System Test Report for Team 7

1st

Project Team

Team 7

Lastest update on:

2018-05-18

Team Information

컴퓨터공학과 201111341 김성민

컴퓨터공학과 201111345 김종우

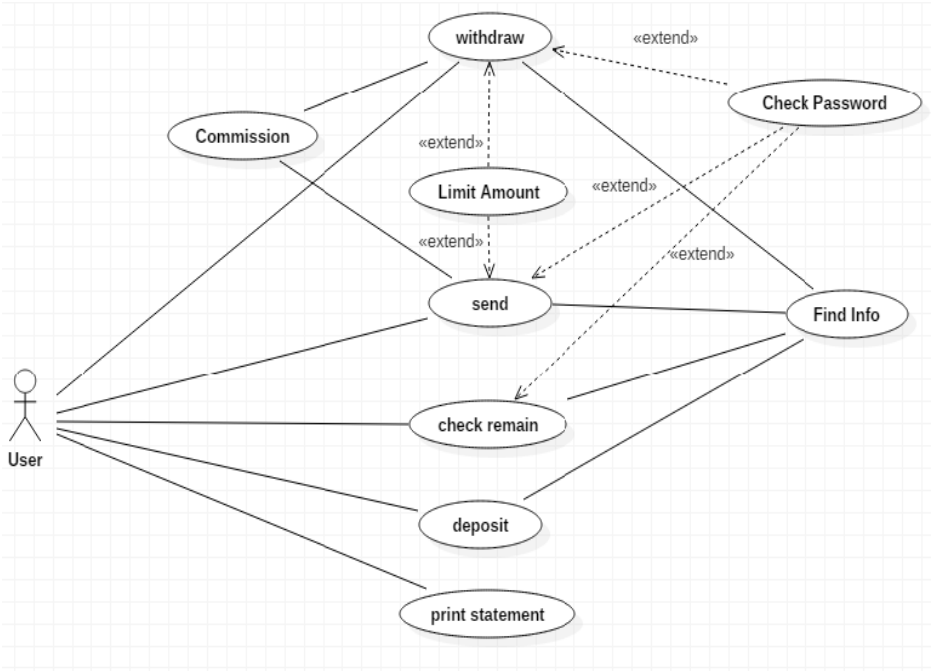
컴퓨터공학과 201211356 송원종

1 Specification Review

1.1 Stage 1000 Planning

1006. Define Business Use Case

6. Draw a Use-Case Diagram



- Event-based Use-Case인 print statement가 User와 직접 연결되어 있음
- Define System Boundary에 User는 Customer와 Admin으로 나누어져 정의되어 있으나 Use-Case Diagram에는 User로 묶어 정의되어있음, 세부적인 명세 필요

7. Describe Use-Case

Use Case	1. Withdraw
Actors	User
Descriptions	-고객은 ATM을 통하여 계좌에서 돈을 뽑는다. -인출을 하려면 해당 계좌의 비밀번호를 입력해야 한다. -인출할 금액보다 잔액이 더 적거나 인출 한도보다 더 많은 금액을 인출할 경우 거래가 취소된다. -인출을 수행할 경우 수수료를 부과된다. -출금이 끝나면 거래 내용을 업데이트한다. -거래가 끝나고 송금에 대한 명세서를 출력한다.

- Actor에 대한 구분이 Customer와 Admin이 아닌 User로 정의되어있음, 세부적인 명세 필요
- 위에서 정의한 Use-Case Number & Name과 일치하지 않는 순서로 명시되어

있음

- Withdraw, Deposit, Send의 '거래가 끝나면 명세서를 출력한다'는 5. Print Statement에서 담당하는 부분임으로 이를 이용해 진행하는 것으로 수정
- 8. Find Info 사용자가 계좌ID를 입력한다는 설명을 보면 Event-Based가 아니라 Actor-Based로 보임

1.2 Stage 2030 Analysis

2031. Define Essential Use case

- 1. Send의 Typical Courses of Events의 '7. (S)는 입력한 출금액이 한도를 내에 있는지 확인한다.'는 어법상 어색하기 때문에 수정 필요
- Use-Case Number & Name의 순서와 일치하는 명시가 필요
- Actor-Based Use-Case의 Typical Courses of Events에 있는 Event-Based Use-Case이 실행하는 부분은 이를 수행한다고 대체

2035. Define System Sequence Diagram

Use Case	Name of Actor-Activated Event
Send	Get Account(account_id) Input Password(password) Get Receiver Account (account_id) Input Amount(amount)
Deposit	Get Account(account_id) Input Amount(amount)
Withdraw	Get Account(account_id) Input Password(password) Input Amount(amount)
Check Remain Amount	Get Account(account_id) Input Password(password) Print Remain Amount()

