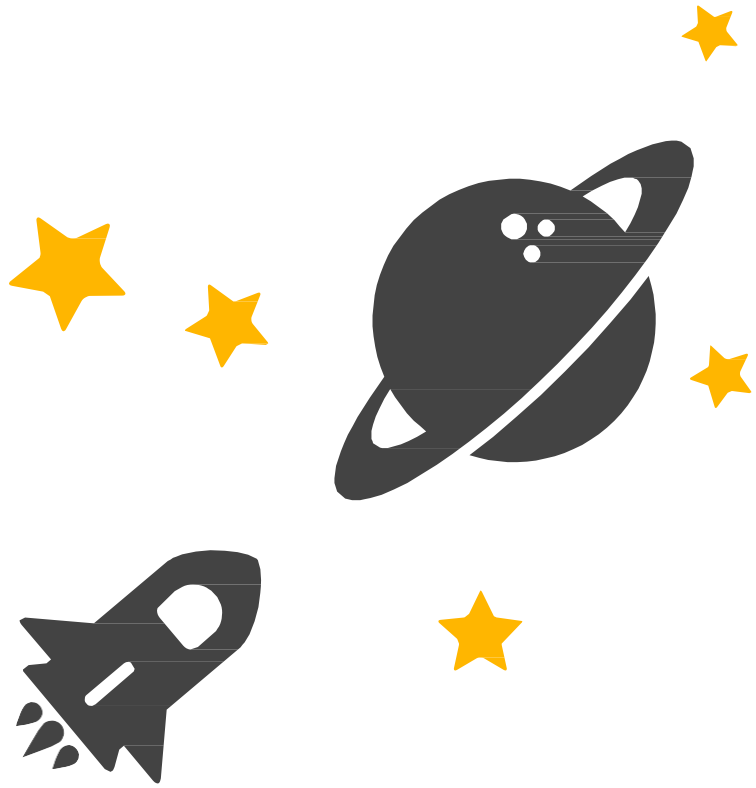
[Software Verification]
1st System Test

Team 4

201411259 고수창
201411314 전소영
201412005 이세라
201511304 하지윤

Index



1. Specification Review

Specification Review

Stage 1000 – 1003. Define Requirements

1. Motivation

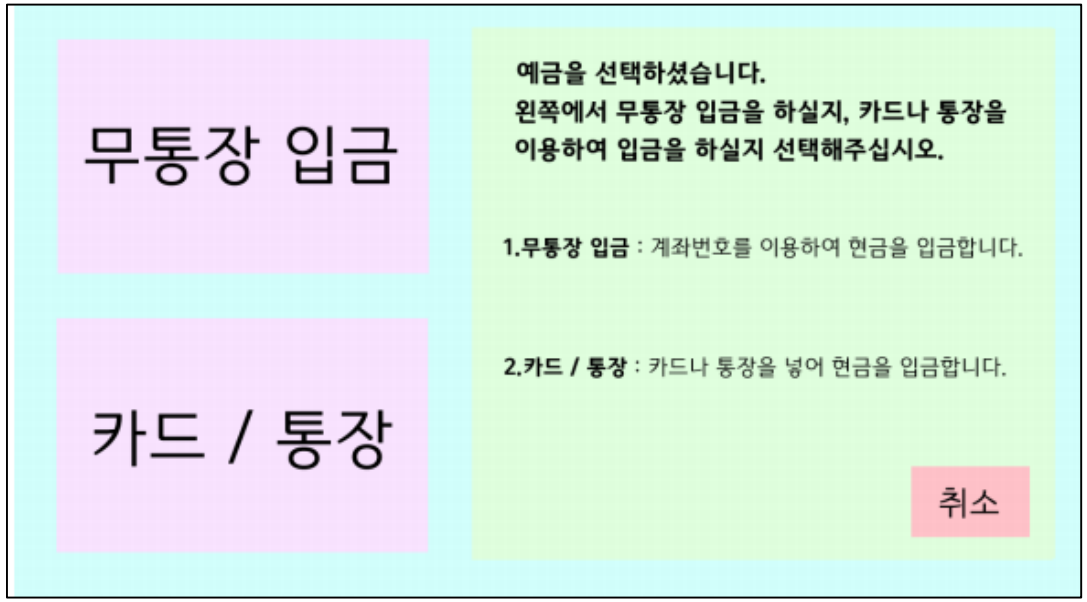
ATM 은 은행과 달리 영업시간의 제약을 받지 않을 수 있지만, 본인 확인 절차가 간소화 되어 있기에 이용가능한 서비스의 영역이 제한되어 있다. 서비스 영역에 제한을 두어도 보이스 피싱, 중고 거래 사기 등의 범죄로부터 ATM 이 이용자를 보호할 방법은 없다. 이용자로 하여금 범죄 이력이 있는 계좌를 조회할 수 있게 하고, 해당 계좌로의 송금을 할 수 없게 하여 보다 안전한 거래를 돕는 ATM 을 개발 하고자 한다.

Transfer	1. 메뉴 선택 화면에서 유저가 거래 메뉴를 선택했을 때 진행된다. 2. 통장이나 카드를 받아 계좌 정보를 BankDB 에서 받아 계좌가 유효한지 확인한다. 3. 계좌의 비밀번호를 받아 비밀번호가 맞는지 확인한다. 4. 비밀번호가 맞는 경우 유저에게서 송금할 계좌를 입력 받고 송금할 계좌가 유효한지 확인한다.

- Motivation, Non-Functional Requirements에 언급된
  안전한 거래에 대한 내용이 Functional Requirements에 **미 반영.**

Specification Review

Stage 1000 – 1005. Implement Prototype



1.1 Functional Requirements

Function	Description
Deposit	1. 메뉴 선택 화면에서 유저가 입금 메뉴를 선택했을 때 진행된다. 2. 통장이나 카드를 받아 계좌 정보를 BankDB 에서 받아 계좌가 유효한지 확인한다. 3. 계좌의 비밀번호를 받아 비밀번호가 맞는지 확인한다. 4. 비밀번호가 맞는 경우 유저에게서 돈을 받을 준비를 하고 유저에게서 돈을 받고 센다. 5. 유저의 계좌에 입금을 진행한다. 이 때, 유저가 입금을 진행하고 있는 ATM 과 계좌의 은행이 다를 경우 수수료를 계산한다. 6. 명세표를 출력할지 유저에게서 확인 받고 명세표를 출력할 경우 명세표를 출력한다. 7. 유저에게 통장 또는 카드를 돌려준다. 8. 메뉴 선택 화면으로 돌아간다.

- Functional Requirements 와 다른 Prototype.

Specification Review

Stage 1000 – 1006. Define Business Use Case

1.2 Reference Number & Categorized and Identified Functions

Reference No.	Function	Category
R 1	Deposit	Evident
R 2	Withdraw	Evident
R 3	Transfer	Evident
R 4	Check Transaction History	Evident
R 5	Check Criminal History	Evident

Stage 1003. Define Requirements

3. Actor - Goal List

Actor	Goal
User	- 입금 / 출금 / 송금 / 거래내역 조회 - 보다 완벽한 프로그램 개발

Stage 1006. Define Business Use Case

- Actor – User에 대한 Goal이 누락됨.

Specification Review

Stage 1000 – 1006. Define Business Use Case

Withdraw	1. 메뉴 선택 화면에서 유저가 출금 메뉴를 선택했을 때 진행된다. 2. 통장이나 카드를 받아 계좌 정보를 BankDB 에서 받아 계좌가 유효한지 확인한다. 3. 계좌의 비밀번호를 받아 비밀번호가 맞는지 확인한다. 4. 비밀번호가 맞는 경우 유저에게서 내보낼 돈의 양을 입력 받는다. 5. 유저의 계좌에서 출금을 진행한다. 이 때, 유저가 출금을 진행하고 있는 ATM 과 계좌의 은행이 다를 경우 수수료를 계산하고, ATM 에 충분한 돈이 있는지 확인한다. 6. 명세표를 출력할지 유저에게서 확인 받고 명세표를 출력할 경우 명세표를 출력한다. 7. 유저에게 통장 또는 카드를 돌려준다. 8. 메뉴 선택 화면으로 돌아간다.

Stage 1003. Define Requirements

Use Case	2. Withdraw
Actor	User
Description	1. 유저가 출금 메뉴를 선택한다. 2. 통장이나 카드를 넣는다. 3. 통장이나 카드에 연결된 계좌의 비밀번호를 넣는다. 4. 비밀번호가 맞을 경우 출금할 돈을 ATM 에 입력한다. 5. 출금을 진행하고 ATM 에게서 현금을 받는다. 6. 명세표를 출력할 지 결정하고, 출력할 경우 명세표를 받는다. 7. ATM 에 삽입한 통장이나 카드를 되돌려 받는다.

Stage 1006. Define Business Use Case

- Use Case에서 Requirements 일부 누락됨.

