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.	UC-1,2,3,4,5
QA-2	Marketability	마케터는 출시 이전 분산 자판기에 대한 설문조사를 한다. 설문조사를 통해 사용자들의 니즈를 파악하여 DVM의 시장 경쟁성을 높인다.	UC-1,2,3,4,5
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-1,2,3,4,5
QA-4	Performance	하나의 자판기에서 여러 자판기로부터 최대 4개의 메시지를 수신한다. 메시지를 수신한 자판기는 60초 이내에 메세지를 모두 처리한다.	UC-3,4,6,7
QA-5	Maintainability	3명의 개발자가 하나의 컴포넌트 코드를 수정하고자 할 때, 3시간 이내에 수정 가능하다.	UC-1,2,3,4,5,6
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-3,4,6,7

1. From QAW - Primary Functionality

No.	Use-Case	Description
UC-1	Select Item	사용자가 선택한 메뉴의 재고가 있을 경우 사용자에게 '구매하시겠습니까?' 등의 메시지로 알리고, 재고가 없을 경우 사용자에게 '다른 자판기의 재고를 찾아보시겠습니까?' 등의 메시지로 알린다.
UC-2	Provide Item	사용자가 결제를 진행하고 나면 상품을 제공한다.
UC-3	Search Other DVM	사용자가 다른 자판기에서 재고를 찾겠다고 선택했을 때, 선택된 상품의 재고 여부를 가장 가까운 자판기부터 순서대로 요청하면, 메시지를 받은 자판기는 상품의 재고 여부를 확인한 후 메시지로 응답한다. 재고가 있는 경우, 사용자에게 해당 자판기의 위치를 알리고, 선결제를 진행할 것인지 묻는다.
UC-4	Provide Verification Code	사용자가 선결제를 하면 인증코드를 생성하고 사용자와 해당 자판기에 제공한다.
UC-5	Verify Verification Code	사용자가 다른 자판기에서 선결제를 통해 받은 인증코드를 입력하면, 제공받았던 인증코드와 동일한 지 그 유효성을 확인한다. 유효한 인증코드라면, 사용자에게 해당 상품을 제공한다.
UC-6	Set Address	관리자가 자판기의 주소를 입력한다.
UC-7	Network Monitoring	관리자가 자판기 사용이력을 본다.

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	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

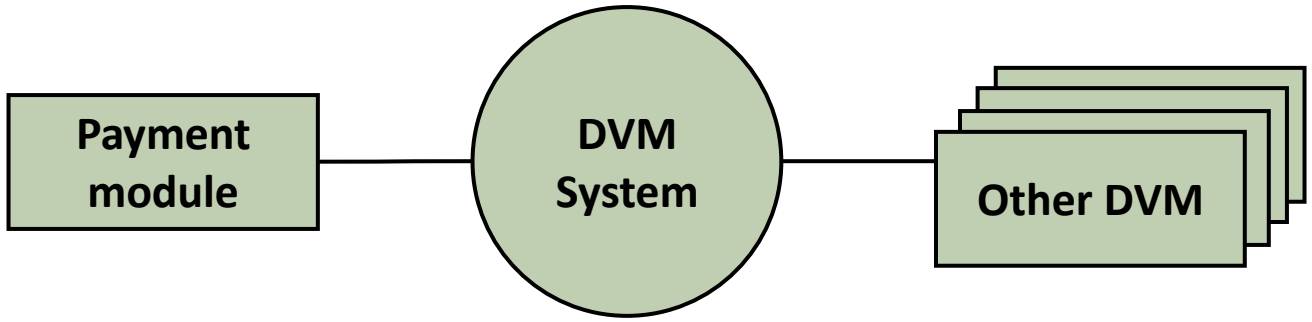
Step 2

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

Step 3

Context diagram for DVM system

(각 DVM에는 payment module이 연결되어 있음)



2. 1st iteration

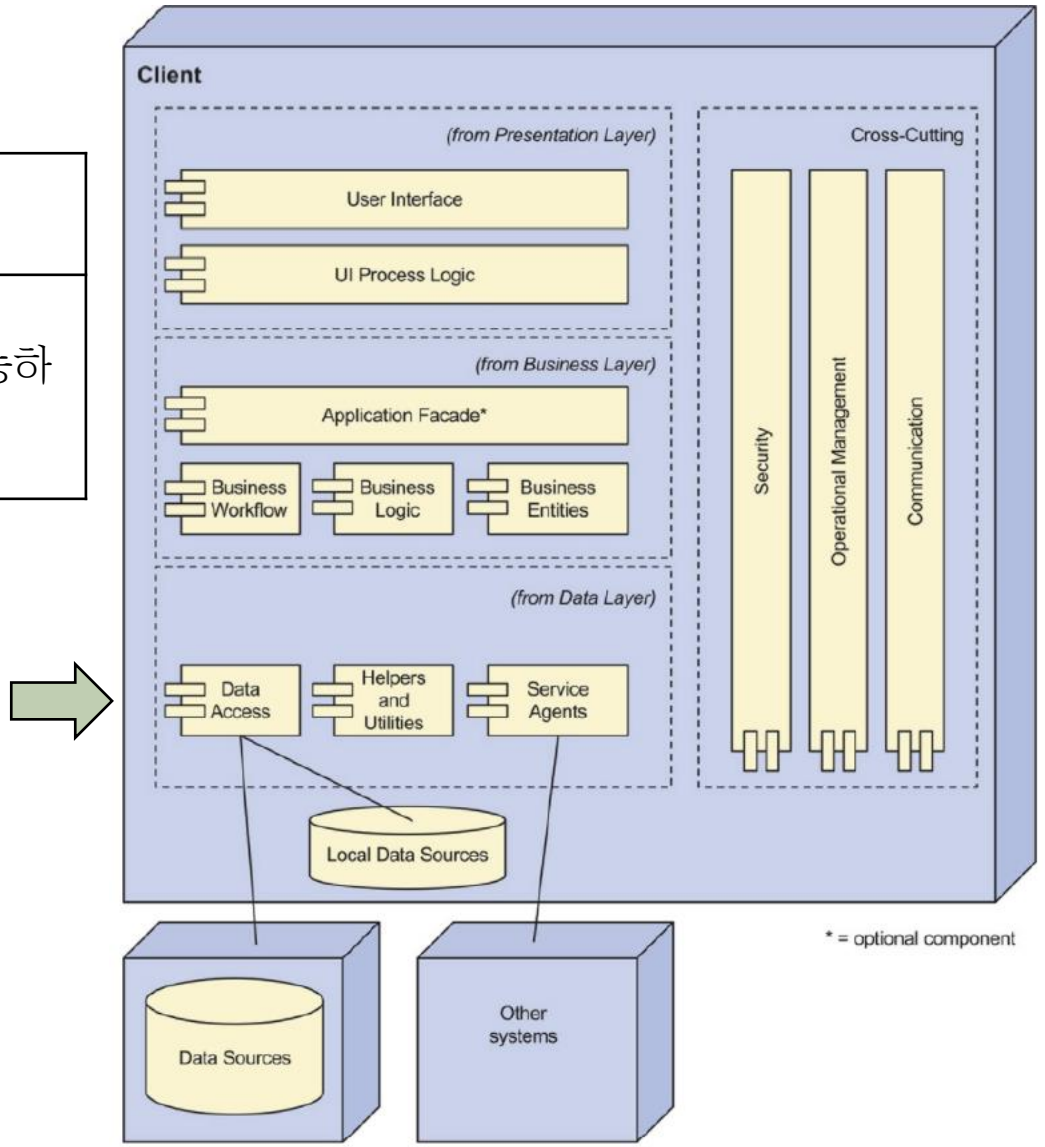
Step 4 Design Concept(reference architecture) 선정

<table>
<tr><th>Design Decisions and Location</th><th colspan="2">Rationale</th></tr>
<tr><td rowspan="6">Logical structure
각 DVM에는
Rich Client Application reference architecture 를 활용한다.</td><td colspan="2">Rich Client application은 자판기에 설치되는 프로그램의 개발에 도움을 준다. 또한, QA-1의 usability를 향상시킬 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>
<tr><td>Service Application</td><td>서비스를 제공해주기 위한 단계가 따로 필요하지 않으므로 사용하지 않는다.</td></tr>
<tr><td>Physical structure
Non-distributed Deployment Pattern을 활용한다.</td><td colspan="2">CON-1에 따라 여러 자판기에서 동일한 동작을 하도록 할 필요가 있고, QA-1에 따라 사용자에게 빠르게 서비스를 제공하도록 할 필요가 있으므로 DVM 자체를 하나의 server로 고려하는 non-distributed deployment pattern을 사용한다.</td></tr>
<tr><td>외부 카드 결제 Module을 사용하여 Payment Module 부분을 채운다.</td><td colspan="2">이미 기존에 존재하는 결제 Module 시스템만으로도 충분히 좋은 성능을 낼 수 있고, 개발하는 데에 추가적인 비용이 발생하므로 외부 Module을 DVM 시스템에 결합하여 사용한다.</td></tr>
</table>

