#2 Attribute Driven Design (ADD)

- 고급소프트웨어공학 Team 4 -

31th May 2021

김상권, 이태영, 배재욱

KU KONKUK UNIVERSITY

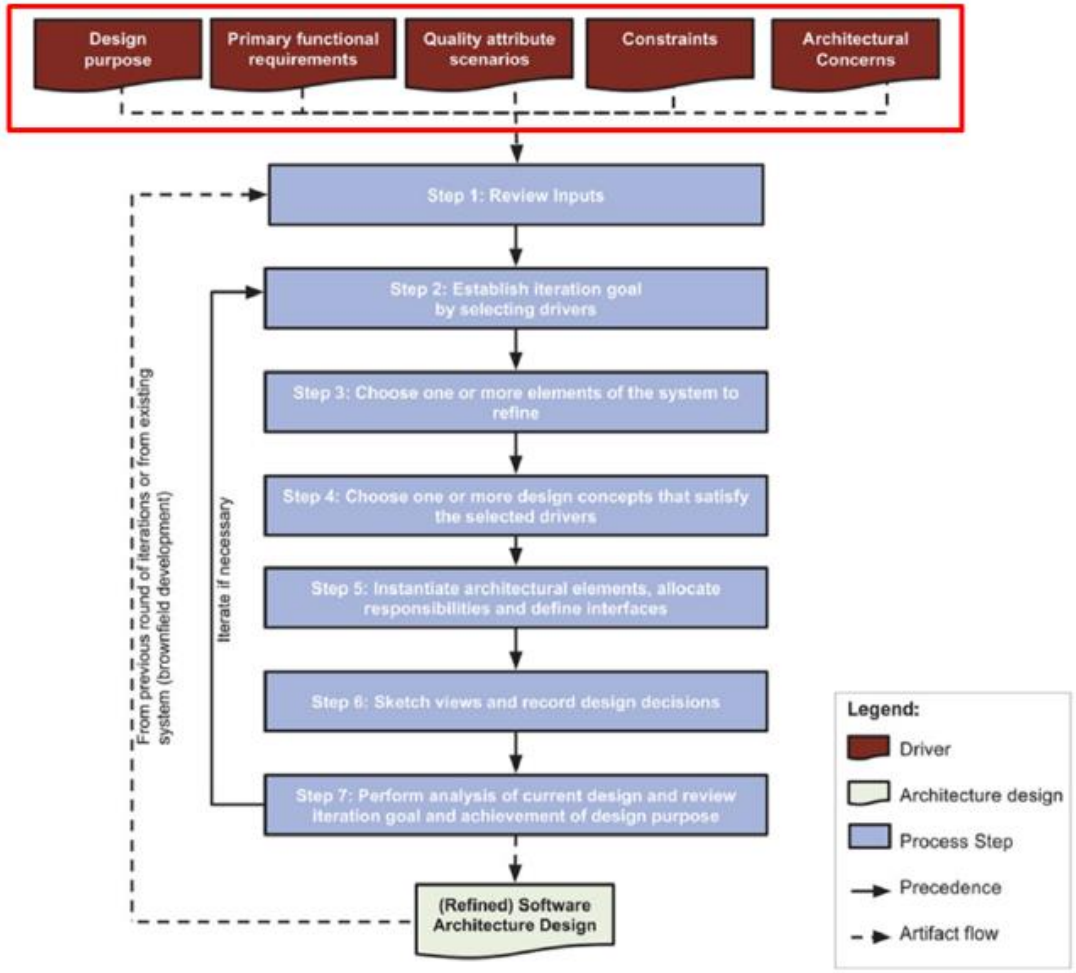
Attribute Driven Design (ADD)

- ADD introduction
- Design process
  - Iteration 1: Establishing an initial overall system structure
  - Iteration 2: Identifying structure to support primary functionality
  - Iteration 3: Refining previously created structures to fully address the remaining drivers

ADD introduction

ADD design process

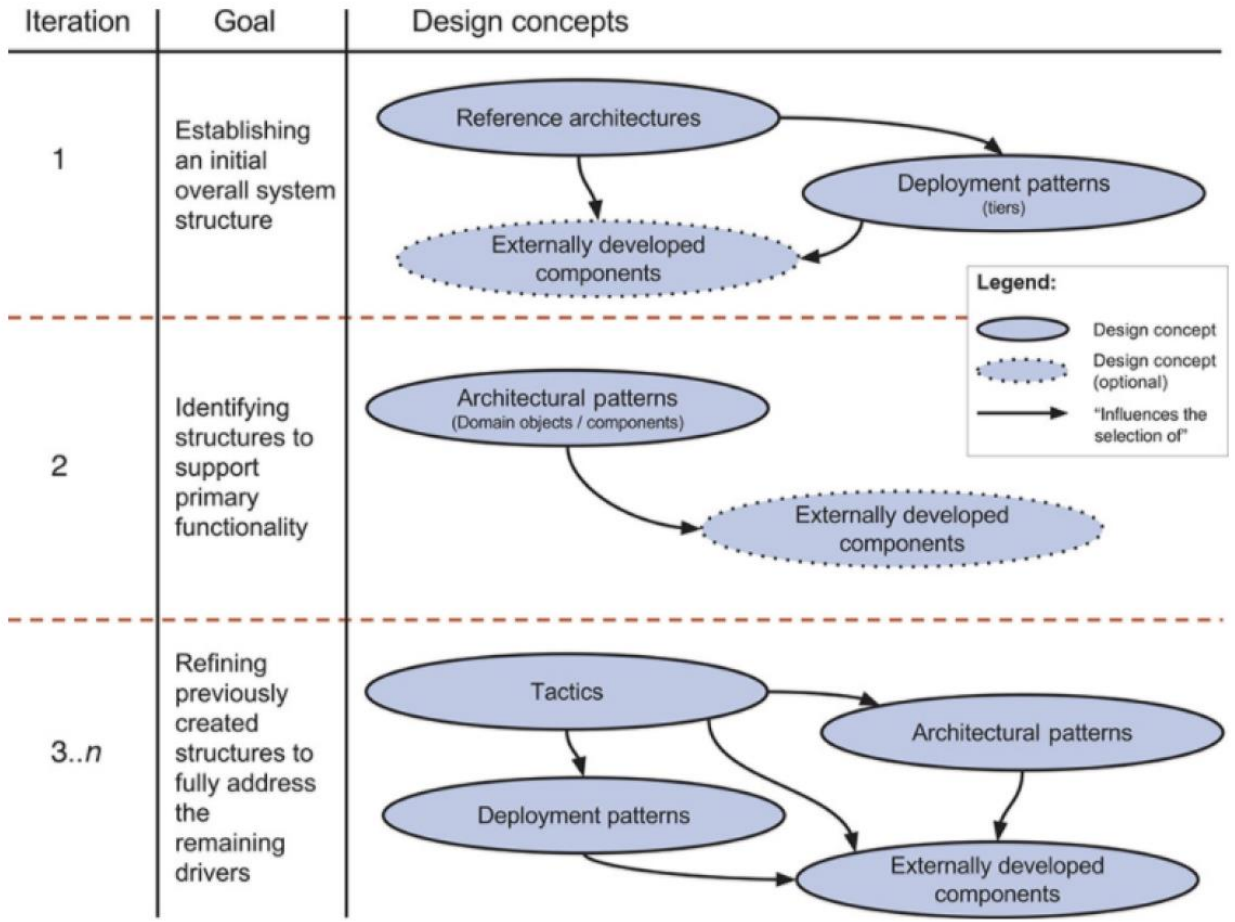
- **5가지 요소로 구성된 Architectural Driver 를 기반으로 requirements 를 architecture structure 로 translate 하는 과정**
  - Architectural Driver 는 input 으로 사용
    - Stakeholders 를 소집하여 Quality Attribute Workshop (QAW) 를 통해 찾음
  - 3 회 정도 iteration 진행
  - ADD 최종 산출물: Software Architecture Design



ADD design process

각 Iteration 별 Goal 및 Design concepts 결정

- Iteration 1
  - Goal: 초기 전반적인 시스템 구조 결정
  - 관련 Design concepts: Ref. architectures, deployment patterns, externally developed components
- Iteration 2
  - Goal: Primary functionality 를 만족하기 위한 구조 선정
  - 관련 Design concepts: Architectural patterns, externally developed components
- Iteration 3
  - Goal: 남은 drivers 요소를 고려하도록 기존 구조 정제
  - 관련 Design concepts: Tactics, architectural patterns, deployment patterns, externally developed components



ADD design process

Iteration 마다 해야 할 단계 별 프로세스

- *(**Step 1**: Review inputs)* ➔ not iteration
- **Step 2**: Establish iteration goal by selecting drivers
- **Step 3**: Choose one or more elements of the system to refine
- **Step 4**: Choose one or more design concepts that satisfy the selected drivers
- **Step 5**: Instantiate architectural elements, allocate responsibilities and define interfaces
- **Step 6**: Sketch views and record design decisions
- **Step 7**: Perform analysis of current design and review iteration goal and achievement of design purpose

ADD Design process
: iteration 1

Selected Architectural Drivers

Design Decisions and Location	Rationale
UC-1: Manage database	총 음료의 개수는 20 종류이다.
UC-2: Display information	한 자판기는 7 종류의 음료를 판매하며, 판매하지 않는 음료도 메뉴는 제공한다.
UC-3: Process tasks	사용자가 음료를 선택 후 결제하면 음료가 제공된다. 결제는 카드로 하고, 잔액이 부족한 경우 결제되지 않는다.
UC-4: Manage network	음료 재고가 부족하거나 자판기에서 판매하지 않는 음료를 구매하려는 경우 다른 자판기 재고 확인 후 위치를 안내한다. 이 때, 네트워크 상의 자판기에 broadcast msg를 통해 재고 확인을 요청하여 확인하고, 네트워크 msg를 통해 대상 자판기의 위치를 확인하여 안내한다.
UC-5: Identification	다른 자판기의 음료 구매에 대해 선결제를 할 수 있다. 이 때, 현재 자판기에서 결제 후 인증 코드를 발급하며 다른 자판기로 가서 인증코드를 입력하면 음료가 나온다.

Quality Attribute	Scenario	Associated Use Case
QA-1: Marketability	기존 코드 중 재활용 가능한 부분의 활용을 통헤 낮은 개발 비용/ 빠른 개발속도를 달성해야 한다.	ALL
QA-2: Usability	SW의 기능이 분명하고 간결하며 사용이 편리해야 한다.	UC-3, 4, 5
QA-3: Usability	모든 구매한 물품과 비용이 한 눈에 보일 수 있게 해야 한다.	UC-1, 2
QA-4: Performance	화면 사이의 연결이나 연동이 사용자로 하여금 불편함을 느끼지 않도록 0.5초 이내로 제한한다.	UC-3, 4, 5
QA-5: Availability	자판기 작동 시 일주일 정도는 이상 없이 동작해야 한다.	ALL
QA-6: Modifiability	음료 메뉴 추가나 삭제가 용이했으면 좋겠다.	UC-1

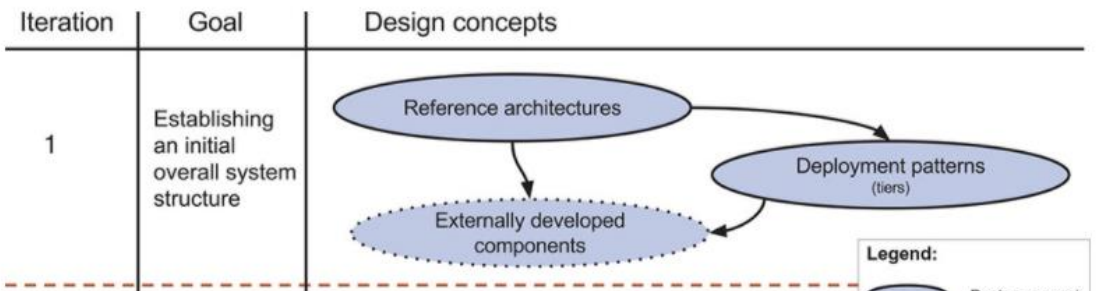
ID	Constraints
CON-1	각 자판기는 모두 네트워크에 연결되어 있고 네트워크 연결 정보는 미리 알고 있다(최대 10대).
CON-2	자판기의 판매 음료 종류는 사전에 결정된다.
CON-3	자판기 사이의 msg protocol은 사전에 결정된다.

