Unit Testing Plan

for Public Transportation System

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

Project Team

Team 2

Date

2014-11-20

Team Information
201111341 김성민
201111391 진청현
201311259 권오승
201311303 이정은

Table of Contents

1 Introduction

1.1 Objectives

이 문서는 T2의 Public Transportation System의 unit test를 수행하기 위해 작성된 계획 문서이며, 본 system이 제대로 작동하는지를 살펴보기 위해 필요한 요소들을 정리해 놓은 문서이다. 앞선 단계에서 작성했던 T2.2014.PTS.SRA-3.0의 State Transition Diagram을 바탕으로 test할 부분을 선정하였고, 각 기능이 제대로 수행되는지 확인하는 것에 중점을 두어 작성하였다.

Test를 수행하기 위해 필요한 활용 및 자원을 정의하고, test approach 및 techniques를 정의한다. 또한 test를 위한 환경적인 요구사항 및 test 도구들을 정의한다.

1.2 Background

Public Transportation System은 대중교통 시스템으로, 사용자가 단말기에 접촉하는 교통카드에 의해 탑승이 가능한지 불가능한지 환승인지 아닌지를 제어하고 하루가 끝나면 하루 동안 접촉된 카드의 요금과 정보를 이용해서 정산을 해주는 시스템이다. 이 시스템은 카드 센서 입력을 가지고 있고 실행 시간도 시스템의 성능을 좌우하는 중요한 요소이다. Unit test는 시스템을 구성하는 최소 단위 모듈들을 대상으로 하는 test이며, 시스템의 성능을 좌우하는 요소들이 요구사항을 만족하는지를 확인 할 수 있는 기본적은 test approach이다.

1.3 Scope

이 계획 문서는 Public Transportation System(이하 PTS)의 unit test를 수행하기 위한 모든 것을 포함한다. 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 test 한다.

1.4 Project plan

1.5 Configuration management plan

Public Transportation System의 program source code 및 unit test를 위한 test code는 Cygwin환경에서 이루어지며, program source code/ test code의 변경 및 수정 사항은 지속적으로 통합되고 test 된다.

(1) Program source code의 변경

Program source code에 변경 및 수정 발생 시, 이를 통합하고 수동적으로 unit test를 수행한다.

(2) 일정주기

program source code는 일정 주기적으로 build 및 unit test를 수행한다.

