# Unit Testing Plan

## for Public Transportation System

Team NO.3

201311269 김제헌
201311275 박상희
201311276 박형민
201311287 엄현식

# Index

# 1 Purpose

- Public Transportation System(이하 PTS)의 unit test를 수행하는 것이 목표
- SRA3.0의 State Transition Diagram을 바탕으로 test할 부분을 선정하였고, 각 기능이 수행되는 지 확인하는 것에 중점을 두어 작성하였다.
- PTS의 unit test를 수행하기 위한 자원과 절차, test approach와 technique과 필요로 하는 환경 및 도구 등을 정의한다.
- PTS의 unit test는 시스템을 구성하는 최소 단위의 모듈들을 대상으로 하며, 구현된 모듈이 요구사항을 만족하는지를 test한다.

# 2 Test process list

- library module(interface process) 제외
- 단순 데이터 흐름 프로세스 제외
- 기타 확인 불가능한 프로세스 제외
- 직접적으로 State Transition Diagram과 연관되는 모든 프로세스 포함

# 2 Test process list

<Test할 Process(DFD) list > [1/4]

| ID | Name | Description |
|---|---|---|
| 1.2 | Card Info load | CID를 indexing하여 해당 Card의 잔액, 탑승 단말기, 승/하차, 태그 시간, 환승상태 정보를 불러온다. |
| 2.1.1.1 | Catch Error Controller | 정산 여부에 관한 정보를 받아온 뒤, 정산이 이루어지지 않았을 경우 경고메시지를 출력하도록 한다.<br>정산이 이루어졌을 경우 Card Info를 받아와서 정상적인 카드 Tag가 이루어졌는지 판별하고, 아닐 경우 경고메시지를 출력하도록 한다. 정상적일 경우 가격을 측정하도록 한다. |
| 2.1.1.2 | Error | 정상적이지 않은 Tag나 정산이 이루어지지 않았을 경우 경고 메시지를 보낸다. |
| 2.1.1.3 | Fix price | Card Info를 받아온 뒤, 해당 조건에 맞는 가격을 측정한다. |
| 2.1.2.1 | Money Check Controller | 측정된 가격을 받아온 뒤, Card Info의 잔액과 비교한 뒤 부족하면 Short, 충분하면 Calculation을 실행한다. |
| 2.1.2.2 | Short | 가격이 부족할 경우 경고메시지를 출력한다. |

# 2 Test process list

<Test할 Process(DFD) list > [2/4]

| | | |
|---|---|---|
| 2.1.2.3 | Calculation | 잔액이 충분할 경우 잔액-결제금액 을 한 뒤, 승/하차 상태 및 환승 상태, 역 단말기 정보, 잔액을 갱신한 뒤 Updated data로 보내준다. |
| 2.1.3 | Payment Controller | Card 결제가 이루어 진 후, 처리된 정보와 현재 시각을 받아와서 결과값을 출력하고, Card에 갱신, 역 단말기에 기록한다. |
| 2.1.4 | Result | Card 결제가 이루어지고 난 뒤의 결제 금액, 잔액, 현재 시각( Display data)을 보내준다. |
| 2.1.5 | Card update | Card 결제가 이루어지고 난 뒤 바뀐 Card Info를 갱신한다. |
| 2.1.6 | Card Reader Record | Card 결제가 이루어지고 난 뒤, 결제 금액을 Card Reader에 기록한다. |

# 2 Test process list

<Test할 Process(DFD) list > [3/4]

| ID | Name | Description |
|---|---|---|
| 1.2 | Card Info Loader | CID를 사용해서 Card Info를 불러와 Recharger Control 에 전달한다. |
| 2.1.1 | Recharger Controller | 입력받은 Card Info, money Data를 종합하여 충전 계산을 한 뒤, 적절한 update와 display를 실행 해준다. |
| 2.1.2 | Update | 충전된 정보로 교통카드를 갱신한다. |
| 2.1.3 | Display | 교통카드에 충전된 정보를 Monitor에 보여준다. |

