Unit Testing Plan

for PTS System

- **Test Plan**
- **Test Design Specification**
- **Test Cases Specification**

Project Team

Class B Team 3

Date

2014-11-20

Team Information

201111333 권태헌

201111375 윤지수

201111379 이한빈

201111384 정국빈

Table of Contents

1 Introduction

1.1 Objectives

Public Transportation System(이하 PTS)의 최종적인 구현을 위해 각 함수별로 unit test를 실행한다. Unit test는 PTS의 버스단말기, 지하철단말기 및 정산시스템의 3가지 시스템에 대해 독립적으로 실행한다. 또한 unit test는 시스템 동작을 위해 필요한 활동과 기준에 대해 정의하고 테스트 도구들에 관한 세부사항을 명시한다.

1.2 Background

PTS는 사용자가 교통카드를 단말기에 태그함으로서, 승하차 및 환승이 이루어지는 시스템이다. 또한 단말기는 여러 사용자의 card data를 관리하며 특정 시간마다 정산 시스템에 데이터를 보낸다. 정산 시스템은 받은 데이터를 분석하고 사용자가 사용한 교통요금을 계산한다. 단말기와 정산시스템은 여러 개의 입력 뿐만 아니라, 시스템 작동에 관련된 다양한 상태 값을 가지고 있다. Unit test를 통해 시스템에 관련된 데이터 및 프로세스들이 요구사항을 만족하고, 올바르게 작동하는지 확인한다.

1.3 Scope

이 계획 문서는 PTS의 Unit Test를 수행하기 위한 모든 것을 포함한다.

1.4 Configuration management plan

PTS의 Program soucrce code 및 Unit Test를 위한 Test Code는 Cygwin 환경에서 이루어진다. Program Source code의 변경 및 수정 사항은 지속적으로 통합되고 test된다.

1.4.1 Program Source code의 변경

Program Source code에 수정사항이 발생 할 경우, 이를 통합하고 수동적으로 Unit Test를 수행한다.

1.4.2 일정 주기

Program Source code는 일정 주기를 가지고 팀원들과 build 및 unit test를 수행한다.