1.6 References

T2.2014.PTS.SRS-1.0

T2.2014.PTS.SRA-3.0

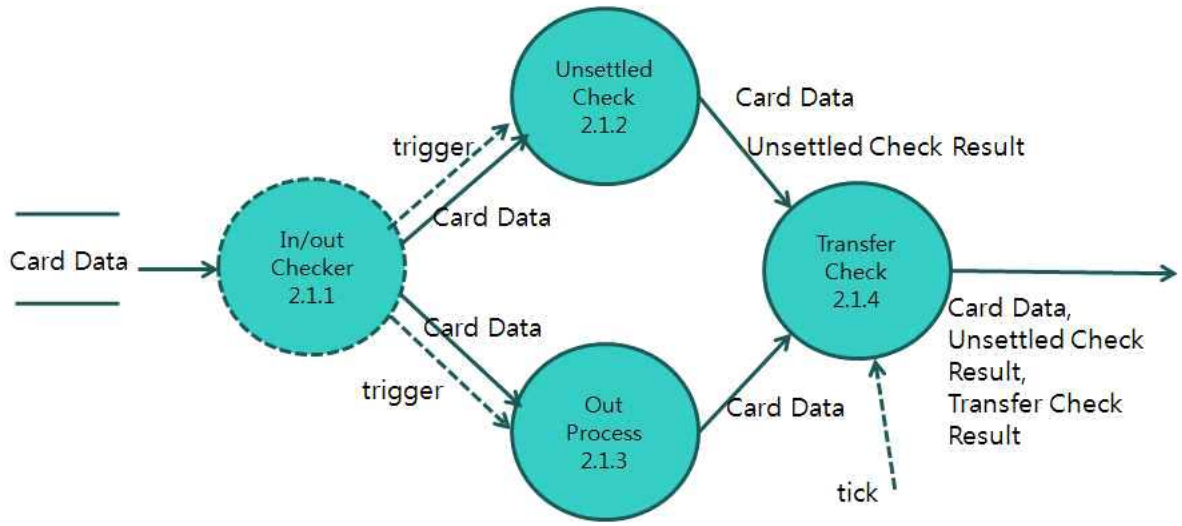
T2.2014.PTS.SDS-1.1

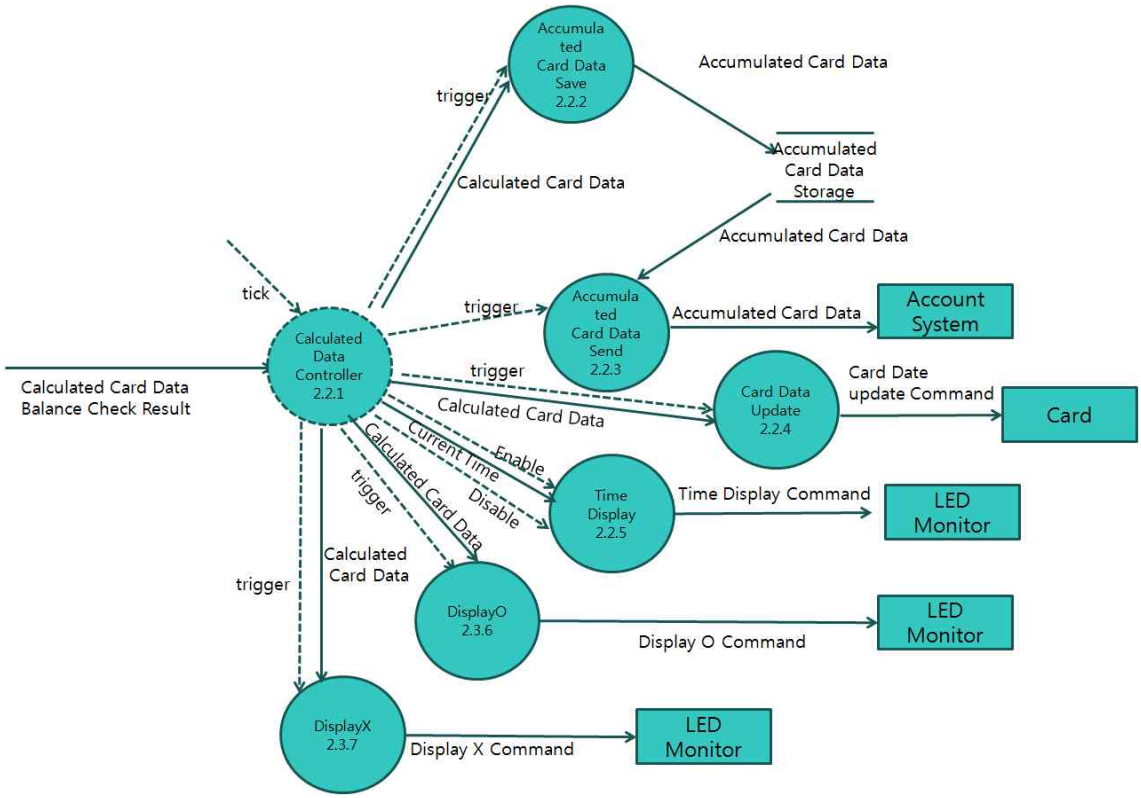
2 Test items

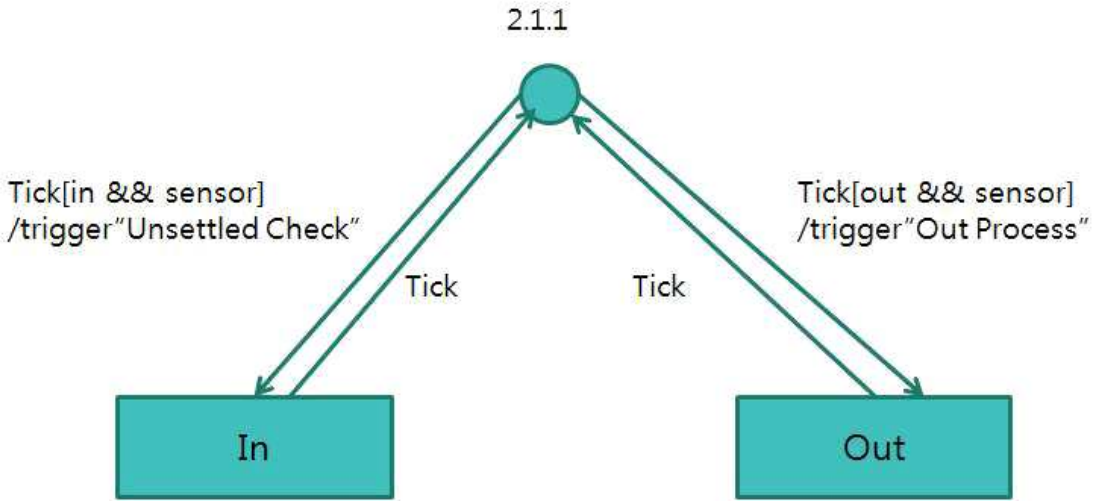
