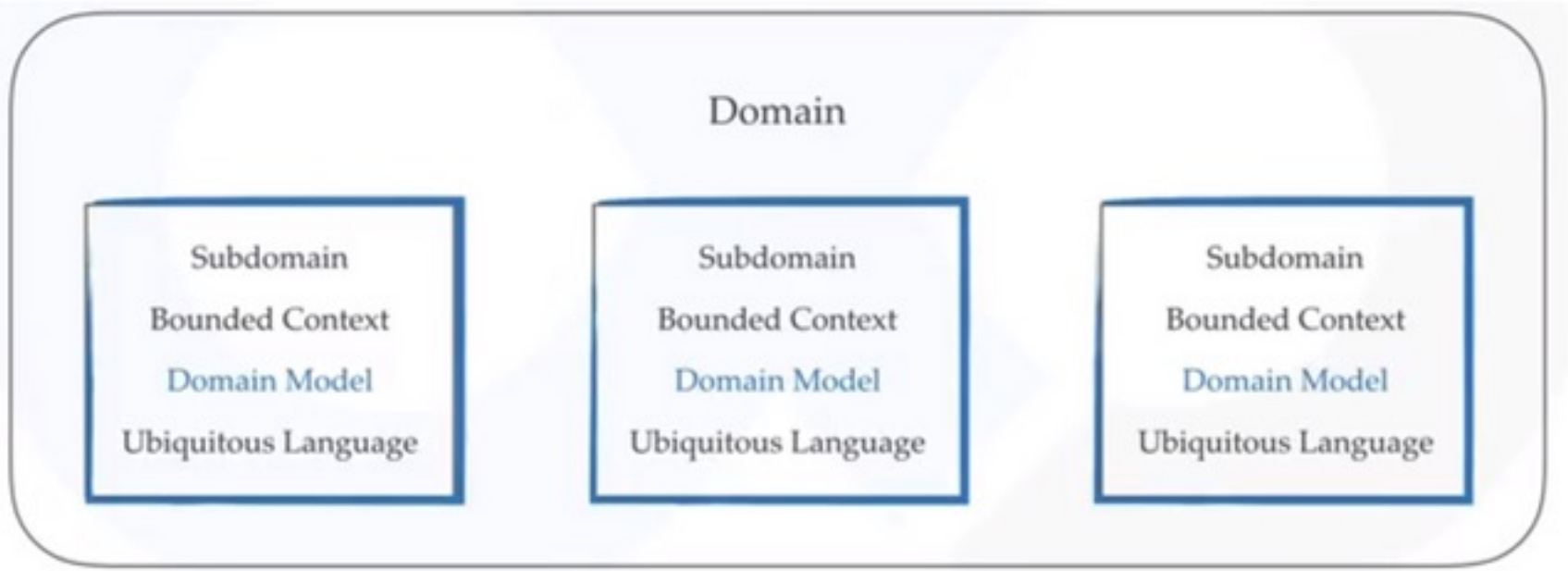
Tactical Design

- Model Driven Design

- Layered Architecture

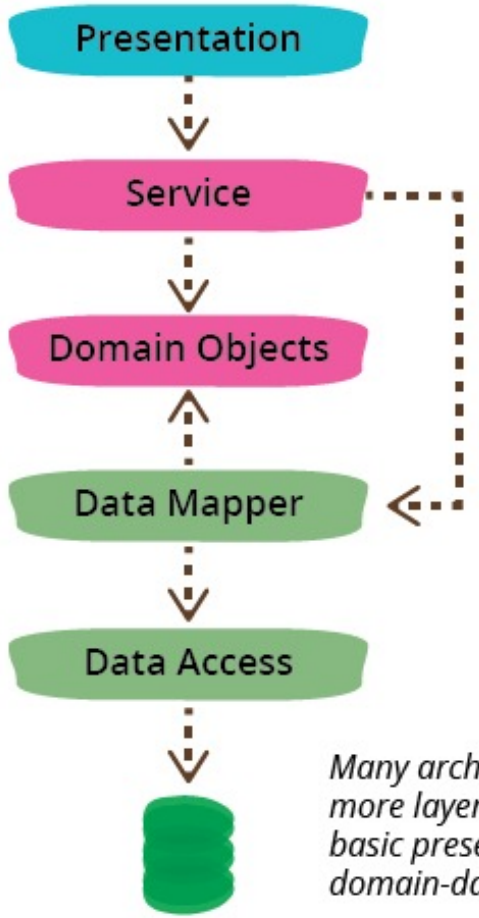
Model Driven Design

- 각 서브도메인별 domain model(context map)을 중심으로 설계



Layered Architecture

- 목적별 계층으로 나누어 설계하는 것
- Presentation Layer : UI Layer
- Service Layer : domain layer와 class들간의 제어 또는 연결
- Domain Layer : domain object별로 비즈니스 로직 처리
- Data Layer : DB와의 CRUD