1.5 References

T3.2014.PTS.SRA

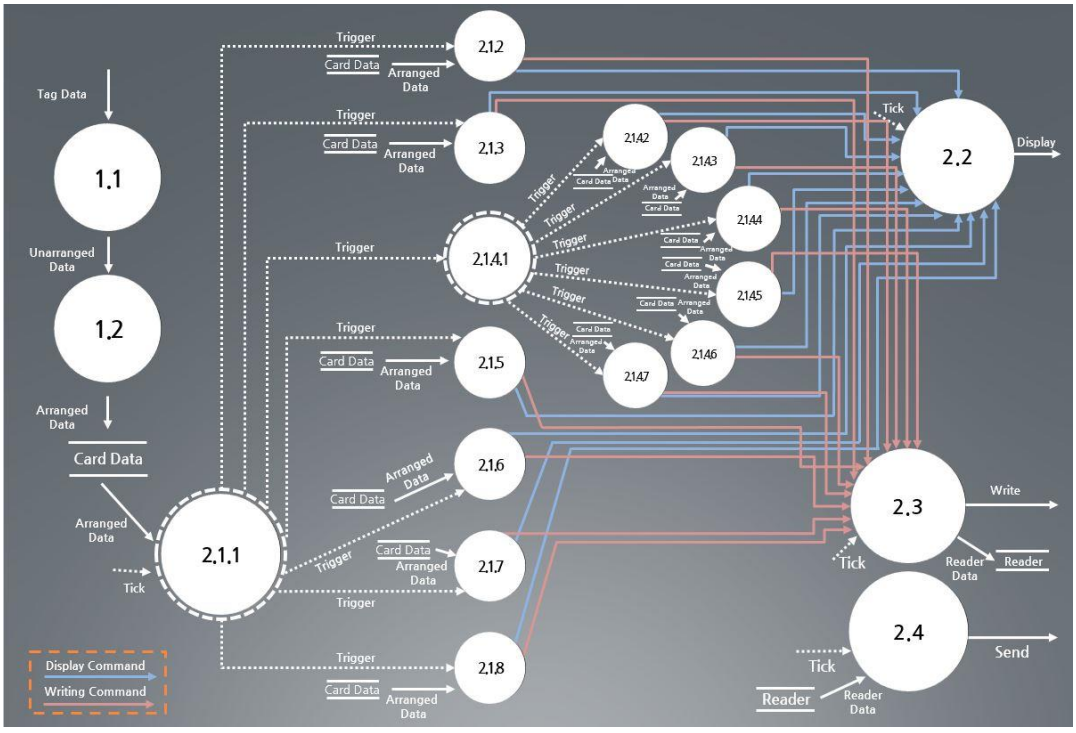
T3.2014.PTS.SDS

2 Test items

PTS를 구성하는 최소 단위의 모듈들이 Unit Test의 대상이 된다. 각 모듈의 input에 임의의 값을 대입 했을 때 원하는 output이 나오는지 test하며 test item 들은 다음 자료들로부터 작성되었다.

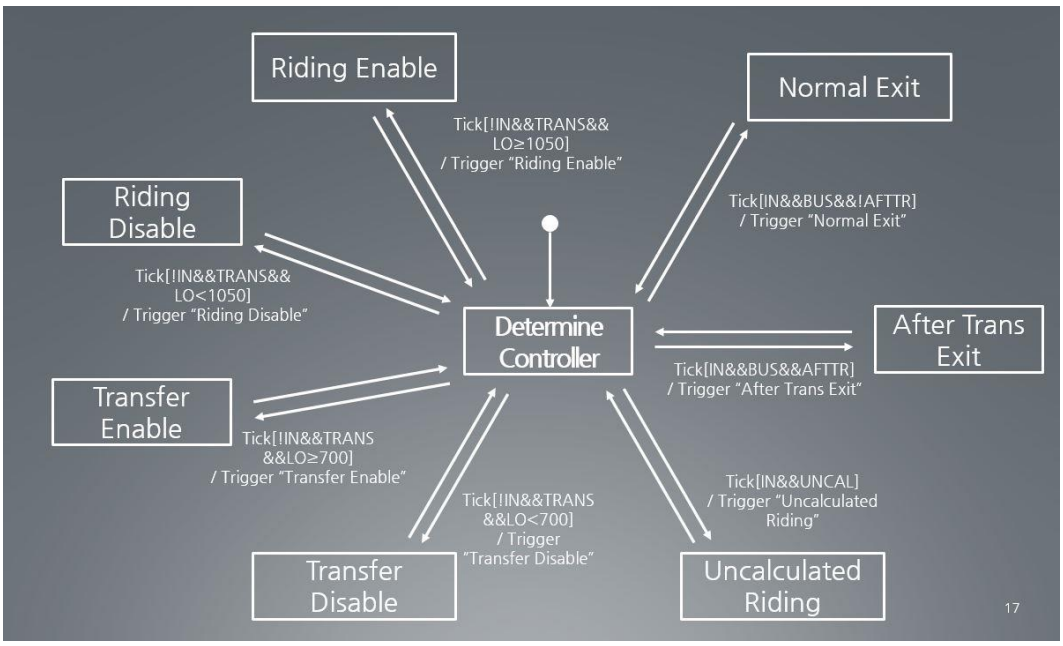
2.1.1 Overall of modules (T3_2014_SRA_3.0 참고)

아래 그림은 일부를 참조한 것이다.



<Bus DFD Overall>

2.1.2 STD of modules (T3.2014.PTS.SRA-3.0 참고)



<Reader State Transition Diagram>

3 Features to be tested

<BUS&SUBWAY> 테스트할 Process (DFD) 리스트

ID	Name	Description
1.1	Receiving Card Interface	카드가 태그 되었을 시 해당 카드의 텍스트파일을 읽어와서 가장 최근 거래내역 2개(Unarranged Data)를 내보낸다.
1.2	Arranging Card Data Interface	Unarranged Data를 받아서 UNAL, TRANS, STA 등 Arranged Data 로 정리 후 내보낸다.
2.1.1	Determine Controller	Arranged Data 를 받아 어떠한 경우인지 판단하여 해당하는 경우의 프로세스를 동작시킨다.
2.1.2	Riding Enable	일반 승차이고 탑승이 가능할 때 실행되며 요금을 계산하여 Command를 내보낸다.