PTS를 구성하는 최소 단위의 모듈들이 unit test의 대상이 된다. 각 모듈의 요구사항을 만족하는지를 test하며, test item은 다음 자료들로부터 작성되었다.

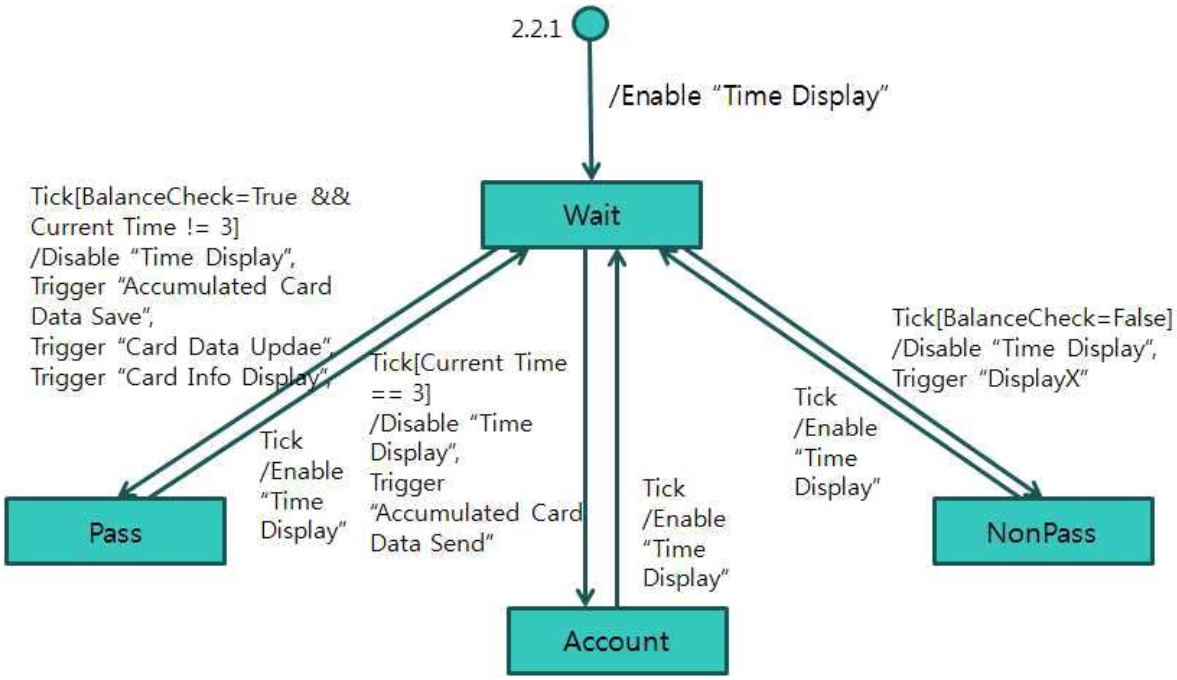
(1) Functionality of modules - T2.2014.PTS.SRA-3.0 : Process specification

아래 그림은 일부를 참조 한 것이다.