# 2 Test process list

<Test할 Process(DFD) list > [4/4]

| ID | Name | Description |
|---|---|---|
| 1 | Card Reader Info Loader | CRID(단말기 고유 ID)를 받아서 해당하는 단말기파일을 열어서 데이터들을 불러와서 데이터 형식에 맞게 각각 저장한다. |
| 2.1 | Fee Calculation Controller | CRID를 통해서 얻은 데이터와 Tick을 받아서 정산을 한 후 상태를 통해서 적절한 프로세스에 명령을 전달한다. |
| 2.2 | Display | Fee Calculation Controller가 정산한 결과를 출력할 때 호출하는 프로세스. 명령을 받으면 adjust_well==0인지 비교(정산이 잘 되었는지)를 하여 정산이 잘 되었으면 adjusted data (fee_bus, fee_metro, time_now)를 출력한다. |
| 2.3 | Send | Fee Calculation Controller가 정산한 결과를 지하철회사와 버스회사에 보낼 때 호출하는 프로세스. Trigger 명령을 받으면 adjust_well==0인지 비교를 하여 정산이 잘 되었으면 버스회사와 지하철 회사에 정산이 잘되었다는 값(c_well)과 정산결과값(fee_bus or fee_metro)을 보낸다. |
| 2.4 | Reset | Fee Calculation Controller가 정산한 결과를 초기화 시킬 때 호출하는 프로세스. Trigger 명령을 받으면 adjust_well==0인지 비교를 하여 정산이 잘 되었으면 모든 단말기 파일을 초기화 시킨다. |

# 2 Test process list

<Test 하지 않을 Process(DFD) list > [1/3]

| ID | Name | Description |
|---|---|---|
| 1.1 | Card Reading Interface | 카드를 Tag했을 때 전달되는 신호를, 컴퓨터가 해석할 수 있는 값을 바꾸어 보내준다. |
| 2.2 | Display interface | 결정된 금액 혹은 경고 메시지, 현재 시각에 관한 정보를 Display data를 통해 받아오고, 정리된 정보를 보내준다. |
| 2.3 | Card interface | 결제 후, 갱신해야 할 Card data를 받아온 뒤, Card에 넘겨준다. |
| 2.4 | Card Reader Recording interface | 결제 후, 결제 금액을 받아온 뒤, 각 Card Reader에 기록하기 위한 정보를 보내준다. |

# 2 Test process list

<Test 하지 않을 Process(DFD) list > [2/3]

| ID | Name | Description |
|---|---|---|
| 1.1 | Card ID Receive Interface | Recharger Sensor로부터 받은 아날로그 신호를 디지털 신호로 변환한다 |
| 2.2 | Update Interface | update card data를 받아서 Card 정보를 Update 시키는 정보를 보내준다 |
| 2.3 | Display interface | display data를 받아서 Monitor에 출력할 display정보를 보내준다. |
| 2.4 | Money Sensor Interface | Money Sensor로부터 받은 아날로그 신호를 디지털 신호로 변환한다. |

# 2 Test process list

<Test 하지 않을 Process(DFD) list > [3/3]

| ID | Name | Description |
|---|---|---|
| 3 | Display Interface | adjusted data(fee_bus, fee_metro, time_now)를 받아서 정산 결과를 출력해주는 Display 장치에 보내준다. |
| 4 | Send Interface | adjusted data(fee_bus, fee_metro, c_well)를 받아서 정산 결과를 버스 회사와 지하철 회사에 보내주는 Send 장치에 보내준다. |

# 3 Composition standard

## Input 설정

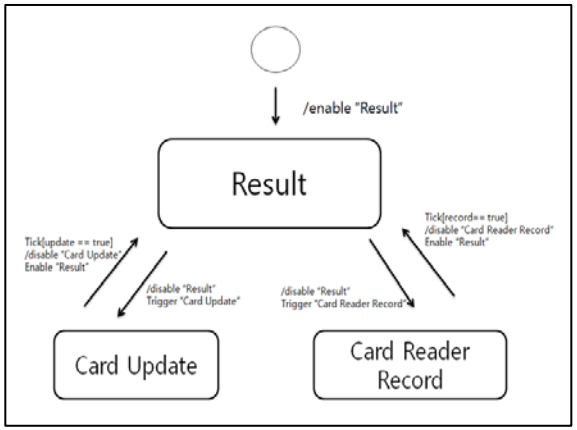
- State Transition Diagram(이하 STD)의 조건에 필요한 input들만 설정
- 의미 있는 데이터만 설정

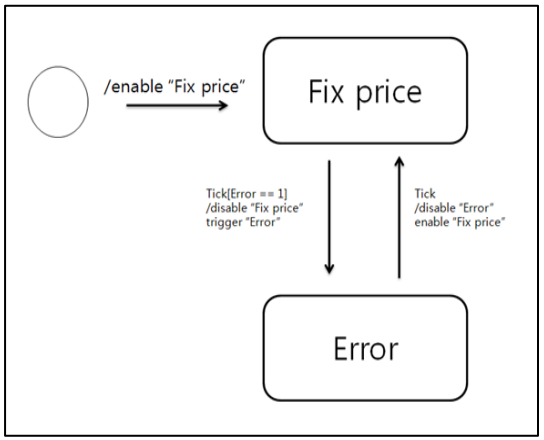
  ex) 정수가 아닌 price는 고려하지 않는다.

