소프트웨어공학특론
Team project - ADD

Team 2
허윤아
전다윤
김두리

Contents

1. From QAW - Quality Attribute Scenarios

ID	Quality Attribute	Scenario	Associated Use Case
QA-1	Usability	사용자는 자판기에서 원하는 상품을 뽑기 위해 버튼을 누른다. 자판기는 재고가 있는 경우 결제 이후 최소 5초 안에 상품을 제공하고 재고가 없는 경우 최소 30초 안에 다른 자판기의 재고를 30초 안에 파악하여 안내한다. 사용자가 상품 선택을 결제 이전 취소하는 경우, 메인 화면으로 5초 이내에 돌아간다.	UC-1,2,4
QA-2	Marketability	마케터는 출시 이전 분산 자판기에 대한 설문조사를 한다. 설문조사를 통해 사용자들의 니즈를 파악하여 DVM의 시장 경쟁성을 높인다.	UC-1,2,3,4
QA-3	Reusability	코드 안에서 코드의 중복성을 최소화하기 위하여 정적분석을 한다. 정적분석을 통하여 코드의 중복성을 최대 10%로 줄여 다른 시스템이나 애플리케이션에서 해당 시스템을 활용할 수 있도록 한다.	UC-1,2,3,4,7,8,10,11
QA-4	Performance	자판기들에서 네트워크에 5개의 메세지를 보낸다. 네트워크에서 60초 이내에 5개의 메세지를 모두 처리하고 응답한다.	UC-7,8,10,11
QA-5	Maintainability	3명의 개발자가 하나의 컴포넌트 코드를 수정하고자 할 때, 3시간 이내에 수정 가능하다.	UC-1,2,3,4,7,8,10,11
QA-6	Testability	테스터가 12시간 이내로 테스트를 위한 가상의 환경을 구축한다. 테스터가 테스트를 위한 가상의 환경이 구축된 상태에서 코드를 테스트하고 자 할 때, 6시간 이내로 85%이상 테스트한다.	모두
QA-7	Security	외부시스템에서 시스템 탈취를 시도하거나 네트워크로 전달되는 메세지에 접근하고자 할 때 , 30초 이내에 접근을 감지 후, 5초 이내에 관리자에게 위험 메세지를 전달한다.	UC-1,7,8,10,11

1. From QAW - Primary Functionality

No.	Use-Case	Description
UC-1	Card Payment	사용자가 상품 구매를 선택하고 해당 상품의 재고가 1 이상인 경우 카드 단말기를 통해 결제를 진행한다.
UC-2	Select Menu	사용자가 선택한 메뉴의 재고가 있을 경우 카드 결제를 요구하고, 재고가 없을 경우 서버에 '다른 자판기 재고 확인'을 요청한다.
UC-3	Input Verification Code	사용자가 다른 자판기에서 선결제를 통해 받은 인증코드를 입력하면, 해당 인증코드의 유효성을 검사한다.
UC-4	Inform Other DVM Address	사용자가 선택한 상품의 재고가 있는 가장 가까운 자판기의 위치를 알려준다.
UC-7	Respond Stock	다른 자판기의 '재고 여부 확인 메시지'를 받아 현재 자판기에 해당 상품의 재고를 확인 후 재고 여부를 메시지로 응답한다.
UC-8	Request Stock	서버가 '다른 자판기 재고 확인 요청'을 받으면, 처음 유저가 선택한 자판기로부터 가까운 자판기 순서대로 선택된 상품의 재고 여부를 요청한다.
UC-10	Receive Verification Code	사용중인 자판기가 다른 자판기에서 선결제를 통해 받은 인증코드를 받는다.
UC-11	Transfer Verification Code	선결제한 상품을 제공해줄 자판기로 인증코드를 전송한다.

1. From QAW - Other architectural drivers

- **Design Purpose**

  1) DVM 도메인에 익숙하지 않으므로, architecture를 먼저 분석한 후 개발을 진행하면 좀 더 체계적으로 시스템을 만들 수 있다.
  2) 개발 시작 전, 도메인 탐구 목적의 프로토타입을 만들어서 큰 그림을 가지고 시작할 수 있다.

- **Constraints**

ID	Constraint
CON-1	여러 자판기에서 같은 동작이 일어날 수 있어야 한다.
CON-2	네트워크는 낮은 대역폭을 가질 수 있지만, 신뢰할 수 있어야 한다.
CON-3	성능 데이터는 5분 이내의 간격으로 수집되어야 한다.
CON-4	매번 동일한 환경에서 동일한 동작을 해야 한다.

- **Architectural Concern**

ID	Concern
CRN-1	전체 초기 System Structure를 수립해야 한다.
CRN-2	구성원이 System Domain에 대한 이해도를 높여야 한다.

2. ADD 3.0 - 1st iteration

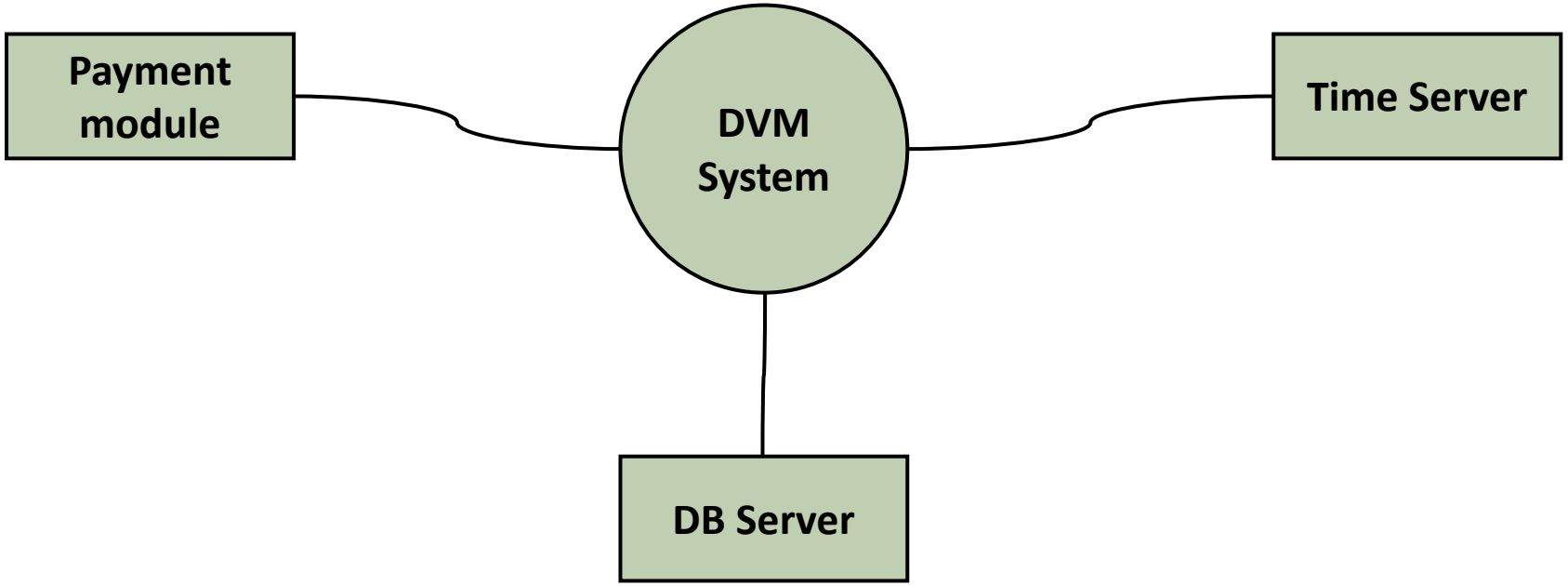
2. 1st iteration

Step 2

Iteration goal: 전체 초기 System Structure를 수립해야 한다. (CRN-1)