Accumulated Card Data Save 2.2.2
Accumulated Card Data
Accumulated Card Data Storage
trigger
Calculated Card Data
Accumulated Card Data
tick
Calculated Data Controller 2.2.1
trigger
Accumulated Card Data Send 2.2.3
Accumulated Card Data
Account System
Calculated Card Data Balance Check Result
trigger
Card Data Update 2.2.4
Card Date update Command
Card
Calculated Card Data
Current Time
Enable
Disable
Time Display 2.2.5
Time Display Command
LED Monitor
trigger
Calculated Card Data
Calculated Card Data
trigger
DisplayO 2.3.6
LED Monitor
Display O Command
DisplayX 2.3.7
Display X Command
LED Monitor


2.1.1
Tick[in && sensor]
/trigger"Unsettled Check"
Tick[out && sensor]
/trigger"Out Process"
Tick
Tick
In
Out



Reference No.	2.1.1
Name	In/Out Check Controller
Input	Card Data
Output	Card Data, Trigger
Process Description	Card Data를 받아서 내가 지금 버스나 지하철에 IN하고 있는지 OUT을 하고 있는지를 판단해서 만약 IN을 하고 있다면 Unsettled Check쪽으로 보내주고 OUT을 하고 있다면 OUT Process쪽으로 보내준다.

Reference No.	2.2.1
Name	Calculated Data Controller
Input	Calculated Card Data, Time Data, Balance Check Result
Output	Calculated Card Data, Time Data, Trigger, Enable, Disable
Process Description	계산된 카드 정보와 시간정보 그리고 잔액확인결과를 확인하여 누적정보를 저장하거나 누적정보를 보내고 LED Monitor 화면에 카드정보를 출력하고 카드의 정보를 업데이트 한다.

(2) Module interface - T2.2014.PTS.SRA-3.0 : Process specification, structure chart

3 Features to be tested

(1) Process in SRA : 각 프로세스가 가지고 있는 요구사항을 만족하는지 test한다.

(2) Module in SDS : 각 모듈이 가지고 있는 데이터 인터페이스를 test한다. <Table 1 테스트 할 Process(DFD) 리스트>의 Process name 참조

<Table 1 테스트 할 Process(DFD) 리스트 - 단말기>

ID	Name	Description
2.1.1	In/Out Checker	Card Data를 받아서 내가 지금 버스나 지하철에 IN하고 있는지 OUT을 하고 있는지를 판단해서 만약 IN을 하고 있다면 Unsettled Check쪽으로 보내주고 OUT을 하고 있다면 OUT Process쪽으로 보내준다.
2.1.2	Unsettled Check	In일 상태의 Calculated Card Data를 받아서 탑승자가 미정산 요금을 갖고 있는지 판단해서 데이터를 수정해서 다음으로 데이터를 넘겨준다.
2.1.3	Out Process	Out인 상태의 Card Data를 받아서 Transfer Check로 Card Data를 넘겨준다.
2.1.4	Transfer Check	In 일 때는 Calculated Card Data에 있는 Time Data를 이용해서 카드를 찍고 내렸을 때 들어있는 시간과 현재 시간을 비교해서 환승이 가능한지 불가능한지를 판단해

		준다. Out 일 때는 지금 내리는 것 전에 환승을 했는지 안했는지를 판단해서 추가요금이 어떻게 결정될지를 정해준다.
2.1.5	Fare Calculate	여태까지 변경 돼서온 Calculated Card Data와 Time Data를 이용해서 버스/지하철 탑승요금이나 하차 시 추가요금을 계산해준다.
2.1.6	Balance Check	Calculated Card Data속에 들어있는 요금 정보와 카드의 잔액정보를 비교해서 잔액이 많으면 True값 적으면 False 값을 넘겨준다.
2.1.7	Minus Fare	Calculated Card Data 속에 True정보가 들어있으면 잔액에서 요금을 빼주고 False 값이 들어있으면 잔액을 바꾸지 않는다.
2.2.1	Data Controller	계산된 카드 정보와 시간정보 그리고 잔액확인결과를 확인하여 누적정보를 저장하거나 누적정보를 보내고 LED Monitor 화면에 카드정보를 출력하고 카드의 정보를 업데이트 한다.
2.2.2	Accumulated Data Save	잔액확인결과가 true일 때 계산된 카드 정보를 누적하여 accumulated card data storage에 저장한다.
2.2.3	Account System Trigger	현재 시간이 정산시간이 되면 잔액확인여부와 상관없이 지금까지 저장했던 accumulated card data storage의 데이터를 가지고 정산기로 accumulated data를 보낸다.
2.2.4	Card Update	잔액확인여부가 true일 때 계산된 카드 정보를 받아와 카드의 정보를 업데이트 하라는 명령을 보낸다.
2.2.5	Time Display	현재 시간 정보를 받아서 LED Monitor에 시간을 계속 출력한다.
2.2.6	Display O	잔액확인여부가 true일 때 계산된 카드 정보를 가지고