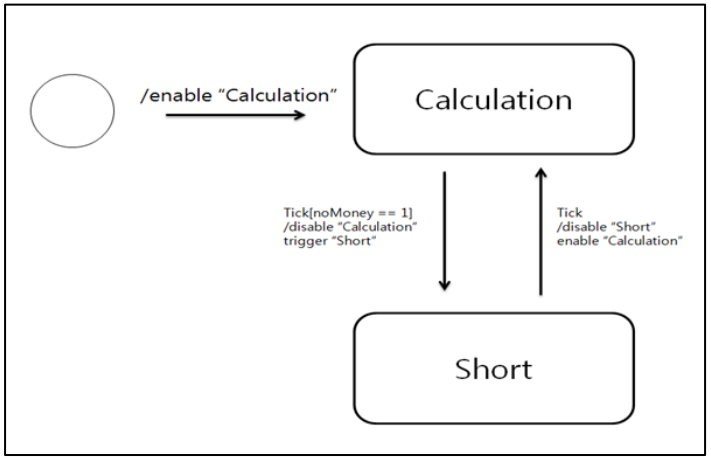
## Controller 뿐만 아니라 STD에서 역할이 존재하는 모든 프로세스 포함

# 4 Unit test design specification

## State Transition Diagram for Controllers in PTS



# 4 Unit test design specification

| | | |
|---|---|---|
| PTS_UTC_2111_000 | 2.1.1.1 Catch Error Controller | card_info의 값과 정수인 price의 값과 stat==Normal인 입력이 들어온다 |
| PTS_UTC_2111_001 | 2.1.1.1 Catch Error Controller | card_info의 값과 정수인 price의 값과 stat!=Normal 인 입력이 들어온다 |
| PTS_UTC_2112_000 | 2.1.1.2 Error | interval_sec <=0 인 입력이 들어온다. |
| PTS_UTC_2112_001 | 2.1.1.2 Error | state==1,interval_sec<15,getout==0이고 CRID의 값이 card_info의 CRID의 값과 같은 입력이 들어온다. |
| PTS_UTC_2112_002 | 2.1.1.2 Error | state==1,interval_sec>=15,getout==0이고 CRID의 값이 card_info의 CRID의 값과 같은 입력이 들어온다. |
| PTS_UTC_2112_003 | 2.1.1.2 Error | state==1,interval_sec<15,getout!=0이고 CRID의 값이 card_info의 CRID의 값과 같은 입력이 들어온다. |
| PTS_UTC_2112_004 | 2.1.1.2 Error | state==1,interval_sec<15,getout==0이고 CRID의 값이 card_info의 CRID의 값과 다른 입력이 들어온다. |
| PTS_UTC_2112_005 | 2.1.1.2 Error | state==1,interval_sec<15,getout!=0이고 CRID의 값이 card_info의 CRID의 값과 다른 입력이 들어온다. |
| PTS_UTC_2112_006 | 2.1.1.2 Error | state==1,interval_sec>=15,getout!=0이고 CRID의 값이 card_info의 CRID의 값과 같은 입력이 들어온다. |
| PTS_UTC_2112_007 | 2.1.1.2 Error | state==1,interval_sec>=15,getout==0이고 CRID의 |