2. 1st iteration

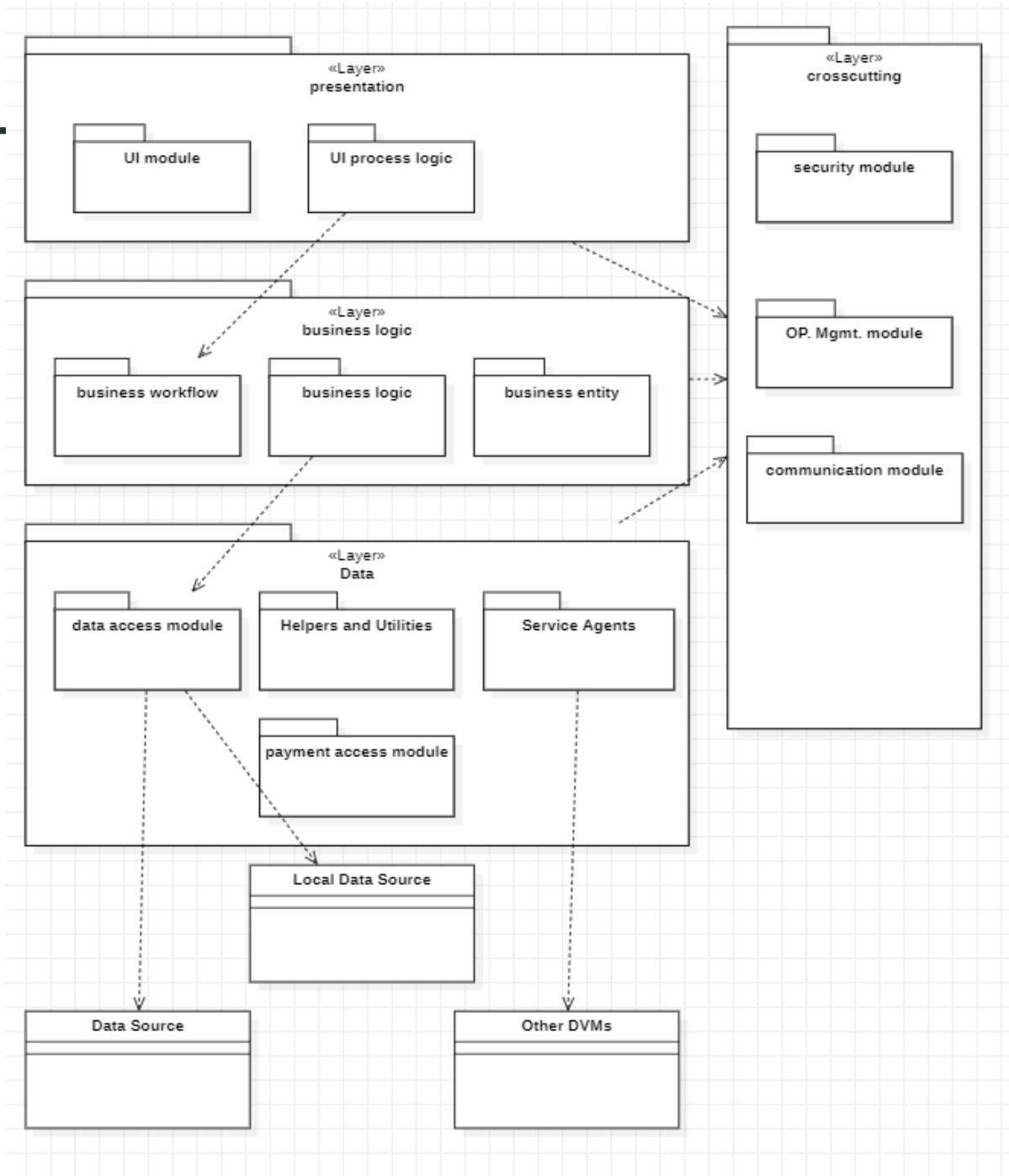
Step 5 Instantiate design decisions

Design Decision and Location	Rationale
Rich client application에 외부 결제 액세스 Module 추가	Data layer에 외부 시스템을 거쳐서 결제 가능하도록 외부 결제 액세스 Module을 추가한다.



2. 1st iteration

Step 6 Sketch Module View



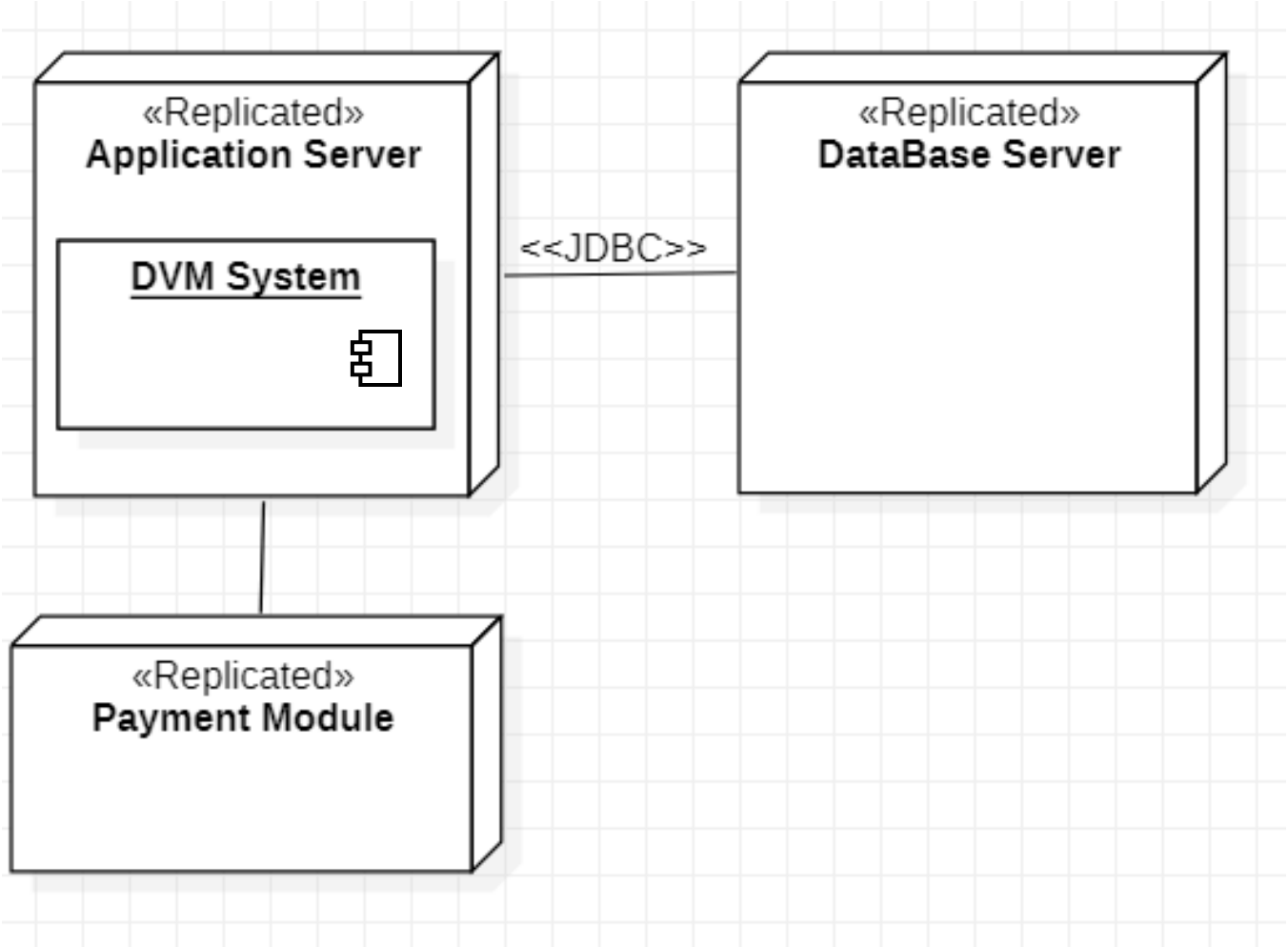
2. 1st iteration

Step 6 Element description – Module View

Element	Responsibility
UI module	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 module	Local DB에 데이터를 저장하고 읽어온다.
Helpers and Utilities	Server Side와 연결한다.
Service Agents	다른 DVM과 연결한다.
Payment Access Module	실제 결제를 수행하기 위하여 결제 모듈과 연결한다.
Security Module	보안 역할을 한다.
Operation Management Module	Operation을 관리한다.
Communication Module	Layer들을 연결하고 소통하게 해준다.

2. 1st iteration

Step 6 Sketch Deployment Diagram



2. 1st iteration

Step 6 Element description **– Deployment Diagram**

Element description

Element	Responsibility
Database Server	DVM Server로부터 받은 데이터를 저장하고, 요청이 있을 경우 제공한다.
Application Server	DVM 자체에 설치되는 UI 및 SW
Payment Module	DVM에서 요청한 결제를 수행한다.

Relationship	Description
Database Server와 Application Server 사이에 JDBC protocol로 연결	Database Server에서는 Application Server로부터 받은 데이터를 실제 데이터로 저장해야 하므로, JDBC protocol을 사용하여 둘 사이를 연결한다.

2. 1st iteration

Step 7 Addressing된 정도 분석

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During Iteration
UC-1			아직 이 항목을 다룰 수 있을 정도로 Architecture가 구체화되지 않음
	UC-3		자판기끼리 서로 연결되어 있음을 Context Diagram과 Module view를 통해 확인할 수 있으므로 일부 달성되었음
UC-4			아직 이 항목을 다룰 수 있을 정도로 Architecture가 구체화되지 않음
	UC-5		아직 이 항목을 다룰 수 있을 정도로 Architecture가 구체화되지 않음
	QA-1		Rich client application을 reference architecture로 선정하여 사용자에게 좀 더 향상된 Usability를 제공할 수 있도록 하였으므로 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 자판기에서 동일한 Architecture을 갖추고 있도록 하였으므로 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
		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-1	Select Item	사용자가 선택한 메뉴의 재고가 있을 경우 사용자에게 '구매하시겠습니까?' 등의 메시지로 알리고, 재고가 없을 경우 사용자에게 '다른 자판기의 재고를 찾아보시겠습니까?' 등의 메시지로 알린다.
UC-3	Search Other DVM	사용자가 다른 자판기에서 재고를 찾겠다고 선택했을 때, 선택된 상품의 재고 여부를 가장 가까운 자판기부터 순서대로 요청하면, 메시지를 받은 자판기는 상품의 재고 여부를 확인한 후 메시지로 응답한다. 재고가 있는 경우, 사용자에게 해당 자판기의 위치를 알리고, 선결제를 진행할 것인지 묻는다.
UC-4	Provide Verification Code	사용자가 선결제를 하면 인증코드를 생성하고 사용자와 해당 자판기에 제공한다.
UC-5	Verify Verification Code	사용자가 다른 자판기에서 선결제를 통해 받은 인증코드를 입력하면, 제공받았던 인증코드와 동일한 지 그 유효성을 확인한다. 유효한 인증코드라면, 사용자에게 해당 상품을 제공한다.

3. 2nd iteration

Step 3

Refine할 System Element 선정: 없음!

Step 4

Architectural driver를 만족하는 Design Concept(Pattern) 선정
Architectural pattern: **Forwarder-Receiver**(DVM간 통신) → 직/간접적으로 연관됨

- 선택한 pattern를 이해하고 module view에 적용하기 위해 structure와 sequence diagram을 파악한다.
- Use case로부터 Class diagram을 작성하여 module view에서 각 layer에 포함되어야 할 class를 파악
- 추가로 사용할 Externally developed component는 **Spring Framework**이다.