Many architectures with more layers follow the basic presentation-domain-data pattern

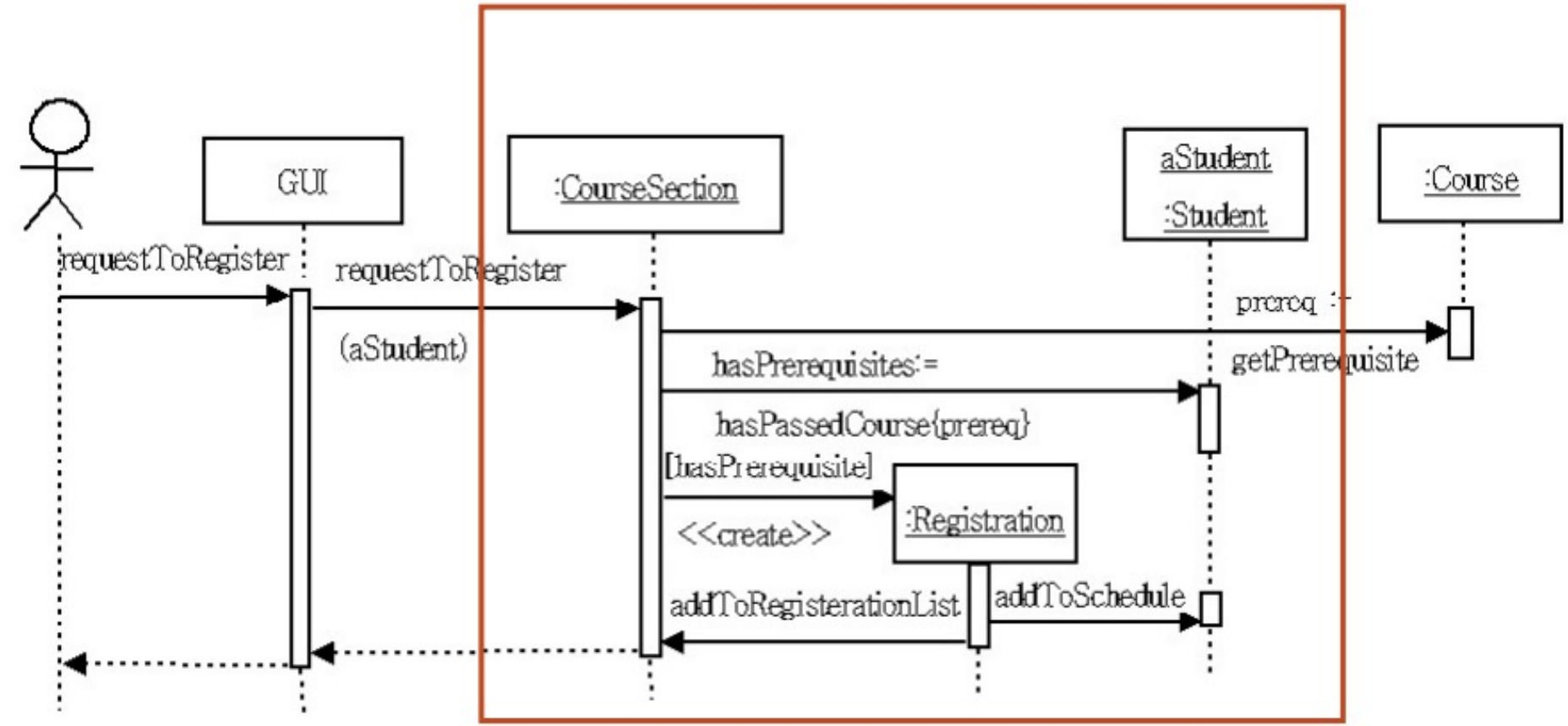
Tactical Design 결과물

- Userstory : B/C 기능과 test case를 사용자 중심으로 기술한 것

APP	ID	Category	User Story	Accept	Conditions	Importance	Score	Sprint	Team	Developer	Test case	Status	Comments
고객게시판	RQ-ACOP-F-003	LNB/목록	고객은 사전정보입력 화면을 통해 개인정보를 입력한다.								0. 입력된 실명확인번호에 대해 행안부 검증을 하는가? 1. 신규 고객이 아니라면 "고객 정보가 이미 있습니다"라고 메시지를 보여주는가? (아래 작업 불필요) 2. 기존에 저장되어 있는 정보가 있는가 확인후 저장되어 있는 정보 이후를 진행할수 있는가? 3. 입력할 개인정보는 다음과 같이 입력할 수 있는가? - 고객명 - 실명번호(주민등록번호) - 국적 (코드값 콤보화면 하드코딩, ex. 000 대한민국) - 주소 - 전화번호 - 이메일 - 거주구분 (코드값 콤보화면 하드코딩, 거주/비거주)		
고객게시판	RQ-ACOP-F-004	LNB/목록	고객은 사전정보입력 화면을 통해 개인정보이용 동의를 진행한다.								1. 기존에 저장되어 있는 정보가 있는가 확인후 저장되어 있는 정보 이후를 진행할 수 있는가? 2. 개인정도이용동의를 입력받아서 처리 할 수 있는가?		
고객게시판	RQ-ACOP-F-005	LNB/목록	고객은 사전정보입력 화면을 통해 금융거래사전정보를 확인 한다.								1. 기존에 저장되어 있는 정보가 있는가 확인후 저장되어 있는 정보 이후를 진행할 수 있는가? 2. 금융거래사전정보를 입력받아서 처리 할 수 있는가?		