# 5 Unit test case specification

| | | |
|---|---|---|
| PTS_UTC_2111_000 | stat==Normal | Error검사를 실행하고 가격을 책정한다. |
| PTS_UTC_2111_001 | stat!=Normal | |
| PTS_UTC_2112_000 | interval_sec <=0 | interval_sec += 60; |
| PTS_UTC_2112_001 | state==1/interval_sec<15/getout==0/CRID==card_info->CRID | interval_sec += 60;<br>stat = HopInProcessing; |
| PTS_UTC_2112_002 | state==1/interval_sec>=15/getout==0/CRID==card_info->CRID | Stat의 값을 변경시키지 않는다. |
| PTS_UTC_2112_003 | state==1/interval_sec<15/getout!=0/CRID==card_info->CRID | Stat의 값을 변경시키지 않는다. |
| PTS_UTC_2112_004 | state==1/interval_sec<15/getout==0/CRID!=card_info->CRID | Stat의 값을 변경시키지 않는다. |
| PTS_UTC_2112_005 | state==1/interval_sec<15/getout!=0/CRID!=card_info->CRID | Stat의 값을 변경시키지 않는다. |
| PTS_UTC_2112_006 | state==1/interval_sec>=15/getout!=0/CRID==card_info->CRID | Stat의 값을 변경시키지 않는다. |
| PTS_UTC_2112_007 | state==1/interval_sec>=15/getout==0/CRID!=card_info->CRID | Stat의 값을 변경시키지 않는다. |
| PTS_UTC_2112_008 | state==1/interval_sec>=15/getout!=0/CRID!=card_info->CRID | Stat의 값을 변경시키지 않는다. |

# 5 Unit test case specification

<Table 2.3 Test Design Identification>

| Identifier | Feature | Valid/Invalid Value |
|---|---|---|
| PTS_UTC_120_000 | 1.2 Card Info Loader | Card.txt파일을 열어 입력받은 CID값이 있는 줄을 찾고, 찾으면 카드의 정보를 덮어씌운다. |
| PTS_UTC_120_001 | 1.2 Card Info Loader | Card.txt파일을 열어 입력받은 CID값이 있는 줄을 찾고, 찾지 못하면 카드의 정보를 초기화해 준다. |
| PTS_UTC_120_002 | 1.2 Card Info Loader | Card.txt파일이 지정한 상대경로에 존재하지 않는다면, 파일열기 실패를 출력하면서 프로그램이 종료된다. |
| PTS_UTC_211_000 | 2.1.1 Recharger Controller | 적합한 CID값을 입력받아서, card_info값을 갱신한 후에는, money값을 입력 받고, 잔액을 충전한 후에, 충전된 정보로 교통카드를 갱신하고, 충전시각과 함께 교통카드에 충전된 정보를 Monitor에 보여준다. |
| PTS_UTC_211_001 | 2.1.1 Recharger Controller | 적합한 CID값을 입력받지 못해서, card_info값을 0으로 모두 초기화한 후일지라도, money값을 입력 받고, 잔액을 충전한다. 충전된 정보로 교통카드를 갱신하려 하지만, Card.txt파일에서 일치하는 CID정보가 없으므로 실질적으로 갱신은 이루어지지 않는다. 이후, 충전시각과 함께 교통카드에 충전된 정보를 Monitor에 보여준다 |

# 5 Unit test case specification

| | | |
|---|---|---|
| FCS.UTC.021.014 | 2.1 Fee Calculation Controller | Adjust상태 일 때, (배열의 처음)|| (배열의 마지막) ||(state==1, transfer==0)이면 다음 state!=1, transfer!=0인 전까지 정산 계산을 할 구간을 정해서 index에 배열 인덱스를 넣는다. |
| FCS.UTC.021.015 | 2.1 Fee Calculation Controller | Adjust상태 일 때, index!=NULL이면 real_fee+= real_CardReader_info[i].price를 수행한다. |
| FCS.UTC.021.016 | 2.1 Fee Calculation Controller | Adjust상태 일 때, index==NULL이면 real_fee+= real_CardReader_info[i].price를 수행한다. |
| FCS.UTC.021.017 | 2.1 Fee Calculation Controller | Adjust상태 일 때, real_CardReader_info[i-1].tp != real_CardReader_info[i].tp이면 total_fee+=real_fee를 수행한다. |
| FCS.UTC.021.018 | 2.1 Fee Calculation Controller | Adjust상태 일 때, real_CardReader_info[i-1].tp == real_CardReader_info[i].tp이면 total_fee+=real_fee를 수행한다. |
| FCS.UTC.021.019 | 2.1 Fee Calculation Controller | Adjust상태 일 때, index!=NULL이면 temp_fee+= real_CardReader_info[i].price를 수행한다. |
| FCS.UTC.021.020 | 2.1 Fee Calculation Controller | Adjust상태 일 때, index==NULL이면 temp_fee+= real_CardReader_info[i].price를 수행한다. |
| FCS.UTC.021.021 | 2.1 Fee Calculation Controller | Adjust상태 일 때, ((real_CardReader_info[j-1].tp != real_CardReader_info[j].tp) || i==j), real_CardReader_info[j].tp==0이면 *bus_fee+=temp_fee/total_fee*real_fee를 수행한다. |