- Name of Actor-Activated Event는 함수에 대한 정의가 아니기 때문에 매개변수에 대한 정의가 들어갈 필요가 없음

2036. Define Operation Contracts

- Operation Name은 실제 사용할 함수의 이름과 같게 작성
- Cross references에 대한 기술이 '2031. Define Essential Use-Cases'와 일치하

지 않음

2037. Define State Diagram

- Send, Withdraw에서 '2031. Define Essential Use Case'의 Typical Courses of Events의 기술과 다르게 Amount나 Payback대상이 맞는지 확인하는 부분이 기술되어있지 않음
- Deposit에서 '2031. Define Essential Use Case'의 Typical Courses of Events의 기술과 다르게 Amount를 Check하는 부분이 기술되어있음
- Check Remain에서는 Password를 잘못 입력했을 때 진행되는 방식에 대해 기술되어있지 않음

1.3 Stage 2040 Design

2041. Define Real Use Case

- Actor가 User가 아닌 Customer, Admin으로 세부적인 명세 필요

2044. Define Interaction Diagrams

- 전체적으로 Event, Operation에 대한 표현은 실선인 것과 Return Value는 점선으로 맞추어 작성이 필요
- Actor가 User가 아닌 Customer, Admin으로 세부적인 명세 필요
- 생명선에 대한 더 명확한 명세가 필요, Return이 되기 전에 끝나거나 생명선이 없는 곳에서 시작하는 Event가 존재
- Actor에게 보여지는 부분은 Return이 Actor까지 반환되어야 함
- Amount나 Commission을 Check하는 부분에 대한 Success와 Fail에 대한 경우를 나눠서 표현해줘야 함

2 Category-partition Testing Report

2.1 Test Case

2.1.1 Testable Units

Mode	key	Category
Withdraw	1	송금/인출 금액

Deposit	2	입금 금액
Send	3	송금할 계좌
		송금/인출 금액
CheckRemain	4	
Check	**Key**	**Category**
ChechPassword	1	Password
Limitedmount	2	
FindInfo	3	로그인 계좌
		송금/인출 금액
		Password
PrintStatement	4	PrintInput

2.1.2 Representative Values

Category	Values	Key
로그인 계좌	범위 이내에 있고, DB에 있는 값	1
	범위 이내에 있고, DB에 없는 값	2
	오버플로우/언더플로우 값	3
	음수값	4
	기타 String	5
송금할 계좌	범위 이내에 있고, DB에 있는 값	1
	범위 이내에 있고, DB에 없는 값	2
	오버플로우/언더플로우 값	3
	음수값	4
	기타 String	5
송금/인출 금액	인출범위 안의 값	1
	음수값	2
	오버플로우/언더플로우 값	3
	기타 String	4
입금 금액	입금범위 안의 값	1
	음수값	2
	오버플로우/언더플로우 값	3
	기타 String	4
Password	범위 이내에 있고, DB에 있는 값	1
	범위 이내에 있고, DB에 없는 값	2
	오버플로우/언더플로우 값	3
	음수값	4
	기타 String	5

PrintInput	허용된 대문자, 소문자	1
	허용되지 않은 대문자, 소문자	2
	"true"/"false"	3
	기타 String	4

2.1.3 Error Constraints 적용

Category	Values	Error
로그인 계좌	범위 이내에 있고, DB에 있는 값	-
	범위 이내에 있고, DB에 없는 값	O
	오버플로우/언더플로우 값	O
	음수값	O
	기타 String	O
송금할 계좌	범위 이내에 있고, DB에 있는 값	-
	범위 이내에 있고, DB에 없는 값	O
	오버플로우/언더플로우 값	O
	음수값	O
	기타 String	O
송금/인출 금액	인출범위 안의 값	-
	음수값	O
	오버플로우/언더플로우 값	O
	기타 String	O
입금 금액	입금범위 안의 값	-
	음수값	O
	오버플로우/언더플로우 값	O
	기타 String	O
Password	범위 이내에 있고, DB에 있는 값	-
	범위 이내에 있고, DB에 없는 값	O
	오버플로우/언더플로우 값	-
	음수값	-
	기타 String	O
PrintInput	허용된 대문자, 소문자	-
	허용되지 않은 대문자, 소문자	O
	"true"/"false"	O
	기타 String	O

2.1.4 Property Constraints 적용

Set	Mode	Property
Enviroments	Withdraw	[property Withdraw]
	Deposit	[property Deposit]
	Check	[property Send]
	CheckRemain	[property CheckRemain]
	Check	**Property**
	CheckPassWord	[property CheckPassWord]
	LimitedAmount	[property LimitedAmount]
	FindInfo	[property FindInfo]
	Printstatement	[property PrintStatement]