Step 3

Context diagram for DVM system



2. 1st iteration

Step 4 Design Concept(reference architecture) 선정

<table>
<tr><th>Design Decisions and Location</th><th colspan="2">Rationale</th></tr>
<tr><td rowspan="5">Logical structure
System의 Client 부분에는
Rich Client Application reference architecture 를 활용한다.</td><td colspan="2">Rich client application은 자판기에 설치되는 프로그램의 개발에 도움을 준다.
또한, 더 풍부한 UI 경험을 사용자에게 제공할 수 있고, 사용자의 요청이 있을 때만 네트워크에 접속하면 된다는 점에서 네트워크 비용을 아낄 수 있다.</td></tr>
<tr><td>Alternative</td><td>Reason for discarding</td></tr>
<tr><td>Rich Internet Application</td><td>빠르게 사용자의 동작에 맞게 응답해야 하고,
로컬 리소스에 접근이 가능해야 하므로 사용하지 않는다.</td></tr>
<tr><td>Web Application</td><td>프로그램을 웹 상에서 사용할 필요가 없으므로
사용하지 않는다.</td></tr>
<tr><td>Mobile Application</td><td>모바일 디바이스에서 동작하는 프로그램이 아니므로
사용하지 않는다.</td></tr>
</table>

2. 1st iteration

Step 4 Design Concept(reference architecture) 선정

Design Decisions and Location	Rationale
Logical structure Server 부분에는 Service Application reference architecture를 활용한다.	Service application은 자판기에서 사용자의 호출에 따라 동작할 수 있다. 자판기 내부에 서버가 존재할 필요가 없으므로 메시지에 따라서 동작하는 별도의 서버를 따로 구축할 필요가 있다.
Physical structure Three-tier Deployment Pattern을 활용한다.	CON-1에 따라 Client와 Server를 분리하여 여러 자판기에서 동일한 동작을 하도록 할 필요가 있고, QA-4를 통해 네트워크 서버와 자판기가 분리되어야 함을 알 수 있다. 그 외에 각 자판기의 시간을 관리해주는 Time Server와 재고와 로그를 관리하는 DB서버, 그리고 추가적인 결제 Module이 필요하므로 Three-tier Deployment Pattern을 사용한다.
외부 카드 결제 Module을 사용하여 Payment Module 부분을 채운다.	이미 기존에 존재하는 결제 Module 시스템만으로도 충분히 좋은 성능을 낼 수 있고, 개발하는 데에 추가적인 비용이 발생하므로 외부 Module을 DVM 시스템에 결합하여 사용한다.

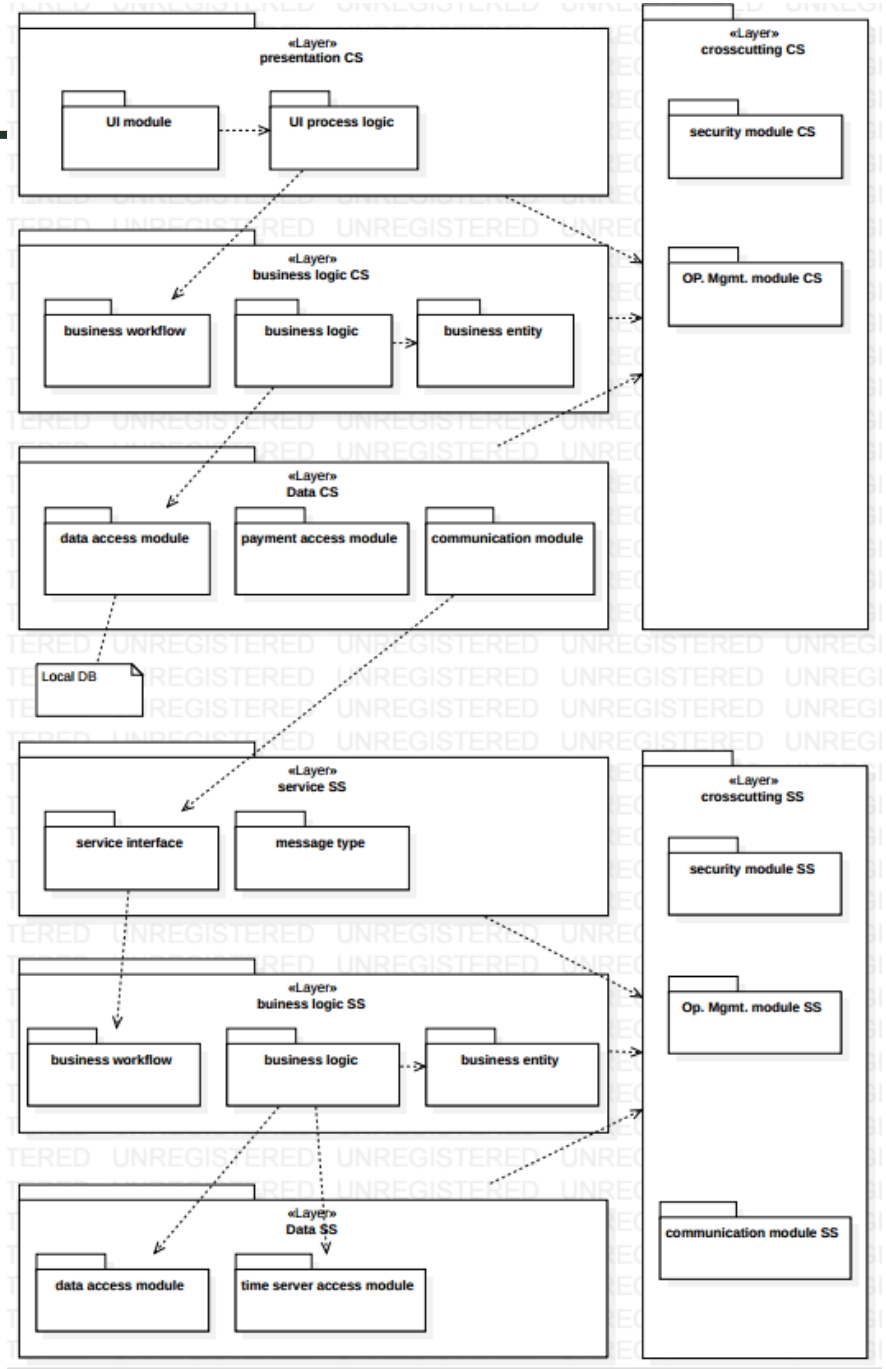
2. 1st iteration

Step 5 Instantiate design decisions

Design Decision and Location	Rationale
Helpers and Utilities In Rich Client Application 삭제	데이터 레이어의 다른 Module과 공통적으로 사용되는 Module이 없어 호환 될 필요가 없기 때문에 DVM 시스템에서는 필요 없다.
Rich client application에 외부 결제 액세스 Module 추가	Data layer에 외부 시스템을 거쳐서 결제 가능하도록 외부 결제 액세스 Module을 추가한다.
Rich client application에 communication module 추가	Service application과 메시지를 주고받기 위한 Module을 추가한다.
Helpers and Utilities in Service Application 삭제	데이터 레이어의 다른 Module과 공통적으로 사용하는 기능이 존재하지 않으므로 필요가 없다.
Service Application 에 Time Server Access Module 추가	Service Application에서 시간 정보를 알 수 있도록 Time Server Access Module을 추가한다.

2. 1st iteration

Step 6 Sketch Module View



2. 1st iteration

Step 6 Element description – Client Side

Element	Responsibility
UI	사용자의 입력(터치 입력, 카드 결제)을 받고 진행사항을 보여준다.
UI Process Logic	UI가 돌아가는 Logic을 관리한다.
Business Workflow	Business Layer안의 연결,흐름을 가지고 있다.
Business Logic	Business logic와 Client내 Operation을 포함한 모듈이다.
Business Entity	Domain Model을 구성하는 Entity
Data Access	Local DB에 데이터를 저장하고 읽어온다.
Local DB	하나의 시스템 안 데이터를 가지고 있다.
Payment Access Module	실제 결제를 수행하기 위하여 결제 모듈과 연결한다.
Communication Module	Server Side와 연결한다.
Security	보안 역할을 한다.
Operation Management	Operation을 관리한다.
Communication	Layer들을 연결하고 소통하게 해준다.