Specification Review

Stage 2030 – 2031. Define Essential Use Case

Withdraw	1. 메뉴 선택 화면에서 유저가 출금 메뉴를 선택했을 때 진행된다. 2. 통장이나 카드를 받아 계좌 정보를 BankDB 에서 받아 계좌가 유효한지 확인한다. 3. 계좌의 비밀번호를 받아 비밀번호가 맞는지 확인한다. 4. 비밀번호가 맞는 경우 유저에게서 내보낼 돈의 양을 입력 받는다. 5. 유저의 계좌에서 출금을 진행한다. 이 때, 유저가 출금을 진행하고 있는 ATM 과 계좌의 은행이 다를 경우 수수료를 계산하고, ATM 에 충분한 돈이 있는지 확인한다. 6. 명세표를 출력할지 유저에게서 확인 받고 명세표를 출력할 경우 명세표를 출력한다. 7. 유저에게 통장 또는 카드를 돌려준다. 8. 메뉴 선택 화면으로 돌아간다.

Stage 1003. Define Requirements

2. Withdraw

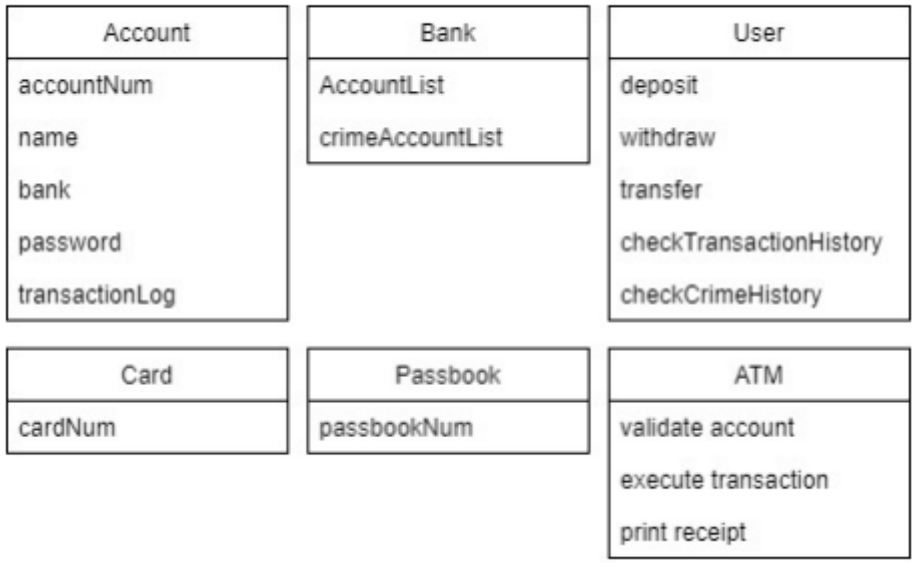
Use Case	2. Withdraw
Actor	User
Purpose	As in the business use case
Overview	As in the business use case
Type	Primary and Essential
Cross Reference	System Functions: R2 Use Cases: "Withdraw"
Pre-Requisites	User select menu
Typical Courses of Events	(A) : Actor, (S) : System (A) 사용자가 출금 메뉴를 선택한다. (S) 카드 혹은 통장을 투입하라고 요구한다 (A) 매체를 투입한다 (S) 매체로부터 계좌 정보를 얻어온다 (S) 계좌 정보가 유효한지 확인한다 (A) 비밀번호를 입력한다. (S) 입력된 비밀번호가 계좌의 비밀번호가 맞는지 확인한다 (S) 출금할 금액을 입력하라고 요구한다 (A) 출금할 금액을 입력한다

Stage 2031. Define Essential Use Case

- Use Case에서 Requirements 일부 누락됨.

Specification Review

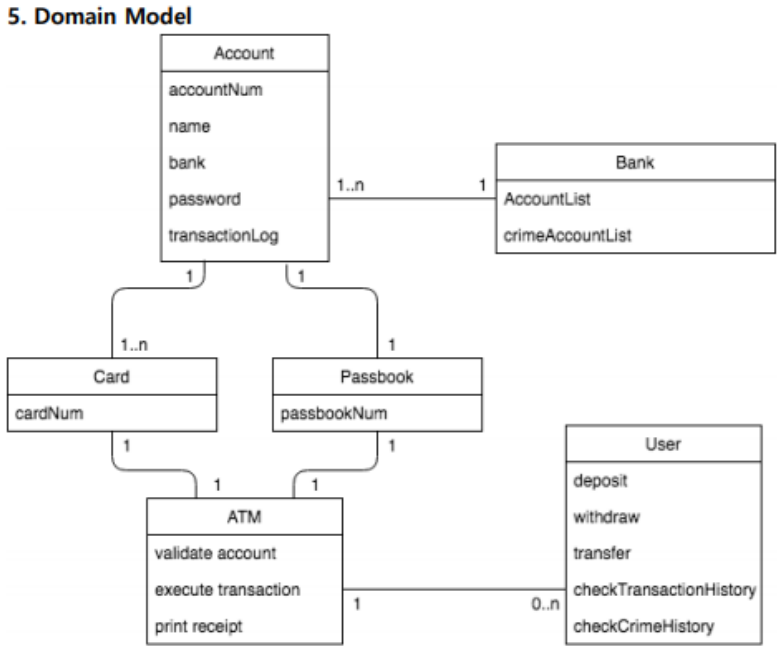
Stage 2030 – 2033. Define Domain Model



- Attribute, Operation의 자료형이 누락됨.

Specification Review

Stage 2030 – 2033. Define Domain Model



- Attribute, Operation의 Type, Parameter가 누락됨.

Specification Review

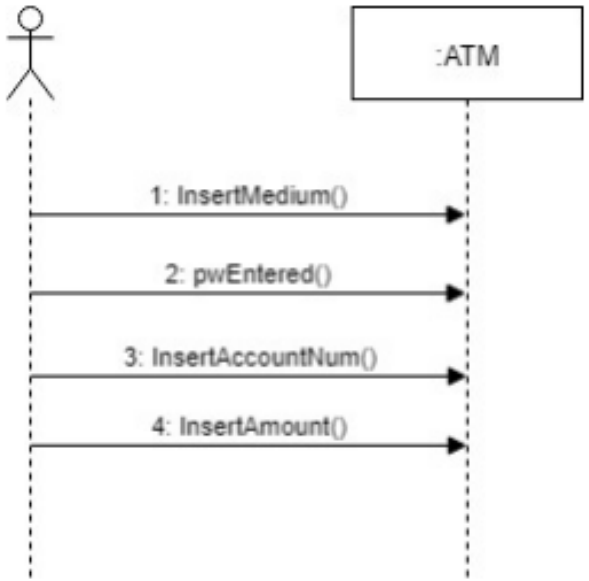
Stage 2030 – 2034. Refine Glossary

User.deposit	Attribute	예금 실행
User.withdraw	Attribute	출금 실행
User.transfer	Attribute	거래 실행
User.checkTransactionHistory	Attribute	거래 내역 조회 실행
User.checkCrimeHistory	Attribute	범죄 이력 내역 조회 실행

- Attribute, Operation을 Attribute로 잘못 표기하였음.

Specification Review

Stage 2030 – 2035. Define System Sequence Diagrams



- InsertMedium(), pwEntered(), InsertAccountNum(), InsertAmount()
  ▷Activity 2036과 명시된 Name of Actor–Activated Event가 다름.

Specification Review

Stage 2030 – 2036. Define Operation Contracts

Name	InsertMedium()
Responsibilities	통장이나 카드를 인식하고 연결된 계좌를 확인한다.
Type	System
Cross Reference	System functions: R1, R2, R3, R4 Use Case: “Deposit”, “Withdraw”, “Transfer”, “Check Transcation History”
Notes	
Exceptions	계좌가 유효하지 않을 경우 에러를 발생시킨다.
Output	Data of Account
Pre-conditions	선택한 메뉴가 Deposit, Withdraw, Transfer또는 Check Transaction History여야한다.
Post-conditions	통장이나 카드에 해당되는 계좌의 정보(잔액, 거래내역, 범죄내역)를 얻는다. 계좌가 유효할 경우 계좌의 비밀번호를 받는다.

- 계좌 정보에 대한 내용이 이전 Stage에서 명시되지 않음.

Specification Review

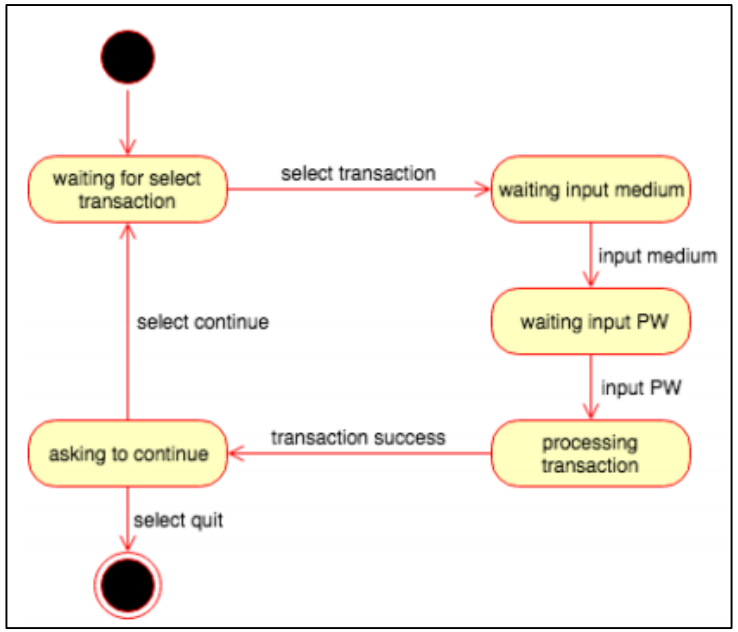
Stage 2030 – 2036. Define Operation Contracts

Name	EnteredPW()
Responsibilities	계좌의 비밀번호를 받고 비밀번호가 일치한지 확인한다.
Type	System
Cross Reference	System Functions: R1, R2, R3, R4 Use Case: “Deposit‘, “Withdraw”, “Transfer”, “Check Transaction History”
Notes	
Exceptions	입력된 비밀번호가 3회 이상 일치하지 않을 경우 에러를 발생시킨다.
Output	N/A
Pre-conditions	입력된 통장이나 카드의 계좌가 유효한지 확인되어 있어야한다.
Post-conditions	Deposit일 경우 돈을 입금받는다. Withdraw일 경우 출금할 돈의 수량을 입력받는다. Transfer일 경우 보낼 돈을 입금받는다. Check Transaction History일 경우 거래 내역을 보여준다.