Category	Values	Property
로그인 계좌	범위 이내에 있고, DB에 있는 값	[if Withdraw][if Findinfo]
	범위 이내에 있고, DB에 없는 값	[if Withdraw][if Findinfo]
	오버플로우/언더플로우 값	[if Withdraw][if Findinfo]
	음수값	[if Withdraw][if Findinfo]
	기타 String	[if Withdraw][if Findinfo]
송금할 계좌	범위 이내에 있고, DB에 있는 값	[if Send]
	범위 이내에 있고, DB에 없는 값	[if Send]
	오버플로우/언더플로우 값	[if Send]
	음수값	[if Send]
	기타 String	[if Send]
송금/인출 금액	인출범위 안의 값	[if Send || Withdraw][if Findinfo]
	음수값	[if Send || Withdraw][if Findinfo]
	오버플로우/언더플로우 값	[if Send || Withdraw][if Findinfo]
	기타 String	[if Send || Withdraw][if Findinfo]
입금 금액	입금범위 안의 값	[if Deposit]
	음수값	[if Deposit]
	오버플로우/언더플로우 값	[if Deposit]
	기타 String	[if Deposit]
Password	범위 이내에 있고, DB에 있는 값	[if CheckPassWord && Findinfo]
	범위 이내에 있고, DB에 없는 값	[if CheckPassWord && Findinfo]
	오버플로우/언더플로우 값	[if CheckPassWord]
	음수값	[if CheckPassWord]
	기타 String	[if CheckPassWord]
PrintInput	허용된 대문자, 소문자	[if PrintStatement]

	허용되지 않은 대문자, 소문자	[if PrintStatement]
	"true"/"false"	[if PrintStatement]
	기타 String	[if PrintStatement]

- 생성된 테스트케이스 43개, 유효 테스트케이스 38개

2.2 Testing Result

Test Case Num.	Key	Result
1	Error	F
2	Error	P
3	Error	P
4	Error	F
5	Error	F
6	Error	F
7	Error	F
8	Error	F
9	Error	F
10	Error	F
11	Error	F
12	Error	F
13	Error	P
14	Error	P
15	Error	F
16	Error	F
17	Error	F
18	Error	F
19	Error	F
20	1.1.0.0.0.0.1.0.	F
21	1.1.0.0.0.0.3.0.	F
22	1.1.0.0.0.0.4.0.	P
23	1.2.0.0.0.0.0.0.	F
24	1.3.1.0.1.0.0.0.	F
25	1.4.0.0.0.0.0.1.	F
26	2.3.1.0.1.1.0.0.	F
27	2.4.0.0.0.1.0.1.	F
28	3.1.0.1.0.0.1.0.	F
29	3.1.0.1.0.0.3.0.	F

30	3.1.0.1.0.0.4.0.	P
31	3.2.0.1.0.0.0.0.	F
32	3.3.1.1.1.0.0.0.	F
33	3.4.0.1.0.0.0.1.	F
34	4.1.0.0.0.0.1.0.	F
35	4.1.0.0.0.0.3.0.	F
36	4.1.0.0.0.0.4.0.	F
37	4.2.0.0.0.0.0.0.	F
39	4.4.0.0.0.0.0.1.	F

6/38 = 15.8% Pass

3 PairWise Testing Report

3.1 Testing Set

Category	Values	Key
로그인 계좌	범위 이내에 있고, DB에 있는 값	1
	범위 이내에 있고, DB에 없는 값	2
	오버플로우/언더플로우 값	3
	음수값	4
	기타 String	5
송금할 계좌	범위 이내에 있고, DB에 있는 값	1
	범위 이내에 있고, DB에 없는 값	2
	오버플로우/언더플로우 값	3
	음수값	4
	기타 String	5
송금/인출 금액	인출범위 안의 값	1
	음수값	2
	오버플로우/언더플로우 값	3
	기타 String	4
입금 금액	입금범위 안의 값	1
	음수값	2
	오버플로우/언더플로우 값	3
	기타 String	4
Password	범위 이내에 있고, DB에 있는 값	1
	범위 이내에 있고, DB에 없는 값	2
	오버플로우/언더플로우 값	3
	음수값	4
	기타 String	5