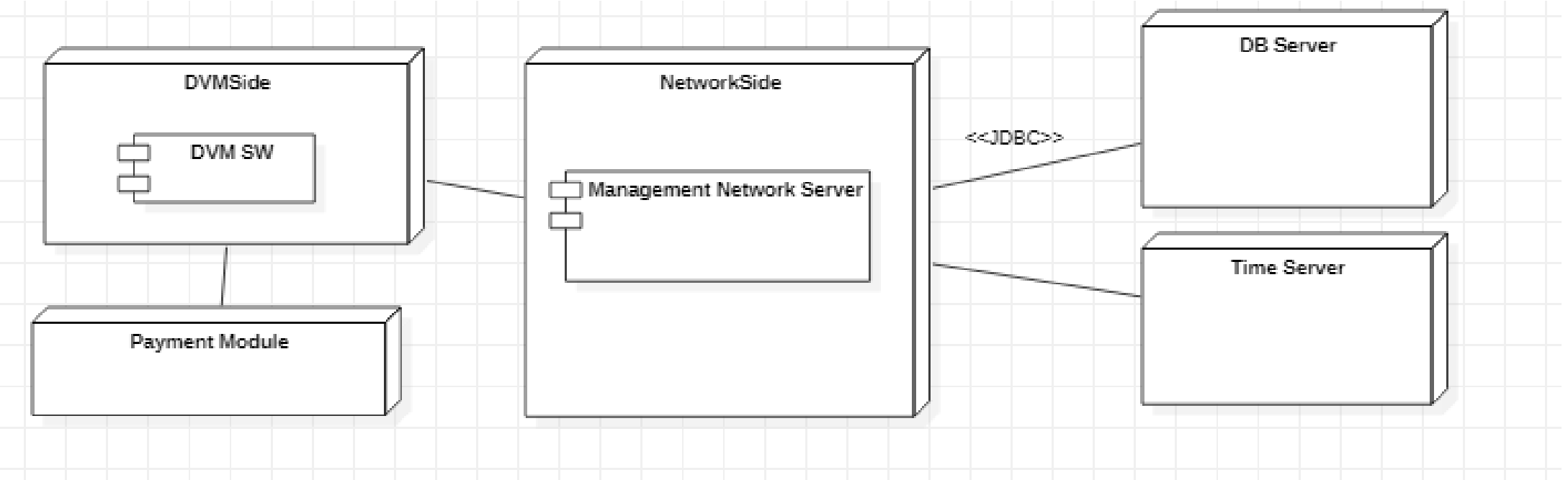
2. 1st iteration

Step 6 Element description – Server Side

Element	Responsibility
Local DB	하나의 시스템 안 데이터를 가지고 있다.
Service Interface	Client Side로 부터 데이터를 주고 받는다.
Message Type	외부와 주고받는 메시지 타입을 가지고 있다.
Business Entity	Domain Model을 구성하는 Entity
Business Logic	Business logic와 Server내 Operation을 포함한 모듈이다.
Business Workflow	Business Layer안의 연결,흐름을 가지고 있다.
Time Server Access Module	시간정보를 받기 위해 Time Server에 연결한다.
Data Access	데이터에 접근한다.
Cross Cutting	다른 레이어들 간의 교차 함수들을 포함한다. Security, I/O, logging 등이 있다.

2. 1st iteration

Step 6 Sketch Deployment Diagram



2. 1st iteration

Step 6 Element description – Deployment Module

Element	Responsibility
Database Server	Network server로부터 받은 데이터를 저장하고, 요청이 있을 경우 제공한다.
Time Server	Server에서 요청한 시간 Data를 Response 로 제공한다.
DVM Software	DVM 자체에 설치되는 UI 및 관리 SW
Payment Module	DVM에서 요청한 결제를 수행한다.
Management & Network Server	DVM으로부터 네트워크를 통해 전달되는 데이터를 관리한다.

Relation	Discription
Server와 DB 사이의 연결	JDBC 프로토콜을 사용하여 DB와 연결한다.

2. 1st iteration

Step 7 Addressing된 정도 분석

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During Iteration
	UC-2		UI에 관한 내용을 다룰 수 있는 reference architecture를 선정하였으므로 일부 달성되었음
	UC-4		Network server와 DB server를 구분하고, 서로 연결되어 있음을 deployment pattern으로 정리하였으므로 일부 달성되었음
	UC-7		Reference architecture와 deployment pattern을 통해 일부 달성되었음을 알 수 있음
	UC-8		Reference architecture와 deployment pattern을 통해 일부 달성되었음을 알 수 있음
	QA-1		UI에 관한 내용을 다룰 수 있는 reference architecture를 선정하였으므로 일부 달성되었음
QA-4			아직 이 항목을 다룰 수 있을 정도로 architecture가 구체화되지 않음
QA-7			아직 이 항목을 다룰 수 있을 정도로 architecture가 구체화되지 않음

2. 1st iteration

Step 7 Addressing된 정도 분석

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During Iteration
	CON-1		Deployment pattern과 reference architecture를 통해 여러 자판기에서도 동일한 동작을 하도록 보장하였으므로 일부 달성되었음
CON-2			아직 이 항목을 다룰 수 있을 정도로 architecture가 구체화되지 않음
CON-3			아직 이 항목을 다룰 수 있을 정도로 architecture가 구체화되지 않음
	CON-4		자판기와 무관하게 전체 시스템이 동일한 architecture 구조를 가지므로 동일한 환경에서 동작하도록 보장하였으므로 일부 달성되었음
		CRN-1	전체 시스템에 대해 iteration 1을 진행했으므로 달성되었음
	CRN-2		Iteration 1에 모든 팀원이 참여하여 수행했으므로 system domain에 대한 이해도가 일정 수준 이상 달성되었음

3. ADD 3.0 - 2nd iteration

3. 2nd iteration

Step 2 Establish iteration goal by selecting drivers

- Iteration goal: Identifying structures to support primary functionality
- Additional iteration goal – design purpose
  1) DVM 도메인에 익숙하지 않으므로, architecture를 먼저 분석한 후 개발을 진행하면 좀 더 체계적으로 시스템을 만들 수 있다.
  2) 개발 시작 전, 도메인 탐구 목적의 프로토타입을 만들어서 큰 그림을 가지고 시작할 수 있다.
- Primary functionality

UC-2	Select Menu	사용자가 선택한 메뉴의 재고가 있을 경우 카드 결제를 요구하고, 재고가 없을 경우 서버에 '다른 자판기 재고 확인'을 요청한다.
UC-4	Inform Other DVM Address	사용자가 선택한 상품의 재고가 있는 가장 가까운 자판기의 위치를 알려준다.
UC-7	Respond Stock	다른 자판기의 '재고 여부 확인 메시지'를 받아 현재 자판기에 해당 상품의 재고를 확인 후 재고 여부를 메시지로 응답한다.
UC-8	Request Stock	서버가 '다른 자판기 재고 확인 요청'을 받으면, 처음 유저가 선택한 자판기로부터 가까운 자판기 순서대로 선택된 상품의 재고 여부를 요청한다.

3. 2nd iteration

Step 3

Refine할 System Element 선정: 없음!

Step 4

Architectural driver를 만족하는 design concept(pattern) 선정
→ Architectural pattern: domain object

1) 전체 System에 대한 **Domain model**을 작성한다.
2) 전체 Primary functionality를 만족시키기 위한 **Domain object**를 생성한다.
3) 각 Object를 **Component**로 decompose한다.
4) 추가로 사용할 Externally developed component는 **Spring Framework**이다.

3. 2nd iteration

Step 4 선택한 Design concepts

Design Decisions and Location	Rationale and Assumptions
Domain Model 생성	Domain의 주요 entity들을 이해하고 식별하는 데에 도움을 줄 수 있기 때문에 Domain Model을 작성해야 한다. 작성하지 않을 경우 유지보수에 어려움을 겪을 수 있다.
Functional Requirement에 Mapping되는 Domain Object 식별	시스템의 각 요소는 Domain Object(이 수준에서는 Class) 내부에 encapsulate 되어야 한다. 데이터의 손상을 최소화하기 위해서 간접적으로 접근 가능하도록 할 필요가 있다.
Domain Object를 Component 단위로 나눔	Domain object는 Class 수준의 요소이므로 각 기능을 더 세분화하여 함수 단위로 구현하기 위해서는 각 domain object를 component 단위로 분해하여 작성할 필요가 있다. 이 시스템은 각 server가 3개의 layer로 구성되어 있으므로 각 layer에 specific한 별도의 component로 작성할 수 있어야 한다.
Spring Framework 사용	Spring은 기업체에서 흔히 쓰는 Framework이다. Spring을 이용하면 본 시스템에서 외부적으로 사용해야 하는 JDBC API와, Payment Module(Paypal) API를 사용할 수 있다.