ID	Concern
CRN-1	처음에 전반적인 system architecture 부터 설계한다.
CRN-2	개발 환경은 프로그램 개발자와 네트워크 개발자가 익숙한 것을 선택한다.
CRN-3	모든 Architectural Driver를 만족하는 Software Architecture 를 세운다.
CRN-4	사용자에게 적합한 UI/UX design 개발
CRN-5	사용자의 요구를 분석하여 그것들을 컴퓨터에 저장할 수 있는 데이터베이스의 구조에 맞게 변형한 후 특정 DBMS로 데이터베이스를 구현
CRN-6	웹 방화벽 구축, 침입탐지시스템 등 정보보호를 위한 관리적 기술적 물리적인 시스템 구축

Step 2: Establish iteration goal by selecting drivers

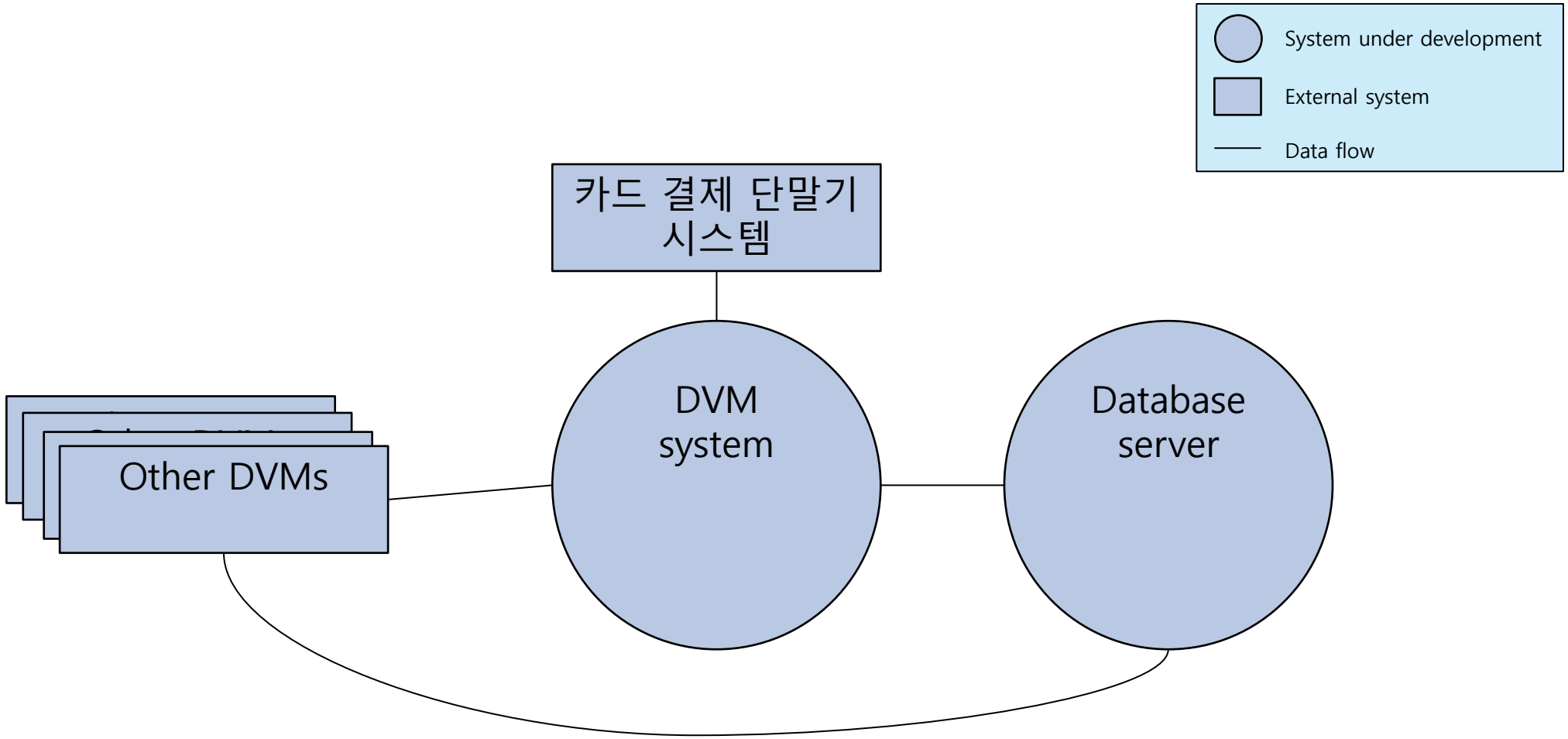
Iteration goal 선택

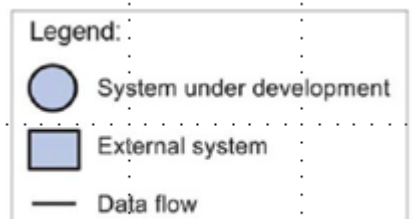
- 전체 Architectural Drivers 를 만족할 수 있는 초기 전반적인 시스템 구조 결정



Step 3: Choose one or more elements of the system to refine

- Iteration goal 인 초기 전반적인 시스템 구조 결정을 위해 DVM 시스템을 elements로 선택





Choosing design concepts

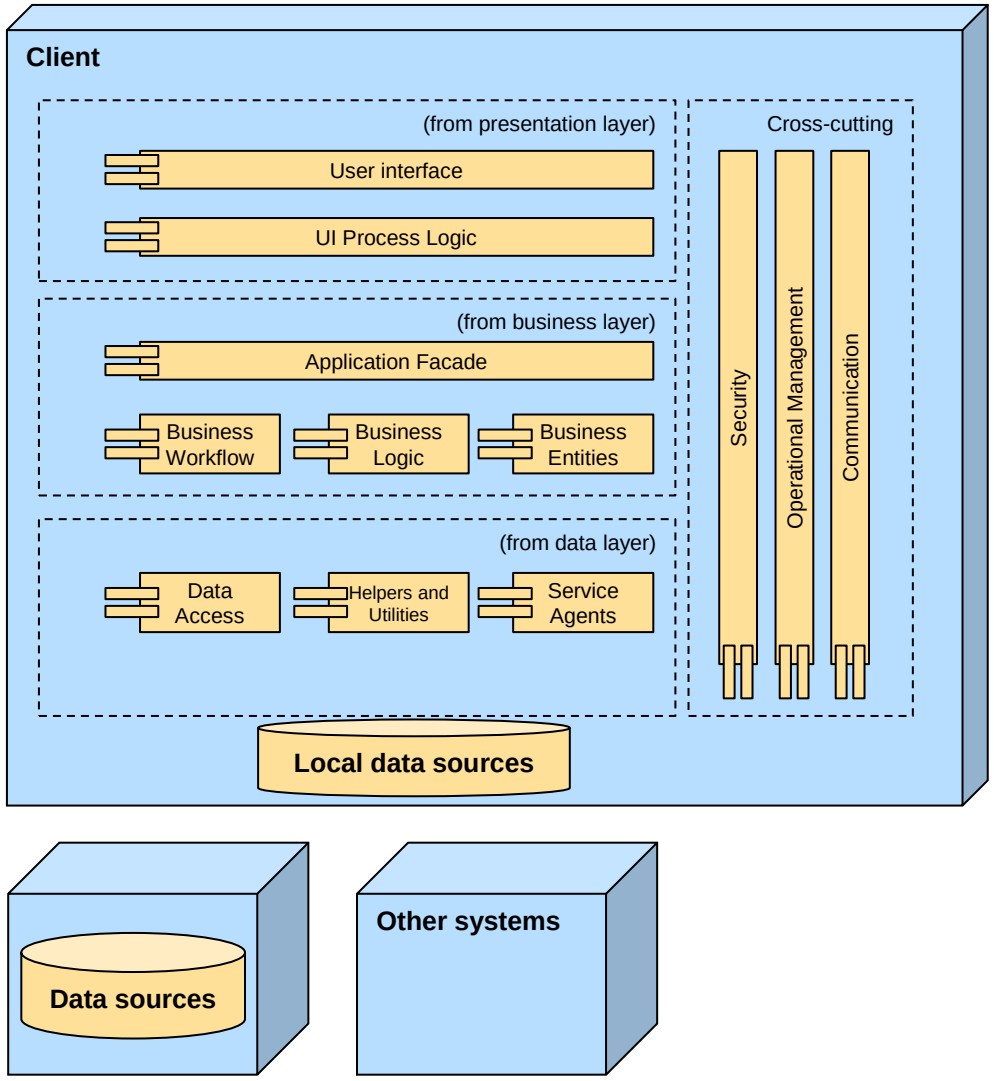
Design Decisions and Location	Rationale	
Logically structure the client part using the Rich Client Application reference architecture	DVM system 은 client part 역할이며, UC-2, UC-3, UC-4, UC-5 처럼 대부분 기능 요구사항을 해결할 수 있는 능력이 필요함. 따라서 수행 capability 가 큰 rich client application 을 선택	
	Discarded alternatives	web application, rich internet application을 고려하였으나, DVM 요구사항에는 인터넷이 필요하지 않기 때문에 선정하지 않음
Logically structure the server part of the system using the Service Application reference architecture	UI 제공이 필요하지 않고, UC-1, QA-6를 만족시키기 위해 데이터베이스 관리를 할 수 있는 application 필요	
	Discarded alternatives	No discarded alternatives

Design Decisions and Location	Rationale
UC-1: Manage database	총 음료의 개수는 20 종류이다.
UC-2: Display information	한 자판기는 7 종류의 음료를 판매하며, 판매하지 않는 음료도 메뉴는 제공한다.
UC-3: Process tasks	사용자가 음료를 선택 후 결제하면 음료가 제공된다. 결제는 카드로 하고, 잔액이 부족한 경우 결제되지 않는다.
UC-4: Manage network	음료 재고가 부족하거나 자판기에서 판매하지 않는 음료를 구매하려는 경우 다른 자판기 재고 확인 후 위치를 안내한다. 이 때, 네트워크 상의 자판기에 broadcast msg를 통해 재고 확인을 요청하여 확인하고, 네트워크 msg를 통해 대상 자판기의 위치를 확인하여 안내한다.
UC-5: Identification	다른 자판기의 음료 구매에 대해 선결제를 할 수 있다. 이 때, 현재 자판기에서 결제 후 인증 코드를 발급하며 다른 자판기로 가서 인증코드를 입력하면 음료가 나온다.

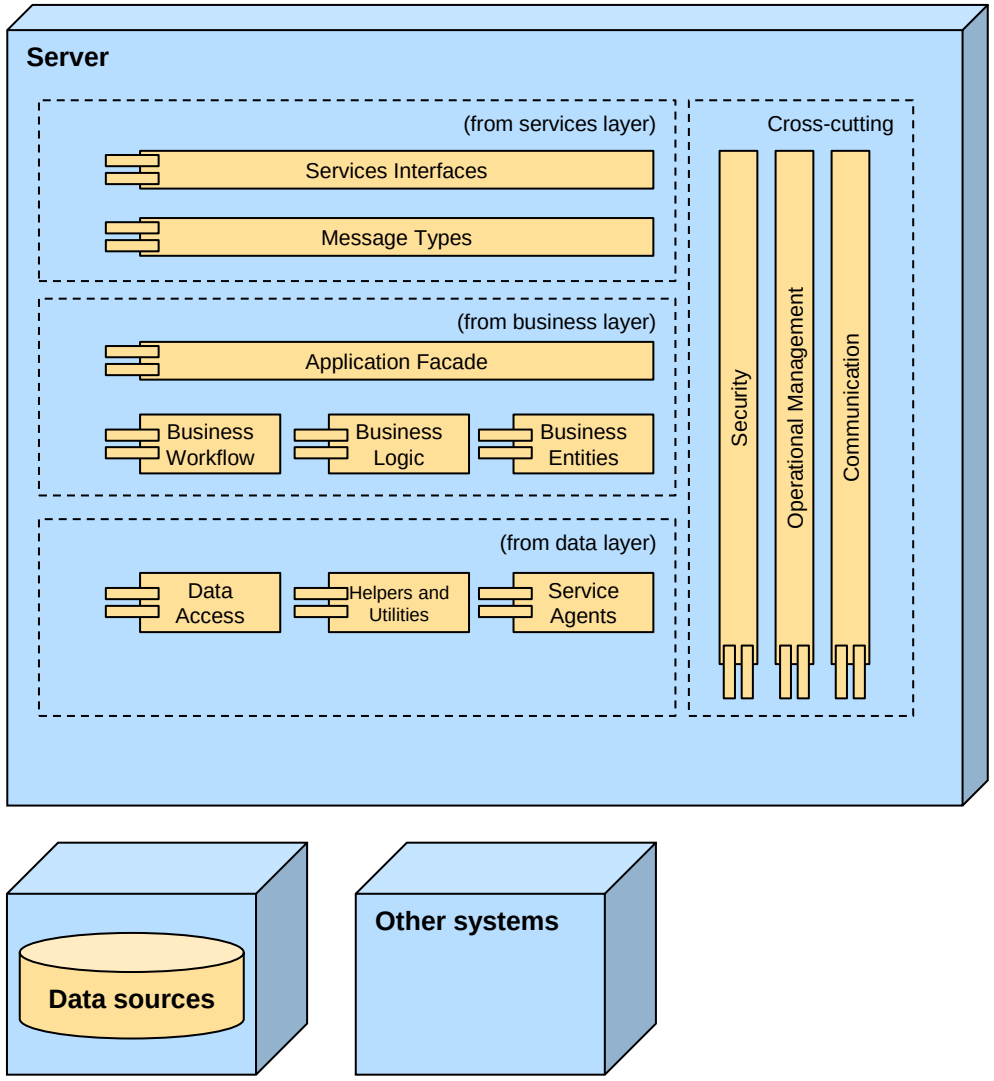
ID	Constraints
CON-1	각 자판기는 모두 네트워크에 연결되어 있고 네트워크 연결 정보는 미리 알고 있다(최대 10대).
CON-2	자판기의 판매 음료 종류는 사전에 결정된다.
CON-3	자판기 사이의 msg protocol은 사전에 결정된다.