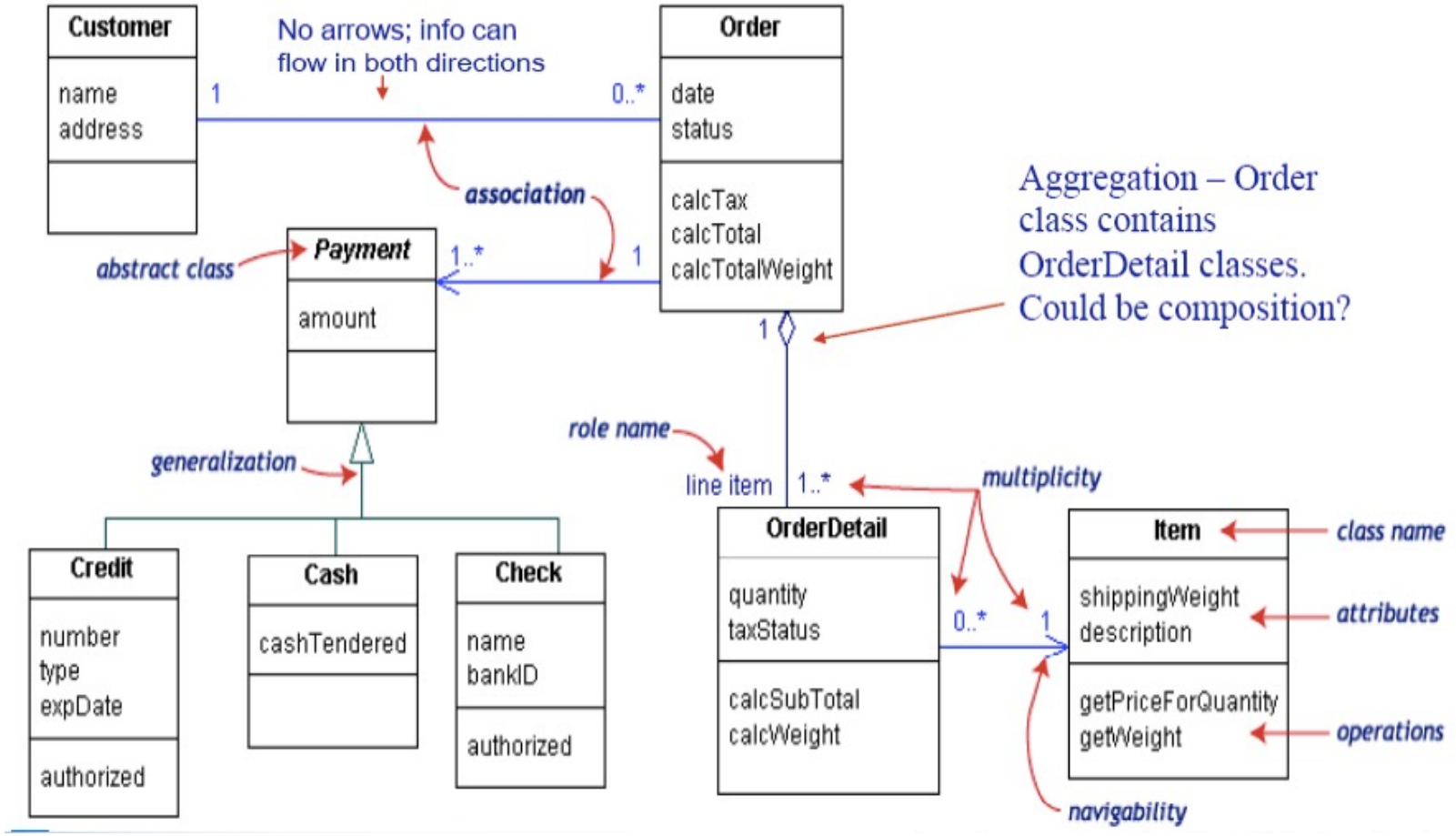
Tactical Design 결과물

- Sequence Diagram : B/C 객체간의 처리 순서를 기술



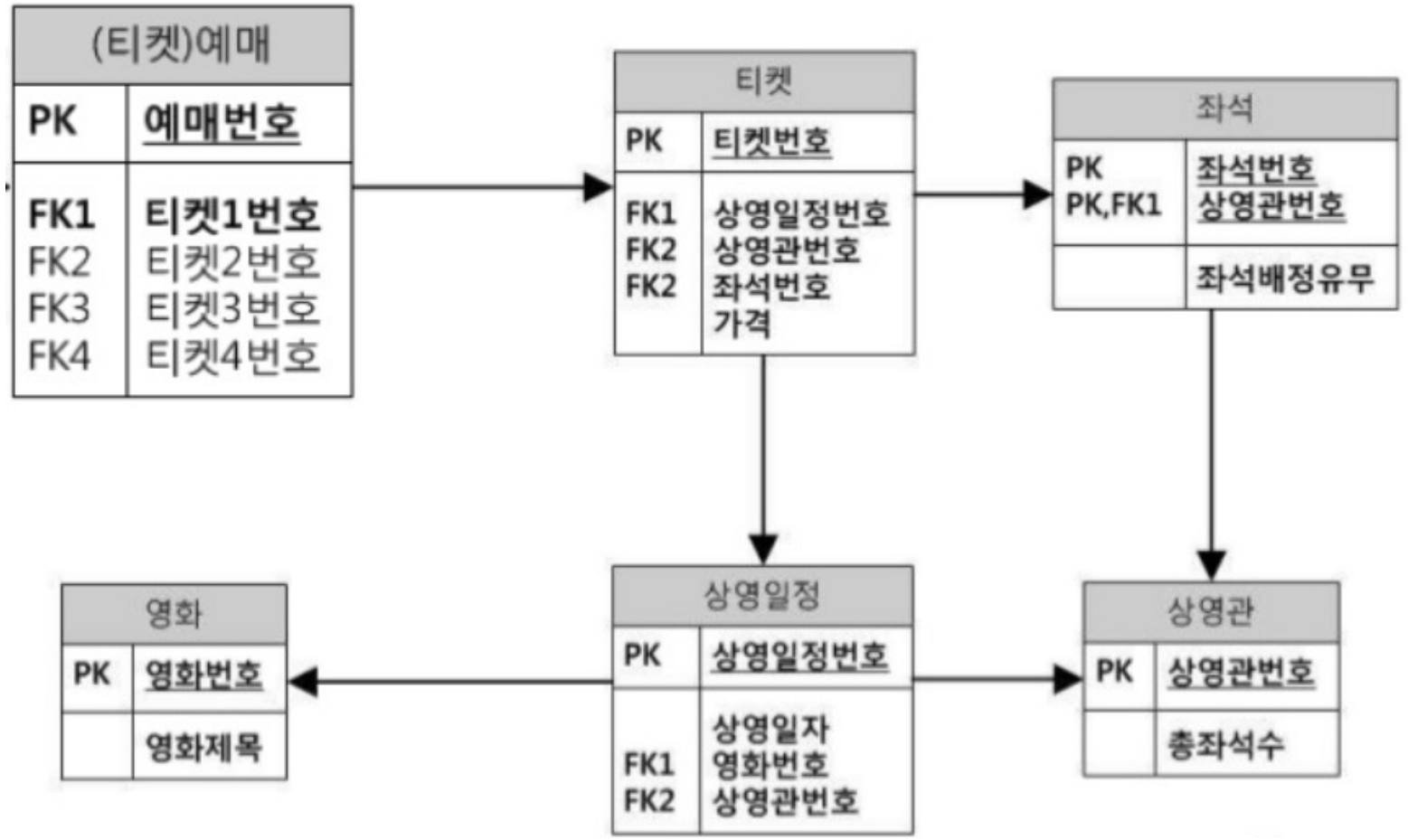
Tactical Design 결과물

- Class diagram : 서비스, 도메인 모델의 클래스 정의