PrintInput	허용된 대문자, 소문자	1
	허용되지 않은 대문자, 소문자	2
	"true"/"false"	3
	기타 String	4

3.2 Testing Result

Test Case Num.	Key	Result
1	4.1.1.1.4.4	F
2	1.3.3.3.2.2	F
3	3.4.4.4.4.1	F
4	2.3.2.2.5.3.	F
5	2.2.4.1.3.2.	F
6	4.2.2.3.4.3.	F
7	5.4.3.2.1.3.	F
8	2.4.1.4.2.4.	F
9	3.1.3.3.5.1.	F
10	4.1.2.4.1.2.	F
11	4.5.2.1.2.1.	F
12	1.5.4.2.1.4.	F
13	2.1.4.3.3.3.	F
14	3.3.1.3.1.4.	F
15	2.2.3.1.1.1.	F
16	4.4.1.2.3.4.	F
17	5.5.1.4.4.2.	F
18	5.2.2.4.3.1.	F
19	1.4.2.1.5.4.	F
20	4.3.3.4.5.3.	F
21	3.5.3.2.4.3.	F
22	1.2.1.2.2.3.	F
23	3.3.4.1.2.2.	F
24	1.1.1.2.4.1.	F
25	5.3.4.2.4.2.	F
26	1.3.3.3.3.1.	F
27	5.1.2.1.2.3.	F
28	2.5.3.3.3.4.	F
29	5.2.4.3.5.4.	F
30	2.4.2.3.4.2.	F

31	4.2.4.1.2.2.	F
32	3.2.2.4.3.2.	F
33	1.5.2.4.5.3.	F

0/33 = 0% Pass

4 Brute Force Testing Report

4.1 Testing Result

Test Case Num	Test Case	Result
1	페이백 종류 범위값 내 번호 입력	F
2	페이백 종류 입력하지 않음	F
3	페이백 종류 범위값 외 번호 입력	F
4	Printstatement의 입력으로 임의의 스트링	F
5	로그인 계좌와 동일한 계좌로 송금 요청	F
6	입금/출금/송금 금액에 음수값 입력	F
7	100000이상의 입금을 수행하고 DB파일 잔액 변화유무 확인	F
8	DB파일 한도를 음수값으로 설정한 후 프로그램 실행	F
9	영수증 출력(y/n) 입력하지 않음	F
10	이체시 DB에 없는 계좌번호값 입력	F
11	페이백 종류, 영수증 출력값 둘 다 입력하지 않음	P
12	입금-금액입력시 특정 값("333")입력	F
13	입력값이 잘못된 경우에 다음으로 진행하는데(이미 Fail인 상태), 그 때 영수증, 페이백 입력	F
14	수수료 발생 유무 확인	F

1/14 = 7% Pass

4.2 Failed Case Report

Num	Report
1	페이백 번호를 어떻게 설정해도 페이백 종류가 일정하다.
2	페이백 종류를 안넣어도, 페이백 종류 2가 아닌 다른 수를 넣어도 2로 고정된다
3	페이백의 조건은 출금/송금 거래시 인데, 잔액조회 이후에도 페이백 여부를 물어봄
4	페이백으로 제시된 번호 이외의 숫자를 넣어도 에러출력 없이 거래 종료
5	영수증 출력 옵션(y/n)에 따른 결과 변화가 없음

6	로그인 계좌와 송금할 계좌가 같은 경우의 예외처리가 안되어있음
7	출금/송금/입금 거래시 음수값 입력에 대한 예외처리가 안되어있음
8	DB내의 한도값이 음수인 경우에 대한 예외처리가 안되어있음
9	영수증 출력 옵션/페이백 번호 중 하나만 입력되어도 거래가 정상종료된것으로 처리
10	계좌번호 검색 알고리즘에 문제가 있는 것으로 보임
11	수수료 기능 미구현으로 보임
12	입금 거래시 5자리 이상의 큰 값이 들어올 경우 DB에서 잔액의 변화가 없음
13	Actor가 Admin인 상태가 구현이 안되어, DB를 직접 조작하지 않는 한 사용자들 한도 변경 불가능 -> 1억 이상의 잔고를 가져도 한도가 0이면 거래 불가능
14	이전 실행에서의 입력사항(String)값의 찌꺼기가 다음 거래에도 남아있음
15	데이터의 저장 및 변화가 이루어지지 않음.
16	문서상 분류했던 4개의 은행이 아닌 2개의 은행에 대한 DB만 존재함

5 Overall

5.1 System Test Result

5.1.1 Category Partition Test

6/38 = 15.8% Pass

5.1.2 Pairwise Test

0/33 = 0% Pass

5.1.3 Brute Force Test

1/14 = 7% pass

5.1.4 Summary

5.1.4.1 문서의 단계마다 일치하지 않는 부분이 다수 존재한다.

5.1.4.2 치명적으로, Requirement Specification 단계에서 존재하는 Admin에 대한 기능은, Use-case 단계에서 전혀 고려되지 않았고, 실제로 구현되지도 않았다.

5.1.4.3 구현이 정확히 되어있지 않아, Test case의 어떤 Category에서 문제가 발생하였는지 판단하기 힘들 정도로 낮은 Pass율을 기록했다.

5.1.4.4 Project의 purpose가 ATM기기이므로, 기본적으로 다중 클라이언트상 동시 접속에 대한 DB접근 우선순위를 고려하여야 했다. 하지만, SRA문서를 보았을 때, 해당 부분을 전혀 고려하지 않은 아쉬움이 존재한다.