Step 4: Choose One or More Design Concepts that Satisfy the Selected Derivers

Rich client application reference architecture



Service client application reference architecture



Step 4: Choose One or More Design Concepts that Satisfy the Selected Derivers

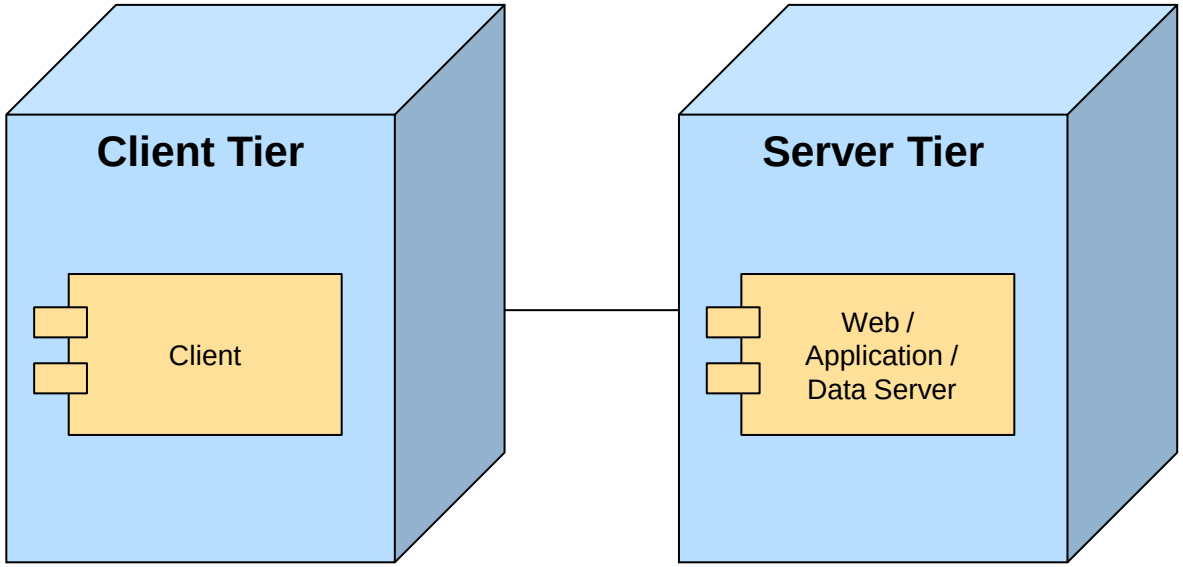
Choosing design concepts

- Physically structure the application using the **Two-tier Deployment pattern**

Design Decisions and Location	Rationale	
Physically structure the application using the Two-tier Deployment pattern	QA-1, 2을 고려하여 배포할 시스템의 아키텍쳐는 간단할 수록 좋고, 각각의 DVM은 Web application을 거치지 않고 직접 database server에 접근하므로 two-tier deployment pattern 이 적절하다고 판단	
	Discarded alternatives	No discarded alternatives

ID	Concern
CRN-1	처음에 전반적인 system architecture 부터 설계한다.
CRN-2	개발 환경은 프로그램 개발자와 네트워크 개발자가 익숙한 것을 선택한다.
CRN-3	모든 Architectural Driver를 만족하는 Software Architecture를 세운다.
CRN-4	사용자에게 적합한 UI/UX design 개발
CRN-5	사용자의 요구를 분석하여 그것들을 컴퓨터에 저장할 수 있는 데이터베이스의 구조에 맞게 변형한 후 특정 DBMS로 데이터베이스를 구현
CRN-6	웹 방화벽 구축, 침입탐지시스템 등 정보보호를 위한 관리적 기술적 물리적인 시스템 구축

Two-tier Deployment pattern

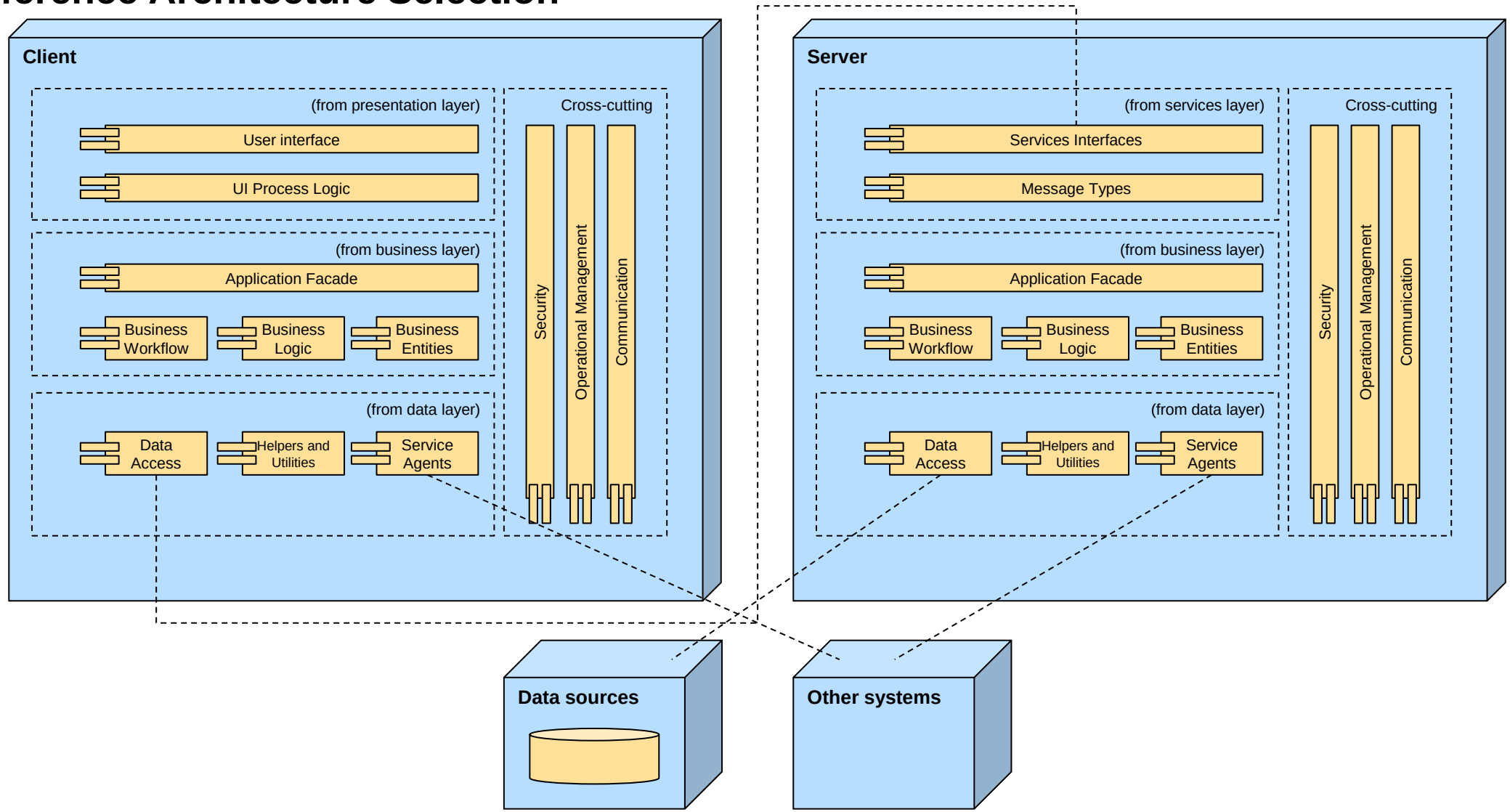


Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

Design Decision and Location	Rationale
Remove **local data sources** in the rich client application	It is believed that there is no need to store data locally, as the network connection is generally reliable. also, communication with the server is handled in the data layer. internal communication between components in the client is managed through local method calls and does not need particular support
Create a module dedicated to **accessing the database servers** of DVM stock in the data layer of the Service Application reference architecture.	The service agents component from the reference architecture is adapted to abstract the access to the database servers of DVM stock. this will play a critical role in the achievement of UC-4 and UC-5. as shown in CON-1, all vending machines are connected to the network, and you need to know the network connection information.

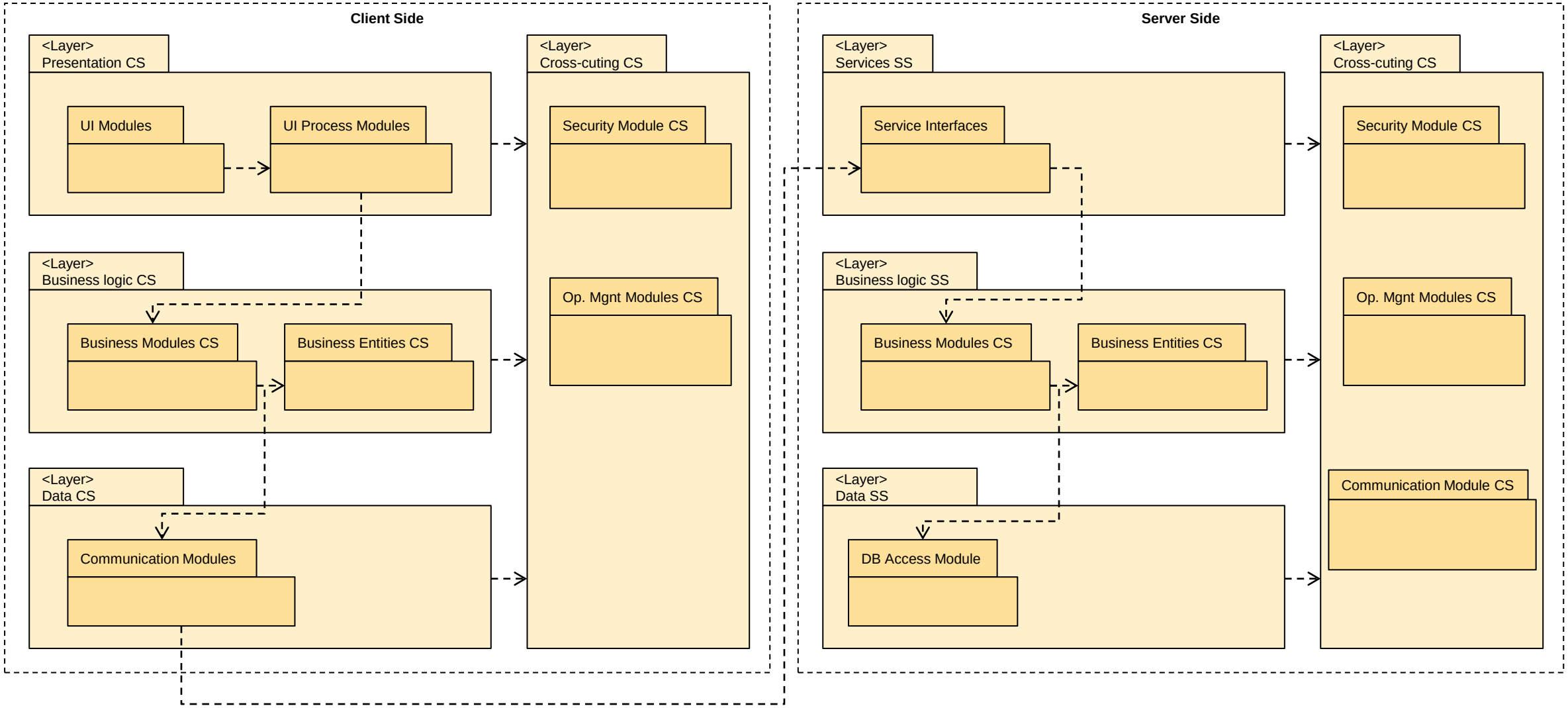
Step 6: Sketch views and record design decisions

Reference Architecture Selection



Step 6: Sketch views and record design decisions

Module View



Step 6: Sketch views and record design decisions

Element Descriptions

Responsibility

Element	Responsibility
Presentation Client side(CS)	이 계층에는 사용자 상호 작용 및 사용 사례 제어 흐름을 제어하는 모듈이 포함됩니다.
Business logic CS	이 계층에는 Client 측에서 내부적으로 실행할 수 있는 business logic operations을 수행하는 모듈이 포함되어 있습니다.
Data CS	이 계층에는 서버와의 통신을 담당하는 모듈이 포함되어 있습니다.
Cross-cutting CS	이 계층에는 보안, 로깅 및 I / O와 같은 여러 계층을 가로 지르는 기능이 있는 모듈이 포함됩니다. 이것은 드라이버 중 하나라도 CRN-6을 달성하는 데 도움이 됩니다.
UI modules	이 모듈은 사용자 인터페이스를 렌더링하고 사용자 입력을 받습니다.
UI process modules	이 모듈은 모든 시스템 사용 사례 (화면 간 탐색 포함)의 제어 흐름을 담당합니다.
Business modules CS	이 모듈은 로컬에서 수행 할 수 있는 비즈니스 운영을 구현하거나 서버 측에서 비즈니스 기능을 노출합니다.
Business entities CS	이 엔티티는 도메인 모델을 구성합니다. 그들은 서버 측보다 덜 상세 할 수 있습니다.
Communication modules CS	이 모듈은 서버 측에서 실행되는 애플리케이션에서 제공하는 서비스를 사용합니다.