<Calculator> 테스트할 Process (DFD) 리스트

ID	Name	Description
1.1	Receiving Reader Data Interface	버스와 지하철로부터 단말기 거래 내역을 받아 종합하여 전송한다.

2.1.1.	Classify Case	종합된 기록을 버스만 이용한 경우, 지하철만 이용한 경우, 버스와 지하철을 환승하여 이용한 경우의 3가지 경우로 분류해서 각각의 계산 프로세스로 전송한다.

4 Features not to be tested

- 외부 장치 드라이버, 단순 데이터조작 프로세스, 등은 test에서 제외한다.

<BUS&SUBWAY> 테스트하지 않을 Process (DFD) 리스트

2.1.3	Riding Disable	일반 승차의 경우지만, 잔액이 부족하여 탑승이 불가한 경우에 실행되며 잔액이 부족하다는 Command를 내보낸다.
2.1.4.1	Uncalculated Controller	Arranged Data를 받아서 어떠한 경우의 미정산 금액인지 판단하여 해당하는 프로세스를 동작시킨다.
2.1.4.2	After Sub-E	지하철 승차 후 하차 시 태그하지 않고 내려 발생한 미정산 요금이 있고 잔액이 충분해 탑승이 가능한 경우 실행되며 추가요금을 계산하여 Command를 내보낸다.
2.1.4.3	After Sub-D	지하철 승차 후 하차 시 태그하지 않고 내려 발생한 미정산 금액이 있으나 잔액이 부족한 경우 실행되며 잔액이 부족하다는 Command를 내보낸다.
2.1.4.4	After Subus-E	지하철에서 버스 환승 후 하차 시 태그하지 않아 발생한 미정산 금액이 있고 잔액이 충분해 탑승이 가능한 경우 실행되며 요금을 계산하여 Command를 내보낸다..
2.1.4.5	After Subus-D	지하철에서 버스 환승 후 하차시 태그하지 않아 발생한 미정산 금액이 있고 잔액이 부족한 경우 실행되며 잔액이 부족하다는 Command를 내보낸다.
2.1.4.6	After Busub-E	버스에서 지하철 환승 후 하차 시 태그하지 않아 발생한 미정산 금액이 있고 잔액이 충분해 탑승이 가능한 경우 실행되며 요금을 계산하여 Command를 내보낸다.
2.1.4.7	After Busub-D	버스에서 지하철 환승 후 하차 시 태그하지 않아 발생한 미정산 금액이 있어 잔액이 부족하여 탑승하지 못하는 경우 실행되며 잔액이 부족한다는 정보를 내보낸다.
2.1.5	Transafter Enable	환승 승차하는 경우 실행되며 요금을 계산하여 Command를 내보낸다..
2.1.6	Transafter Disable	환승 승차 시, 잔액이 부족하여 탑승하지 못하는 경우 실행되며 잔액이 부족하다는 Command를 내보낸다.
2.1.7	Normal Exit	일반 승차 후 하차의 경우 실행되며 요금을 계산하여 Command를 내보낸다.