3. 2nd iteration

Step4

Forwarder-and Receiver Pattern :

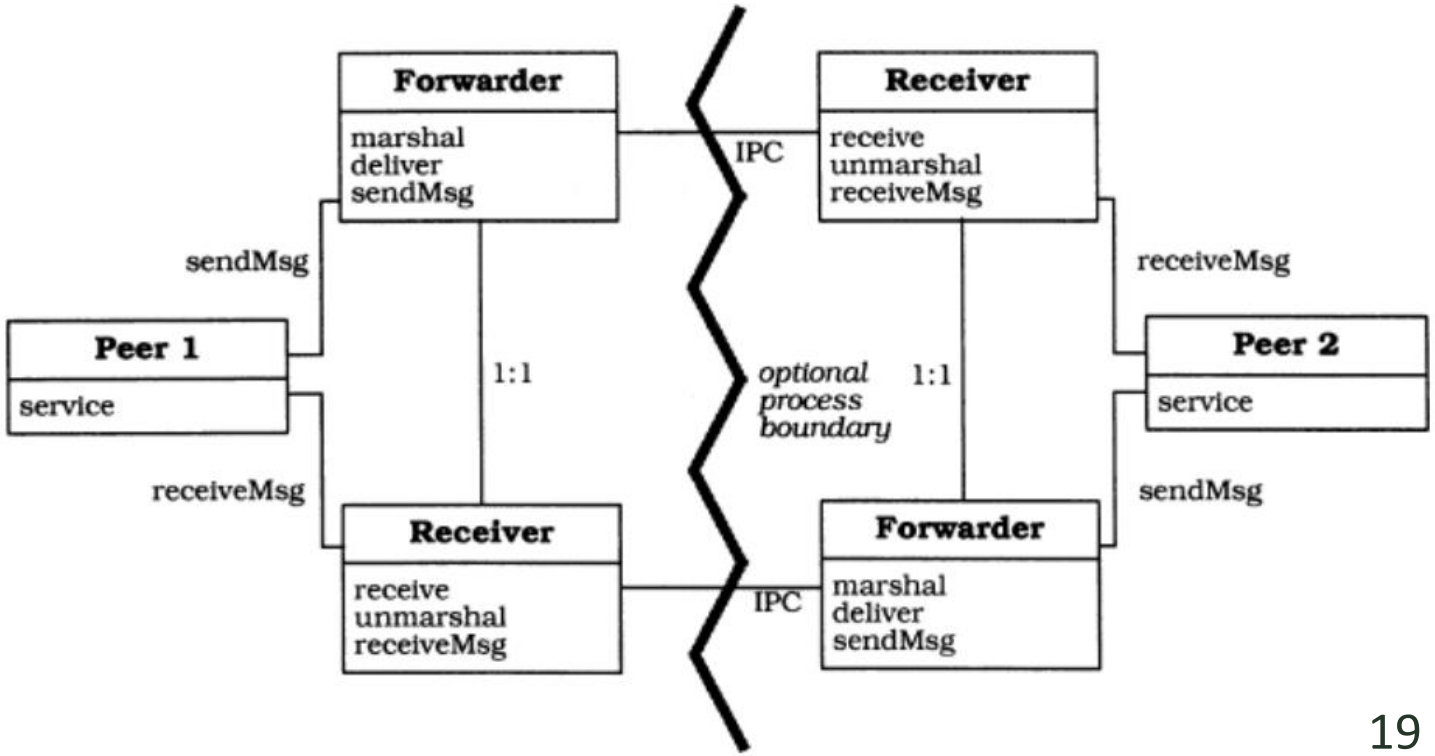
Forwarder Component

- 메시지를 Marshaling 하고 Optional Process boundary에 전달
- 메시지 전달을 위한 General Interface를 제공
- Name 와 Physical Address를 매핑.

Receiver Component

- 메시지를 받기 위한 General Interface를 제공
- Unmarshalling

Structure(static)



3. 2nd iteration

Step 4 선택한 Design concepts

Design Decisions and Location	Rationale and Assumptions
DVM간 통신에서 **Forwarder-Receiver** Pattern 사용	각 DVM간 peer-to-peer 통신을 하도록 하면 DVM간 통신에 있어서 효율적인 inter-process communication이 가능하고 IPC 기능을 encapsulate 이에 가장 적절한 forwarder-receiver pattern을 사용한다.
각 Use-Case를 위한 **Class Diagram** 작성	각 use case를 가지고 class diagram을 작성한다. 이때 각 class는 use case에 관여하는 object들을 먼저 파악한 후 작성된다.
Spring Framework 사용	Spring은 기업체에서 흔히 쓰는 Framework이다. Spring을 이용하면 본 시스템에서 외부적으로 사용해야 하는 JDBC API와, Payment Module(Paypal) API를 사용할 수 있다.

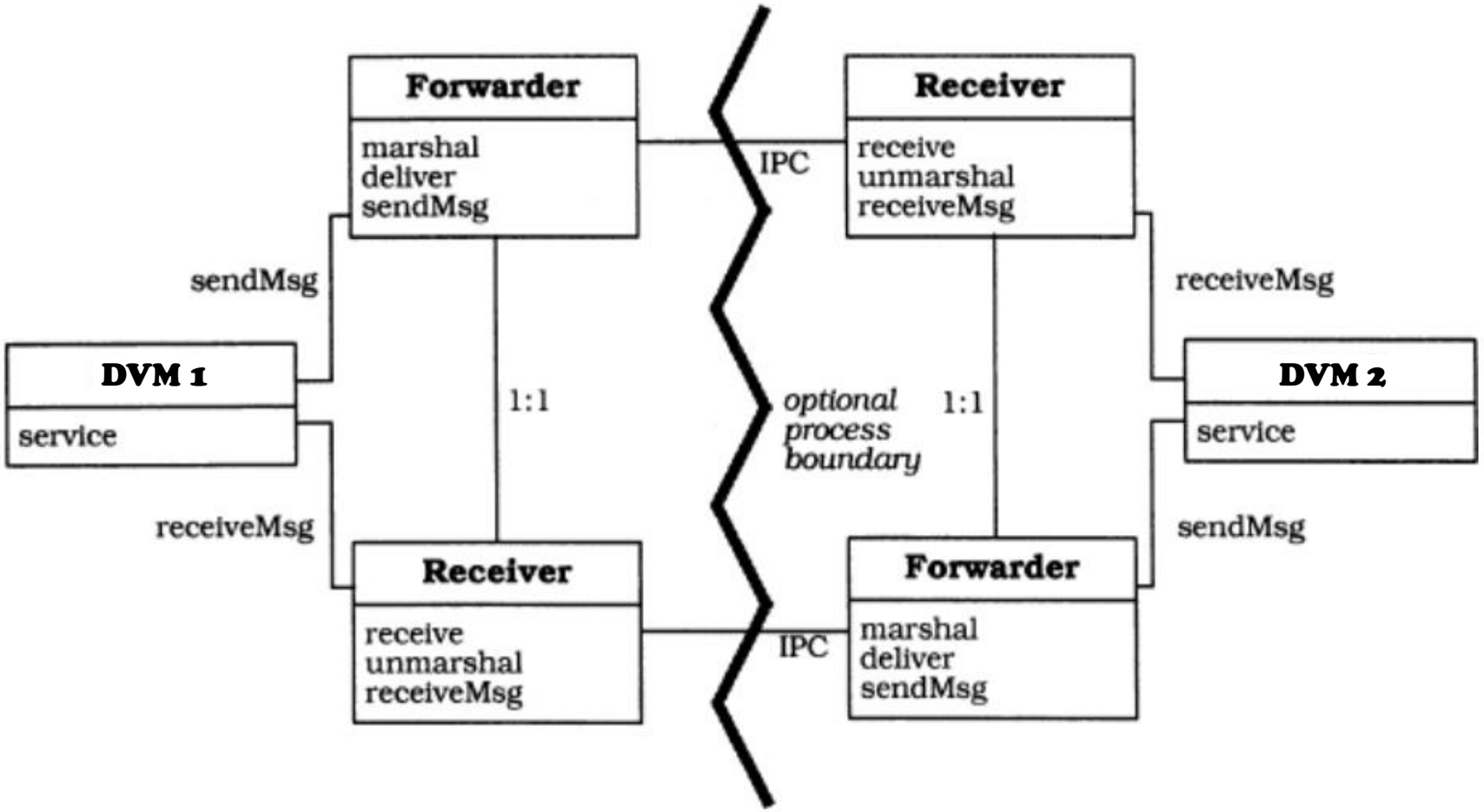
3. 2nd iteration

Step 5 Instantiate design decisions

Design Decisions and Location	Rationale
DVM System에 사용될 **Forwarder-Receiver Sequence Diagram** 작성	Forwarder-Receiver Pattern에 있는 Sequence Diagram을 응용하여 작성한다. 위 Diagram을 통해 DVM간의 통신을 설명하고, 효율적인 통신을 가능하게 한다.
DVM System의 **Forwarder-Receiver Structure Diagram** 작성	Forwarder-Receiver Pattern에 있는 Structure Diagram을 응용하여 작성한다. 위 Diagram을 통해 DVM간의 통신을 설명하고, 효율적인 통신을 가능하게 한다. Structure Diagram을 이용해 Class Diagram을 작성한다.
Class diagram을 작성하여 Module view에서의 각 layer를 class로 치환	Use-Case를 참고하여 Class Diagram을 그린다. 작성된 Class Diagram을 참고하여 1 iteration에서 생성된 Module View 수정한다. Module View와 Class Diagram을 통해 DVM System에 대한 이해도를 높인다.
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.