Element	Responsibility
Services server side (SS)	이 계층에는 클라이언트가 사용하는 서비스를 노출하는 모듈이 포함되어 있습니다.
Business logic SS	이 계층에는 서버 측에서 처리해야하는 비즈니스 논리 작업을 수행하는 모듈이 포함되어 있습니다.
Data SS	이 계층에는 데이터 지속성 및 시간 서버와의 통신을 담당하는 모듈이 포함되어 있습니다. 이것은 QA-5을 달성하는 데 도움이 됩니다.
Cross-cutting SS	이러한 모듈에는 보안, 로깅 및 I / O와 같은 여러 계층에 걸친 기능이 있습니다.
Service Interfaces SS	이 모듈은 클라이언트가 사용하는 서비스를 노출합니다.
Business modules CS	이 모듈은 비즈니스 운영을 구현합니다.
Business entities CS	이 엔티티는 도메인 모델을 구성합니다.
DB access module	이 모듈은 관계형 데이터베이스에 대한 비즈니스 항목 (객체)의 지속성을 담당합니다. 그것은 객체 지향 관계형 매핑을 수행하고 지속성 세부 사항에서 응용 프로그램의 나머지 부분을 보호한다.
Time Server access module	이 모듈은 시간 서버와의 통신을 담당합니다. 다양한 유형의 시간 서버와의 통신을 지원하기 위해 시간 서버와의 작업을 격리하고 추상화합니다. (UC-4 참조).

Step 7: Perform analysis of current design and review iteration goal and achievement of design purpose

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During the Iteration
	UC-1		Server의 Reference architecture를 선정함에 있어 부분적으로 고려됨
	UC-2		Reference architecture를 rich client application으로 선정함에 있어 부분적으로 고려되었음
	UC-3		Reference architecture를 rich client application으로 선정함에 있어 부분적으로 고려되었음
	UC-4		Reference architecture를 rich client application으로 선정함에 있어 부분적으로 고려되었음
	UC-5		Reference architecture를 rich client application으로 선정함에 있어 부분적으로 고려되었음
	QA-1		Reference architecture와 Deploy pattern을 선정함에 있어 부분적으로 고려되었음
	QA-2		Reference architecture와 Deploy pattern을 선정함에 있어 부분적으로 고려되었음
QA-3			
QA-4			
QA-5			
	QA-6		Server의 Reference architecture를 선정함에 있어 부분적으로 고려됨
		CRN-1	해당 Concern을 고려하여 System의 architecture는 Green field에서 설계되었음.
CRN-2			
	CRN-3		해당 Concern을 고려하여 architecture 설계 과정에서 모든 Architectural Driver를 고려함
	CRN-4		해당 Concern을 고려하여 Client architecture 선정
	CRN-5		Server architecture에 해당 Concern 부분 반영
	CRN-6		해당 Concern을 반영하여 Server architecture 선정
	CON-1		부분 반영
CON-2			
CON-3			

ADD Design process
: iteration 2

Selected Architectural Drivers

Design Decisions and Location	Rationale
UC-1: Manage database	총 음료의 개수는 20 종류이다.
UC-2: Display information	한 자판기는 7 종류의 음료를 판매하며, 판매하지 않는 음료도 메뉴는 제공한다.
UC-3: Process tasks	사용자가 음료를 선택 후 결제하면 음료가 제공된다. 결제는 카드로 하고, 잔액이 부족한 경우 결제되지 않는다.
UC-4: Manage network	음료 재고가 부족하거나 자판기에서 판매하지 않는 음료를 구매하려는 경우 다른 자판기 재고 확인 후 위치를 안내한다. 이 때, 네트워크 상의 자판기에 broadcast msg를 통해 재고 확인을 요청하여 확인하고, 네트워크 msg를 통해 대상 자판기의 위치를 확인하여 안내한다.
UC-5: Identification	다른 자판기의 음료 구매에 대해 선결제를 할 수 있다. 이 때, 현재 자판기에서 결제 후 인증 코드를 발급하며 다른 자판기로 가서 인증코드를 입력하면 음료가 나온다.

Quality Attribute	Scenario	Associated Use Case
QA-1: Marketability	기존 코드 중 재활용 가능한 부분의 활용을 통헤 낮은 개발 비용/ 빠른 개발속도를 달성해야 한다.	ALL
QA-2: Usability	SW의 기능이 분명하고 간결하며 사용이 편리해야 한다.	UC-3, 4, 5
QA-3: Usability	모든 구매한 물품과 비용이 한 눈에 보일 수 있게 해야 한다.	UC-1, 2
QA-4: Performance	화면 사이의 연결이나 연동이 사용자로 하여금 불편함을 느끼지 않도록 0.5초 이내로 제한한다.	UC-3, 4, 5
QA-5: Availability	자판기 작동 시 일주일 정도는 이상 없이 동작해야 한다.	ALL
QA-6: Modifiability	음료 메뉴 추가나 삭제가 용이했으면 좋겠다.	UC-1

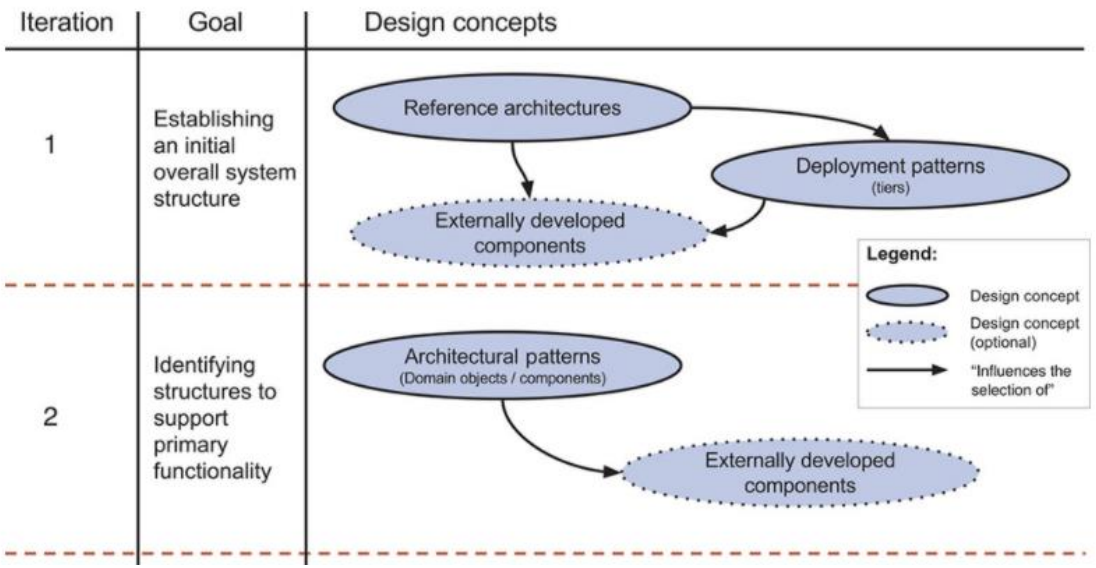
ID	Constraints
CON-1	각 자판기는 모두 네트워크에 연결되어 있고 네트워크 연결 정보는 미리 알고 있다(최대 10대).
CON-2	자판기의 판매 음료 종류는 사전에 결정된다.
CON-3	자판기 사이의 msg protocol은 사전에 결정된다.

ID	Concern
CRN-1	처음에 전반적인 system architecture 부터 설계한다.
CRN-2	개발 환경은 프로그램 개발자와 네트워크 개발자가 익숙한 것을 선택한다.
CRN-3	모든 Architectural Driver를 만족하는 Software Architecture 를 세운다.
CRN-4	사용자에게 적합한 UI/UX design 개발
CRN-5	사용자의 요구를 분석하여 그것들을 컴퓨터에 저장할 수 있는 데이터베이스의 구조에 맞게 변형한 후 특정 DBMS로 데이터베이스를 구현
CRN-6	웹 방화벽 구축, 침입탐지시스템 등 정보보호를 위한 관리적 기술적 물리적인 시스템 구축

Step 2: Establish iteration goal by selecting drivers

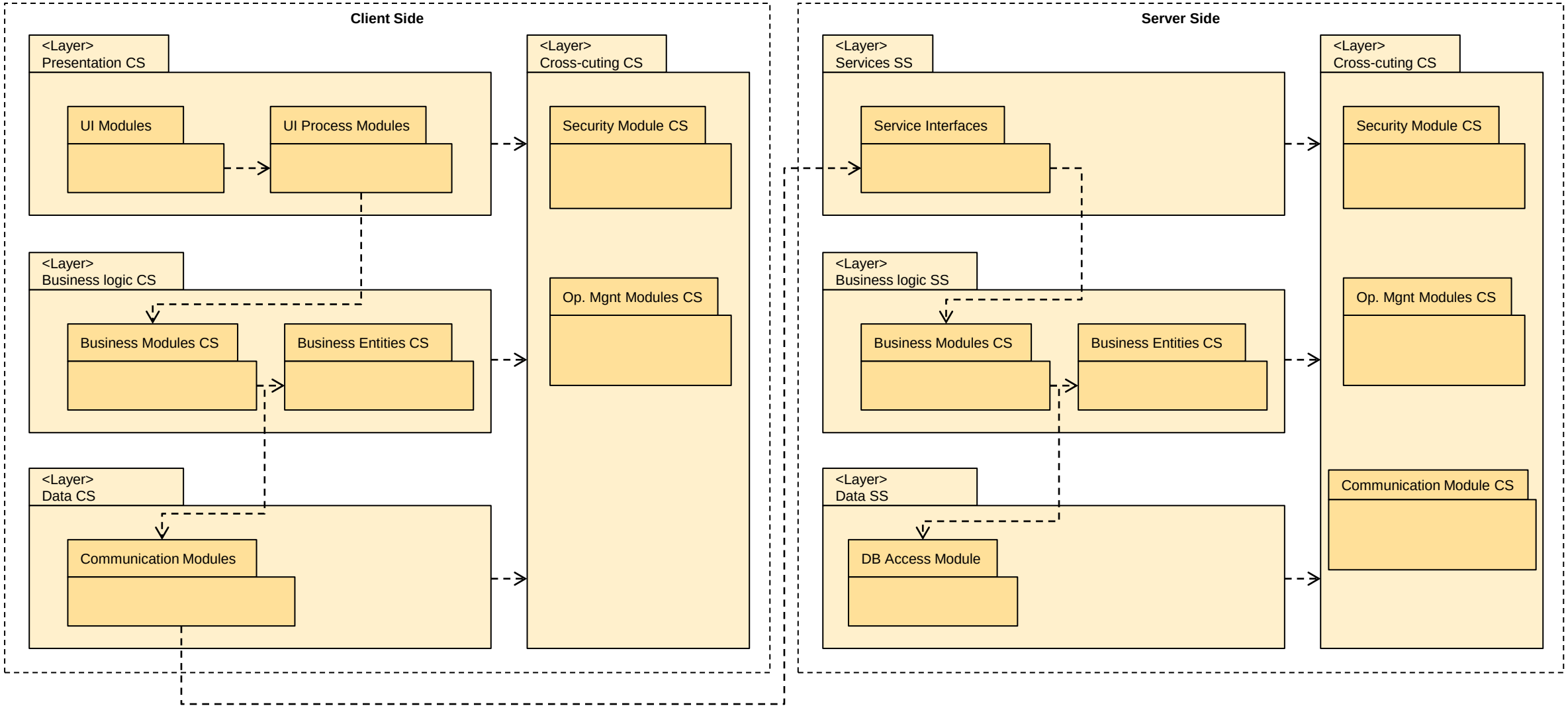
Iteration goal

- Identifying structures to support primary functionality
  - Use case 모두 completely addressed 되도록 하는 것을 목표!



Step 3: Choose one or more elements of the system to refine

- 세부 구조를 Use case 고려하기 위해 각 layers 및 modules를 element로 선택



Step 4: Choose One or More Design Concepts that Satisfy the Selected Drivers

- **Design concept: architectural design patterns**
- **Selection of design concepts**
  - Domain Model
  - Domain Objects
  - Decompose Domain Objects into general and specialized Components

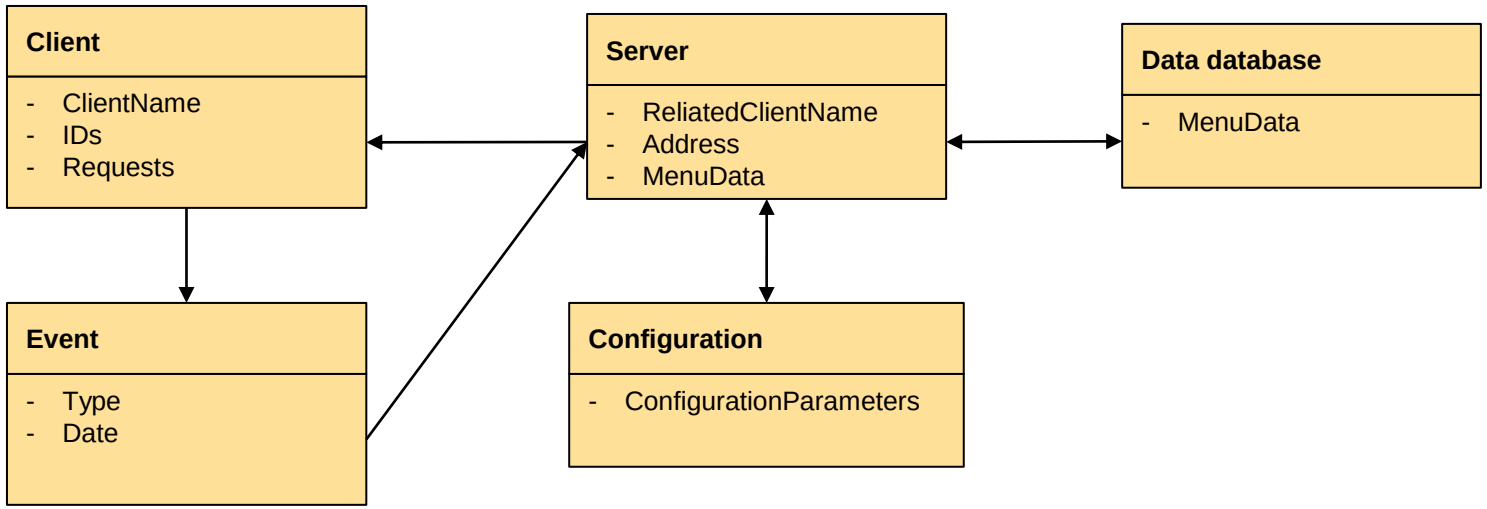
Design Decision and Location	Rationale
Create a **Domain model**	기능 세분화 전에 시스템에 대한 초기 domain model 을 생성하여 도메인의 주요 elements 를 식별해야 한다.
Identify **Domain Objects** that map to functional requirements	Use cases 를 빌딩 블록 하나에 캡슐화하기 위해 domain objects 를 결정
Decompose Domain Objects into general and specialized Components	생성한 Domain objects를 각각 계층 내에 있는 세분화된 elements 로 분해가 필요하다.

Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

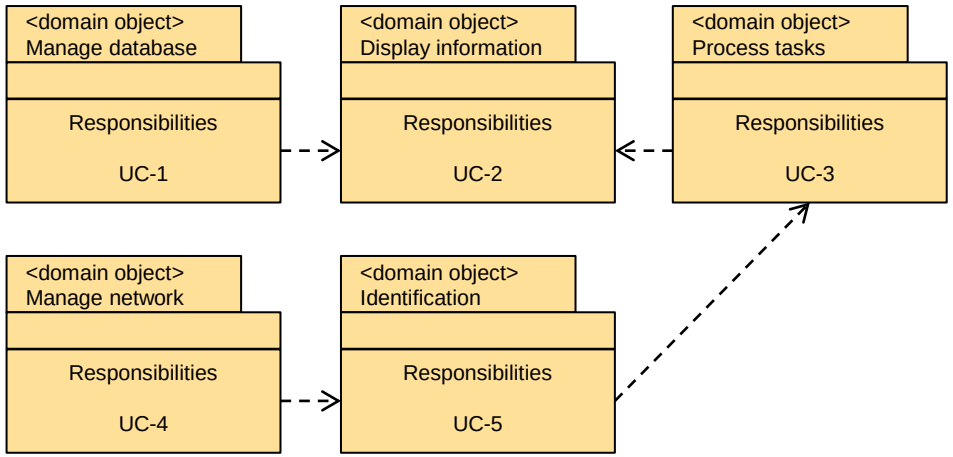
Design Decision and Location	Rationale
Create a **Domain model**	Primary use cases와 관련된 초기 domain model 을 생성하여 디자인 단계를 가속화 한다.
Identify **Domain Objects** that map to functional requirements	도메인 개체의 초기 식별은 시스템의 use cases 를 분석하여 만들 수 있습니다.
Decompose Domain Objects into general and specialized Components by layer-specific modules with an explicit interface	모든 Primary use cases에 대해 앞서 식별해둔 domain objects 를 세분화하고 연결하기 위해 layer-specific modules 을 식별한다.

Step 6: Sketch views and record design decisions

Initial domain model

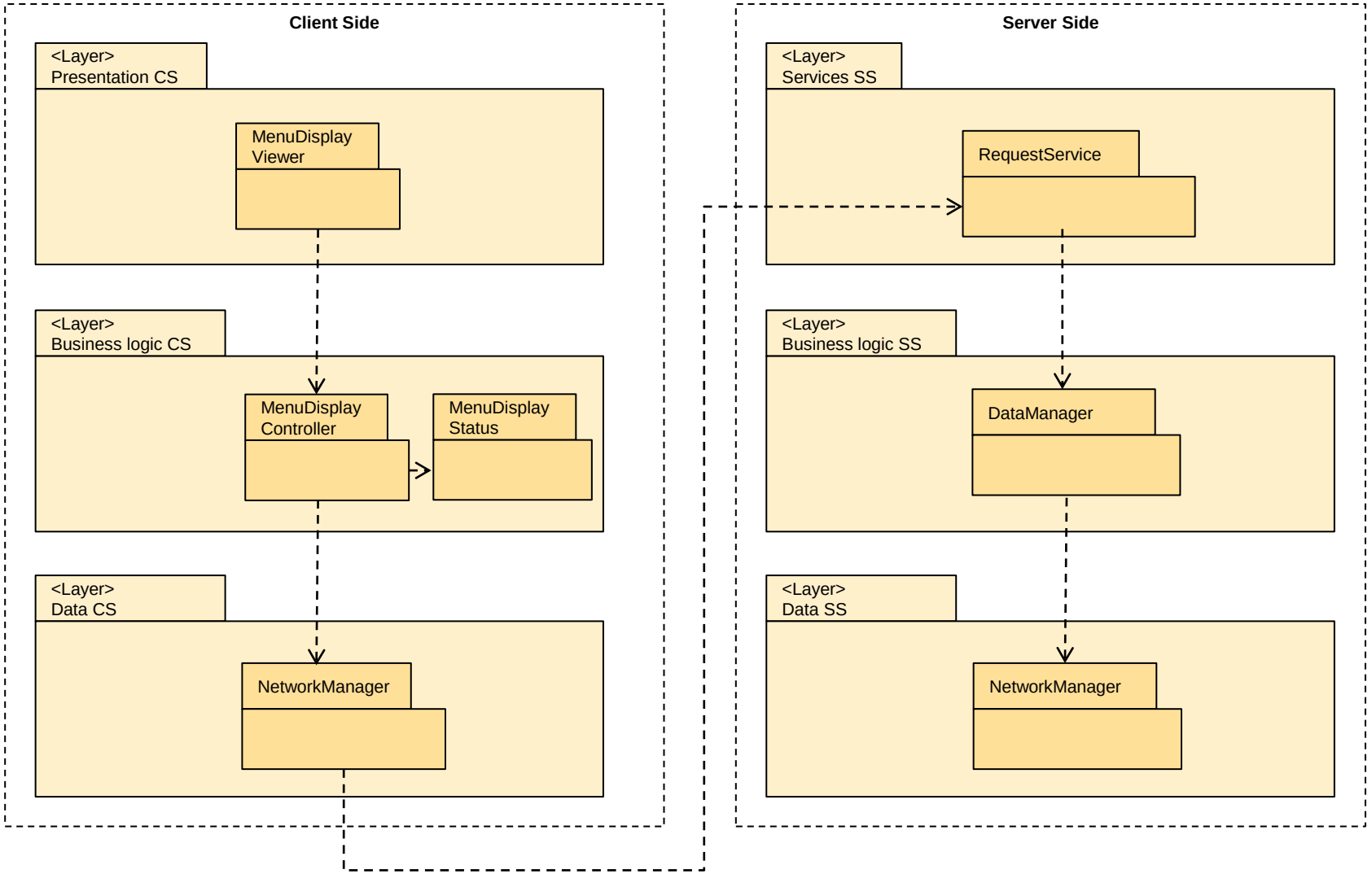


Domain Objects Associated with Use Cases



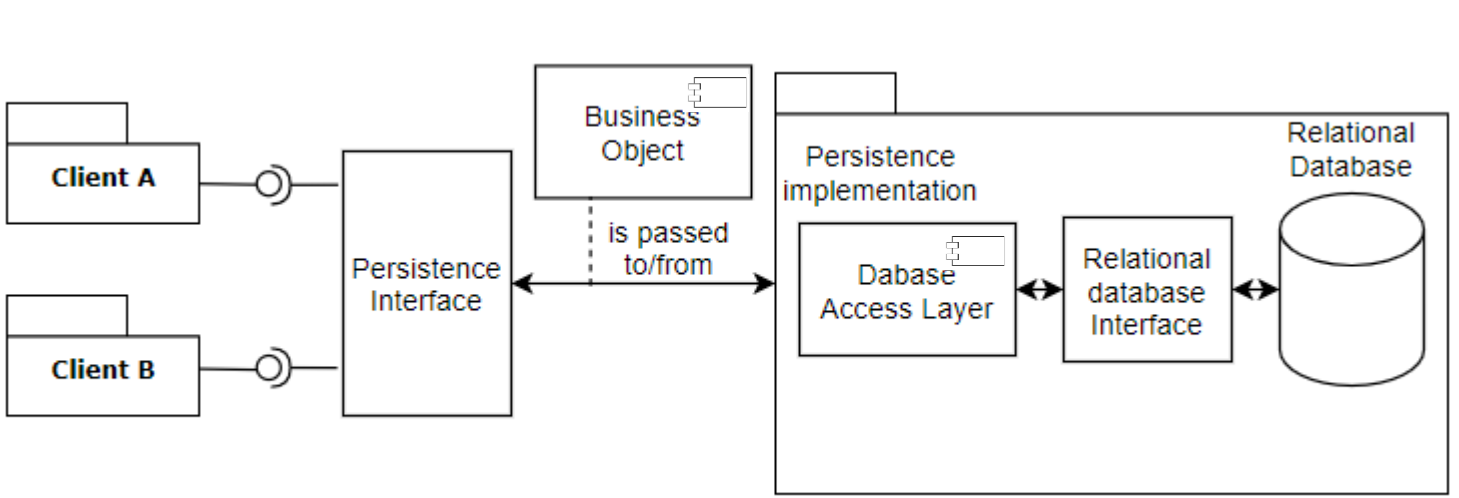
Step 6: Sketch views and record design decisions

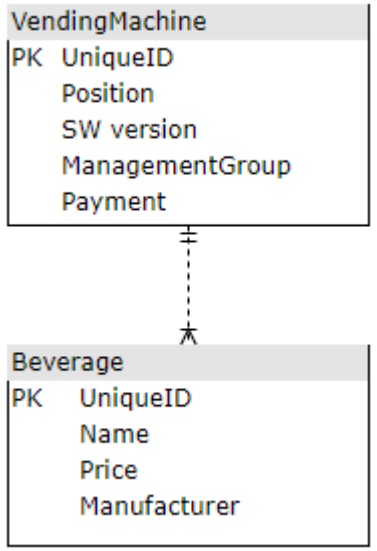
Module view



Sequence Diagram for UC-1: Manage database

- 총 음료의 개수는 20 종류이다.





Step 6: Sketch views and record design decisions

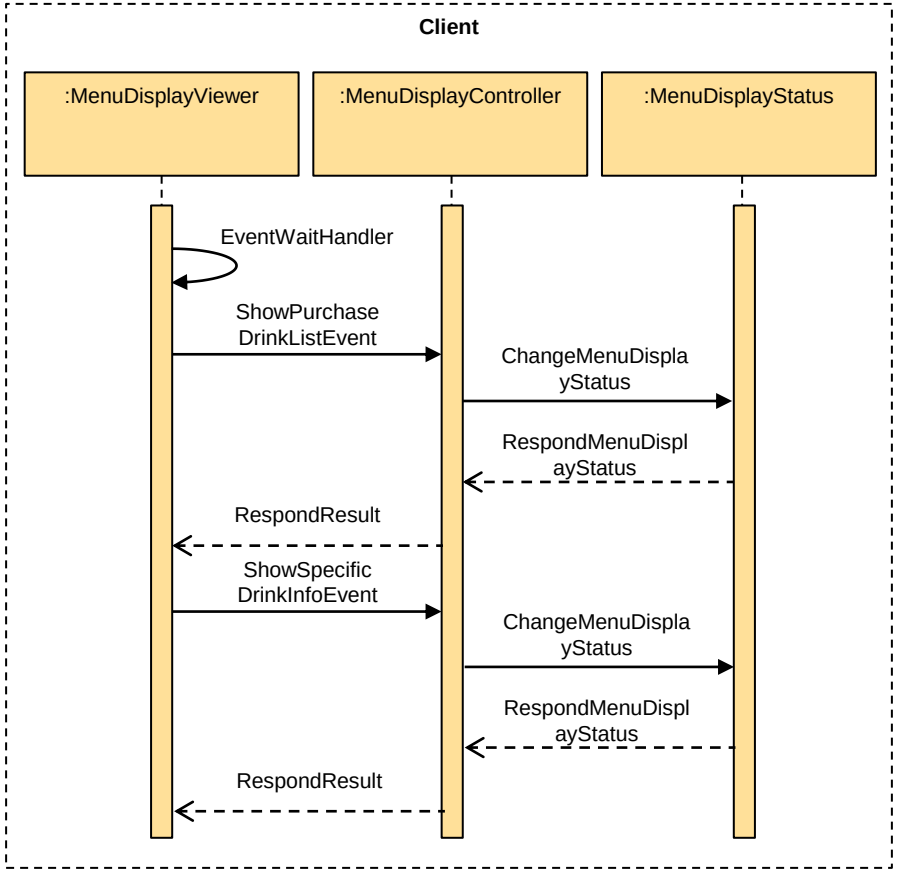
Sequence Diagram for UC-1: Manage database

- 총 음료의 개수는 20 종류이다.