- Requirements에 없던 내용이 새로 생김.

Specification Review

Stage 2030 – 2037. Define State Diagrams



- System Sequence Diagram과 일치하지 않고, 명시된 Operation들과 일치하지 않음.

Specification Review

Stage 2030 – 2038. Refine System Test Case

ATM_STC_004_001	Check Transaction History	사용자가 Check Transaction History버튼을 눌렀을 때 정상적으로 진행되는지 확인한다.
ATM_STC_004_002		정보를 조회할 카드나 통장이 정상적으로 투입되었는지 확인한다.
ATM_STC_004_003		거래 내역 조회가 정상적으로 이루어지는지 확인한다.
ATM_STC_004_004		거래 내역 조회 종료 후 카드나 통장이 정상적으로 반환되는지 확인한다.

Typical Courses of Events	(A) : Actor, (S) : System (A) 사용자가 계좌 조회 메뉴를 선택한다 (S) 카드 혹은 통장을 투입하라고 요구한다 (A) 매체를 투입한다 (S) 매체로부터 계좌 정보를 얻어온다 (S) 계좌 정보가 유효한지 확인한다 (S) 비밀번호를 입력하라고 요구한다 (A) 비밀번호를 입력한다 (S) 비밀번호를 확인한다 (S) 계좌 잔액을 표시한다 (S) 거래내역 조회 버튼을 표시한다 (S) 거래내역을 조회해서 화면에 표시한다 (S) 통장 정리 버튼을 표시한다

- 2031, 2035의 Check Transaction에서 비밀번호 입력 내용이 있으나 System Test Case에서 누락됨.

Specification Review

Stage 2030 – 2038. Refine System Test Case

ATM_STC_002_001	Withdraw	사용자가 Withdraw버튼을 눌렀을 때 정상적으로 진행되는지 확인한다.
ATM_STC_002_002		출금을 할 카드나 통장이 정상적으로 투입되었는지 확인한다.
ATM_STC_002_003		타행 거래시 수수료가 정상적으로 계산되는지 확인한다.
ATM_STC_002_004		출금이 정상적인 상황에서 정상적으로 이루어지는지 확인한다.
ATM_STC_002_005		명세표가 정상적으로 출력되는지 확인한다.
ATM_STC_002_006		출금 종료 후 카드나 통장이 정상적으로 반환되는지 확인한다.

ATM_STC_003_001	Transfer	사용자가 Transfer버튼을 눌렀을 때 정상적으로 진행되는지 확인한다.
ATM_STC_003_002		송금할 카드나 통장이 정상적으로 투입되었는지 확인한다.
ATM_STC_003_003		타행 거래시 수수료가 정상적으로 계산되는지 확인한다.
ATM_STC_003_004		송금이 정상적인 상황에서 정상적으로 이루어지는지 확인한다.
ATM_STC_003_005		명세표가 정상적으로 출력되는지 확인한다.
ATM_STC_003_006		송금 종료 후 카드나 통장이 정상적으로 반환되는지 확인한다.

- 계좌 유효성 체크 여부 누락됨.

Specification Review

Stage 2040 – 2041. Design Real Use Cases

Typical Courses of Events	(A): Actor (S): System 1. (A) 유저가 입금 버튼을 누른다. 2. (S) 통장이나 카드 번호를 입력하는 화면을 띄운다. 3. (A) 통장이나 카드 번호를 입력한다. 4. (S) 입력한 계좌번호가 실제로 유효한지 확인한다. 5. (S) 계좌번호가 유효할 경우 비밀번호를 입력하는 화면을 띄운다. 6. (A) 계좌의 비밀번호를 입력한다.

- 2030까지는 통장이나 카드를 넣는다고 명시되어 있었으나,
  2041부터 통장, 카드번호를 입력하는 방식으로 변경.

Specification Review

Stage 2040 – 2041. Design Real Use Cases

Typical Courses of Events	(A): Actor (S): System 1. (A) 유저가 출금 버튼을 누른다. 2. (S) 통장이나 카드 번호를 입력하는 화면을 띄운다. 3. (A) 통장이나 카드 번호를 입력한다. 4. (S) 입력한 계좌번호가 실제로 유효한지 확인한다. 5. (S) 계좌번호가 유효할 경우 비밀번호를 입력하는 화면을 띄운다. 6. (A) 계좌의 비밀번호를 입력한다. 7. (S) 입력된 비밀번호가 계좌의 비밀번호와 일치하는지 확인한다. 8. (S) 비밀번호가 일치할 경우 돈을 입력할 수 있는 화면을 띄운다. 9. (A) 출금할 돈을 입력한다. 10. (S) 입력한 계좌가 타행의 계좌일 경우, 타행 거래 간의 수수료를 계산한다.

- Withdraw – 2030에서 존재하는 ‘ATM에 충분한 돈이 있는지 확인한다’가 누락됨.

Specification Review

Stage 2040 – 2041. Design Real Use Cases

	띄운다. 9. (A) 송금할 계좌를 입력한다. 10. (S) 송금할 계좌가 실제로 유효한지 확인한다. 11. (S) 계좌가 유효할 경우 송금할 금액을 입력하는 화면을 띄운다. 12. (A) 송금할 금액을 입력한다. 13. (S) 입력한 계좌가 타행의 계좌일 경우, 타행 거래 간의 수수료를 계산하고 확인한다. 14. (S) 타행 거래일 경우, 수수료를 빼고 송금시키고 아니면 입력된 돈을 입력된 계좌에 송금한다. 15. (S) 송금 후의 잔고를 화면에 표시하고 유저에게 명세표를 출력할지 묻는 화면을 띄운다. 16. (A) 명세표를 출력할 경우 예 버튼을 누르고 아닐 경우 아니오 버튼을 누른다. 17. (S) 명세표를 출력할 경우 송금 내역을 명세표로 출력한다. 18. (S) 초기 메뉴 선택 화면으로 돌아간다.

Stage 2041. Design Real Use Cases

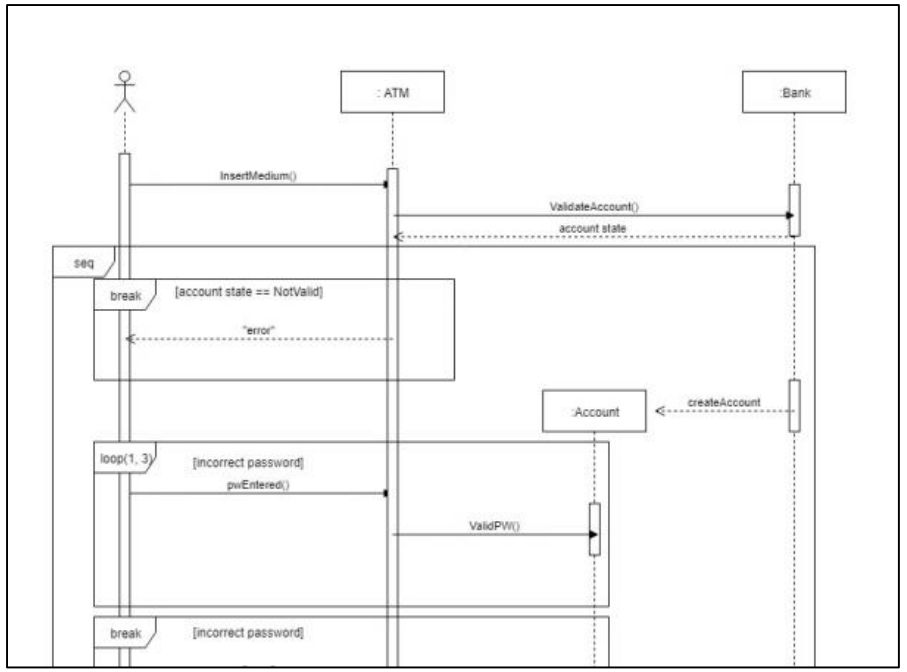
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) 유저가 거래 메뉴를 선택한다 2. (A) 유저가 카드나 통장을 삽입한다. 3. (S) 삽입된 카드나 통장의 계좌번호가 유효한지 확인한다. 4. (A) 비밀번호를 입력한다. 5. (S) 입력된 비밀번호가 계좌의 비밀번호가 맞는지 확인한다 6. (A) 송금할 계좌를 입력한다. 7. (S) 송금할 계좌가 유효한지 확인한다. 8. (A) 송금할 돈의 양을 넣는다. 9. (S) 타행 거래시 수수료를 계산한다. 10. (S) 송금할 계좌에 송금한다. 11. (A) 명세표를 출력할지 확인한다.