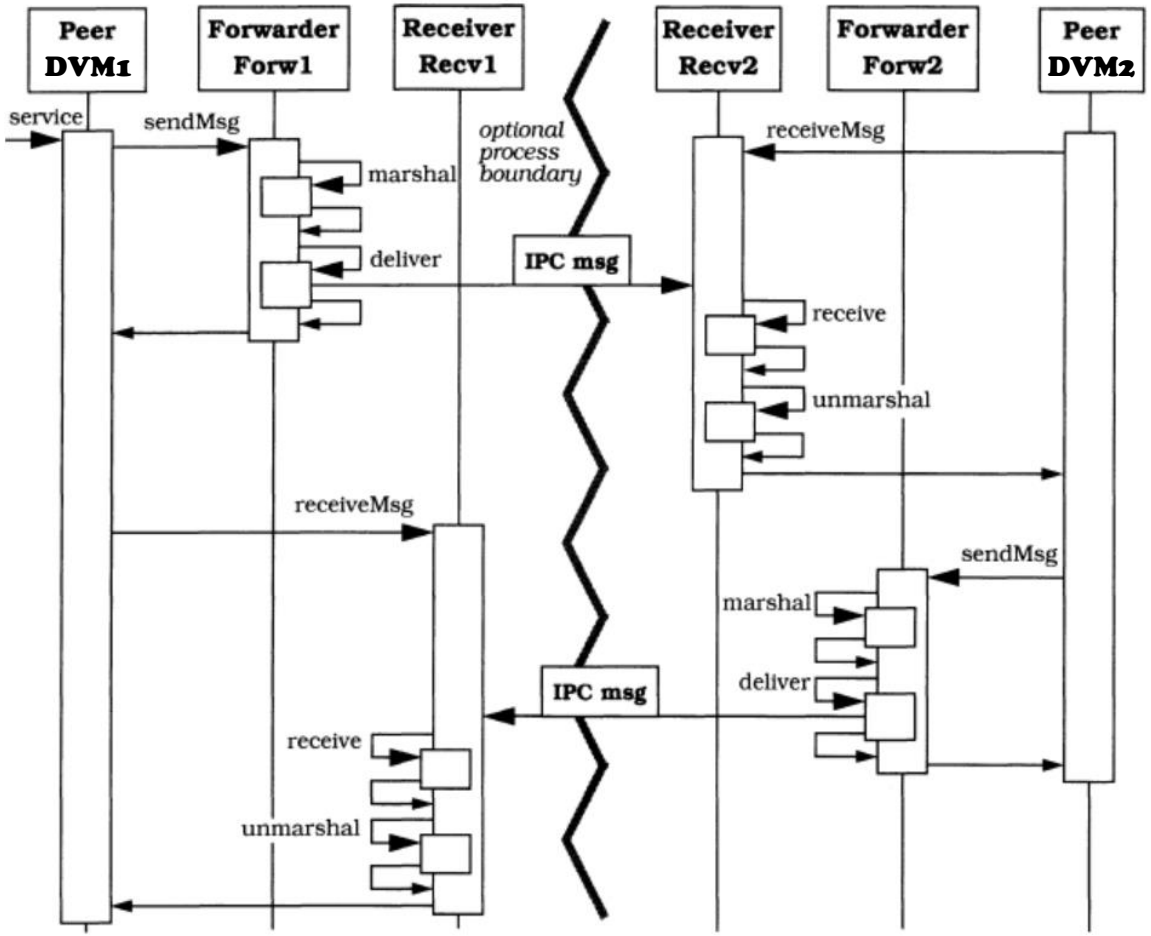
3. 2nd iteration

Step 6 Sketch Structure Diagram



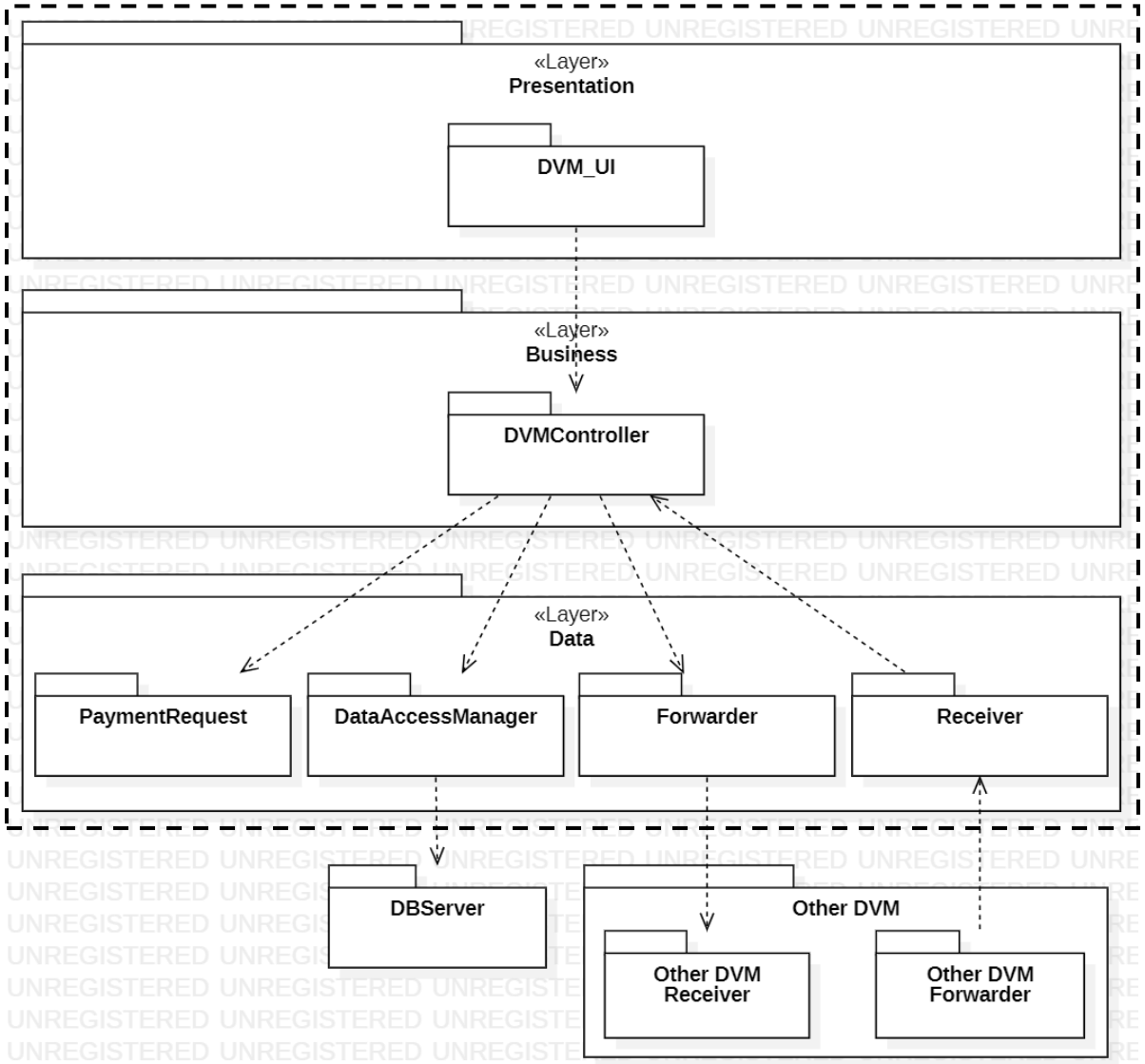
3. 2nd iteration

Step 6 Sketch Sequence diagram



3. 2nd iteration

Step 6 Sketch Module View(Refined)



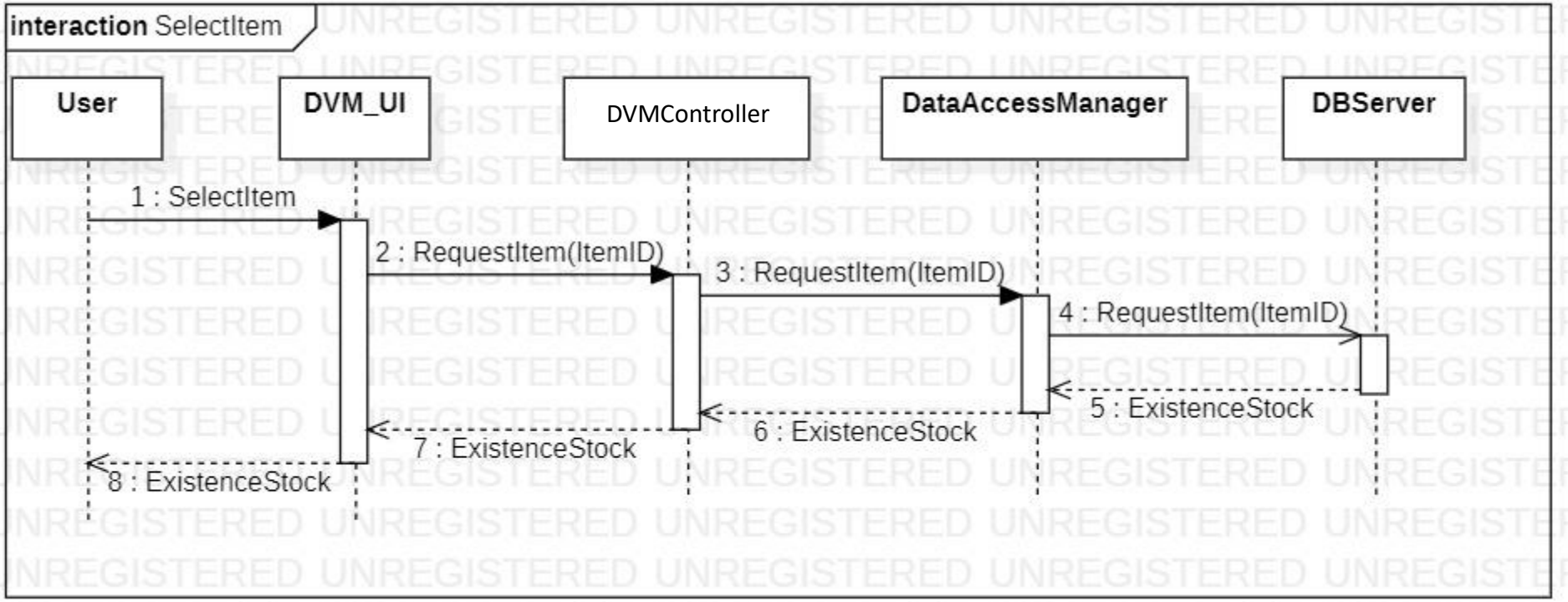
3. 2nd iteration

Step 6 Element Description – Module View(Refined)

Element	Responsibility
DVM_UI	사용자와 직접 interaction이 가능한 UI로, 사용자가 선택할 수 있는 상품을 보여주고, 결제를 완료한 상품을 제공하거나 다른 자판기에서 사용 가능한 인증코드를 제공한다.
DVMController	사용자의 입력에 따른 전체적인 동작을 위한 기능을 포함한다.
PaymentRequest	사용자가 (선)결제를 요청할 경우, 외부 결제 모듈에 결제를 요청을 한다.
DataAccessManager	각 DVM의 위치, DVM의 ID, 취급하는 상품 종류, 각 상품의 재고에 대한 정보와 송수신한 인증번호를 DBServer에 저장하거나 불러온다.
Forwarder	전송할 정보를 파일의 형태로 marshalling하여 다른 자판기의 receiver로 전송한다.
Receiver	다른 자판기의 forwarder로부터 파일의 형태로 수신한 정보를 unmarshalling하여 자판기에 전달한다.
DBServer	각 자판기의 log 정보와 위치, 식별번호, 상품 종류 정보를 가지고 있다.