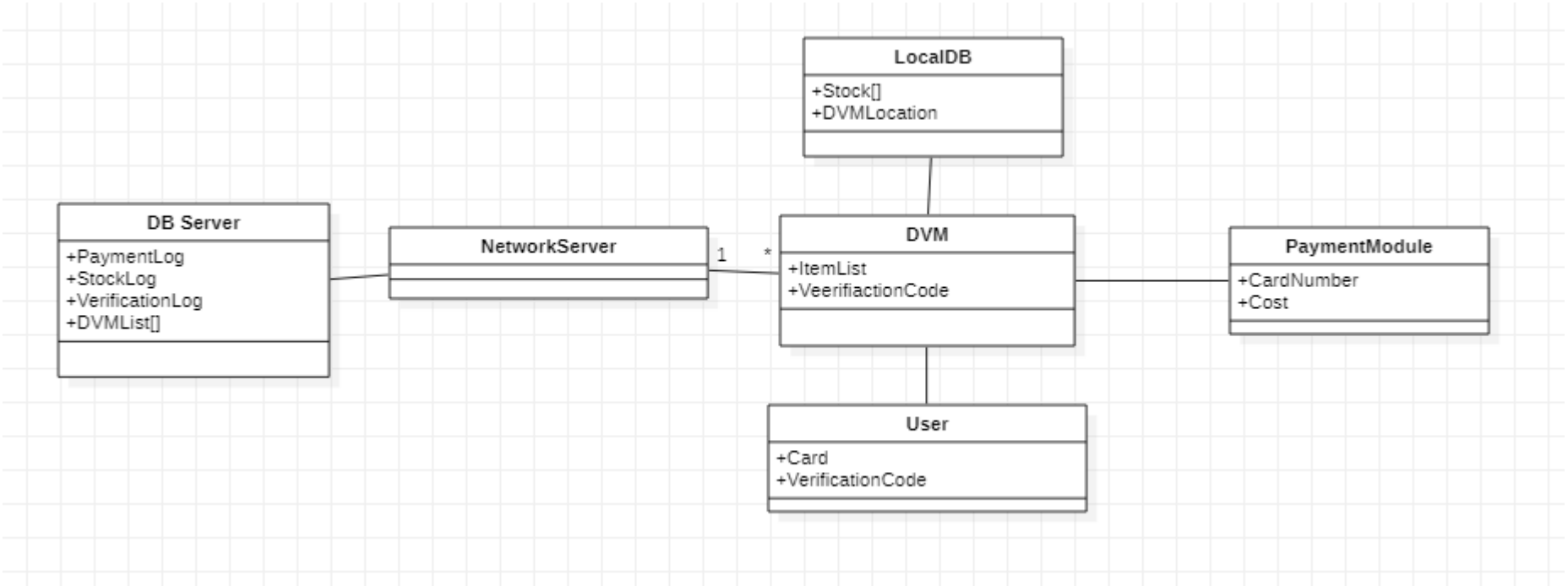
3. 2nd iteration

Step 5 Instantiate design decisions

Design Decisions and Location	Rationale
초기 Domain Model 생성	각 Primary Functionality를 만족시키고, Domain에 대한 이해도를 향상시키기 위해서 작성해야 한다. 단, 너무 오랜 시간이 소요되지 않도록 간단하게 작성해야 한다.
각 Primary Functionality를 만족시키기 위한 Domain Object 식별	시스템의 Primary Functionality를 만족시키기 위한 Domain Object를 작성한다. 기본적으로 Domain Object Architectural Pattern은 Object-Oriented Architecture를 따르므로, CON-1, 4를 만족시킬 수 있고, 이 과정을 통해 System Domain을 이해할 수 있으므로 CRN-2를 만족시킬 수 있다.
Domain object를 Component 단위로 나누고, 특정 Layer에 귀속되는 Component 식별	각 layer에서 모든 primary functionality가 고려될 수 있도록 해야 한다. 이때 Quality attribute나 constraint, architectural constraint는 고려하지 않는다. 각 Domain object를 class로 간주하고, 각 component를 각 class 내부의 함수로 간주하여 작성한다.
Spring Framework와 연결되어야 하는 component 식별	본 system에서는 DB server 관리와 외부 결제 module 사용을 위해서 spring framework를 사용한다. 이를 위해서 spring framework와 연결되어야 하는 DB server 접근 및 결제와 관련되는 component를 미리 식별할 필요가 있다.