		LED Monitor에 카드 정보를 출력할 수 있도록 신호를 보낸다.
2.2.7	Display X	잔액확인여부가 false일 때 탑승이 가능하지 않으므로 LED Monitor에 X표가 뜨도록 신호를 보낸다.

<Table 1 테스트 할 Process(DFD) 리스트 - 정산기>

ID	Name	Description
2.1	Account	Accumulated Data를 받아서 계산식에 맞게 정보를 계산해서 정산을해서 Account Data로 만들어준다.
2.5	Finish Signal Send	2.1에서 정산이 완료가 되면 그 신호를 받아서 Bus / Metro 단말기로 정산이 완료되었다는 신호를 전해줘서 운행이 다시 시작하게 해준다.

4 Features not to be tested

(1) Process in SRA : 외부 장치 드라이버, 단순 데이터 전달 프로세스 등은 test에서 제외한다.

(2) Modules in SDS : <Table 2 테스트하지 않을 Process(DFD) 리스트>의 Process name 참조

<Table 2 테스트하지 않을 Process(DFD) 리스트 - 정산기>

ID	Name	Description
2.2	Account Data Save	Account Data를 받아서 그 정보를 저장하게 하는 Trigger를 보내줘서 그 정보를 날짜별로 구별해서 누적해서 저장하게 해둔다.
2.3	Account Data Display	Account Data를 받아서 그것을 LED Monitor에 Trigger를 줘서 LED Monitor에 정산 결과를 출력하게 해준다.
2.4	Bus Metro Send	2.1에서 정산이 완료된 Account Data를 받아서 거기서 계산된 Bus의 금액과 Metro의 금액을 각각의 회사로 지급해준다.

5 Approach

Public Transportation의 Program source code 및 unit test를 위한 test code는 Cygwin 환경에서 이루어지며, program code/test code의 변경 및 수정 사항은 지속적으로 통합되고 test 된다.

(1) Brute force testing : 각 모듈의 요구사항을 만족하는지를 확인할 수 있는 test case를 작성한다. 그 이외의 예외사항에 대해서는 test하지 않는다.

6 Item pass/fail criteria

Functional test pass/fail criteria : 각 모듈은 요구사항을 모두 만족하여야 한다.

7 Unit test design specification

7.1 Test design specification identifier

PTS.UTD.000.000

7.2 Features to be tested

7.2.1 Process in SRA

<Table 1 테스트할 Process(DFD) 리스트> 참조

7.3 Approach refinements

7.3.1 Brute force testing

PTS의 각 모듈이 요구사항을 만족하는지를 확인하기 위하여, 요구사항에 정의된 내용에 기반하여 test case를 작성한다. 그 이외의 예외 상항에 대해서는 test case를 작성하지 않는다.

7.4 Test identification

<Table 3 Test Design Identification>

단말기 Test Identification

Identifier	Feature(Process ID in DFD)	valid/Invalid value
PTS_UTC_000_000	2.1.1 IN/OutCheck	카드 정보와 단말기누적태그정보가 tag==1과 동시에 들어온다.