3. 2nd iteration

Step 6 Sequence Diagram for UC-1



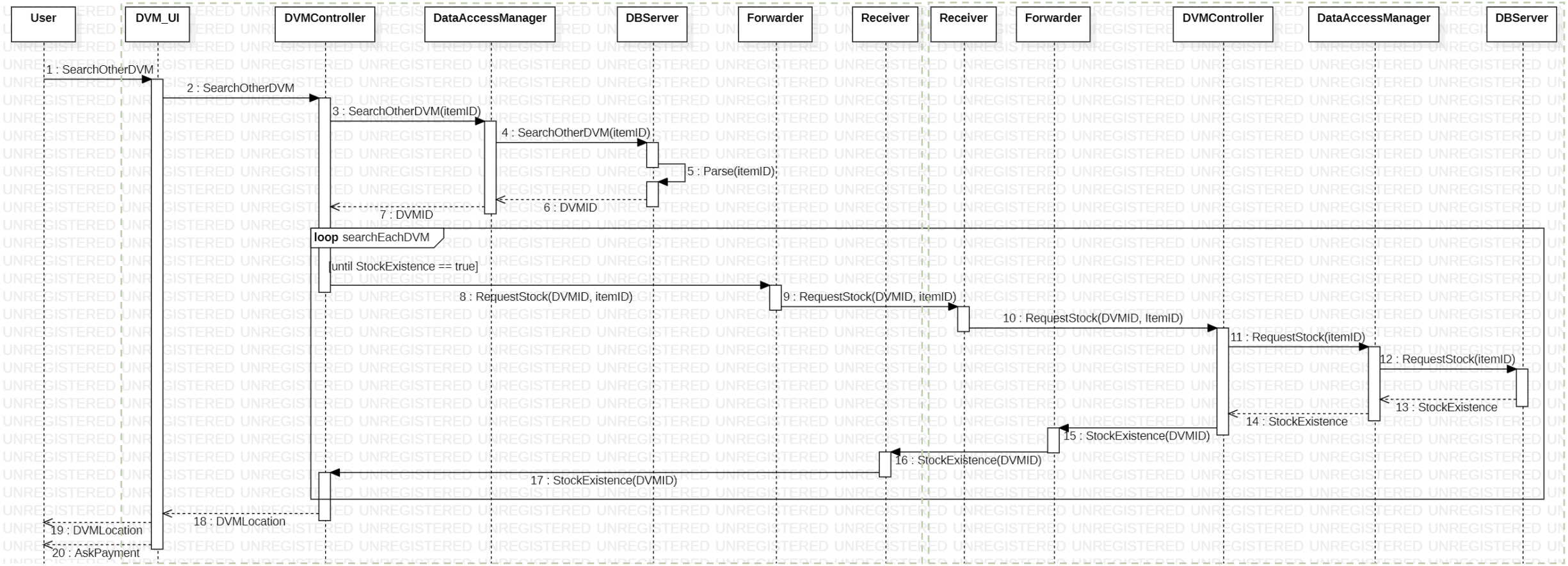
3. 2nd iteration

Step 6 Interface Description for UC-1

Element	Method Name	Responsibility
DVM_UI	SelectItem	User가 원하는 품목의 재고여부를 알기 위해 DVM_Controller에 요청하고 재고 여부를 리턴한다.
DVMController	RequestItem (itemID)	요청 받은 품목의 재고여부를 알기 위해 DataAccessManager에 요청하고 재고 여부를 리턴한다.
DataAccessManager	RequestItem (itemID)	요청 받은 품목의 재고여부를 알기 위해 DBServer에 요청하고 재고 여부를 리턴한다.
DBServer	RequestItem (itemID)	DVM에 선택된 품목의 재고가 남아있는지 확인하고 있으면 ExistenceItem을 True로 , 없으면 False로 하여 리턴 한다.

3. 2nd iteration

Step 6 Sequence Diagram for UC-3



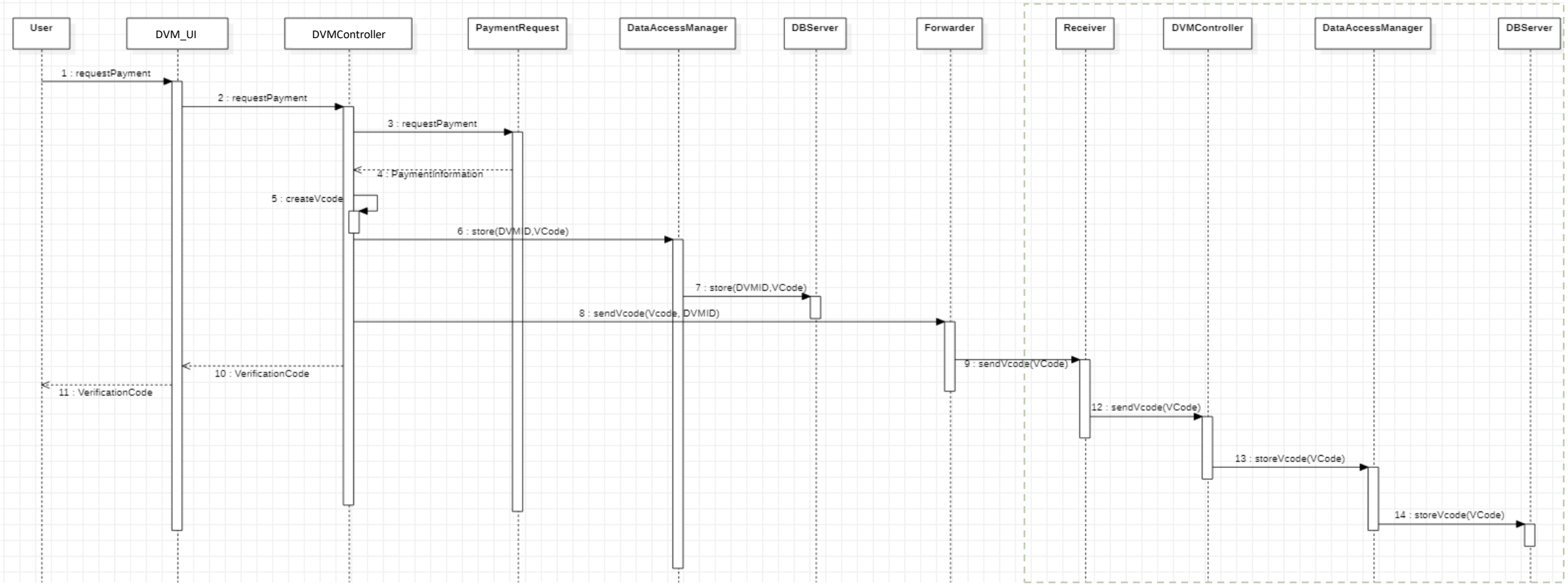
3. 2nd iteration

Step 6 Interface Description for UC-3

Element	Method Name	Responsibility
DataAccessManager DBServer	SearchOtherDVM(itemID)	사용자가 선택한 item을 취급하고, 해당 item의 재고가 있는 다른 DVM을 itemID를 통해 찾는다.
DBServer	Parse(itemID)	itemID를 가지는 item을 취급하는 DVM을 찾는다.
Forwarder	RequestStock(DVMID, itemID)	Item을 취급하는 DVM의 DVMID를 가지고 해당 DVM의 Receiver에 재고 여부를 요청한다.
Receiver DVMController	RequestStock(DVMID, itemID)	Item의 재고 여부를 요청한 DVM의 ID와 재고 여부를 확인해야 하는 itemID를 가진다.
DataAccessManager DBServer	RequestStock(itemID)	itemID를 가지고 재고 여부를 요청한다.
Forwarder	StockExistence(DVMID)	재고 여부를 전달받아야 하는 DVM의 DVMID를 가지고 해당 DVM의 Receiver에 재고 여부를 전달한다.
Receiver DVMController	StockExistence(DVMID)	재고 여부를 전달한 DVM의 DVMID를 가지고 해당 DVM이 재고를 가지고 있음을 알린다.

3. 2nd iteration

Step 6 Sequence Diagram for UC-4



DVM2

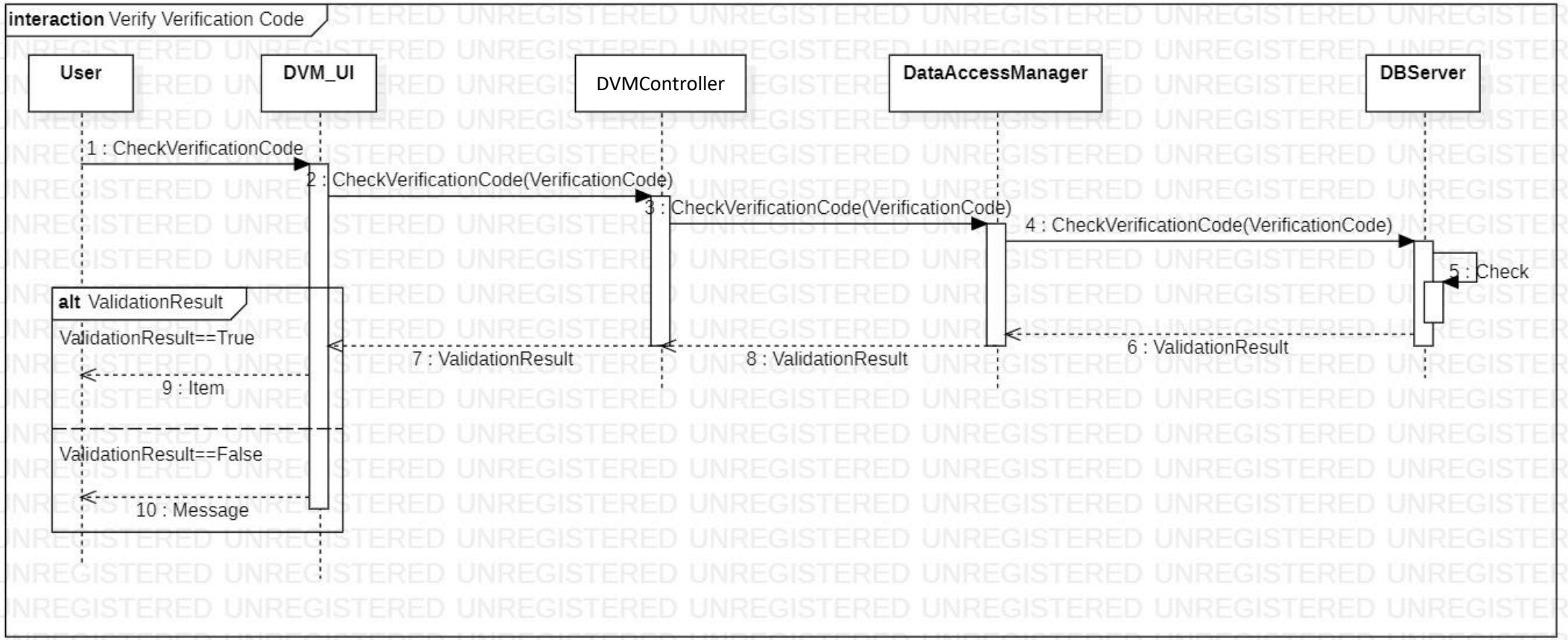
3. 2nd iteration

Step 6 Interface Description for UC-4

Element	Method Name	Responsibility
DVM_UI	requestPayment	User가 인증코드를 발급받기 위하여 선결제를 진행하고 인증코드를 리턴한다.
DVMController	requestPayment	User의 결제 요청을 전달하고 인증코드를 리턴한다.
	createVcode	결제 정보를 바탕으로 인증코드를 생성한다.
PaymentRequest	requestPayment	User의 결제를 요청하고 결제에 대한 정보를 리턴한다.
DataAccessManager	store(DVMID,Vcode)	User가 음료를 받을 DVM의 ID와 인증코드를 저장한다.
DBServer	store(DVMID,Vcode)	User가 음료를 받을 DVM의 ID와 인증코드를 저장한다.
Forwarder	sendVcode(Vcode,DVMID)	User가 음료를 받을 DVM의 ID와 인증코드를 전달한다.
Receiver	sendVcode(Vcode)	다른 DVM에서 해당 DVM으로 인증코드를 전달한다.
DVMController	sendVcode(Vcode)	다른 DVM에서 해당 DVM으로 인증코드를 전달한다.
DataAccessManager	storeVcode(Vcode)	User가 음료를 받기 위한 인증코드를 저장한다.
DBServer	storeVcode(Vcode)	User가 음료를 받기 위한 인증코드를 저장한다.