# 5 Unit test case specification

| | | |
|---|---|---|
| | if(file != NULL)↵ | tagTime={2014, 11, 20, 19, 52, 18}, tp=1, state=1, cash=30900, transfer=0, getout=1}↵ |
| PTS_UTC_120_001↵ | CID=100 /↵ if(file != NULL)↵ | card_info={CID=0, CRID=0, tagTime={?, ?, ?, ?, ?, ?}, tp=0, state=0, cash=0, transfer=0, getout=0}↵ |
| PTS_UTC_120_002↵ | if(file == NULL)↵ | "파일이 없습니다. 파일 열기 실패"↵ exit(1);exit(1);↵ |
| PTS_UTC_211_000↵ | CID=1000 /↵ money=20000↵ | card_info.cash+=money / trigger "Update"/↵ trigger "Display"↵ |
| PTS_UTC_211_001↵ | CID=100 /↵ money=20000↵ | card_info.cash+=money / trigger "Update"/↵ trigger "Display"↵ |
| PTS_UTC_212_000↵ | card_info={CID=1000, CRID=11, tagTime={2014, 11, 20, 19, 52, 18}, tp=1, state=1, cash=50900, transfer=0, getout=1}↵ | fprintf(newFile, "....", card_info->CID, .... card_info->getout);이 한번 수행↵ |
| PTS_UTC_212_001↵ | card_info={CID=0, CRID=0, tagTime={?, ?, ?, ?, ?, ?}, tp=0, state=0, cash=0, transfer=0, getout=0}↵ | fprintf(newFile, "....", card_info->CID, .... card_info->getout);이 한번도 수행되지 못함. ↵ |

# 5 Unit test case specification

<Table 3.4 Test Case Identification>

| Test Case Identifier | Input Specification | Output Specification |
|---|---|---|
| FCS.UTC.010.000 | Tick()==0/ *.txt(단말기)!=NULL | fscanf( , ,&CardReader_info[i].CID,,,,,,,) |
| FCS.UTC.010.001 | Tick()==0/ *.txt(단말기)==NULL | |
| FCS.UTC.010.002 | Tick()==1/ *.txt(단말기)!=NULL | |
| FCS.UTC.010.003 | Tick()==1/ *.txt(단말기)==NULL | |
| FCS.UTC.010.004 | CardReader_info[i].CID==1000 | fprint(file_CID_1000 , ,CardReader_info[i].CID,,,,,,,) |
| FCS.UTC.010.005 | CardReader_info[i].CID==1001 | fprint(file_CID_1001, ,CardReader_info[i].CID,,,,,,,) |
| FCS.UTC.010.006 | CardReader_info[i].CID==1002 | fprint(file_CID_1002, ,CardReader_info[i].CID,,,,,,,) |
| FCS.UTC.010.007 | CardReader_info[i].CID==1003 | fprint(file_CID_1003, ,CardReader_info[i].CID,,,,,,,) |
| FCS.UTC.010.008 | CardReader_info[i].CID==1004 | fprint(file_CID_1004, ,CardReader_info[i].CID,,,,,,,) |
| FCS.UTC.010.009 | CardReader_info[i].CID==1005 | fprint(file_CID_1005, ,CardReader_info[i].CID,,,,,,,) |
| FCS.UTC.010.010 | CardReader_info[i].CID==1006 | fprint(file_CID_1006, ,CardReader_info[i].CID,,,,,,,) |
| FCS.UTC.010.011 | CardReader_info[i].CID>1006 \|\|<br>CardReader_info[i].CID<1000 | |
| FCS.UTC.021.000 | AdjustStart() | c_well=1 |
| FCS.UTC.021.001 | !AdjustStart() | c_well=1 |
| FCS.UTC.021.002 | CardReader_info[i].state==1 | CardReader_info[i].getout=1 |
| FCS.UTC.021.003 | CardReader_info[i].state==1 | CardReader_info[i].getout=0 |
| FCS.UTC.021.004 | CardReader_info[i].state==0 | CardReader_info[i].getout=1 |

# QnA

Thank

You