PTS_UTC_000_001	2.1.1 IN/OutCheck	카드 정보와 단말기누적태그정보가 tag==2과 동시에 들어온다.
PTS_UTC_000_002	2.1.1 IN/OutCheck	카드 정보와 단말기누적태그정보가 tag==3과 동시에 들어온다.
PTS_UTC_000_003	2.1.1 IN/OutCheck	카드 정보와 단말기누적태그정보가 tag==4과 동시에 들어온다.
PTS_UTC_000_004	2.1.1 IN/OutCheck	카드 정보와 단말기누적태그정보가 tag==5과 동시에 들어온다.
PTS_UTC_000_005	2.1.1 IN/OutCheck	카드 정보와 단말기누적태그정보가 tag==6과 동시에 들어온다.
PTS_UTC_000_006	2.1.1 IN/OutCheck	카드 정보와 단말기누적태그정보가 tag==7과 동시에 들어온다.
PTS_UTC_000_007	2.1.1 IN/OutCheck	카드 정보와 단말기누적태그정보가 tag==8과 동시에 들어온다.
PTS_UTC_000_008	2.1.1 IN/OutCheck	카드 정보와 단말기누적태그정보가 tag==9과 동시에 들어온다.
PTS_UTC_000_009	2.1.1 IN/OutCheck	카드 정보와 단말기누적태그정보가 tag==10과 동시에 들어온다.
PTS_UTC_001_000	2.1.2 UnsettledCheck	바로 전의 탑승정보가 in인 상태로 카드정보가 들어온다.
PTS_UTC_001_001	2.1.2 UnsettledCheck	바로 전의 탑승정보가 out인 상태로 카드정보가 들어온다.
PTS_UTC_002_000	2.1.3 OutProcess	바로 전의 탑승 정보가 in인 상태로 카드정보가 들어온다.
PTS_UTC_002_001	2.1.3 OutProcess	바로 전의 탑승 정보가 out인 상태로 카드정보

		가 들어온다.
PTS_UTC_003_000	2.1.4 TransferCheck	unsettledCheckResult가 0이고, 바로 전의 탑승수단이 버스인 상태로 카드정보가 들어온다.
PTS_UTC_003_001	2.1.4 TransferCheck	unsettledCheckResult가 0이고, 바로 전전의 탑승수단이 버스이고 전전과 전의 탑승시간 차가 120이하인 상태로 카드정보가 들어온다.
PTS_UTC_003_002	2.1.4 TransferCheck	unsettledCheckResult가 1이고, 바로 전의 탑승수단이 지하철이고 전전의 탑승수단이 버스이고 전전과 전의 탑승시간 차가 120이하인 상태로 카드정보가 들어온다.
PTS_UTC_003_003	2.1.4 TransferCheck	unsettledCheckResult가 1이고, 바로 전의 탑승수단이 버스이고 전전의 탑승수단이 지하철이고 전전과 전의 탑승시간 차가 120이하인 상태로 카드정보가 들어온다.
PTS_UTC_003_004	2.1.4 TransferCheck	unsettledCheckResult가 1이고, 바로 전의 탑승수단이 지하철인 상태로 카드정보가 들어온다.
PTS_UTC_003_005	2.1.4 TransferCheck	unsettledCheckResult가 2이고, 바로 전의 탑승수단이 버스이고 바로 전과 현재 탑승시간의 차가 120이하인 상태로 카드정보가 들어온다.
PTS_UTC_003_006	2.1.4 TransferCheck (Bus)	unsettledCheckResult가 1이고, 바로 전의 탑승수단이 지하철이고 전전의 탑승수단이 버스인 상태로 카드정보가 들어온다.
PTS_UTC_003_007	2.1.4 TransferCheck (Bus)	unsettledCheckResult가 2이고, 바로 전의 탑승수단이 지하철이고 바로 전과 현재 탑승시간의 차가 120이하인 상태로 카드정보가 들어온다.
PTS_UTC_004_000	2.1.5 FareCalculator	unsettledCheckResult가 0이고, transferCheckResult가 0이고 현재 단말기 ID와

		바로 전의 단말기 ID의 차의 절댓값이 2이상인 상태로 카드정보가 들어온다.
PTS_UTC_004_001	2.1.5 FareCalculator	unsettledCheckResult가 0이고, transferCheckResult가 1이고 현재 단말기 ID와 바로 전의 단말기 ID의 차의 절댓값*300이 600 미만인 상태로 카드정보가 들어온다.
PTS_UTC_004_002	2.1.5 FareCalculator	unsettledCheckResult가 0이고, transferCheckResult가 1이고 현재 단말기 ID와 바로 전의 단말기 ID의 차의 절댓값*300이 600 이상인 상태로 카드정보가 들어온다.
PTS_UTC_004_003	2.1.5 FareCalculator	unsettledCheckResult가 1이고, transferCheckResult가 1인 상태로 카드정보가 들어온다.
PTS_UTC_004_004	2.1.5 FareCalculator	unsettledCheckResult가 1이고, transferCheckResult가 2인 상태로 카드정보가 들어온다.
PTS_UTC_004_005	2.1.5 FareCalculator	unsettledCheckResult가 1이고, transferCheckResult가 3인 상태로 카드정보가 들어온다.
PTS_UTC_004_006	2.1.5 FareCalculator	unsettledCheckResult가 1이고, transferCheckResult가 1,2,3이 아닌 상태로 카드정보가 들어온다.
PTS_UTC_004_007	2.1.5 FareCalculator	unsettledCheckResult가 2이고, transferCheckResult가 1인 상태로 카드정보가 들어온다.
PTS_UTC_004_008	2.1.5 FareCalculator	unsettledCheckResult가 2이고, transferCheckResult가 1이 아닌 상태로 카드정보가 들어온다.