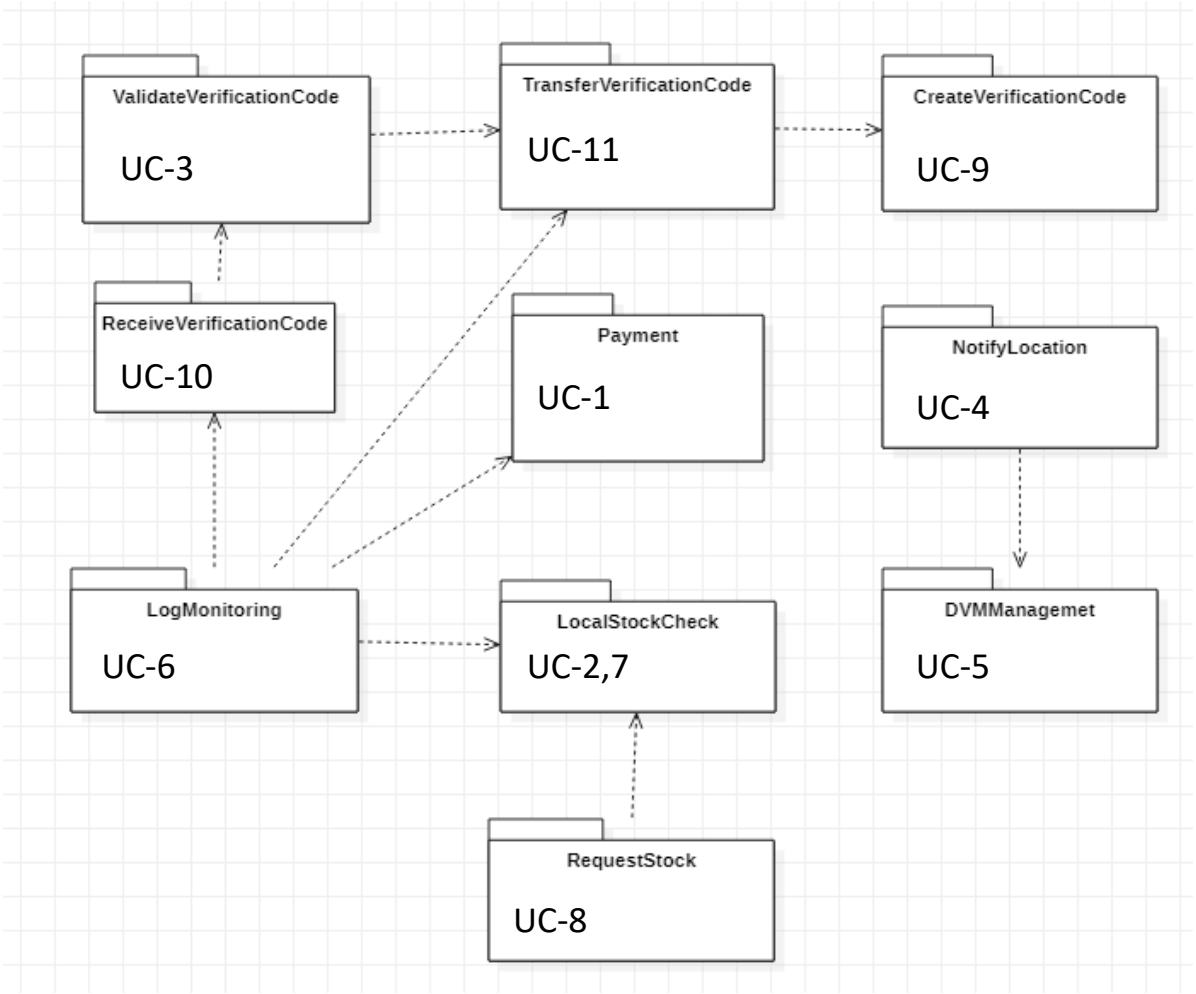
3. 2nd iteration

Step 6 Sketch Domain Model



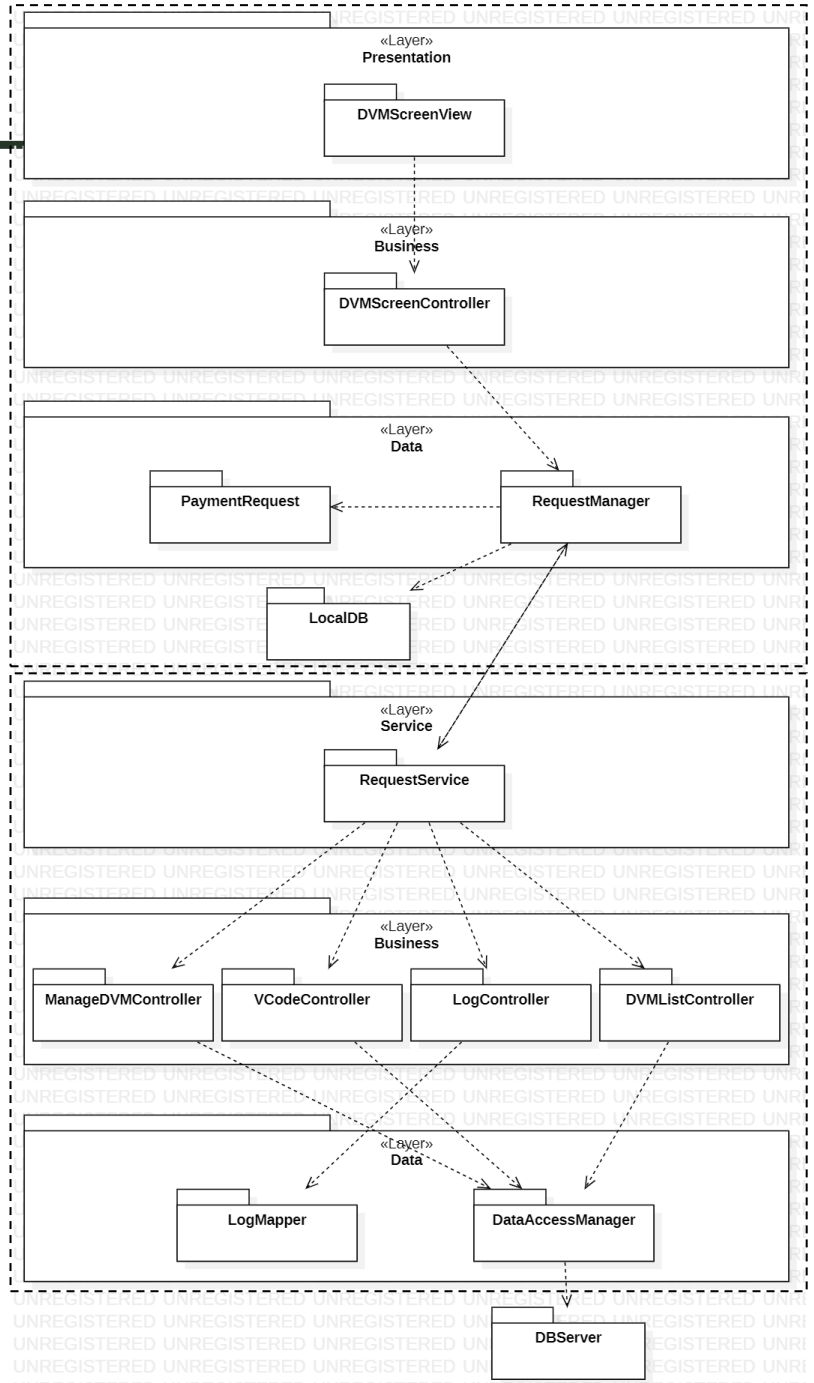
3. 2nd iteration

Step 6 Sketch Domain Object



3. 2nd iteration

Step 6 Sketch Module View(Refined)



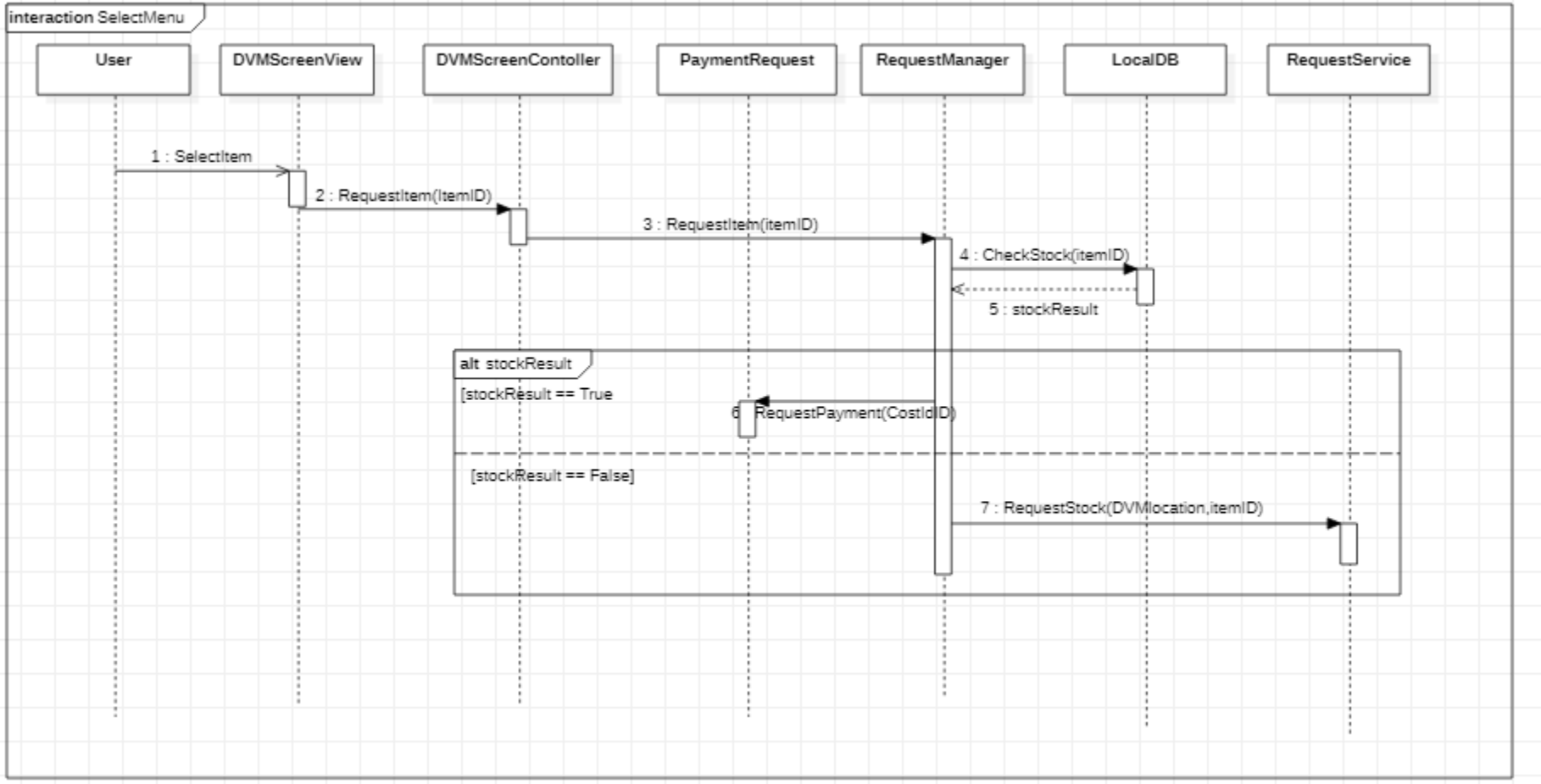
3. 2nd iteration

Step 6 Element Description – Module View(Refined)