<domain object>
Manage database

Responsibilities

UC-1



Interfaces from UC-1 Sequence Diagram

Method Name	Description
	Element: MenuDisplayViewer
EventWaitHandler	사용자가 이벤트를 발생시키는 것을 기다림
ShowPurchaseDrinkListEvent	구매할 수 있는 모든 음료리스트 보기 이벤트 발생
ShowSpecificDrinkInfoEvent	판매 음료 중 세부 정보 보기
	Element: MenuDisplayController
ChangeMenuDisplayStatus	내부 State machine 상태를 모든 음료 리스트 보기로 변경
RespondResult	사용자에게 모든 음료리스트 보기
	Element: MenuDisplayStatus
RespondMenuDisplayStatus	내부 State machine 상태 응답

Step 6: Sketch views and record design decisions

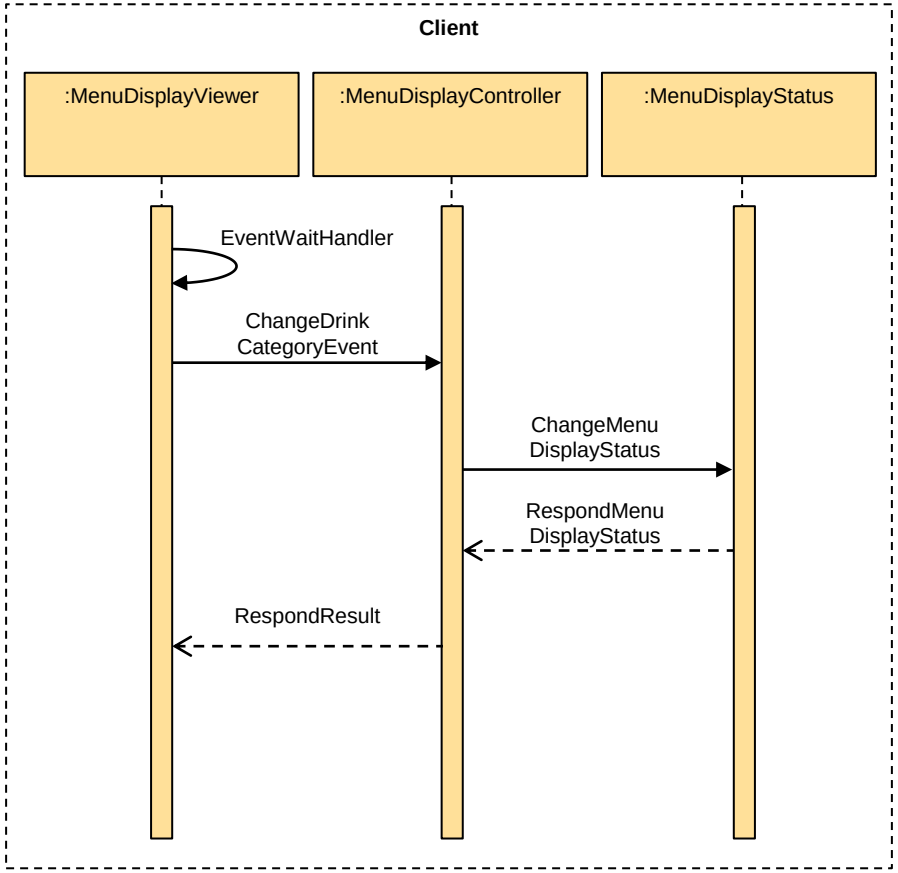
<domain object>
Display information

Responsibilities

UC-2

Sequence Diagram for UC-2: Display information

- 한 자판기는 7종류의 음료 판매
- 판매하지 않는 음료도 메뉴는 제공

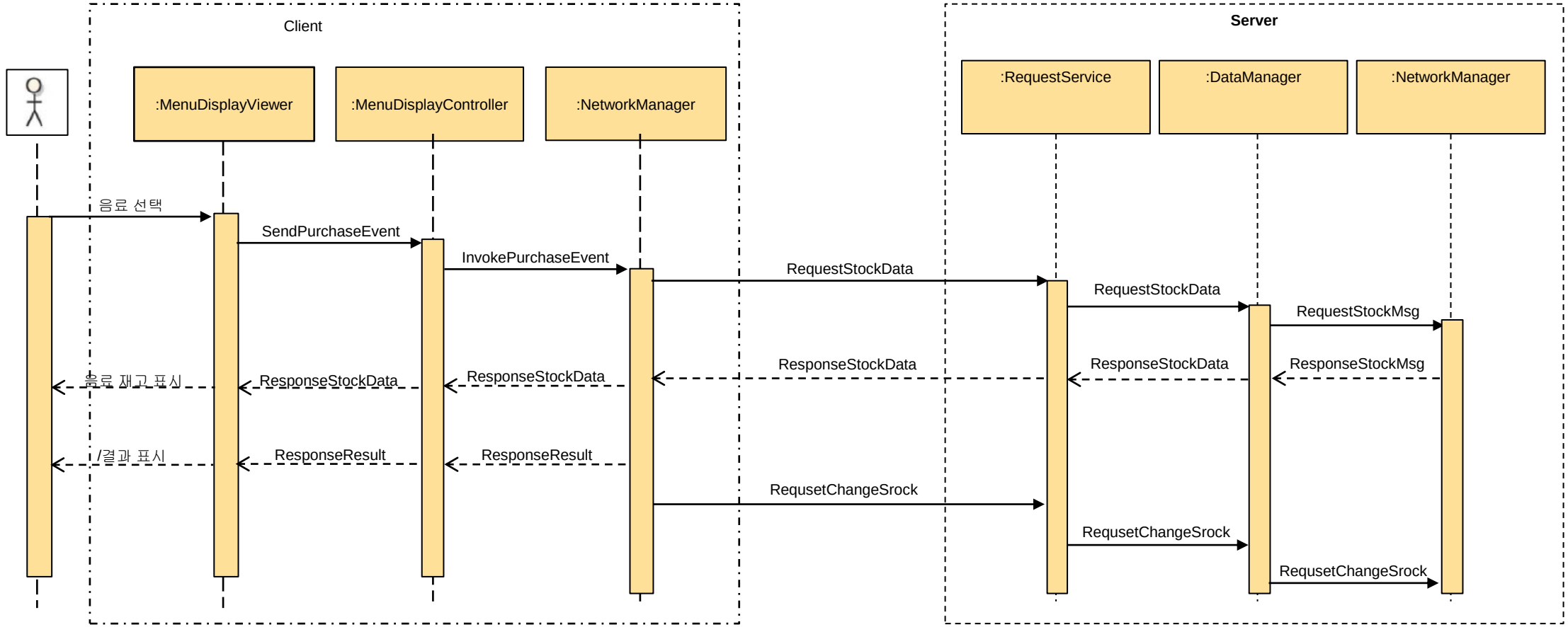


Interfaces from UC-2 Sequence Diagram

Method Name	Description
Element: MenuDisplayViewer	
EventWaitHandler	사용자가 이벤트를 발생시키는 것을 기다림
ChangeDrinkCategoryEvent	현재 자판기 판매하는(7 종류)/ 판매하지 않는, 이온/탄산 등 category 분류 이벤트 발생
Element: MenuDisplayController	
ChangeMenuDisplayStatus	Category 분류에 따른 내부 State machine 상태 변경
RespondResult	사용자에게 Category에 따른 음료리스트 보기
Element: MenuDisplayStatus	
RespondMenuDisplayStatus	내부 State machine 상태 응답

Sequence Diagram for UC-3: Process tasks

- 사용자가 음료를 선택 후 결제하면 음료가 제공된다.
- 결제는 카드로 하고, 잔액이 부족한 경우 결제되지 않는다.

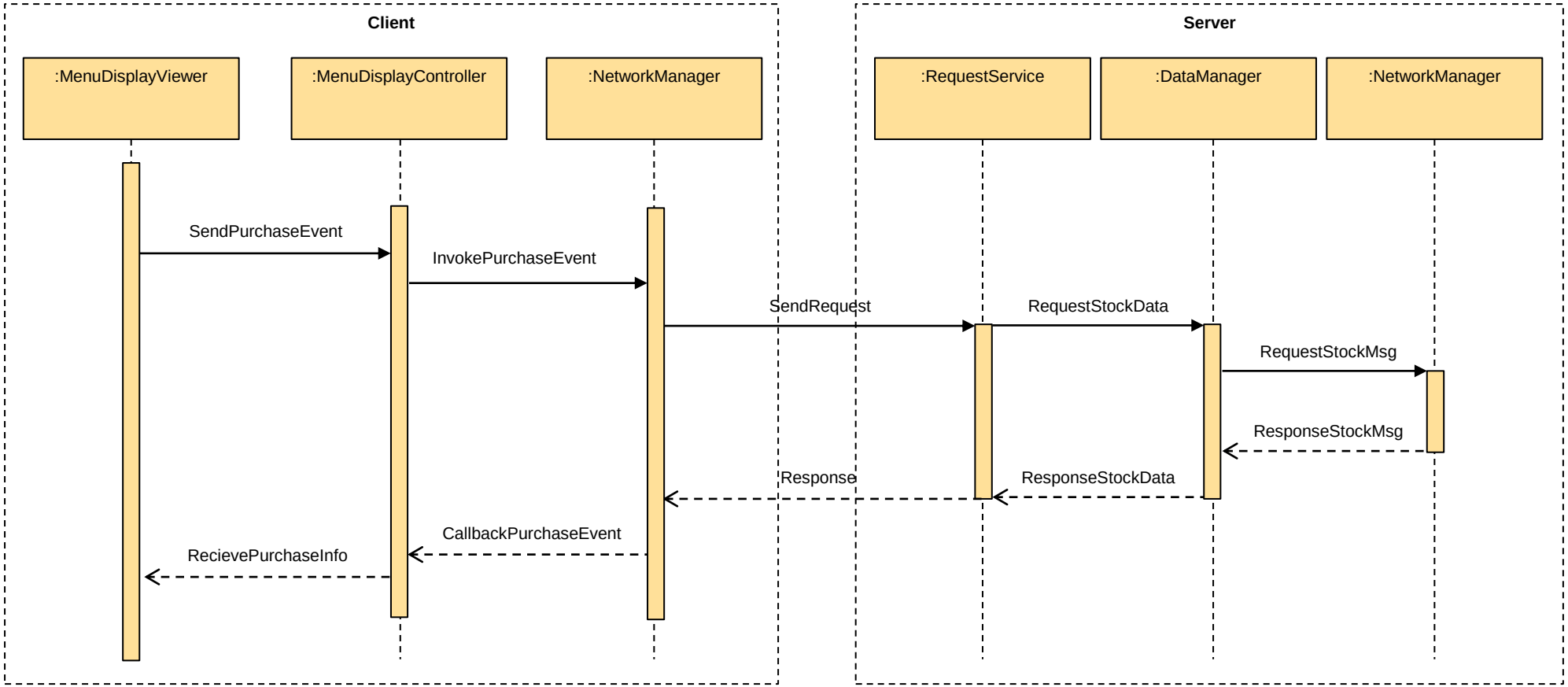


Interfaces from UC-3 Sequence Diagram

Method Name	Description
Element: MenuDisplayViewer	
SendPurchaseEvent	사용자 구매 요청 이벤트 발생
음료 재고 표시	음료 재고 표시
결과 표시	음료 구매에 대한 처리 결과 응답
Element: MenuDisplayController	
InvokePurchaseEvent	구매 요청 이벤트 호출
ResponseStockData	음료 재고에 대한 처리 결과 응답
ResponseResult	음료 구매에 대한 처리 결과 응답
Element: NetworkManager	
RequestStockData	서버에 음료 재고 정보 요청
ResponseStockData	서버에서 음료 재고 정보를 받아 응답
ResponseResult	음료 구매에 대한 처리 결과를 응답
RequsetChangeSrock	구매로 인한 재고 변동사항애 대한 DB 수정 요청
Element: RequestService	
RequestStockData	음료 재고 정보 요청
ResponseStockData	음료 재고 정보를 자판기에 응답
RequsetChangeSrock	구매로 인한 재고 변동사항을 DB로 전달
Element: DataManager	
RequestStockMsg	DB에 음료 재고 정보 요청
ResponseStockData	음료 재고 정보를 받아 응답
RequsetChangeSrock	구매로 인한 재고 변동사항애 대한 DB 수정 요청
Element: NetworkManager	
ResponseStockMsg	DB에서 음료 재고 정보를 받아 응답

Sequence Diagram for UC-4: Manage network

- 음료 재고가 부족하거나 자판기에서 판매하지 않는 음료를 구매하려는 경우 다른 자판기 재교 확인 후 위치를 안내한다. 이 때, 네트워크 상의 자판기에 broadcast msg를 통해 재고 확인을 요청하여 확인하고, 네트워크 msg를 통해 대상 자판기의 위치를 확인하여 안내한다.