PTS_UTC_004_009	2.1.5 FareCalculator (Bus)	unsettledCheckResult가 0이고, transferCheckResult가 1이고 (timeCalculate(dpp,2)/30)*100>700인 상태로 카드정보가 들어온다
PTS_UTC_004_010	2.1.5 FareCalculator (Bus)	unsettledCheckResult가 0이고, transferCheckResult가 1이고 (timeCalculate(dpp,2)/30)*100<700인 상태로 카드정보가 들어온다
PTS_UTC_005_000	2.1.6 BalanceCheck	unsettledCheckResult가 1이고, transferCheckResult가 1이고 카드의 잔액이 1650이상인 상태로 카드정보가 들어온다.
PTS_UTC_005_001	2.1.6 BalanceCheck	unsettledCheckResult가 1이고, transferCheckResult가 1이고 카드의 잔액이 1650인 상태로 카드정보가 들어온다.
PTS_UTC_005_002	2.1.6 BalanceCheck	unsettledCheckResult가 1이고, transferCheckResult가 2이고 카드의 잔액이 1750이상인 상태로 카드정보가 들어온다.
PTS_UTC_005_003	2.1.6 BalanceCheck	unsettledCheckResult가 1이고, transferCheckResult가 2이고 카드의 잔액이 1750미만인 상태로 카드정보가 들어온다.
PTS_UTC_005_004	2.1.6 BalanceCheck	unsettledCheckResult가 1이고, transferCheckResult가 3이고 카드의 잔액이 1250이상인 상태로 카드정보가 들어온다.
PTS_UTC_005_005	2.1.6 BalanceCheck	unsettledCheckResult가 1이고, transferCheckResult가 3이고 카드의 잔액이 1250미만인 상태로 카드정보가 들어온다.
PTS_UTC_005_006	2.1.6 BalanceCheck	unsettledCheckResult가 1이고, transferCheckResult가 1,2,3이 아니고 카드의

		잔액이 1050이상인 상태로 카드정보가 들어온다.
PTS_UTC_005_007	2.1.6 BalanceCheck	unsettledCheckResult가 1이고, transferCheckResult가 1,2,3이 아니고 카드의 잔액이 1050미만인 상태로 카드정보가 들어온다.
PTS_UTC_005_008	2.1.6 BalanceCheck	unsettledCheckResult가 2이고, transferCheckResult가 1이고 카드의 잔액이 600이상인 상태로 카드정보가 들어온다.
PTS_UTC_005_009	2.1.6 BalanceCheck	unsettledCheckResult가 2이고, transferCheckResult가 1이고 카드의 잔액이 600미만인 상태로 카드정보가 들어온다.
PTS_UTC_005_010	2.1.6 BalanceCheck	unsettledCheckResult가 1과 2가 아니고, 카드의 잔액이 1050이상인 상태로 카드정보가 들어온다.
PTS_UTC_005_011	2.1.6 BalanceCheck	unsettledCheckResult가 1과 2가 아니고, 카드의 잔액이 1050미만인 상태로 카드정보가 들어온다.
PTS_UTC_006_000	2.1.7 MinusFare	balanceResult가 1인 상태로 카드정보가 들어온다.
PTS_UTC_006_001	2.1.7 MinusFare	balanceResult가 1이 아닌 상태로 카드정보가 들어온다.
PTS_UTC_007_000	2.2.1 DataController	현재 탑승정보가 in이거나 현재 탑승정보가 out이고 현재 탑승날짜와 전의 탑승날짜가 같은 상태로 카드정보가 들어온다.
PTS_UTC_008_000	2.2.2 AccumulatedDataSave	계산된 카드정보가 들어온다.