Element	Responsibility
DVMScreenView	DVM의 화면으로, 선택할 수 있는 상품을 보여주고 사용자가 선택할 수 있다.
DVMScreenController	DVM 화면 UI에 필요한 동작을 하도록 기능을 담고 있다.
PaymentRequest	결제 모듈에 결제를 요청을 한다. 외부 모듈과 연결되어 있으나 나타내지 않음.
RequestManager	서버측과 커뮤니케이션을 한다.
LocalDB	각 자판기마다 각 상품에 대한 재고를 가지고 있다.
RequestService	각 자판기로부터 요청을 받은 서비스를 관리/제공한다.
ManageDVMController	관리자가 DB 서버에 각 자판기의 위치와 식별번호, 상품 종류를 입력한다.
VcodeController	각 자판기가 생성하고 전달받는 인증코드에 대한 정보를 관리한다.
LogController	각 자판기의 log에 대한 정보를 관리한다.
DVMListController	각 자판기의 위치, 식별번호, 상품 종류에 대한 정보를 관리한다.
LogMapper	각 자판기의 log 정보를 DB 서버에 저장한다.
DataAccessManager	각 DVM의 위치, 식별번호, 상품 종류에 대한 정보를 저장하거나 불러온다.
DBServer	각 자판기의 log 정보와 위치, 식별번호, 상품 종류 정보를 가지고 있다.

3. 2nd iteration

Step 6 Sequence Diagram for UC-2



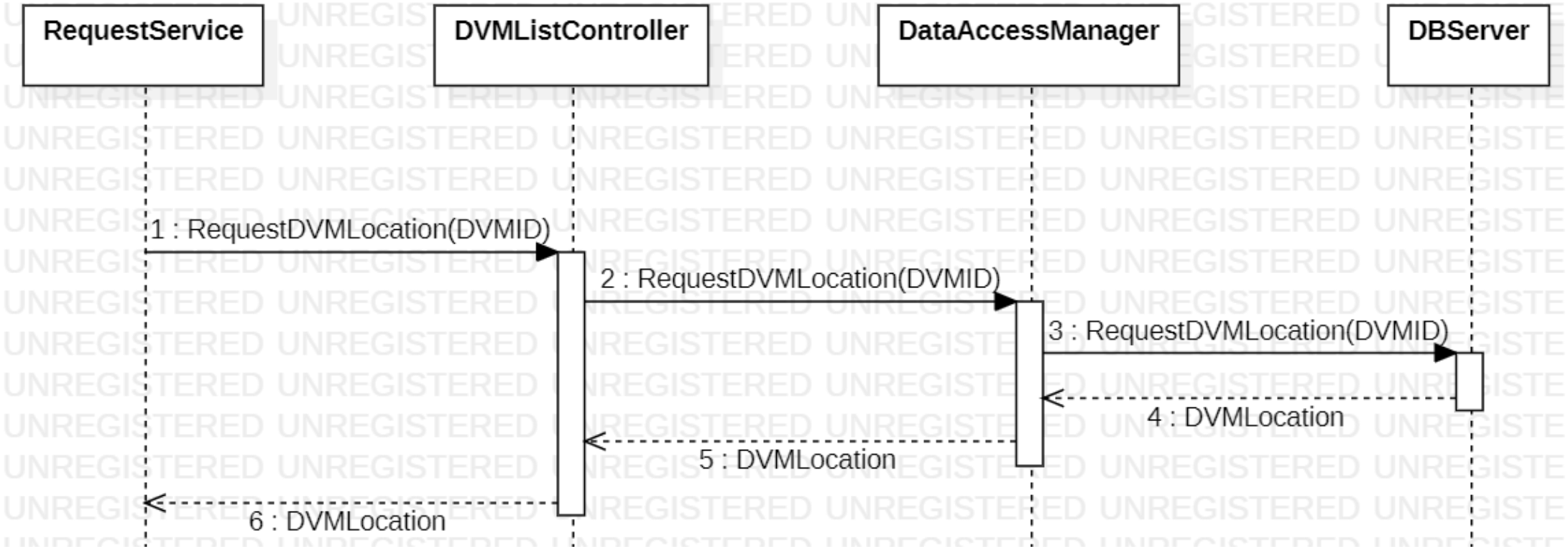
3. 2nd iteration

Step 6 Interface Description UC-2

Element	Method Name	Responsibility
DVMScreen Contoller	RequestItem (itemID)	요청 받은 품목의 재고여부를 알기 위해 DVMScreenController에 요청한다.
RequestManger	RequestItem (itemID)	요청 받은 품목의 재고여부를 알기 위해 RequestManger에 요청하고 재고 여부를 리턴한다.
LocalDB	CheckStock (itemID)	DVM에 선택된 품목의 재고가 남아있는지 확인하고 있으면 StockResult로 True를 , 없으면 False를 리턴 한다.
RequestService	RequestStock (DVMlocation, itemID)	네트워크 서버에 선택된 품목의 재고가 남아있는 DVM을 요청한다. 이때 DVM의 위치를 같이 보내 가까운 순으로 찾도록 한다.
PaymentRequest	ShowCardPayment (itemCost)	선택된 품목의 가격을 결제하도록 카드 결제 창에 결제 가능 화면을 띄운다.