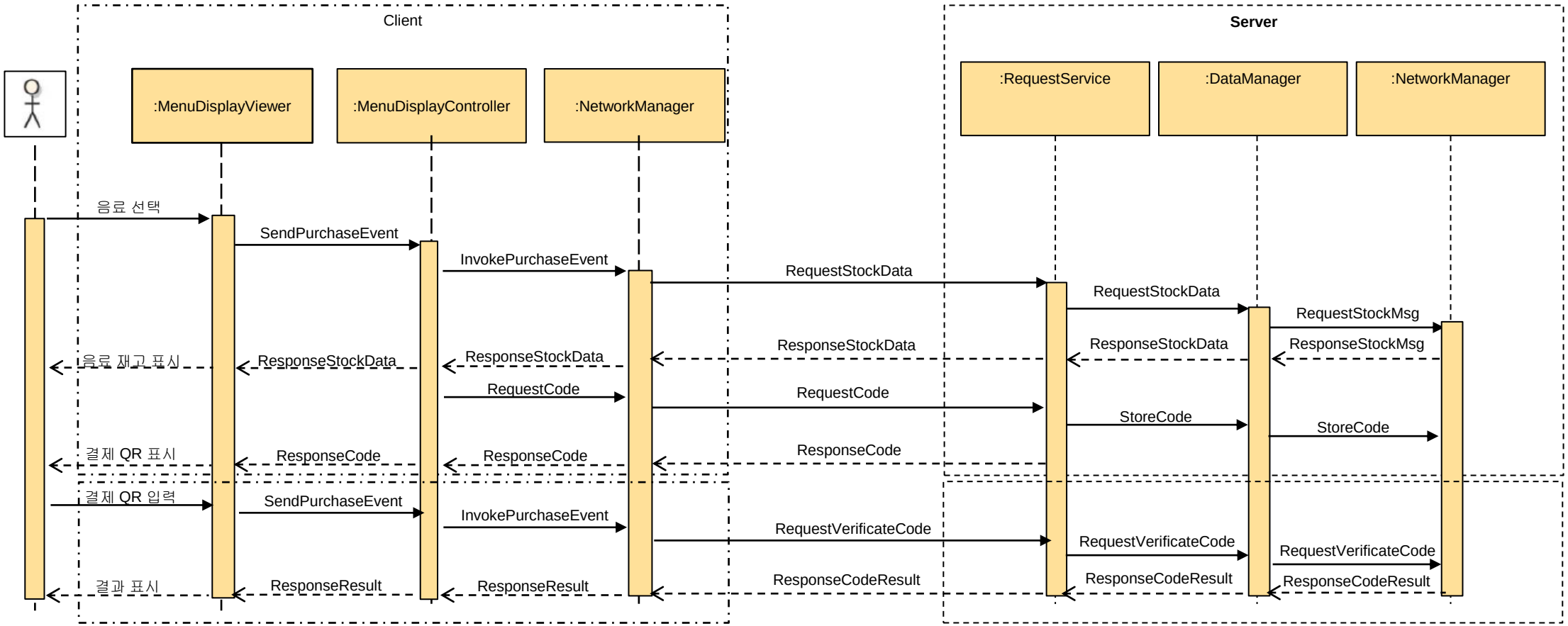


Interfaces from UC-4 Sequence Diagram

Method Name	Description
Element: MenuDisplayViewer	
SendPurchaseEvent	사용자 구매 요청 이벤트 발생
Element: MenuDisplayController	
InvokePurchaseEvent	구매 요청 이벤트 호출
RecievePurchaseInfo	사용자 구매 정보 받기
Element: NetworkManager	
SendRequest	구매한 음료 정보(수량, 비용 등) 요청
CallbackPurchaseEvent	구매 요청 이벤트 호출에 따른 콜백 함수 실행
Element: RequestService	
RequestStockData	구매 음료 재고 상태 송신
Response	요청에 따른 결과 응답
Element: DataManager	
RequestStockMsg	구매 음료 재고 상태 메세지 송신
ResponseStockData	구매 음료 재고 상태 수신
Element: NetworkManager	
ResponseStockMsg	구매 음료 재고 상태 메세지 수신

Sequence Diagram for UC-5

- 다른 자판기의 음료 구매에 대해 선결제를 할 수 있다. 이 때, 현재 자판기에서 결제 후 인증 코드를 발급하며 다른 자판기로 가서 인증코드를 입력하면 음료가 나온다.



Interfaces from UC-5 Sequence Diagram

Method Name	Description
Element: MenuDisplayViewer	
SendPurchaseEvent	사용자 구매 요청 이벤트 발생
음료 재고 표시	음료 재고 표시
결재 QR 표시	결제 QR을 사용자에게 표시
SendPurchaseEvent	사용자 구매 요청 이벤트 발생
결과 표시	음료 구매에 대한 처리 결과 응답
Element: MenuDisplayController	
InvokePurchaseEvent	구매 요청 이벤트 호출
ResponseStockData	음료 재고에 대한 처리 결과 응답
RequestCode	결제 코드 요청
ResponseCode	결제 코드 응답
InvokePurchaseEvent	구매 요청 이벤트 호출
ResponseResult	음료 구매에 대한 처리 결과 응답
Element: NetworkManager	
RequestStockData	서버에 음료 재고 정보 요청
ResponseStockData	서버에서 음료 재고 정보를 받아 응답
RequestCode	결제 코드 요청
ResponseCode	서버에서 결제 코드를 받아 전달
RequestVerificateCode	QR코드에 대한 무결성 확인 및 구매에 따른 재고 변동 전달
ResponseResult	음료 구매에 대한 처리 결과 응답

Method Name	Description
Element: RequestService	
RequestStockData	음료 재고 정보 요청
ResponseStockData	음료 재고 정보를 자판기에 응답
StoreCode	결제 코드를 DB에 저장
ResponseCode	결제 코드를 클라이언트에 전달
RequestVerificateCode	결제코드 무결성 확인 및 구매에 따른 재고 변동 전달
ResponseCodeResult	결제 코드 검사 결과 응답
Element: DataManager	
RequestStockMsg	DB에 음료 재고 정보 요청
ResponseStockData	음료 재고 정보를 받아 응답
StoreCode	결제 코드를 DB에 저장
RequestVerificateCode	결제코드 무결성 확인 및 구매에 따른 재고 변동 전달
ResponseCodeResult	결제 코드 검사 결과 응답
Element: NetworkManager	
ResponseStockMsg	DB에서 음료 재고 정보를 받아 응답
ResponseCodeResult	결제 코드 검사 결과 응답

Step 7: Perform analysis of current design and review iteration goal and achievement of design purpose

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During the Iteration
		UC-1	Reference architecture 를 sequence diagram 을 통해 고려 완료함
		UC-2	Reference architecture 를 sequence diagram 을 통해 고려 완료함
		UC-3	Reference architecture 를 sequence diagram 을 통해 고려 완료함
		UC-4	Reference architecture 를 sequence diagram 을 통해 고려 완료함
		UC-5	Reference architecture 를 sequence diagram 을 통해 고려 완료함
	QA-1		Reference architecture와 Deploy pattern을 선정함에 있어 부분적으로 고려되었음
	QA-2		Reference architecture와 Deploy pattern을 선정함에 있어 부분적으로 고려되었음
QA-3			
QA-4			
QA-5			
	QA-6		Server의 Reference architecture를 선정함에 있어 부분적으로 고려됨
		CRN-1	해당 Concern을 고려하여 System의 architecture는 Green field에서 설계되었음.
CRN-2			
	CRN-3		해당 Concern을 고려하여 architecture 설계 과정에서 모든 Architectural Driver를 고려함
		CRN-4	Sequence diagram 세분화 중 고려 완료
		CRN-5	Sequence diagram 세분화 중 고려 완료
		CRN-6	해당 Concern을 반영하여 Server architecture 선정
		CON-1	부분 반영
		CON-2	Sequence diagram 을 통해 고려 완료함
		CON-3	Requirement 문서대로 반영

ADD Design process
: iteration 3

Selected Architectural Drivers

Design Decisions and Location	Rationale
UC-1: Manage database	총 음료의 개수는 20 종류이다.
UC-2: Display information	한 자판기는 7 종류의 음료를 판매하며, 판매하지 않는 음료도 메뉴는 제공한다.
UC-3: Process tasks	사용자가 음료를 선택 후 결제하면 음료가 제공된다. 결제는 카드로 하고, 잔액이 부족한 경우 결제되지 않는다.
UC-4: Manage network	음료 재고가 부족하거나 자판기에서 판매하지 않는 음료를 구매하려는 경우 다른 자판기 재고 확인 후 위치를 안내한다. 이 때, 네트워크 상의 자판기에 broadcast msg를 통해 재고 확인을 요청하여 확인하고, 네트워크 msg를 통해 대상 자판기의 위치를 확인하여 안내한다.
UC-5: Identification	다른 자판기의 음료 구매에 대해 선결제를 할 수 있다. 이 때, 현재 자판기에서 결제 후 인증 코드를 발급하며 다른 자판기로 가서 인증코드를 입력하면 음료가 나온다.

Quality Attribute	Scenario	Associated Use Case
QA-1: Marketability	기존 코드 중 재활용 가능한 부분의 활용을 통헤 낮은 개발 비용/ 빠른 개발속도를 달성해야 한다.	ALL
QA-2: Usability	SW의 기능이 분명하고 간결하며 사용이 편리해야 한다.	UC-3, 4, 5
QA-3: Usability	모든 구매한 물품과 비용이 한 눈에 보일 수 있게 해야 한다.	UC-1, 2
QA-4: Performance	화면 사이의 연결이나 연동이 사용자로 하여금 불편함을 느끼지 않도록 0.5초 이내로 제한한다.	UC-3, 4, 5
QA-5: Availability	자판기 작동 시 일주일 정도는 이상 없이 동작해야 한다.	ALL
QA-6: Modifiability	음료 메뉴 추가나 삭제가 용이했으면 좋겠다.	UC-1

ID	Constraints
CON-1	각 자판기는 모두 네트워크에 연결되어 있고 네트워크 연결 정보는 미리 알고 있다(최대 10대).
CON-2	자판기의 판매 음료 종류는 사전에 결정된다.
CON-3	자판기 사이의 msg protocol은 사전에 결정된다.

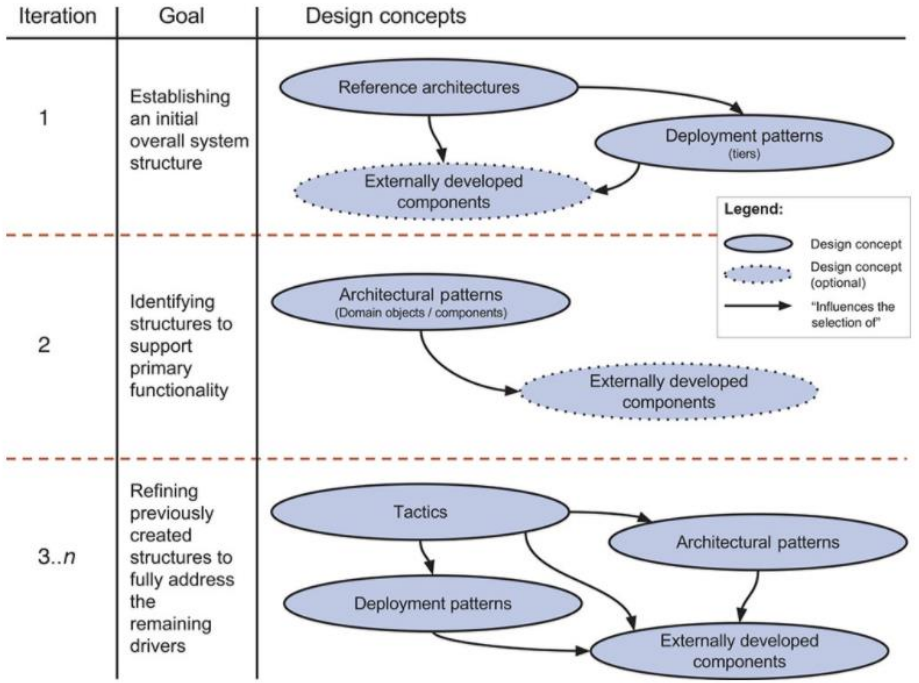
ID	Concern
CRN-1	처음에 전반적인 system architecture 부터 설계한다.
CRN-2	개발 환경은 프로그램 개발자와 네트워크 개발자가 익숙한 것을 선택한다.
CRN-3	모든 Architectural Driver를 만족하는 Software Architecture 를 세운다.
CRN-4	사용자에게 적합한 UI/UX design 개발
CRN-5	사용자의 요구를 분석하여 그것들을 컴퓨터에 저장할 수 있는 데이터베이스의 구조에 맞게 변형한 후 특정 DBMS로 데이터베이스를 구현
CRN-6	웹 방화벽 구축, 침입탐지시스템 등 정보보호를 위한 관리적 기술적 물리적인 시스템 구축

Step 2: Establish iteration goal by selecting drivers

Iteration goal

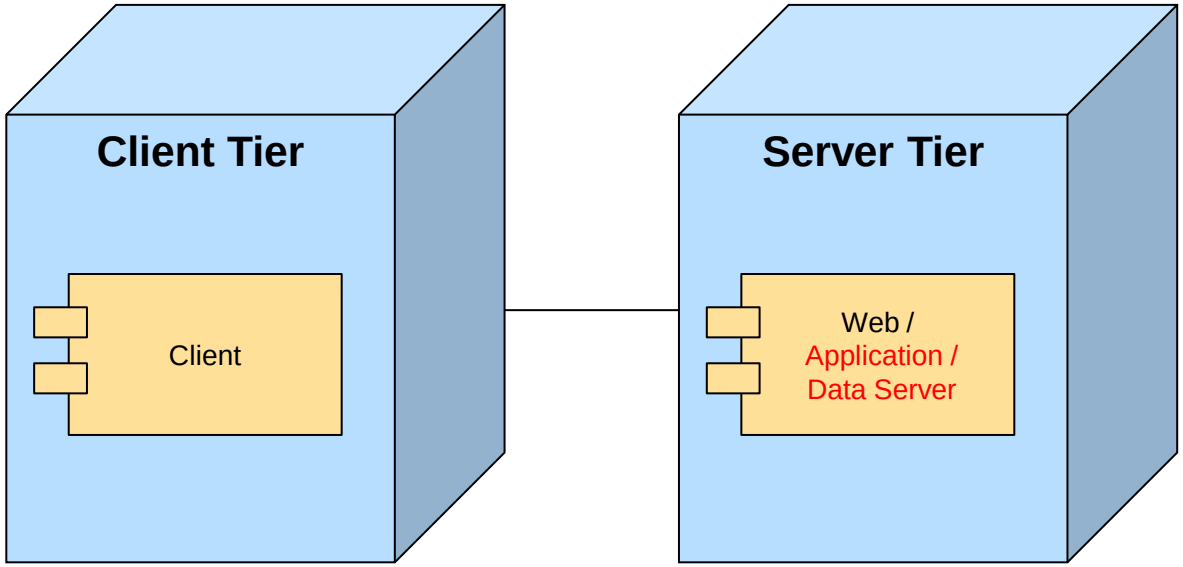
- Reasoning about the fulfillment of the important QA.
  - 가장 중요한 QA를 **"Availability"** 로 선정