2.1.7.2	Normal Exit Controller	Arranaged Data 를 받아와서 추가 요금이 있는지 판단하여 해당 프로세스를 동작시킨다.
2.1.7.3	Add Fee	일반 승차 후 두 개 정거장 이상 이동한 경우 실행되며 추가요금을 계산하여 Command를 내보낸다.
2.1.7.4	No Add Fee-N	일반 승차 후 한 개 정거장 이하로 이동한 경우 실행되며 추가 요금을 부과하지 않고 하차 처리하는 Command를 내보낸다.
2.1.8.2	After Trans Exit Controller	Arranged Data 를 받아와서 추가요금이 있는지 판단하여 해당하는 경우의 프로세스를 동작시킨다.
2.1.8.3	Add Fee One	환승 승차 후 한 개 정거장을 이동한 경우 실행되며 추가요금을 계산하여 Command를 내보낸다.
2.1.8.4	Add Fee Two	환승 승차 후 두 개 정거장 이상 이동한 경우 실행되며 추가요금을 계산하여 Command를 내보낸다.
2.1.8.5	No Add Fee-T	환승 승차 후 정거장을 이동하지 않은 경우 실행되며 추가요금을 부과하지 않고 하차 처리하는 Command를 내보낸다..
2.2	Monitor Interface	Main Control 이 보내주는 Display Command를 받아 모니터가 출력을 하도록 한다
2.3	Writing Interface	Main Control이 보내주는 Writing Command를 받아 카드와 단말기에 기록하도록 한다.

<Calculator> 테스트하지 않을 Process (DFD) 리스트

2.1.2	Trans Calculator	환승한 모든 경우를 정산 공식을 적용해서 계산한다.
2.1.3	Bus Calculator	버스만 이용한 경우의 정산 방법을 적용해서 계산한다.
2.1.4	Subway Calculator	지하철만 이용한 경우의 정산 방법을 적용해서 계산한다.
2.1.5	Final Calculator	3가지 경우에서 정산된 정보들을 받아 합산하여 Display, Sending, Signal Command를 전송한다.
2.2	Display Interface	Main Control 이 보내주는 Display Command를 받아 모니터 화면에 출력한다.
2.3	Sending Data Interface	Main Control 이 보내주는 Sending Command를 받아 각 회사로 정산 기록을 전송한다.
2.4	Signal Interface	Main Control 로부터 정산이 완료되었다는 정보를 받아 단말기에 보낸다.

5 Approach

PTS의 Program Source code 및 Unit test를 위한 Test Code 는 Visual Studio 환경에서 이루어지며, Program Source code 의 변경 및 수정 사항은 지속적으로 통합되고 test된다.

6 Item pass/fail criteria

각 모듈은 요구사항을 모두 만족하여야 한다.

7 Unit test design specification

7.1 Test design specification identifier

PTS_UTC_000_000

7.2 Features to be tested

- 본 문서 3. Features to be tested 참조

7.3 Test identification

<BUS&SUBWAY>

Identifier	Feature (Process ID in DFD)	Valid value
PTS_UTC_000	1.2 ArrangingCardDataInterface	UnarrangedData를 입력받아 해당하는 ArrangedData를 생성하여 출력한다.
PTS_UTC_001	2.1.1 DetermineController	ArrangedData를 받아 해당하는 프로세스를 동작시킨다.
PTS_UTC_002	2.1.2 RidingEnable	ArrangedData를 받아 처리하여 Command를 생성하여 출력한다.
PTS_UTC_003	1.1 ReceivingCardDataInterface	텍스트파일을 읽어와 UnarrangedData를 생성하여 출력한다.

<Calculator>

Identifier	Feature (Process ID in DFD)	Valid value
PTS_UTC_004	1.1 ReceivingReaderDataInterface	Bus.txt, subway1.txt, subway2.txt, subway3.txt, subway4.txt, subway5.txt를 읽어서 inputdata.txt를 만든다.
PTS_UTC_005	2.1.1 ClassifyCase	Inputdata.txt를 읽어서 un_trans.txt, un_bus.txt, un_sub.txt를 만든다.