3. 2nd iteration

Step 6 Sequence Diagram for UC-4



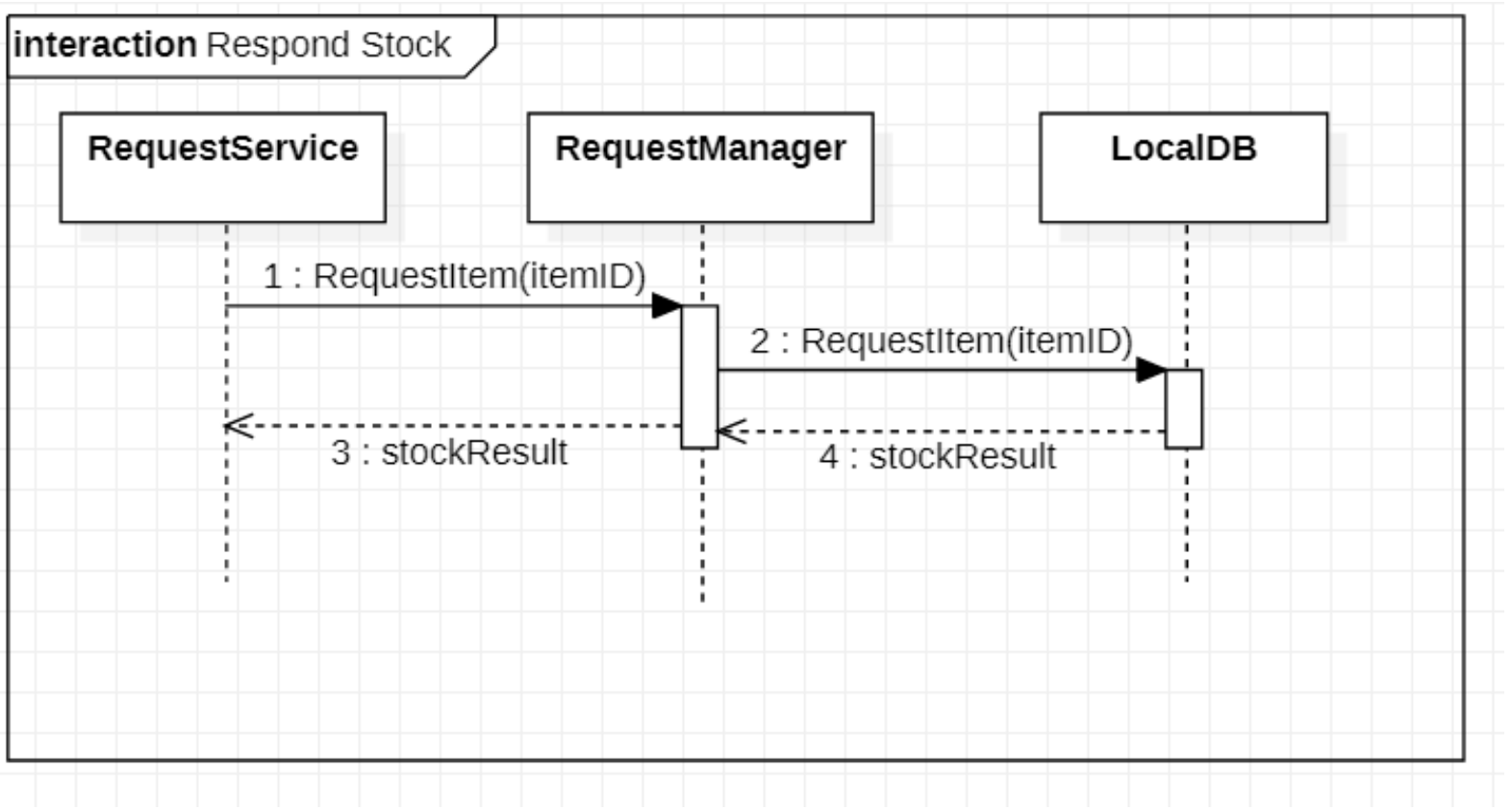
3. 2nd iteration

Step 6 Interface Description UC-4

Element	Method Name	Responsibility
DVMListController	RequestDVMLocation (DVMID)	UC-7 이후 재고를 가지고 있는 자판기 ID를 알고 있는 상태에서 DVMListController → DataAccessManager → DBServer로 DVM의 위치 정보를 요청한다. 재고를 가지고 있는 자판기의 위치를 리턴한다. 이 정보는 이후 사용자가 접속 중인 DVM의 화면을 통해 전달되어 표출된다.
DataAccessManager	RequestDVMLocation (DVMID)	
DBServer	RequestDVMLocation (DVMID)	

3. 2nd iteration

Step 6 Sequence Diagram for UC-7



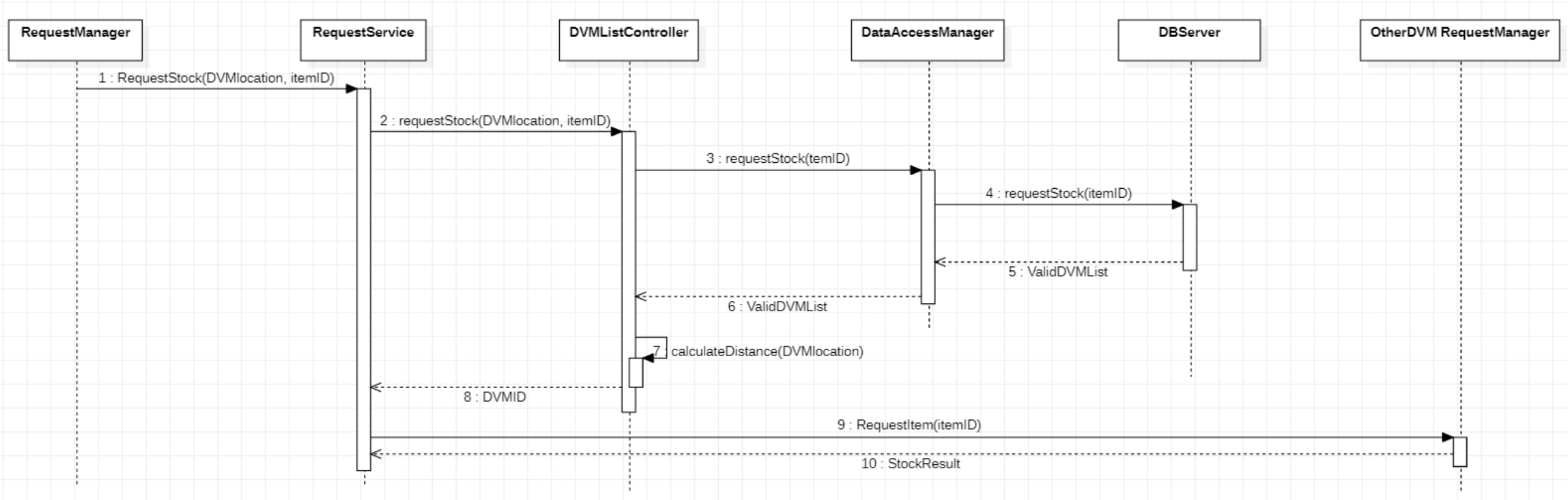
3. 2nd iteration

Step 6 Interface Description – UC-7

Element	Method Name	Responsibility
RequestManager	RequestItem (itemID)	요청받은 품목의 재고여부를 알기 위해 RequestManger에 요청하고 재고 여부를 리턴한다.
LocalDB	RequestItem (itemID)	요청받은 품목의 재고 여부를 확인한다. 확인한 품목의 재고 여부를 리턴한다.

3. 2nd iteration

Step 6 Sequence Diagram for UC-8



3. 2nd iteration

Step 6 Interface Description – UC-8

Element	Method Name	Responsibility
OtherDVM RequestManager	RequestItem (itemID)	다른 DVM의 재고를 요청한다. (UC-7과 이어짐)
DVMListController	RequestStock (DVMlocation, itemID)	현재 요청한 DVM에서 가장 가까운 DVM의 요청받은 품목의 재고여부를 확인한다.
	CalaulateDistance (DVMlocation)	현재 요청한 DVMlocation을 제외한 DVM중 DVMlocation에 가장 가까운 DVM이 무엇인지 계산한다.
DataAccessManager	RequestStock (itemID)	요청받은 품목의 재고여부를 확인한다.
DBServer	RequestStock (itemID)	요청받은 품목의 재고여부를 확인한다.

3. 2nd iteration

Step 7 Addressing된 정도 분석

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During Iteration
		UC-2	본 use case를 달성하기 위한 각 layer의 module과 interface들을 전부 식별하였음
		UC-4	본 use case를 달성하기 위한 각 layer의 module과 interface들을 전부 식별하였음
		UC-7	본 use case를 달성하기 위한 각 layer의 module과 interface들을 전부 식별하였음
		UC-8	본 use case를 달성하기 위한 각 layer의 module과 interface들을 전부 식별하였음
	QA-1		해당 quality attribute와 연관된 use case 중 일부를 이번 iteration에서 전부 달성하였으므로 일부 달성되었음
	QA-4		해당 quality attribute와 연관된 use case 중 일부를 이번 iteration에서 전부 달성하였으므로 일부 달성되었음
	QA-7		해당 quality attribute와 연관된 use case 중 일부를 이번 iteration에서 전부 달성하였으므로 일부 달성되었음

3. 2nd iteration

Step 7 Addressing된 정도 분석

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During Iteration
	CON-1		각 레이어 모듈에 따른 동작이 모든 DVM이 동일하게 작동하게 설계되었음
CON-2			아직 이 항목을 다룰 수 있을 정도로 Architecture가 구체화되지 않음
CON-3			아직 이 항목을 다룰 수 있을 정도로 Architecture가 구체화되지 않음
	CON-4		각 레이어 모듈에 따른 동작이 어떠한 환경이든 동일하게 작동하게 설계되었음
	CRN-2		Iteration 1과 2에 모든 팀원이 참여하여 수행했으므로 system domain에 대한 이해도가 일정 수준 이상 달성되었음