Stage 2031. Define Essential Use Cases

- 2030에 존재하지 않았던 송금 후 잔고 확인 과정이 추가됨.

Specification Review

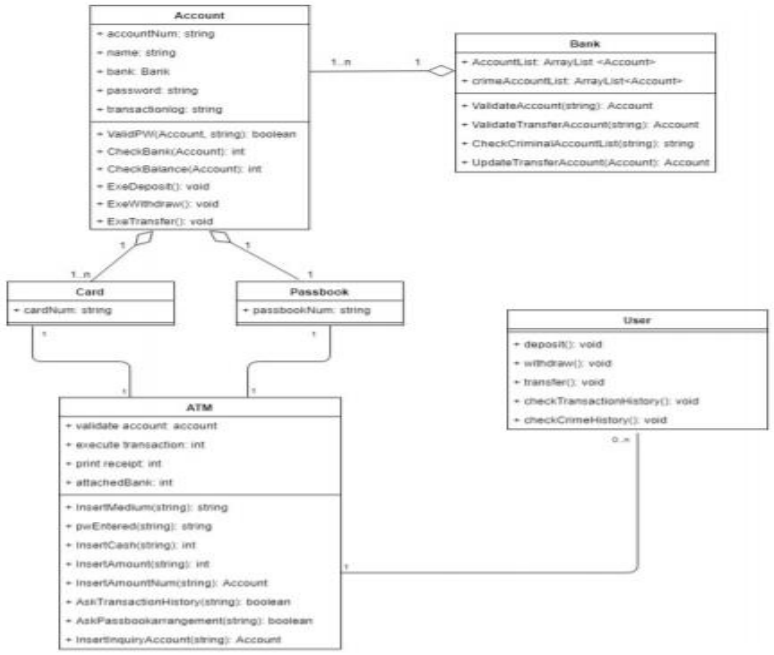
Stage 2040 – 2044. Define Interaction Diagrams



- 2036에 명시된 Actor–Activated Event와 System Operation과 상이함.

Specification Review

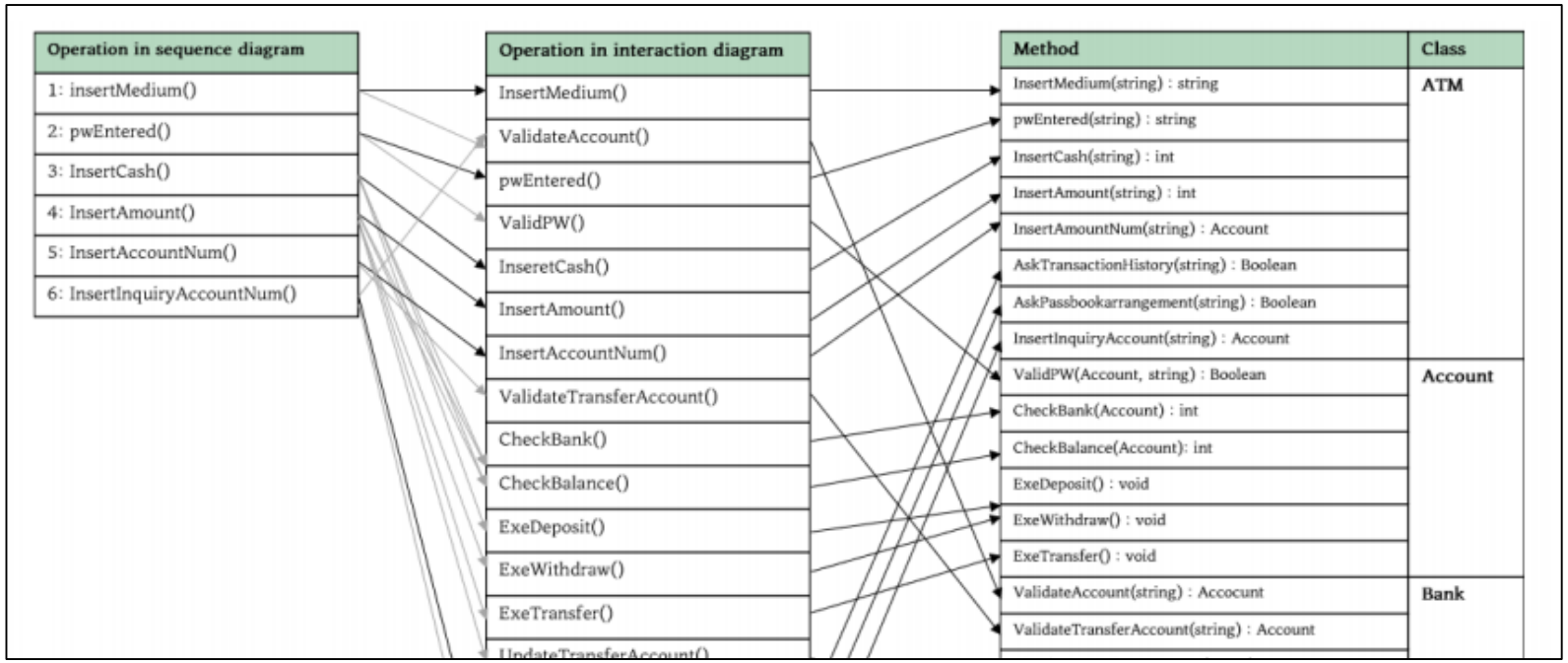
Stage 2040 – 2045. Define Design Class Diagrams



- 사용하지 않는 변수 및 함수들이 다수 존재함.

Specification Review

Stage 2040 – 2046. Design Traceability Analysis



- Operation in Sequence Diagram의 전체 항목이 잘못되었음.
  ▷Deposit, Withdraw, Transfer, Check Transaction History, Check Criminal History로 바뀌어야 함.

Specification Review

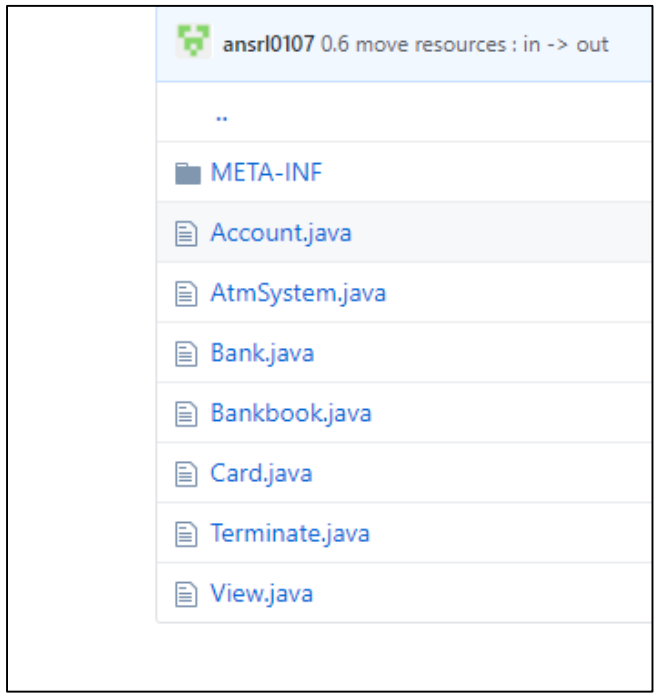
Stage 2050 – 2051. Implement Class & Methods Definition

Type	class
Name	AtmSystem
Purpose	ATM 정보를 모아두는 클래스
Overview(class)	Attribute : Bank own, Account src, Account des, amount, fee, banks, transaction Method : AtmSystem(), tansaction(), checkBalance(), checkPassword(), deposit(), withdraw(), transfer(), getLogs(), getCriminalLogs(), getTransaction(), setSrv(), validateSrc(), setDes(), validateDes(), setAmount(), findAccount()
CrossReference	Use Case : R 1.1, R 2.1, R.3.1, R 3.2, R 3.3, R 4.1, R 4.2, R 4.3, R 4.4, R 5.1, R 5.2, R 5.3, R 5.4, R 5.5, R 6.1
Exceptional Courses of Events	N/A

- 자료형에 대한 명시가 되어 있지 않으며 실제 코드와 상이한 함수들이 존재함.

Specification Review

Stage 2050 – 2051. Implement Class & Methods Definition



- Bankbook, Card, View, Terminate에 대한 Class Definition이 되어있지 않음.

Specification Review

Stage 2050 – 2051. Implement Class & Methods Definition

Type	Method
Name	getBalance
Purpose	계좌 잔액을 확인한다.
CrossReference	Use Case
Input(Method)	-
Output(Method)	int bank.getBalance(account: this)
Abstract Operation(Method)	bank DB 에서 balance 를 불러와 반환한다.
Exceptional Courses of Events	-

- Bank DB에 대한 정의가 없음.

Specification Review

Stage 2050 – 2051. Implement Class & Methods Definition

Type	Method
Name	transaction()
Purpose	이용할 서비스를 선택한다.
CrossReference	Use Case R 1.1
Input(Method)	-
Output(Method)	-
Abstract Operation(Method)	button 입력에 따라 선택한 서비스를 실행한다.
Exceptional Courses of Events	-

- 문서상 정의는 존재하나 실제 코드상에 있지 않음.