3. 2nd iteration

Step 6 Sequence Diagram for UC-5



3. 2nd iteration

Step 6 Interface Description for UC-5

Element	Method Name	Responsibility
DVM_UI	CheckVerificationCode	User가 입력한 Verification Code가 DVM이 가지고 있는 올바른 Verification Code인지 확인하고 맞으면 Verification Code의 Item을, 틀리면 잘못된 Verification Code라는 메시지를 보낸다.
DVMController	CheckVerificationCode (VerificationCode)	Verification Code가 DVM이 가지고 있는 올바른 Verification Code인지 DataAccessManger에게 요청한다. 올바른 코드이면 True를 리턴한다.
DataAccessManager	CheckVerificationCode (VerificationCode)	Verification Code가 DVM이 가지고 있는 올바른 Verification Code인지 DBServer에게 요청한다. 올바른 코드이면 True를 리턴한다.
DBServer	CheckVerificationCode (VerificationCode)	Verification Code가 DVM이 가지고 있는 올바른 Verification Code이면 True를, 아닌 경우 False를 리턴한다.
	Check	Verification Code가 DB Server가 가지고 있는 Verification Code인지 확인한다.

3. 2nd iteration

Step 7 Addressing된 정도 분석

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During Iteration
		UC-1	본 use case를 달성하기 위한 각 layer의 module과 interface들을 전부 식별하였음
		UC-3	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-5	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	Peer-to-Peer 통신을 하므로 각 DVM간 구분이 없기 때문에 모든 DVM에서 동일 동작을 할 것이라고 예상할 수 있음
	CON-2		Forwarder-Receiver pattern을 사용하여 특정 interface를 충족할 때만 메시지를 수신하고 unmarshalling하게 되었으므로 일부 달성되었음
		CON-3	Peer-to-Peer 통신을 하므로 각 DVM간 구분이 없기 때문에 동일한 환경에서 동일한 동작을 할 것이라고 예상할 수 있음
	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
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.
QA-4	Performance	하나의 자판기에서 여러 자판기로부터 최대 4개의 메시지를 수신한다. 메시지를 수신한 자판기는 60초 이내에 메세지를 모두 처리한다.
QA-7	Security	외부시스템에서 시스템 탈취를 시도하거나 네트워크로 전달되는 메시지에 접근하고자 할 때 , 30초 이내에 접근을 감지 후, 5초 이내에 관리자에게 위험 메세지를 전달한다.

ID	Constraint
CON-2	네트워크는 낮은 대역폭을 가질 수 있지만, 신뢰할 수 있어야 한다.

ID	Concern
CRN-2	구성원이 System Domain에 대한 이해도를 높여야 한다.

Step 3 시스템에서 수정 되어야 할 Element

Quality Attribute와 constraint를 만족시키기 위해 Application Server가 수정되어야 한다

4. 3rd iteration

Step 4

Design goal을 만족하기 위한 Design concept 선정

Design Decisions and Location	Rationale and Assumptions
UI 차원에서 **Cancel**을 사용	사용자가 사용 중간에 취소할 수 있게 만들어 Usability를 향상시킨다. 사용자가 결제하기 전 취소 가능하도록 한다.
Receiver로 수신한 메시지에 **Prioritize Event** 적용	다른 자판기로부터 메시지를 수신할 때, Receiver가 처리해야 하는 메시지의 우선순위를 부여하여 처리함으로써 Performance를 향상시킨다. 처리 우선 순위: 재고 확인 메시지 > 인증코드 수신
Forwarder-Receiver와 DataAccessManager에서 **Encrypt Data** 적용	Asymmetric Encryption을 사용하여 Data를 저장할 때는 각 DVM에서 Private key를 통해 Encryption을 진행하고, Data를 전송할 때는 Public key를 통해 Encryption을 진행하여 전송한다. 이를 통해 Security를 향상시킨다.

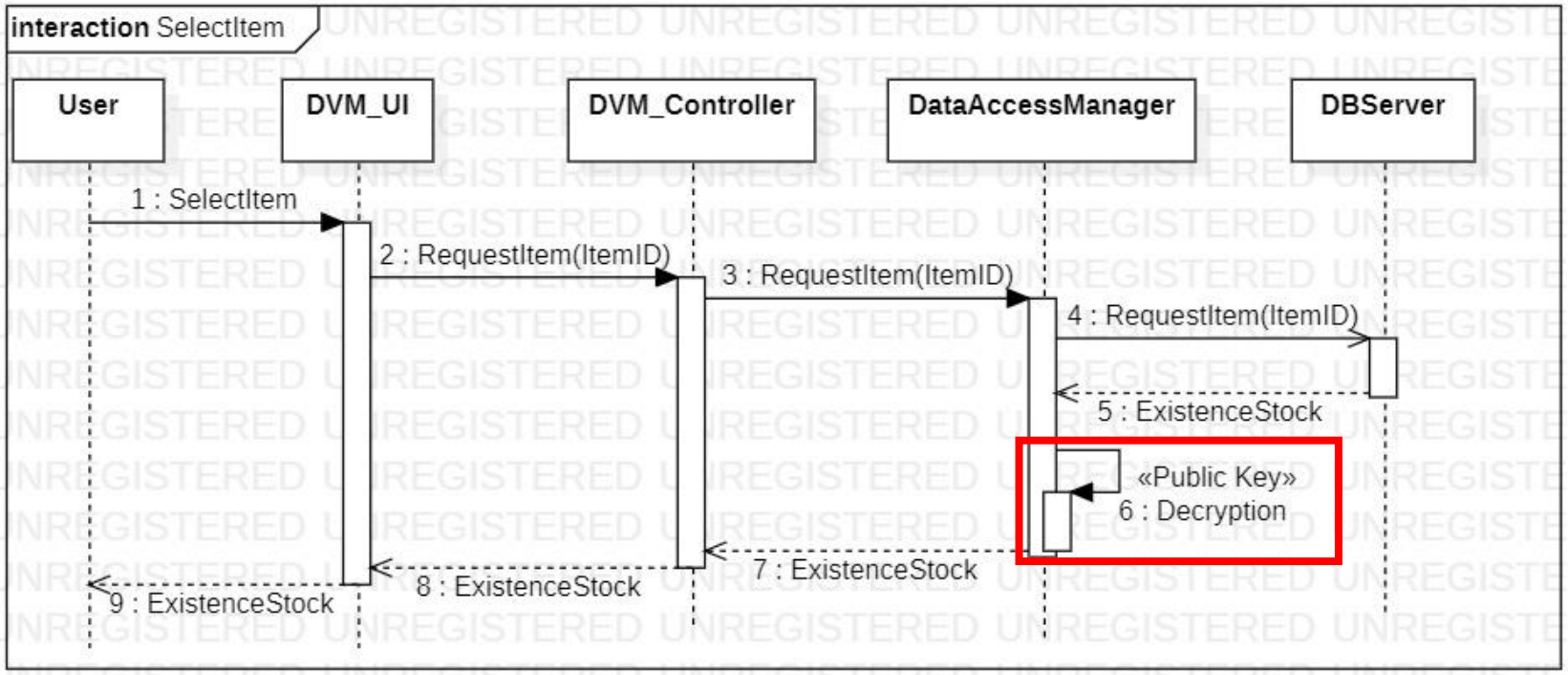
4. 3rd iteration

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 입력한 정보들을 전부 초기화하고 메인 화면으로 돌아가야 하므로, UI는 내부의 controller와는 Asynchronous하게 동작해야 한다. Cancel에 대한 Use-Case를 추가하고 해당 Use-Case의 Sequence diagram를 생성한다. Usability가 향상된다.
Receiver로 수신한 메시지를 처리할 때 **Priority**를 부여하여 처리	다른 자판기로부터 메시지를 수신할 때, Receiver가 처리해야 하는 메시지의 우선순위를 부여하여 처리한다. Receiver는 메시지를 처리할 때 아래와 같은 우선순위를 토대로 처리하게 되며, sequence diagram에 변화가 생기고, Performance가 향상된다. 처리 우선 순위: 재고 확인 메시지 > 인증코드 수신
Forwarder-Receiver사이와 DataAccessManger 에 메시지 무결성 확인을 위해 **Private Key, Public Key** 적용	Forwarder가 메시지를 보내기 전 Public Key를 통해 Encryption을 진행하고, Receiver가 메시지를 수신할 때 Public Key를 통해 수신한 메시지의 Decryption을 진행한다. Data를 DBServer에 저장할 때, DataAccessManager가 Private Key를 통해 Encryption을 진행하고, Data를 DBServer로부터 읽어 오려고 할 때, DataAccessManager가 Private Key를 통해 Decryption을 진행한다. Data를 암호화할 때, DVMID와 itemID를 제외한 내용만 암호화하여 데이터의 접근성은 유지하되, security는 향상시킬 수 있도록 한다. Sequence Diagram, Module View에 변화가 생긴다.