Quality Attribute	Scenario	Associated Use Case
QA-1: Marketability	기존 코드 중 재활용 가능한 부분의 활용을 통해 낮은 개발비용/ 빠른 개발속도를 달성해야 한다.	ALL
QA-2: Usability	SW의 기능이 분명하고 간결하며 사용이 편리해야 한다.	UC-3, 4, 5
QA-3: Usability	모든 구매한 물품과 비용이 한 눈에 보일 수 있게 해야 한다.	UC-1, 2
QA-4: Performance	화면 사이의 연결이나 연동이 사용자로 하여금 불편함을 느끼지 않도록 0.5초 이내로 제한한다.	UC-3, 4, 5
QA-5: Availability	**자판기 작동 시 일주일 정도는 이상 없이 동작해야 한다.**	ALL
QA-6: Modifiability	음료 메뉴 추가나 삭제가 용이했으면 좋겠다.	UC-1



Step 3: Choose one or more elements of the system to refine

- **Availability QA 고려를 통해 physical node 를 refine 하기로 결정**
  - Two-tier deployment model 서버 tier 의 Application & data server 선택



Step 4: Choose One or More Design Concepts that Satisfy the Selected Drivers

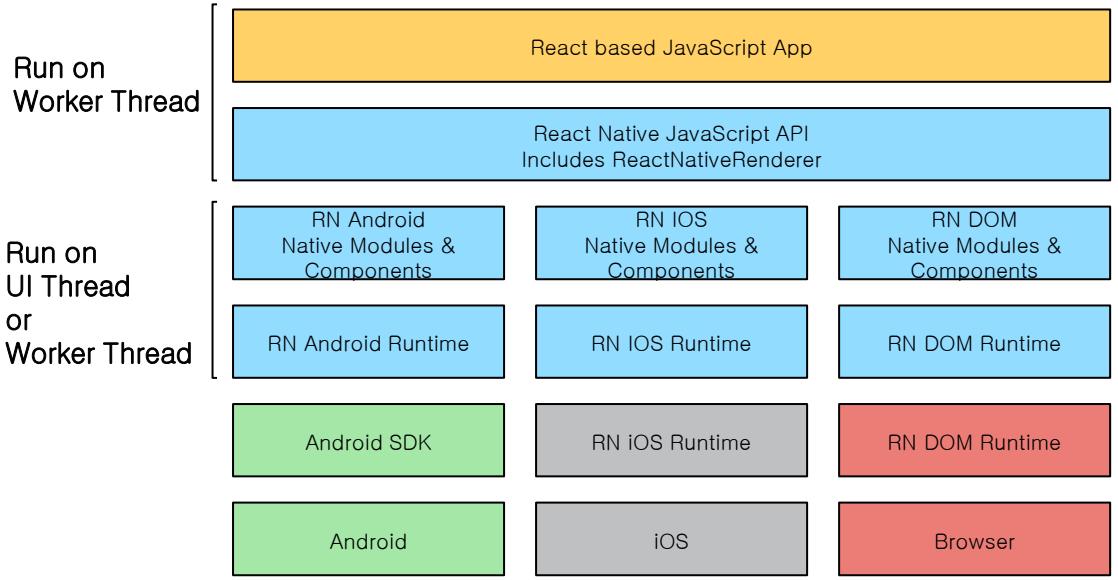
- **Design concept: Tactics & Externally developed components**
- **Selection of design concepts**
  - Tactic: active redundancy
  - Externally developed components:
    - message queue technology
    - React native framework

Design Decision and Location	Rationale
Apply the Active redundancy tactic by refining the application server and other critical components such as the network management	중요한 elements 에 대해 시스템 redundancy 를 높일 수 있는 전략을 추가(load balancing)
Introduce an element from the **message queue technology** family	Queue 구조의 elements 를 시스템 failure 발생 시 치명적인 부분에 추가하여 trapping 하고자 하며, 이는 QA-5 Availability 를 고려한 사항
React native framework	Portable local user interface 를 구축하기 위한 JavaScript 언어의 framework 를 기반으로 개발

Design Decision and Location	Rationale
Deploy message queue on a separate	Deploying the message queue on a separate node will guarantee that no traps are lost in case of application failure. This node is replicated using the tactic of active redundancy, but only one copy receives and treats events coming from the network devices
Use active redundancy and load balancing in the application server	Because two replicas of the application server are active at any time, it makes sense to distribute and balance the load among the replicas. This tactic can be achieved through the use of the Load-Balanced Cluster pattern
Implement load balancing and redundancy using technology support	Many technological options for load balancing and redundancy can be implemented without having to develop an ad hoc solution that would be less mature and harder to support

React Native Framework

- Portable UI 개발에 적합
  - Ex.) Facebook
  - 인증코드 사용을 고려하여 추후 확장성 기대

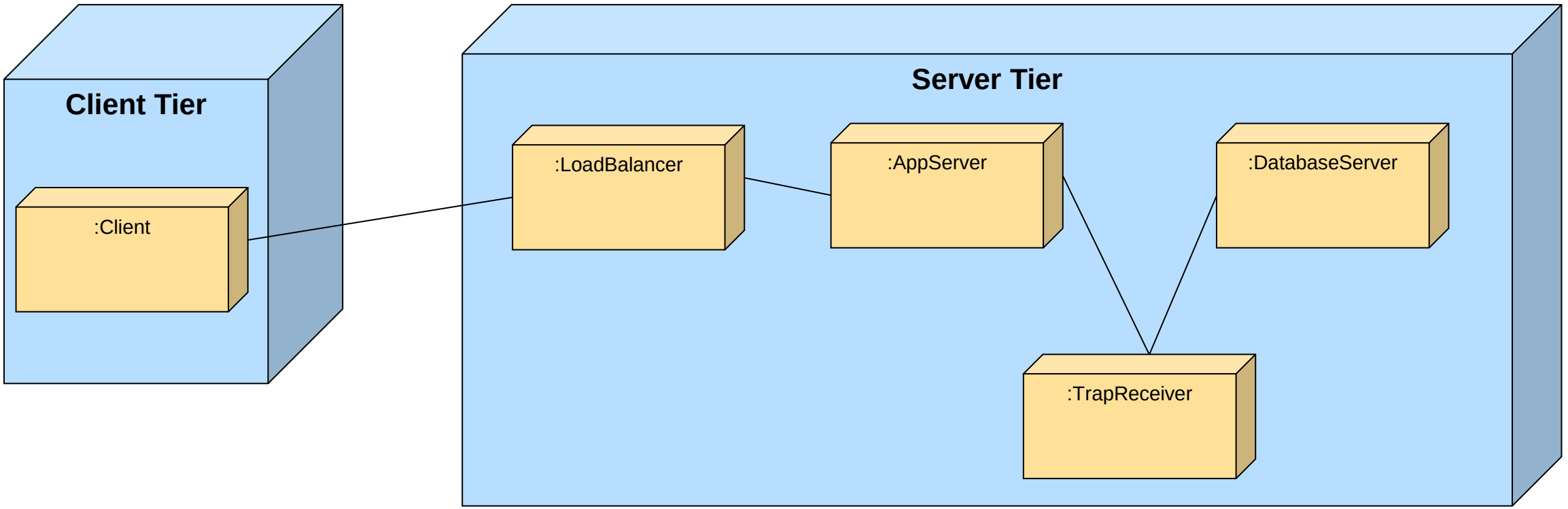


Division	React Native Description
Technology family	Local user interface
Language	JavaScript
URL	https://reactnative.dev/
Purpose	Framework to support the creation of portable local user interface.
Overview	React Native is an open source application framework developed by Facebook. With react native, you can develop real native apps. By using JavaScript and React library, you can develop for both Android and iOS. ** Characteristic 1. A collection of "special" React components 2. Components compiled to Native Widgets 3. Native platform APIs exposed to JavaScript
Development Environment	React Native CLI
Benefits	1. Productivity: When the source code is modified, the changed contents can be checked immediately. 2. Open Source: MIT License
Limitations	1. Performance: A hybrid app method that uses a native bridge to connect a JavaScript thread and a native thread. lower than the native development method. 2. Native Functions Development: Services with many unique native features can be a bit difficult to develop.

Step 6: Sketch views and record design decisions

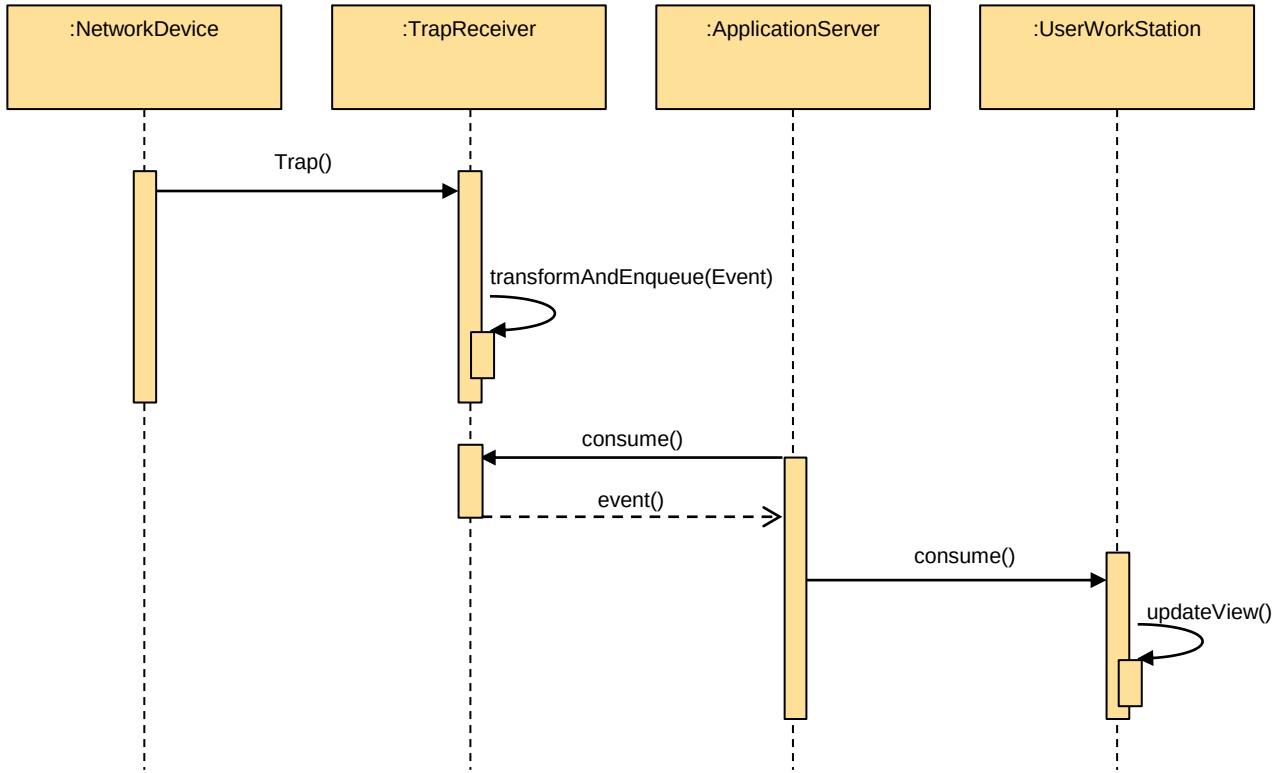
Refining deployment diagram with new elements

- LoadBalancer
- TrapReciever



Sequence Diagram for UC-4: Manage network

- Illustrating how the TrapReceiver element exchanges messages with other elements to support UC-4 (Manage network)



Step 7: Perform analysis of current design and review iteration goal and achievement of design purpose

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During the Iteration
		UC-1	
		UC-2	
		UC-3	
		UC-4	
		UC-5	
	QA-1		
	QA-2		
	QA-3		Tactic 채택을 통해 부분 고려되었음
	QA-4		Tactic 채택을 통해 부분 고려되었음
		QA-5	Important QA 로 선택되어 Tactic, queue, framework 를 통해 고려 완료
	QA-6		
		CRN-1	
		CRN-2	ADD iteration 을 끝냄으로써 고려 완료
	CRN-3		
		CRN-4	
		CRN-5	
		CRN-6	
		CON-1	
		CON-2	
		CON-3	

THANK YOU
FOR YOUR ATTENTION!