Specification Review

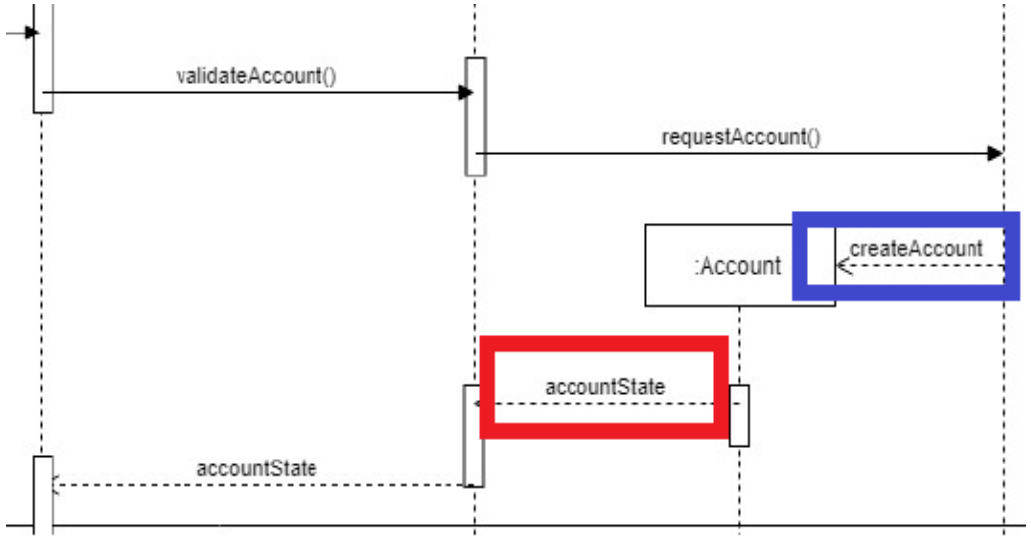
Stage 2050 – 2051. Implement Class & Methods Definition

Type	Method
Name	transfer()
Purpose	src 에서 des 로 송금한다.
CrossReference	Use Case 5.3
Input(Method)	-
Output(Method)	-
Abstract Operation(Method)	src 계좌의 balance 를 amount + fee 만큼 감소시키고 log 에 msg 와 해당 거래내용을 업데이트한다 des 계좌의 balance 를 amount 만큼 증가시키고 log 에 msg 와 해당 거래내용을 업데이트한다
Exceptional Courses of Events	-

- 수수료에 대한 정의가 존재하지 않음
- 코드상으로 타행거래시 1%만큼 차감하는 것으로 되어있음.

Specification Review

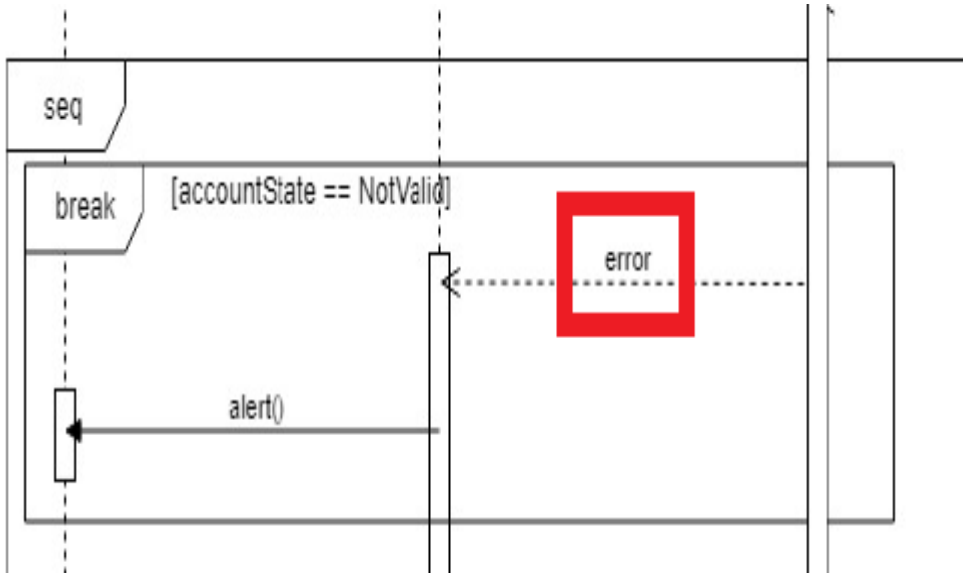
Stage 2050 – 2052. Implement Windows



- accountState 반환이 명시되어 있지만 실제 소스코드 및 문서에 Description이 존재하지 않음.

Specification Review

Stage 2050 – 2052. Implement Windows



- accountState == not Valid한 경우 AtmSystem에서 Error를 반환한다고 명시되어 있음 .
  ▷But,소스코드에서 Error가 아니라 Method별 False 혹은 Null를 반환하도록 구현되어 있음.

Specification Review

Stage 2050 – 2055. Write Unit Test Code

```
class ActivateTest {

    @Test
    void testActivate_1() {
        Activate Acti = new Activate();
        int x = Acti.activate();
        assertEquals(1, x);
    }

    @Test
    void testActivate_2() {
        Activate Acti = new Activate();
        int x = Acti.activate();
        assertEquals(2, x);
    }

}
```

- 무엇을 테스트하는지 불분명하고, 실제 프로젝트의 소스코드를 테스트하지 않음.

Specification Review

Stage 2050 – 2055. Write Unit Test Code

```
class SelectMenuTest {

    @Test
    void testSelectMenu_1() {
        SelectMenu Sele = new SelectMenu();
        int result = Sele.selectMenu();
        assertEquals(1, result);
    }

    @Test
    void testSelectMenu_2() {
        SelectMenu Sele = new SelectMenu();
        int result = Sele.selectMenu();
        assertEquals(2, result);
    }

    @Test
    void testSelectMenu_3() {
        SelectMenu Sele = new SelectMenu();
        int result = Sele.selectMenu();
        assertEquals(3, result);
    }
```

- 테스트코드에서 각각의 작업을 Case로 나누어 진행하지만,
  실제 코드에서 위와 같이 Transaction 작업을 수행하지 않음.

Specification Review

Stage 2050 – 2055. Write Unit Test Code

```
public class InputAccountNum {
    public int inputAccount() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("input AccountNum: ");
        String accountNum = scanner.nextLine();
        Account ret = new Account();
        ret = findAccount(accountNum);

        if(ret != null) {
            return 1;
        }else {
            return 2;
        }
    }
```

- 실제 테스트 코드는 FindAccount를 수행함.

Specification Review

Stage 2050 – 2055. Write Unit Test Code

```java
public class CheckTransactionHistory {
    public int checkTransactionHistory() {
            Account src = inputMedium();
            if (src == null) {
                return 1; //!1
            }
            //Bank bank = src.getBank();
            //bank.showLogs(src);
            System.out.println("Complete CheckTransactionHistory");
            return 2;//!2
    }
```

```java
class CheckTransactionHistoryTest {

    @Test
    void testCheckTransactionHistory_1() {
        CheckTransactionHistory CT = new CheckTransactionHistory();
        int result = CT.checkTransactionHistory();
        assertEquals(1, result);
    }

    @Test
    void testCheckTransactionHistory_2() {
        CheckTransactionHistory CT = new CheckTransactionHistory();
        int result = CT.checkTransactionHistory();
        assertEquals(2, result);
    }
}
```

- 실제 CheckTransactionHistory에 대한 검사를 수행하지 않음.

Specification Review

Stage 2060 – 2063. System Testing

1_6	selectMenuTest	오류처리가 되었는지 확인한다.	R 1.1	P
2_1	inputMediumTest	매체를 입력하는 화면이	R 2.1	P

- 오류 처리에 대한 Definition이 없음.

4_1	insertMoneyTest	돈을 입력하고 그만큼 제대로 계산되는지 확인한다.	R 4.1	P

- “제대로 계산되는지” 라는 불분명한 표현을 사용함.

Specification Review

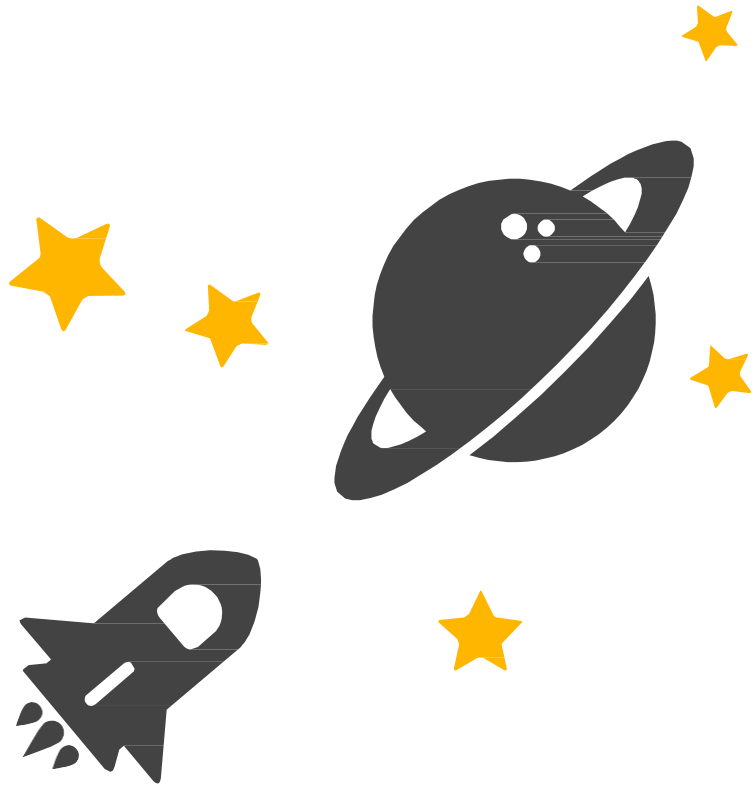
Stage 2060 – 2063. System Testing

4_4_1	CalculateBalance	1.당행 간의 거래 시 수수료가 정확히 계산된다. 2. 잔고를 넘지 않는 선에서 거래를 할 때 제대	R 4.4	P

- 실제 코드상으로 당행 거래 시 수수료가 차감되지 않음.