8 Unit test case specification

8.1 Test case specification identifier

<BUS&SUBWAY> Test case Identification

Identifier	Input Specification	Output Specification
PTS_UTC_000_000	Unarranged_data[2] = {20141117195032, "BUS", "OUT", 5950, "B_1"}, {20141117195035, "BUS", "IN", 5950, "B_2"}	Arranged_data의 In == True
PTS_UTC_000_001	Unarranged_data[2] = {20141117195032, "BUS", "OUT", 5950, "B_1"}, {20141117195035, "BUS", "IN", 5950, "B_2"}	Arranged_data의 Bus == True
PTS_UTC_000_002	Unarranged_data[2] = {20141117195032, "SUBWAY", "IN", 5950, "S1_1"}, {20141117195035, "SUBWAY", "OUT", 5950, "S2_2"} (and 마지막 태그시간이 현재시간과 15초이내)	Arranged_data의 Trans == True
PTS_UTC_000_003	Unarranged_data[2] = {20141117195032, "SUBWAY", "OUT", 5950, "S1_1"}, {20141117195035, "BUS", "IN", 5950, "B_1"}	Arranged_data의 After == True
PTS_UTC_000_004	Unarranged_data[2] = {20141117195032, "SUBWAY", "OUT", 5950, "S1_1"}, {20141117195035, "BUS", "IN", 5950, "B_1"} 이고 하루(3분)가 지났을 때	Arranged_data의 Uncal_day == True
PTS_UTC_000_005	Unarranged_data[2] = {20141117195032, "SUBWAY", "OUT", 5950, "S1_1"}, {20141117195035, "SUBWAY, "IN", 5950, "S1_2"}	Arranged_data의 Uncal_s == True
PTS_UTC_000_006	Unarranged_data[2] = {20141117195032, "BUS", "OUT", 5950, "B_1"}, {20141117195035, "SUBWAY, "IN", 5950, "S1_2"}	Arranged_data의 Uncal_bs == True
PTS_UTC_000_007	Unarranged_data[2] = {20141117195032, "SUBWAY", "OUT", 5950, "B_1"}, {20141117195035, "BUS, "IN", 5950, "S1_2"	Arranged_data의 Uncal_sb == True
PTS_UTC_000_008	Unarranged_data[2] = {20141117195032, "BUS", "OUT", 5950, "B_1"}, {20141117195035, "SUBWAY, "IN", 5950, "S1_2"}	Arranged_data의 Uncal == True
PTS_UTC_000_009	Unarranged_data[2] =	Arranged_data의

	{20141117195032, "BUS", "OUT", 5950, "B_1"}, {20141117195035, "SUBWAY, "IN", 5950, "S1_2"}	Time == 20141117195035
PTS_UTC_000_010	Unarranged_data[2] = {20141117195032, "BUS", "OUT", 5950, "B_1"}, {20141117195035, "SUBWAY", "IN", 5950, "S1_2"}	Arranged_data의 Lo == 5950
PTS_UTC_000_011	Unarranged_data[2] = {20141117195032, "BUS", "OUT", 5950, "B_1"}, {20141117195035, "SUBWAY", "IN", 5950, "S1_2"}	Arranged_data의 Info == S1_2
PTS_UTC_001_000	Unarranged_data[2] = {20141119133000, ",SUBWAY", "IN", 10000, "S1_1"}, {20141119133010, "SUBWAY, "OUT", 10000, "S2_1"}	RidingEnable함수 호출
PTS_UTC_001_001	Unarranged_data[2] = {20141119133000, ",SUBWAY", "IN", 10000, "S1_1"}, {20141119133010, "SUBWAY", "OUT", 200, "S2_1"}	ridingDisable함수 호출
PTS_UTC_001_002	Unarranged_data[2] = {20141119133000, ",SUBWAY", "IN", 10000, "S1_1"}, {20141119133010, "SUBWAY", "OUT", 10000, "S2_1"} (and 마지막 태그시간과 현재시간이 15초 이내)	transferEnable함수 호출
PTS_UTC_001_003	Unarranged_data[2] = { 20141119133000, ",SUBWAY", "IN", 10000, "S1_1"}, { 20141119133010, "SUBWAY", "OUT", 100, "S2_1"}	transferDisable함수 호출
PTS_UTC_001_004	Unarranged_data[2] = {20141119133000, "BUS", "OUT", 10000, "B_1"}, {20141119133010, "BUS," "IN", 10000, "B_2"}	normalExit함수 호출
PTS_UTC_001_005	Unarranged_data[2] = {20141119133000, "SUBWAY", "OUT", 10000, "S1_1"}, {20141119133010, "BUS," "IN", 10000, "B_2"}	afterTransExit함수 호출
PTS_UTC_001_006	Unarranged_data[2] = {20141119133000, "BUS", "OUT", 10000, "B_1"}, {20141119133110, "SUBWAY" "IN", 10000, "S1_2"}	afterSub_E함수 호출
PTS_UTC_001_007	Unarranged_data[2] = {20141119133000, "BUS", "OUT", 10000, "B_1"}, {20141119133110, "SUBWAY" "IN", 100, "S1_2"}	afterSub_D함수 호 출
PTS_UTC_001_008	Unarranged_data[2] = {20141119133000, "SUBWAY", "OUT", 10000, "S1_1"}, {20141119133010, "BUS" "IN", 10000, "B_2"}	afterSubus_E함수 호 출
PTS_UTC_001_009	Unarranged_data[2] = {20141119133000, "SUBWAY", "OUT", 10000, "S1_1"},	afterSubus_D함수 호출