Drivers that are completely addressed in previous iteration is removed

3. ADD 3.0 - 3rd iteration

4. 3rd iteration

Step 2 달성해야 할 iteration goal

ID	Quality Attribute	Scenario	Associated Use Cases
QA-1	Usability	사용자는 자판기에서 원하는 상품을 뽑기 위해 버튼을 누른다. 자판기는 재고가 있는 경우 결제 이후 최소 5초 안에 상품을 제공하고 재고가 없는 경우 최소 30초 안에 다른 자판기의 재고를 30초 안에 파악하여 안내한다. 사용자가 상품 선택을 결제 이전 취소하는 경우, 메인 화면으로 5초 이내에 돌아간다.	UC-1,2,4
QA-4	Performance	자판기들에서 네트워크에 5개의 메세지를 보낸다. 네트워크에서 60초 이내에 5개의 메세지를 모두 처리하고 응답한다.	UC-7,8,10,11
QA-7	Security	외부시스템에서 시스템 탈취를 시도하거나 네트워크로 전달되는 메시지에 접근하고자 할 때 , 30초 이내에 접근을 감지 후, 5초 이내에 관리자에게 위험 메세지를 전달한다.	UC-1,7,8,10,11

Step 3 시스템에서 수정 되어야 할 Element

Security를 만족하기 위해서 Network Server가 수정되어야 한다

4. 3rd iteration

Step 4
Design goal을 만족하기 위한 Design concept 선정

Design Decisions and Location	Rationale and Assumptions
DVM SW 차원에서 **Cancel**을 사용	사용자가 사용 중간에 취소할 수 있게 만들어 Usability를 높인다. 사용자가 결제하기 전 취소 가능하도록 한다.
Network server 차원에서 **Introduce concurrency**를 사용	여러 DVM에서 사용자가 동시에 주문을 진행하여도 지연없이 원활하게 돌아갈 수 있도록 Performance를 높인다. 여러 DVM에서 주문이 진행될 때 Multi-thread를 사용한다.
Network server 차원에서 **Verify message integrity**를 사용	Message를 주고 받을 때, 메시지 확인을 통해 외부 접근을 탐지하여 Security를 높인다. Message를 보낼 때 , Hash Value를 추가하고 받을 때 검사 한다.
Network server 차원에서 **Inform actors**를 사용	외부접근을 탐지하면, 관리자에게 Message를 보내 Security를 높인다. 관리자에게 메시지를 보내는 부분을 추가한다.

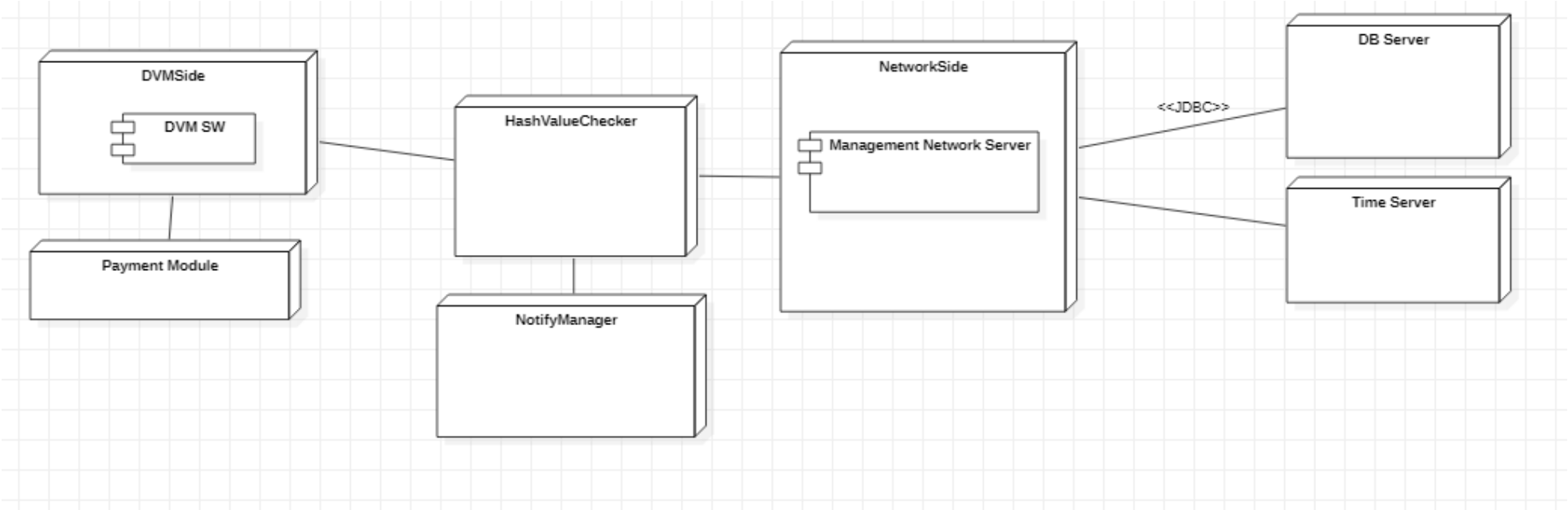
4. 3rd iteration

Step 5 Instantiate design decision

Design Decisions and Location	Rationale and Assumptions
DVM SW에 **Cancel** 기능을 위해 UI와 system을 분리	사용자가 결제 전 진행 중이던 프로세스를 취소할 수 있도록 취소 기능을 추가하여 사용자의 편의성을 증가시킨다. 언제든 user가 cancel을 요청했을 때 메인 화면으로 돌아가야 하므로, UI는 asynchronous해야 한다. Deployment diagram과 sequence diagram에 변화는 없고, usability는 향상된다.
Network server에서 여러 메시지를 동시에 처리하기 위해 **multi-thread** 사용	여러 DVM에서 여러 사용자가 동시에 특정 기능을 실행하고자 할 때, 입력된 메시지들을 동시에 처리하기 위해 multi-thread 기능을 사용한다. Deployment diagram과 sequence diagram에 변화는 없고, 성능이 향상된다.
Network server와 DVM 사이에 메시지 무결성 확인을 위해 **HashValueChecker** 추가	DVM과 네트워크 서버 사이에서 메시지를 주고받을 때, 각 메시지의 hash value를 확인해서 외부 접근을 감지한다. Message를 송신하는 쪽에서 hash value를 추가하고, 이를 수신하기 전 hash value를 검사하여 메시지의 무결성을 확인한다. Deployment diagram을 변화시키며, sequence diagram 상에 변화는 있으나 작성하지 않는다.
외부 접근을 탐지하면, 관리자에게 알리기 위해 **NotifyManager** 추가	Hash value 검사를 통해 외부로부터 접근이 있었음을 탐지하면, 관리자에게 이를 알린다. 네트워크 관리자에게 실시간으로 알림으로써 DVM 시스템의 보안을 향상시킬 수 있다. Deployment diagram을 변화시키며, sequence diagram 상에 변화는 있으나 별도로 작성하지 않는다.

4. 3rd iteration

Step 6 Deployment diagram



Element	Responsability
HashValueChecker	Message를 주고 받을 때 Hash 값을 Message의 Integerity를 검사하여 외부 침입을 감지하면 NotifyManager로 전달한다.
NotifyManager	HashValueChecker에서 전달된 외부 침입을 관리자에게 전송한다.

4. 3rd iteration

Step 7

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During Iteration
		QA-1	이번 iteration에서 해당 Quality attribute를 만족하게 위하여 SW차원에서 기능을 일부 수정하였으므로 달성되었음
		QA-4	이번 iteration에서 멀티스레드를 사용하여 해당 Quality attribute를 만족시켰으므로 달성되었음
		QA-7	이번 iteration에서 Deployment pattern을 수정하면서 해당 Quality attribute를 만족시켰으므로 달성되었음
	CON-1		각 레이어 모듈에 따른 동작이 모든 DVM 에서 동일하게 작동하도록 설계되었음
	CON-2		Security와 관련된 QA-7의 영향을 받아, 일부 만족되었음
CON-3			아직 이 항목을 다룰 수 있을 정도로 Architecture가 구체화되지 않음
	CON-4		각 레이어 모듈에 따른 동작이 어떤 환경이든 모든 DVM 에서 동일하게 작동하도록 설계되었음
		CRN-2	Iteration 1,2,3 을 수행하면서 팀원들의 이해도를 높였으므로 달성되었음

Drivers that are completely addressed in previous iteration is removed

감사합니다 ☺