5_4_2	exeCheckTransaction HistoryTest	2. CheckTransactionHistory 화면이 뜬 후 알맞은 매체를 입력한뒤 거래 내역이 제대로 뜨는지 확인한다.	R 5.4	P

- “알맞은 매체를 입력한다” 라고 명시되어 있으나,
  실제로 계좌번호만 입력 받게 구현 되어있음.



2. Category-Partition Test

Category-Partition Test

Testable Units & Representative Values

Group	Category	Description	Value	Number
Validate	Account	계좌의 존재 유무 확인	계좌번호가 존재할때 / 계좌번호가 존재하지 않을때	1000 / 1001
	Password	비밀번호 일치 확인	일치하는 비밀번호/일치하지 않는 비밀번호/다른 계좌의 비밀번호	1100 / 1101
	ATM Balance	ATM 기계 내 잔고 확인	ATM의 잔고가 출금하려는 잔고보다 많다(충분) / ATM의 잔고가 출금하려는 잔고보다 적다(불충분)	1200 / 1201
	Account Balance	계좌 내 잔고 확인	계좌의 잔고가 출금하려는 잔고보다 많다(충분) / 계좌의 잔고가 출금하려는 잔고보다 적다(불충분)	1300 / 1301
Transaction	Deposit	입금	타행인경우 거래 금액에서 수수료를 제한 금액을 입금한다(잔액 추가) / 당행인경우 거래 금액을 입금한다(잔액 추가)	2000 / 2001
	Withdraw	출금	타행인경우 거래 금액에서 수수료를 더한 금액을 출금한다(잔액 감소) / 당행인경우 거래 금액을 출금한다(잔액 감소)	2100 / 2101
	Transfer	송금	타행인경우 거래 금액에서 수수료를 더한 금액을 출금하고(잔액 감소), 송금 대상 계좌의 잔액을 거래 금액만큼 증가시킨다 / 당행인경우 거래 금액을 출금하고(잔액 감소), 송금 대상 계좌의 잔액을 거래 금액만큼 증가시킨다	2200 / 2201
Print	Transaction Result	입,출, 송금 결과 출력	입금, 출금, 송금 후 계좌에 남은 실제 잔액을 화면에 표시한다	3000
	Statement	명세표 출력	명세표 출력을 원하는 경우 / 명세표 출력을 원하지 않는 경우	3100 / 3101
	Criminal History	범죄기록 출력	범죄기록이 있는 경우 / 범죄기록이 없는 경우	3200 / 3201
	Transaction History	거래내역 출력	거래내역이 있는 경우 / 거래내역이 없는 경우	3300 / 3301
Input	Account Number	계좌번호 입력	숫자로만 입력한 경우 / 기타 문자와 숫자를 섞어 쓴 경우(숫자 이외의 문자가 들어간 경우) / 미입력	4000 / 4001 / 4002
	Password Number	계좌 비밀번호 입력	계좌 비밀번호를 입력받는다 / 입력받지 않은 경우 / 미입력	4100 / 4101 / 4102
	Transaction Amount	거래할 금액 입력	숫자로만 입력한 경우 / 기타 문자와 숫자를 섞어 쓴 경우(숫자 이외의 문자가 들어간 경우) / 미입력	4200 / 4201 / 4203
Access	Transaction Log	txt 파일에 접근	거래기록 파일이 있는 경우 / 거래기록 파일이 없는 경우	5000 / 5001
	Criminal Log	txt 파일에 접근	범죄기록 파일이 있는 경우 / 범죄기록 파일이 없는 경우	5100 / 5101

Category-Partition Test

Error Constraints 적용

Group	Category	Error Constraints
Validate	Account	Not in database [error]
	Password	Incorrect [error]
	ATM Balance	Not Enough [error]
	Account Balance	Not Enough [error]
Input	Account Number	!Number [error] Null [error]
	Password Number	!Number [error] Null [error]
	Transaction Amount	!Number [error] Null [error]

- **Test Frame 개수**

  적용 전: 22814개
  Error Constraints 적용 후: 777개

  ▷**96.5% 감소**

Category-Partition Test

Single Constraints 적용

Group	Category	Single Constraints
OS	Window 7	
	Window 10	[single]
	Mac	[single]
Print	Statment	!Print [single]

- **Test Frame 개수**

  적용 전: 777개
  Single Constraints 적용 후: 111개

  ▷**85.71% 감소**

Category-Partition Test

Property Constraints 적용

Group	Category	Property Constraints	Description
Transaction	Deposit	당행 property SAME 타행 property DIFER 입금 property D	if SAME: 당행거래인 경우 if DIFER: 타행거래인 경우 if D: 입금활동인 경우
	Withdraw	property W	if W: 출금활동인 경우
	Transfer	property T	if T: 송금활동인 경우
Access	Transaction Log	proerty HISTORY	if HISTORY: 거래내역이 존재하는 경우
	Criminal Log	property CRIM	if CRIM: 범죄내역이 존재하는 경우

- **Test Frame 개수**

적용 전: 111개
Property Constraints 적용 후: 51개 ▷**54.05% 감소**

Category-Partition Test

Testing Result (1/3)

Test Case No.	Test Case	Result
1	1001	F
2	1101	F
3	1201	F
4	1301	F
5	4001	F
6	4002	P
7	4101	P
8	4201	F
9	4202	F
10	1000.1100.1200.2000.3000.3100.3200.4000.4100.4200.5000.5100.	F
11	1000.1100.1200.2100.3000.3100.3200.4000.4100.4200.5000.5100.	F
12	1000.1100.1200.2200.3000.3100.3200.4000.4100.4200.5000.5100.	F
13	1000.1100.1200.2001.3000.3100.3200.4000.4100.4200.5000.5100.	F
14	1000.1100.1200.2101.3000.3100.3200.4000.4100.4200.5000.5100.	F
15	1000.1100.1200.2201.3000.3100.3200.4000.4100.4200.5000.5100.	F

Category-Partition Test

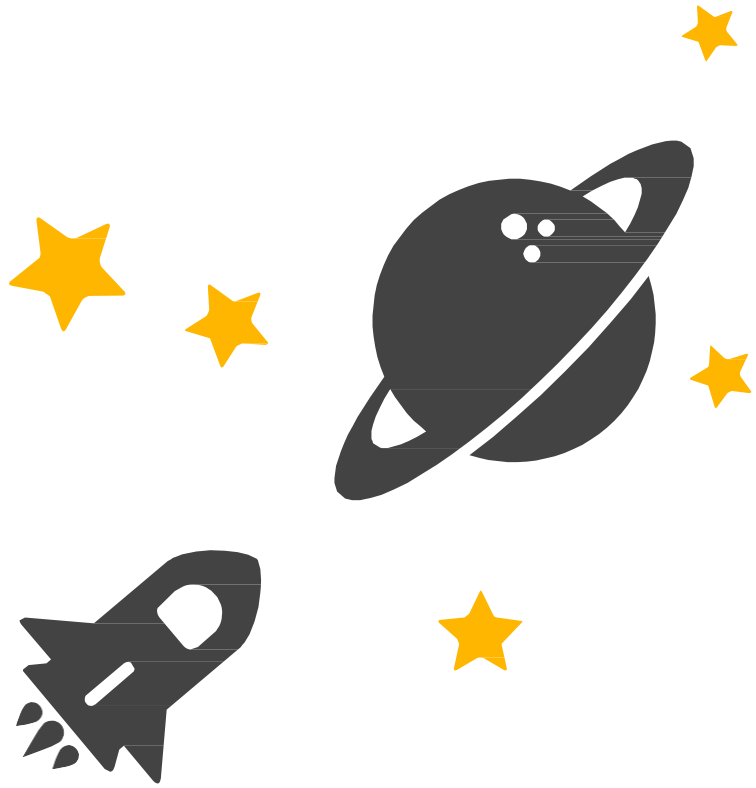
Testing Result (2/3)