	{20141119133010, "BUS" "IN", 100, "B_2"}	
PTS_UTC_001_010	Unarranged_data[2] = {20141119133000, "BUS", "OUT", 10000, "B_1"}, {20141119133010, "SUBWAY" "IN", 10000, "S1_2"}	afterBusub_E함수 호출
PTS_UTC_001_011	Unarranged_data[2] = {20141119133000, "BUS", "OUT", 10000, "B_1"}, {20141119133010, "SUBWAY" "IN", 100, "S1_2"}	afterBusub_D함수 호출
PTS_UTC_001_012	Unarranged_data[2] = {20141119133000, "BUS", "OUT", 10000, "B_1"}, {20141119133110, "SUBWAY" "IN", 10000, "S3_2"}	addFee함수 호출
PTS_UTC_001_013	Unarranged_data[2] = {20141119133000, "BUS", "OUT", 10000, "B_1"}, {20141119133110, "SUBWAY" "IN", 10000, "S4_2"}	noAddFee_N함수 호출
PTS_UTC_001_014	Unarranged_data[2] = {20141119133000, "BUS", "OUT", 10000, "B_1"}, {20141119133010, "SUBWAY" "IN", 10000, "S4_2"}	addFeeOne함수 호출
PTS_UTC_001_015	Unarranged_data[2] = {20141119133000, "BUS", "OUT", 10000, "B_1"}, {20141119133010, "SUBWAY" "IN", 10000, "S3_2"}	addFeeTwo함수 호출
PTS_UTC_001_016	Unarranged_data[2] = {20141119133000, "BUS", "OUT", 10000, "B_1"}, {20141119133010, "SUBWAY" "IN", 10000, "S5_2"}	noAddFee_T함수 호출
PTS_UTC_002_000	Arranged_data {cname[20]="input.txt", time=20141117195035, info[10]=B_2, lo=5950, in=0, bus=1, trans=0, after=0, uncal_day=0, uncal_s=0 uncal_bs=0 uncal_sb=0 uncal=0, sta=0} 위에 ad는 밑에 있는 경우에 해당함 {20141117195032, "BUS", "IN", 5950, "B_1"}, {20141117195035, "BUS, "OUT", 5950, "B_2"}	Display_command (fee=1050, lo=4900) Writing command {fee=1050, time=20141117195035, vechicle="BUS", inout="IN", lo=4900, readerInfo="B_1", cardInfo="B_1", name="input.txt"}
PTS_UTC_003_000	임의의 카드정보가 담긴 텍스트파일	unarranged_data(텍스트파일의 가장 밑의 두줄)

<Calculator> Test case Identification

Identifier	Input Specification	Output Specification
PTS_UTC_004_000	Bus.txt, subway1.txt, subway2.txt, subway3.txt, subway4.txt, subway5.txt	Inputdata.txt
PTS_UTC_005_000	Inputdata.txt	un_trans.txt, un_bus.txt, un_sub.txt

8.2 Test items

- 8.1의 Test case Identification 참조

8.3 Input specifications

- 8.1의 Test case Identification 참조

8.4 Output specifications

- 8.1의 Test case Identification 참조

9 Environmental needs

PTS의 unit test를 위한 환경적 요구사항은 다음과 같다

IDE : Visual Studio