4. 3rd iteration

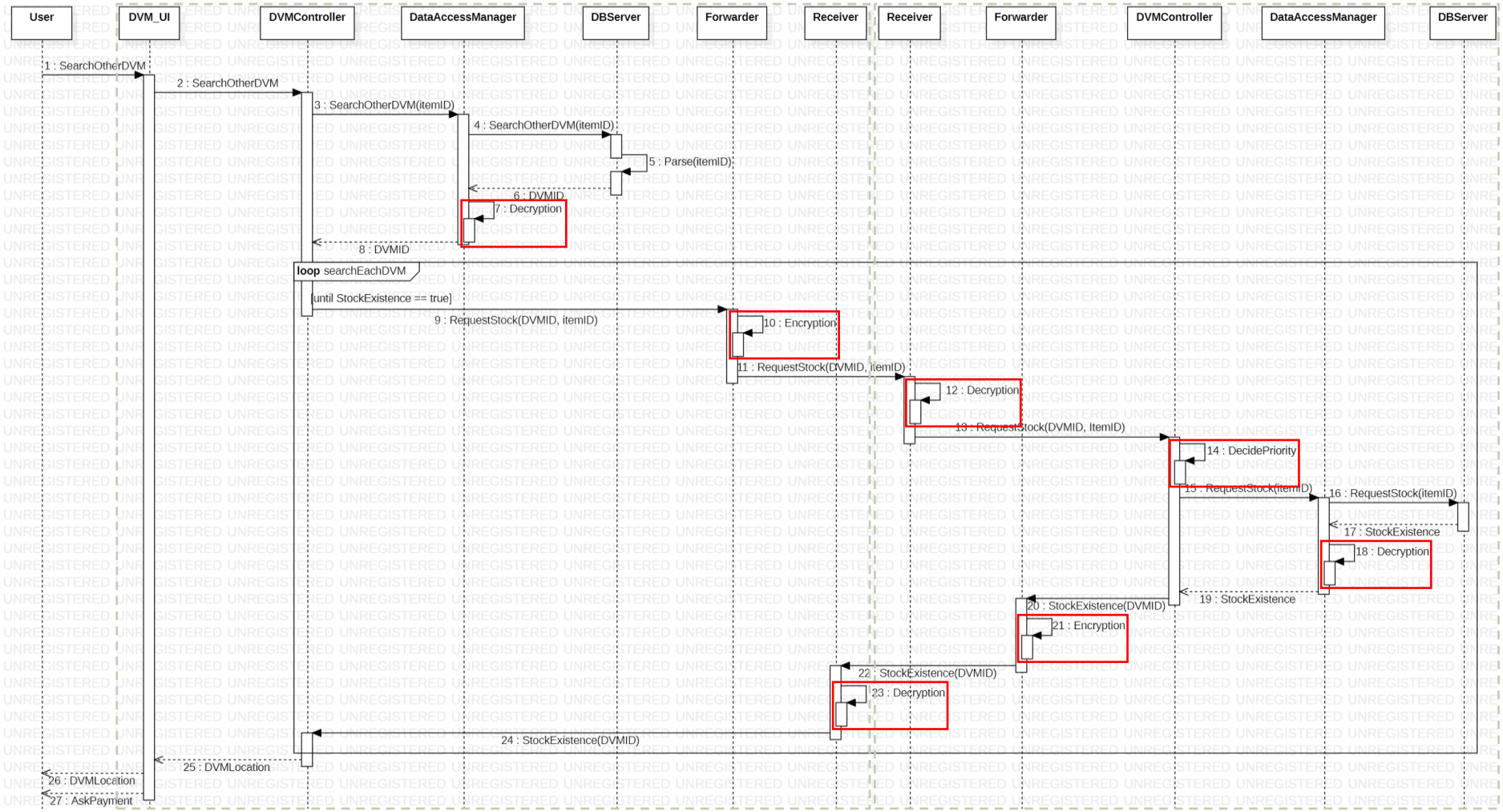
Step 6 Refine Sequence diagram(UC-1)



Element	Method Name	Responsibility
DataAccessManager	Decryption	DataServer에서 받아온 재고 여부가 암호화 되어 있기 때문에, 이를 읽고 쓰기 위해 Private Key로 풀어준다.

4. 3rd iteration

Step 6 Refine Sequence diagram(UC-3)



DVM1 DVM2

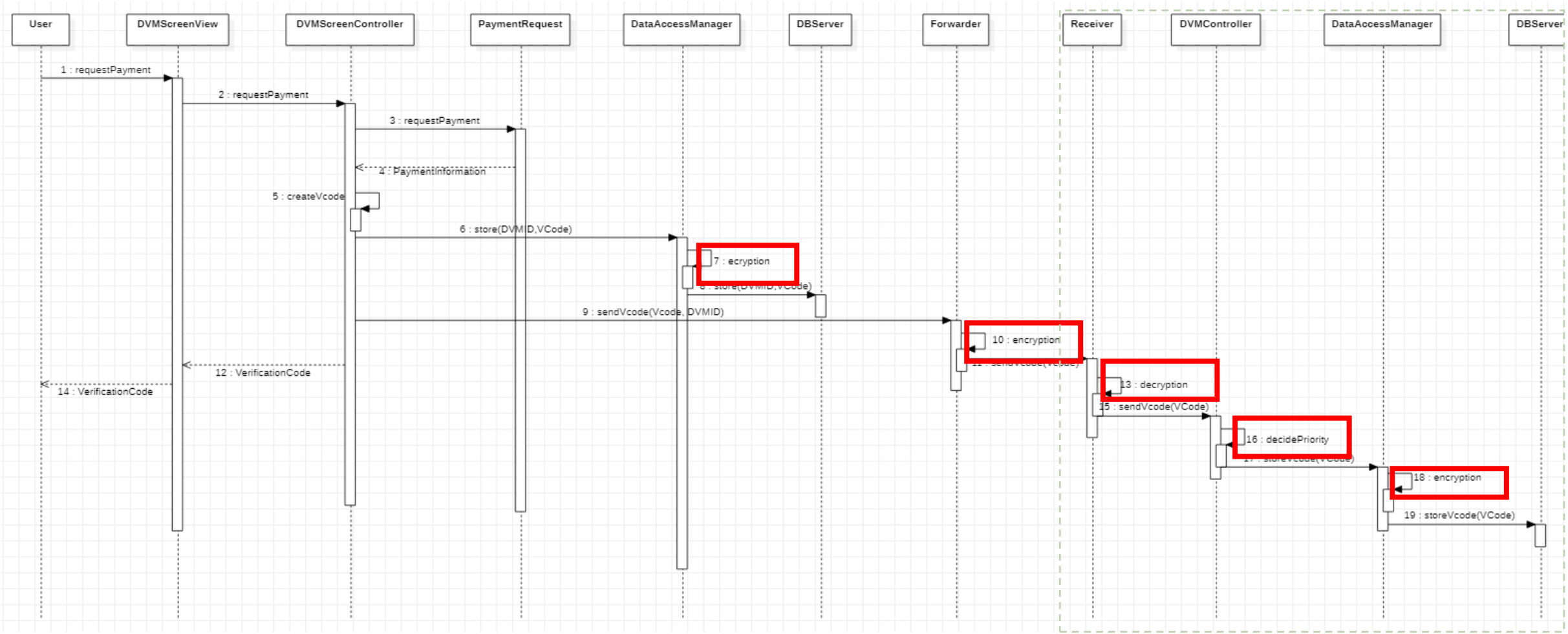
4. 3rd iteration

Step 6 Refine Sequence diagram(UC-4)

Element	Method Name	Responsibility
DataAccessManager DataAccessManager	Decryption	DBServer에 암호화하여 저장해둔 정보를 읽어오기 위해 private key로 복호화한다.
Forwarder Forwarder	Encryption	다른 DVM의 Receiver로 정보를 보내기 위해 public key로 암호화한다.
Receiver Receiver	Decryption	다른 DVM의 Forwarder로부터 받은 정보를 public key로 복호화한다.
DVMController	decidePriority	다른 DVM으로부터 들어온 요청들의 처리에 관한 우선순위를 정한다.

4. 3rd iteration

Step 6 Refine Sequence diagram(UC-4)



DVM2

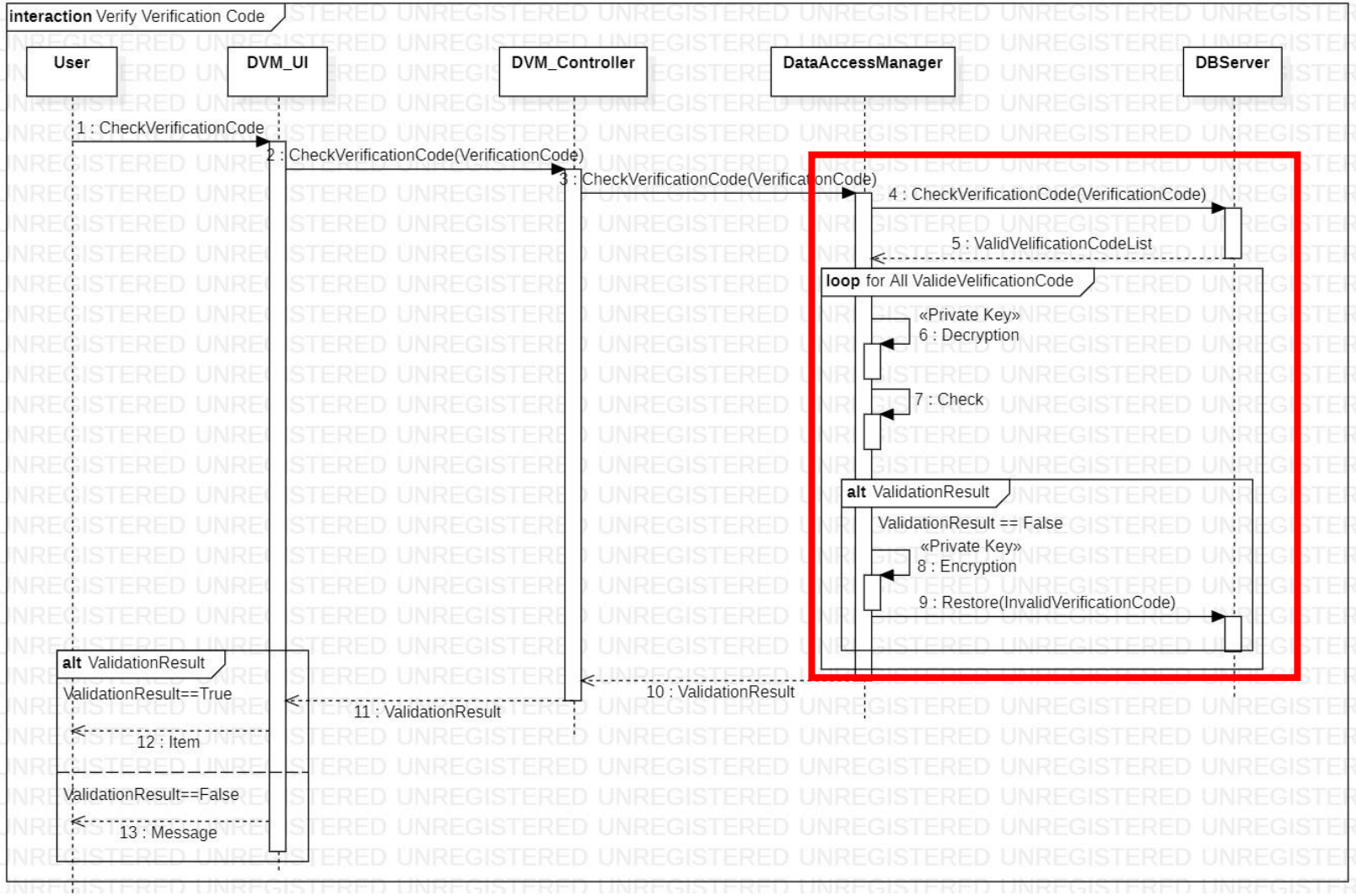
4. 3rd iteration

Step 6 Refine Sequence diagram(UC-4)

Element	Method Name	Responsibility
DataAccessManager	encryption	DBServer에 저장하기 전 정보를 private key로 암호화한다.
Forwarder	encryption	다른 DVM의 Receiver로 정보를 보내기 위해 public key로 암호화한다.
Receiver	decryption	다른 DVM의 Forwarder로부터 받은 정보를 public key로 복호화한다.
DVMController	decidePriority	다른 DVM으로부터 들어온 요청들의 처리에 관한 우선순위를 정한다.
DataAccessManager	encryption	DBServer에 저장하기 전 정보를 private key로 암호화한다.