16	1000.1100.1200.2000.3000.3101.3200.4000.4100.4200.5000.5100.	P
17	1000.1100.1200.2100.3000.3101.3200.4000.4100.4200.5000.5100.	P
18	1000.1100.1200.2200.3000.3101.3200.4000.4100.4200.5000.5100.	P
19	1000.1100.1200.2001.3000.3101.3200.4000.4100.4200.5000.5100.	P
20	1000.1100.1200.2101.3000.3101.3200.4000.4100.4200.5000.5100.	P
21	1000.1100.1200.2201.3000.3101.3200.4000.4100.4200.5000.5100.	P
22	1000.1100.1200.2000.3000.3100.3201.4000.4100.4200.5000.5100.	F
23	1000.1100.1200.2100.3000.3100.3201.4000.4100.4200.5000.5100.	F
24	1000.1100.1200.2200.3000.3100.3201.4000.4100.4200.5000.5100.	F
25	1000.1100.1200.2001.3000.3100.3201.4000.4100.4200.5000.5100.	F
26	1000.1100.1200.2101.3000.3100.3201.4000.4100.4200.5000.5100.	P
27	1000.1100.1200.2201.3000.3100.3201.4000.4100.4200.5000.5100.	P
28	1000.1100.1200.2000.3000.3101.3201.4000.4100.4200.5000.5100.	P
29	1000.1100.1200.2100.3000.3101.3201.4000.4100.4200.5000.5100.	P
30	1000.1100.1200.2200.3000.3101.3201.4000.4100.4200.5000.5100.	P

Category-Partition Test

Testing Result (3/3)

31	1000.1100.1200.2001.3000.3100.3200.4000.4100.4200.5000.5100.	P
32	1000.1100.1200.2101.3000.3100.3200.4000.4100.4200.5000.5100.	P
33	1000.1100.1200.2201.3000.3100.3200.4000.4100.4200.5000.5100.	P
34	1000.1100.1200.2000.3000.3101.3200.4000.4100.4200.5000.5101.	P
35	1000.1100.1200.2100.3000.3101.3200.4000.4100.4200.5000.5101.	P
36	1000.1100.1200.2200.3000.3101.3200.4000.4100.4200.5000.5101.	P
37	1000.1100.1200.2001.3000.3101.3200.4000.4100.4200.5000.5101.	P
38	1000.1100.1200.2101.3000.3101.3200.4000.4100.4200.5000.5101.	P
39	1000.1100.1200.2201.3000.3101.3200.4000.4100.4200.5000.5101.	P
40	1000.1100.1200.2000.3000.3101.3201.4000.4100.4200.5000.5101.	P
41	1000.1100.1200.2100.3000.3101.3201.4000.4100.4200.5000.5101.	P
42	1000.1100.1200.2200.3000.3101.3201.4000.4100.4200.5000.5101.	F
43	1000.1100.1200.2001.3000.3101.3201.4000.4100.4200.5000.5101.	P
44	1000.1100.1200.2101.3000.3101.3201.4000.4100.4200.5000.5101.	F
45	1000.1100.1200.2201.3000.3101.3201.4000.4100.4200.5000.5101.	F
46	1000.1100.1200.2000.3001.3100.3201.4000.4100.4200.5001.5100.	P
47	1000.1100.1200.2100.3001.3100.3201.4000.4100.4200.5001.5100.	P
48	1000.1100.1200.2200.3001.3100.3201.4000.4100.4200.5001.5100.	P
49	1000.1100.1200.2001.3001.3100.3201.4000.4100.4200.5001.5100.	P
50	1000.1100.1200.2101.3001.3100.3201.4000.4100.4200.5001.5100.	P
51	1000.1100.1200.2201.3001.3100.3201.4000.4100.4200.5001.5100.	P



3. Pairwise Test

Pairwise Test

Test Case 생성 (1/2)

1

pairwise.txt - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
Bank: Same, Different
Action: Deposit, Withdraw, Transfer, History, Criminal
Statement: Print, No Print
Log File: Exist both, Exist Criminal Only, Exist History Only, Non
PW: Vaild, Not Vaild
Account: Vaild, Not Vaild
```

2