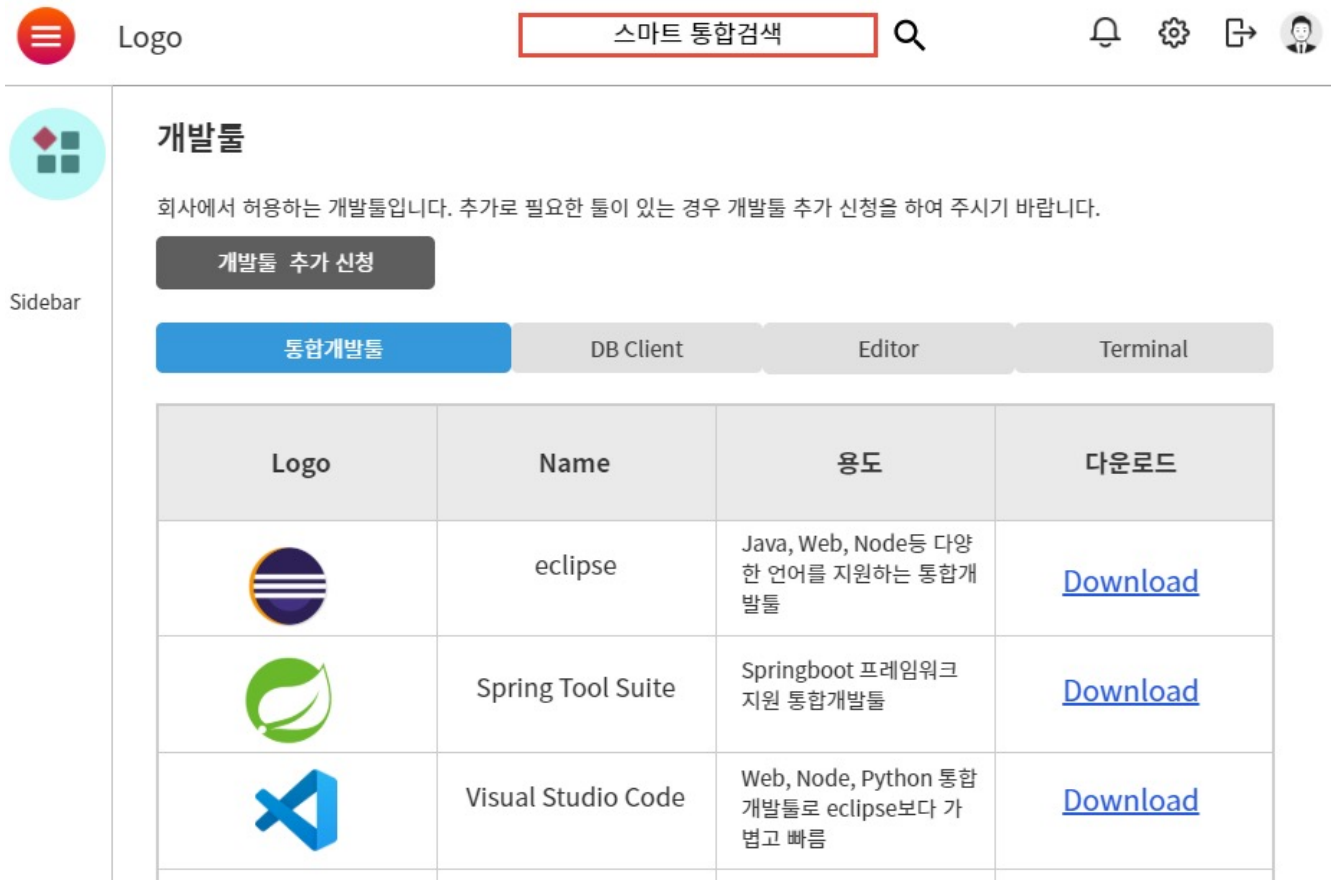
Tactical Design 결과물

- Data diagram : 데이터 구조 정의 (ERD와 동일)



Tactical Design 결과물

- Storyboard : 화면 설계서



Tactical Design 결과물

- API 설계서
- Message 설계서 (마이크로서비스 간 통신 방법을 기술)
- 마이크로서비스패턴 적용 설계서

결론

- DDD는 어플리케이션 설계에 대한 방법론

- DDD를 통해 더욱 효율적인 설계 및 개발 가능

참고 URL

- https://happycloud-lee.tistory.com/94
- https://docs.microsoft.com/ko-kr/archive/msdn-magazine/2009/february/best-practice-an-introduction-to-domain-driven-design

Q & A