4. 3rd iteration

Step 6 Refine Sequence diagram(UC-5)



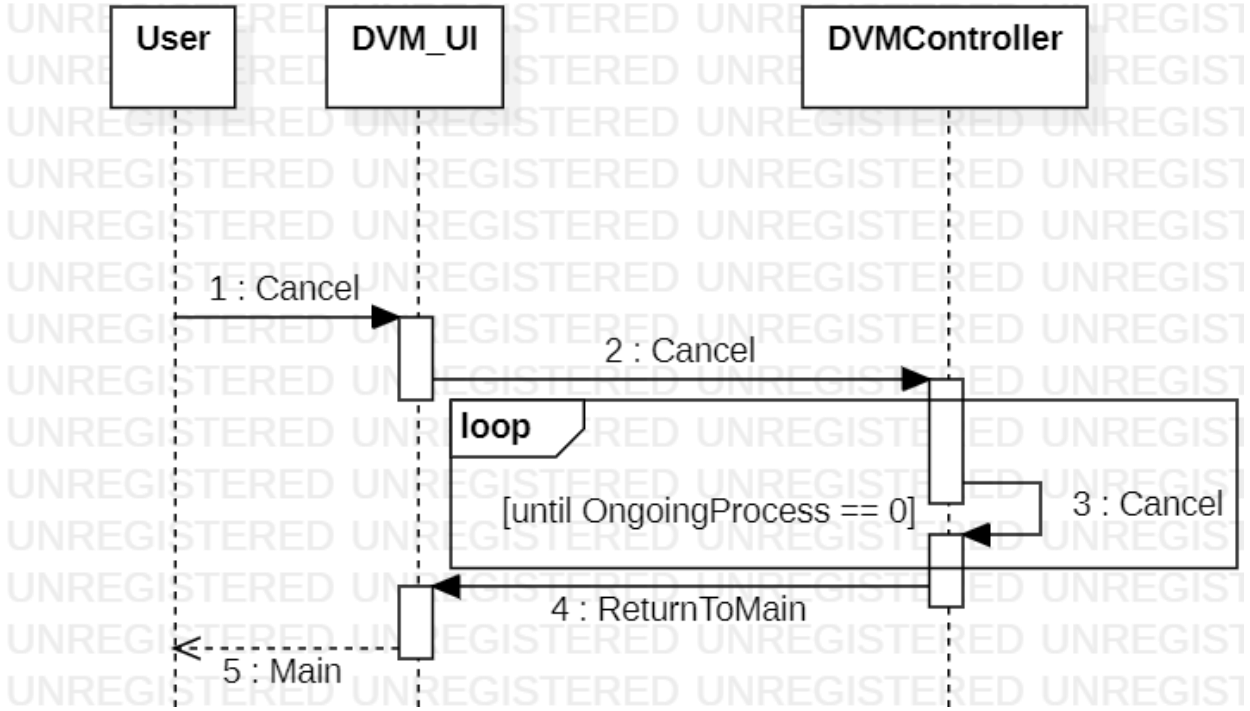
4. 3rd iteration

Step 6 Refine Sequence diagram(UC-5)

Element	Method Name	Responsibility
DBServer	CheckVerificationCode (VerificationCode)	Verification Code속 ItemID, DVMID를 읽어 이와 일치하는 Verification Code들(ValidVerificaitonCodeList)을 리턴 하고, DBServer에서 지운다. * Verification Code는 Verification Code를 생성한 DVM ID , Item ID, Key로 구성되었다고 설정하였으며, Key 부분만 암호화 된다고 가정한다.
	Check	암호화된 Verification Code로는 Private Key가 없는 DBServer에서는 올바른 Verification Code인지 확인할 수 없다고 판단하여 Sequence Diagram에서 지우고, Check기능을 DataAccessManager가 하도록 하였다.
	Restore	암호화 된 VerificationCode를 다시 DBServer에 저장한다.
DataAccessManager	Decryption	DBServer에서 받은 암호화된 Valid Verification Code들의 암호화를 푼다.
	Check	처음에 입력 받은 Verification Code와 Decryption한 Verification Code가 일치하는지 검사하고, 결과를 ValidationResult로 저장하고 ValidationResult가 True이면 Loop를 빠져 나온다.
	Encryption	ValidationResult가 False일 때 하는 동작으로, 와 Decryption한 Verification Code를 암호화 한다.

4. 3rd iteration

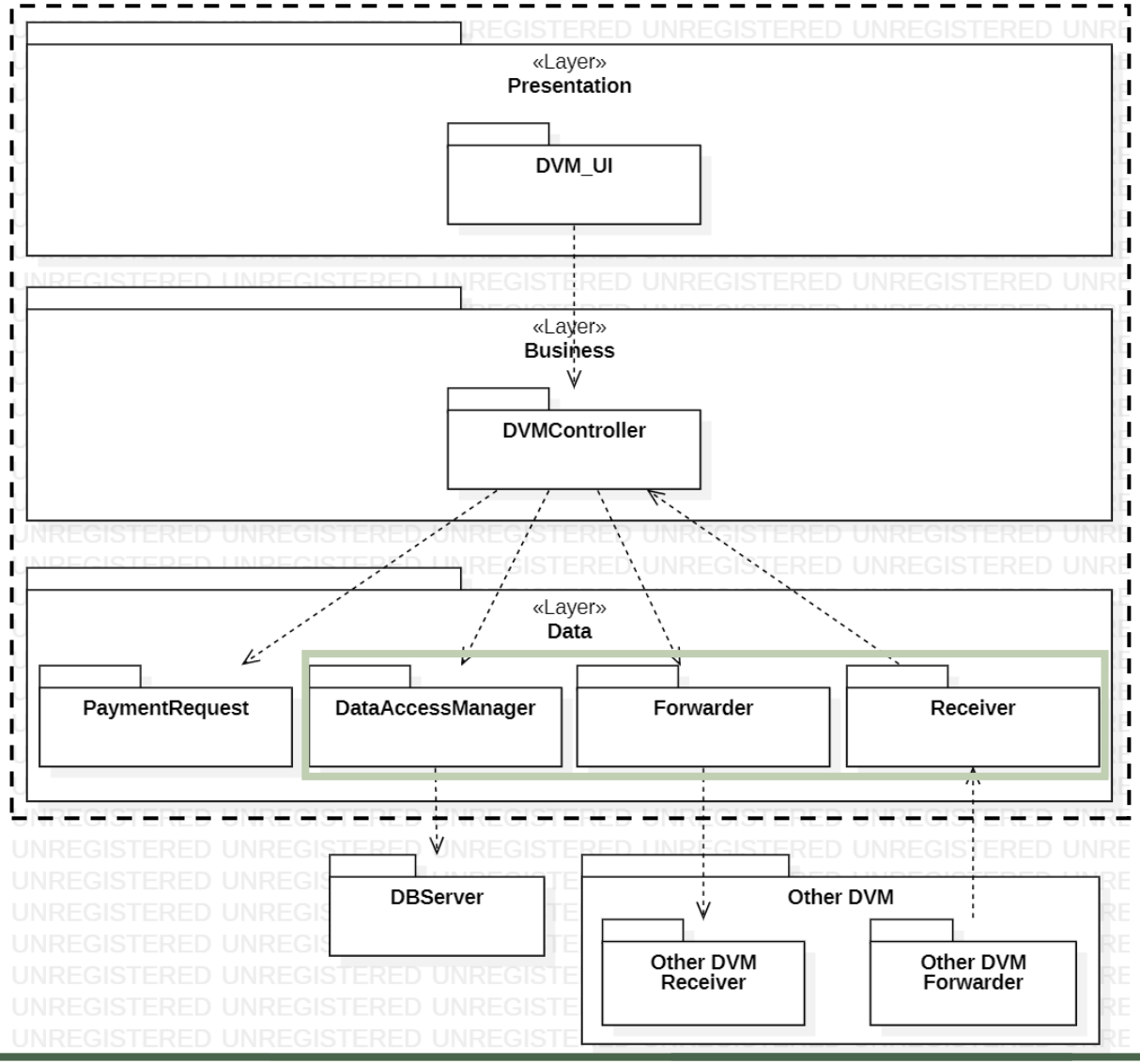
Step 6 Sequence diagram(UC-8) ← New! Cancel



Element	Method Name	Responsibility
DVM_UI	Cancel	사용자가 취소 버튼을 누르면 controller에 취소를 요청한다.
	ReturnToMain	진행 중이던 모든 process가 취소되면 메인화면으로 되돌아간다.
DVMController	Cancel	취소 요청을 받으면 loop을 돌면서 연관된 모든 process를 취소한다.

4. 3rd iteration

Step 6 Module View(Refined) – Refine Element Description



Element	Responsibility
DataAccess Manager	각 DVM의 위치, DVM의 ID, 취급하는 상품 종류, 각 상품의 재고에 대한 정보와 송수신한 인증번호를 DBServer에 저장하거나 불러온다. DBServer에 저장하고 불러올 때 private key를 활용하여 암/복호화를 진행한다.
Forwarder	전송할 정보를 파일의 형태로 marshalling하여 다른 자판기의 receiver로 전송한다. 파일을 전송하기 전, public key를 활용하여 암호화를 진행한다.
Receiver	다른 자판기의 forwarder로부터 파일의 형태로 수신한 정보를 unmarshalling하여 자판기에 전달한다. 파일을 수신하기 전, public key를 활용하여 복호화를 진행한다.

4. 3rd iteration

Step 7 Addressing된 정도 분석

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During Iteration
	UC-8		
		QA-1	이번 iteration에서 적절한 tactics(cancel)를 찾아 적용하고 이에 적절한 use case를 추가하여 sequence diagram을 작성함으로써 해당 Quality attribute를 만족시켰으므로 달성되었음
		QA-4	이번 iteration에서 적절한 tactics(prioritize event)를 찾아 적용하고 sequence diagram을 수정하면서 해당 Quality attribute를 만족시켰으므로 달성되었음
		QA-7	이번 iteration에서 적절한 tactics(encrypt data)를 찾아 적용하고 sequence diagram을 수정하면서 해당 Quality attribute를 만족시켰으므로 달성되었음
	CON-2		Security와 관련된 QA-7의 영향을 받아, 일부 만족되었음
		CRN-2	Iteration 1,2,3 을 수행하면서 팀원들의 이해도를 높였으므로 달성되었음

Drivers that are completely addressed in previous iteration is removed

감사합니다 ☺