```
D:₩2018_1₩소프트웨어 검증론>pict pairwise.txt > pair_result_1.txt
```

- PICT Tool을 이용한 Pairwise Test Case 생성.

Pairwise Test

Test Case 생성 (2/2)

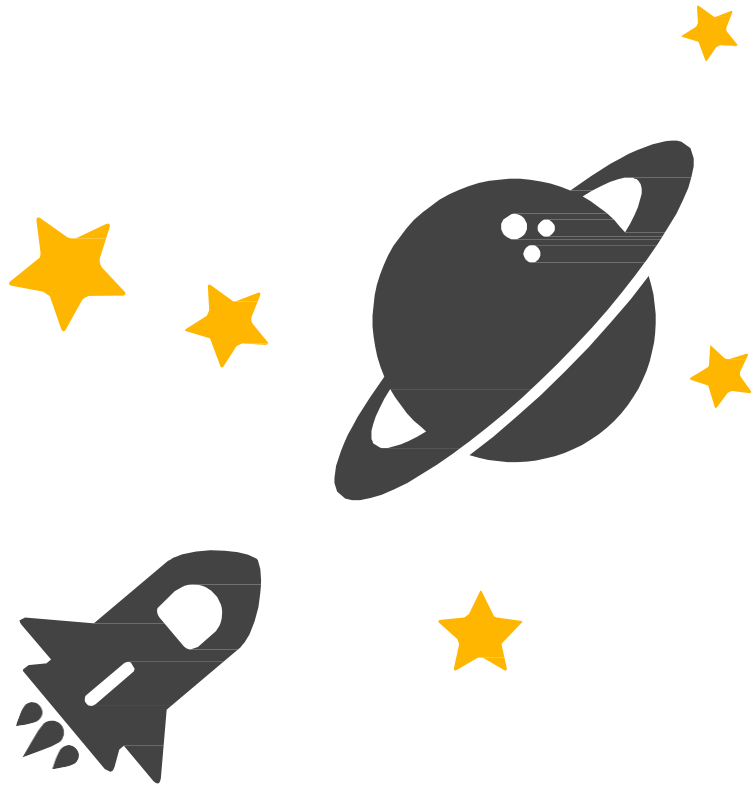
3	Pairwise Test Case:20				
	Bank	**Action**	**Log File**	**PW**	**Account**
1	Same	Criminal	Non	-	Not Vaild
2	Different	Criminal	Exist both	-	Vaild
3	Same	History	Exist History Only	Vaild	Vaild
4	Different	Deposit	Exist History Only	Not Vaild	Not Vaild
5	Same	Withdraw	Exist both	Vaild	Not Vaild
6	Different	Criminal	Exist History Only	-	Not Vaild
7	Same	Criminal	Exist Criminal Only	-	Vaild
8	Different	Withdraw	Exist Criminal Only	Not Vaild	Vaild
9	Different	History	Non	Not Vaild	Vaild
10	Same	Transfer	Exist Criminal Only	Vaild	Not Vaild
11	Different	Withdraw	Non	Not Vaild	Vaild
12	Same	History	Exist both	Vaild	Not Vaild
13	Same	History	Exist Criminal Only	Not Vaild	Not Vaild
14	Same	Deposit	Non	Vaild	Vaild
15	Different	Transfer	Exist both	Not Vaild	Vaild
16	Same	Withdraw	Exist History Only	Vaild	Vaild
17	Same	Deposit	Exist both	Vaild	Vaild
18	Same	Deposit	Exist Criminal Only	Vaild	Vaild
19	Same	Transfer	Exist History Only	Vaild	Not Vaild
20	Different	Transfer	Non	Not Vaild	Not Vaild

- 총 20개의 Pairwise Test Case 생성.

Pairwise Test

Testing Result

	Pairwise Test Case:20					4
	Bank	**Action**	**Log File**	**PW**	**Account**	**Result**
1	Same	Criminal	Non	-	Not Vaild	F
2	Different	Criminal	Exist both	-	Vaild	F
3	Same	History	Exist History Only	Vaild	Vaild	F
4	Different	Deposit	Exist History Only	Not Vaild	Not Vaild	F
5	Same	Withdraw	Exist both	Vaild	Not Vaild	F
6	Different	Criminal	Exist History Only	-	Not Vaild	F
7	Same	Criminal	Exist Criminal Only	-	Vaild	F
8	Different	Withdraw	Exist Criminal Only	Not Vaild	Vaild	F
9	Different	History	Non	Not Vaild	Vaild	F
10	Same	Transfer	Exist Criminal Only	Vaild	Not Vaild	F
11	Different	Withdraw	Non	Not Vaild	Vaild	F
12	Same	History	Exist both	Vaild	Not Vaild	F
13	Same	History	Exist Criminal Only	Not Vaild	Not Vaild	F
14	Same	Deposit	Non	Vaild	Vaild	F
15	Different	Transfer	Exist both	Not Vaild	Vaild	F
16	Same	Withdraw	Exist History Only	Vaild	Vaild	F
17	Same	Deposit	Exist both	Vaild	Vaild	F
18	Same	Deposit	Exist Criminal Only	Vaild	Vaild	F
19	Same	Transfer	Exist History Only	Vaild	Not Vaild	F
20	Different	Transfer	Non	Not Vaild	Not Vaild	F



4. Brute Force Test

Brute Force Test

Test Case & Result

Number	Test Case	Result
1	입금 시 입금 금액에 음수를 넣고 확인버튼을 누른다.	Fail
2	출금 시 출금 금액에 음수를 넣고 확인버튼을 누른다.	Fail
3	송금 시 송금 금액에 음수를 넣고 확인버튼을 누른다.	Fail
4	입금 시 입금 금액에 숫자 이외의 문자를 넣고 확인버튼을 누른다.	Fail
5	출금 시 출금 금액에 숫자 이외의 문자를 넣고 확인버튼을 누른다.	Fail
6	송금 시 송금 금액에 숫자 이외의 문자를 넣고 확인버튼을 누른다.	Fail
7	입금 시 입금 금액에 int 범위 이외의 숫자를 넣고 확인버튼을 누른다.	Fail
8	출금 시 출금 금액에 int 범위 이외의 숫자를 넣고 확인버튼을 누른다.	Fail
9	송금 시 송금 금액에 int 범위 이외의 숫자를 넣고 확인버튼을 누른다.	Fail
10	입금 시 입금 금액에 아무것도 넣지 않고 확인버튼을 누른다.	Fail
11	출금 시 출금 금액에 아무것도 넣지 않고 확인버튼을 누른다.	Fail
12	송금 시 송금 금액에 아무것도 넣지 않고 확인버튼을 누른다.	Fail
13	입금 시 비밀번호를 3회 이상 틀렸을 때 예외처리를 확인 한다.	Fail
14	출금 시 비밀번호를 3회 이상 틀렸을 때 예외처리를 확인 한다.	Fail
15	송금 시 비밀번호를 3회 이상 틀렸을 때 예외처리를 확인 한다.	Fail
16	입금 내역이 많을 경우 1초 이내에 처리 되는지 확인 한다.	Fail
17	출금 내역이 많을 경우 1초 이내에 처리 되는지 확인 한다.	Fail
18	송금 내역이 많을 경우 1초 이내에 처리 되는지 확인 한다.	Fail
19	거래 내역이 많을 경우 1초 이내에 처리 되는지 확인 한다.	Fail
20	범죄 이력이 많을 경우 1초 이내에 처리 되는지 확인 한다.	Fail
21	거래 내역이 많을 경우, 거래 내역 조회 기능에서 Simple한 UI/UX를 가지는지 확인한다.	Fail
22	범죄 내역이 많을 경우, 범죄 이력 조회 기능에서 Simple한 UI/UX를 가지는지 확인한다.	Fail

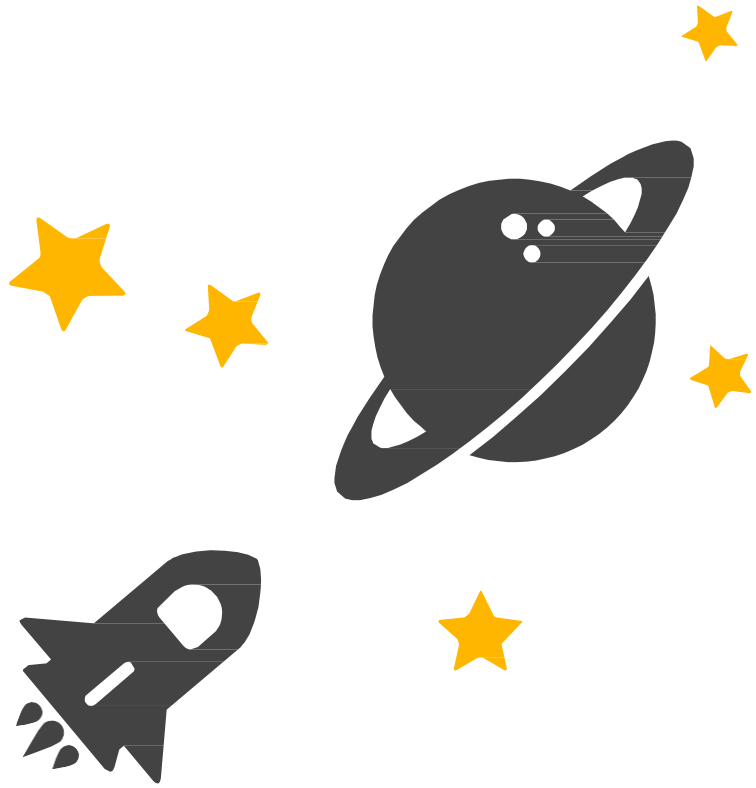
- ATM의 주요 기능을 중심으로 Brute Force Test를 진행함.

Brute Force Test

Failed Case Report

1	입력 금액에 음수를 넣고 입금을 진행할 경우 잔고가 감소한다.	F
2	출금 금액에 음수를 넣고 출금을 진행할 경우 잔고가 증가한다.	F
3	송금 금액에 음수를 넣고 송금을 진행할 경우 돈을 받는 사람의 잔고는 감소하고, 돈을 보내는 사람의 잔고는 증가한다.	F
4	입금 시 입금 금액에 숫자 이외의 문자를 넣고, 확인 버튼을 눌렀을 때아무 반응이 없다.	F
5	출금 시 출금 금액에 숫자 이외의 문자를 넣고, 확인 버튼을 눌렀을 때아무 반응이 없다.	F
6	송금 시 송금 금액에 숫자 이외의 문자를 넣고, 확인 버튼을 눌렀을 때아무 반응이 없다.	F
7	입금 시 입금 금액에 int 범위 이외의 숫자를 넣고 확인 버튼을 눌렀을 때아무 반응이 없다.	F
8	출금 시 출금 금액에 int 범위 이외의 숫자를 넣고 확인 버튼을 눌렀을 때아무 반응이 없다.	F
9	송금 시 송금 금액에 int 범위 이외의 숫자를 넣고 확인 버튼을 눌렀을 때아무 반응이 없다.	F
10	입금 시 입금 금액에 아무것도 넣지 않고 확인 버튼을 눌렀을 때아무 반응이 없다.	F
11	출금 시 출금 금액에 아무것도 넣지 않고 확인 버튼을 눌렀을 때아무 반응이 없다.	F
12	송금 시 송금 금액에 아무것도 넣지 않고 확인 버튼을 눌렀을 때아무 반응이 없다.	F
13	입금 시 비밀번호를 3회 이상 틀렸을 때 에러를 발생시키지 않고, 비밀번호가 틀렸을 때의 에러와 동일한 에러가 발생한다.	F
14	출금 시 비밀번호를 3회 이상 틀렸을 때 에러를 발생시키지 않고, 비밀번호가 틀렸을 때의 에러와 동일한 에러가 발생한다.	F
15	송금 시 비밀번호를 3회 이상 틀렸을 때 에러를 발생시키지 않고, 비밀번호가 틀렸을 때의 에러와 동일한 에러가 발생한다.	F
16	입금메뉴 선택 후, log.txt 파일이 50MB가 넘은 계좌번호를 입력하고 확인 버튼을 누르면 약 5초 후 메뉴가 표시된다.	F
17	출금메뉴 선택 후, log.txt 파일이 50MB가 넘은 계좌번호를 입력하고 확인 버튼을 누르면 약 5초 후 메뉴가 표시된다.	F
18	송금메뉴 선택 후, log.txt 파일이 50MB가 넘은 계좌번호를 입력하고 확인 버튼을 누르면 약 5초 후 메뉴가 표시된다.	F
19	조회메뉴 선택 후, log.txt 파일이 50MB가 넘은 계좌번호를 입력하고 확인 버튼을 누르면 약 5초 후 메뉴가 표시된다.	F
20	범죄이력 조회 메뉴 선택 후, criminalLog.txt 파일이 50MB가 넘은 계좌번호를 입력하고 확인 버튼을 누르면 약 5초 후 메뉴가 표시된다.	F
21	조회 버튼을 눌러 1,622,925개 내역이 있는 거래 계좌를 입력하고 확인 버튼을 눌렀을 경우 페이징 처리가 되어 있지 않고 단순 스크롤로 약 160만여개의 거래 내역을 보여주기 때문에 Simple한 UI/UX를 가진다고 보기 어렵다.	F
22	범죄 이력 조회 버튼을 눌러 1,563,312개 내역이 있는 거래 계좌를 입력하고 확인 버튼을 눌렀을 경우 페이징 처리가 되어 있지 않고 단순 스크롤로 약 150만여개의 거래 내역을 보여주기 때문에 Simple한 UI/UX를 가진다고 보기 어렵다.	F

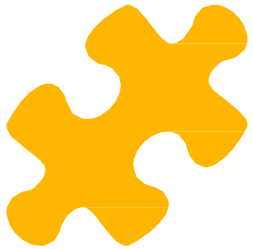
5. Overall



1st System Test Result

- Category–partition Test
  31/51 = 60.78% Pass
- Pairwise Test
  0/20 = 0% Pass
- Brute Force Test
  0/22 = 0% Pass

Summary



- 문서의 Stage별로 누락되거나 새로 생기는 부분들이 존재한다.
- 입금액을 입력할 때 음수 값을 입력해주면 계좌 잔액이 차감되는 등의 기본적인 계산에서 오류가 발생하므로, Super Safe ATM이라는 Non Functional Requirement에 적합하지 않다.
- Performance, User Interface와 같은 Non Functional Requirement를 만족시키지 못했다.
- 문서가 일관성 없이 작성되어 각 단계마다 글을 작성하는 방식, 내용 등 달라지는 것이 많다.
- Refine 부분을 명확히 할 필요가 있으며 이에 대한 문서 작성자와 개발자의 명확한 인식이 필요해 보인다.
- 문서에서 존재하나 코드에서 구현이 전혀 되지 않은 부분이 상당수 존재한다.

Q & A

Thank you☺