PTS_UTC_009_000	2.2.3 AccumulatedDataSend	이전 시간이 초기값이 아니고 현재날짜와 이전 날짜가 다른 정보가 들어온다.
PTS_UTC_010_000	2.2.4 CardUpdate	계산된 카드정보와 계산된 단말기누적태그정보가 들어온다.
PTS_UTC_011_000	2.2.6 DisplayO	balanceResult가 1인 상태로 카드정보가 들어온다.
PTS_UTC_012_000	2.2.6 DisplayX	balanceResult가 1이 아닌 상태로 카드정보가 들어온다.

정산기 Test Identification

PTS_UTC_013_000	2 AccountSystem	트리거가 1인 상태로 들어온다.
PTS_UTC_013_001	2 AccountSystem	트리거가 1이 아닌 상태로 들어온다.
PTS_UTC_014_000	2.1 Account	탑승수단이 버스이고 탑승요금이 100이상 700이하인 상태로 카드정보가 들어온다.
PTS_UTC_014_001	2.1 Account	탑승수단이 버스이고 탑승요금이 1650인 상태로 카드정보가 들어온다.
PTS_UTC_014_002	2.1 Account	탑승수단이 버스이고 탑승요금이 1750인 상태로 카드정보가 들어온다.
PTS_UTC_014_003	2.1 Account	탑승수단이 버스이고 탑승요금이 1250인 상태로 카드정보가 들어온다.
PTS_UTC_014_004	2.1 Account	탑승수단이 버스이고 탑승요금이 1050인 상태로 카드정보가 들어온다.
PTS_UTC_014_005	2.1 Account	탑승수단이 지하철이고 탑승요금이 200인 상태로 카드정보가 들어온다.
PTS_UTC_014_006	2.1 Account	탑승수단이 지하철이고 탑승요금이 300인 상태

		로 카드정보가 들어온다.
PTS_UTC_014_007	2.1 Account	탑승수단이 지하철이고 탑승요금이 600인 상태로 카드정보가 들어온다.
PTS_UTC_014_0.08	2.1 Account	탑승수단이 지하철이고 탑승요금이 1650인 상태로 카드정보가 들어온다.
PTS_UTC_014_009	2.1 Account	탑승수단이 지하철이고 탑승요금이 1750인 상태로 카드정보가 들어온다.
PTS_UTC_014_010	2.1 Account	탑승수단이 지하철이고 탑승요금이 1250인 상태로 카드정보가 들어온다.
PTS_UTC_014_011	2.1 Account	탑승수단이 지하철이고 탑승요금이 1050인 상태로 카드정보가 들어온다.
PTS_UTC_015_000	2.6 finishSignal	trigger가 1인 상태로 들어온다.

7.5 Feature pass/fail criteria

PTS의 각 모듈(프로세스)은 SRA에 정의되어 있는 요구사항 (입력 / 출력 및 동작)을 모두 만족해야 한다. 각 모듈(프로세스)의 입력 / 출력 및 동작은 SRA의 process description 항목을 참조한다.

8 Unit test case specification

8.1 Test case specification identifier

<Table 4 Test Case Identification>

TestCase Identifier	Input specification	Output specification
PTS_UTC_000_000	CARD *dpp / count[] / tag==1	dpp[0].ID=1 / count[0]++ / dpp[0].means=Metro (Metro) dpp[0].ID=6 / count[5]++ / dpp[0].means=Bus(Bus)

PTS_UTC_000_001	CARD *dpp / count[] / tag==2	dpp[0].ID=1 / count[0]++ / dpp[0].means=Metro (Metro) dpp[0].ID=6 / count[5]++ / dpp[0].means=Bus(Bus)
PTS_UTC_000_002	CARD *dpp / count[] / tag==3	dpp[0].ID=2 / count[1]++ / dpp[0].means=Metro (Metro)
PTS_UTC_000_003	CARD *dpp / count[] / tag==4	dpp[0].ID=2 / count[1]++ / dpp[0].means=Metro (Metro)
PTS_UTC_000_004	CARD *dpp / count[] / tag==5	dpp[0].ID=3 / count[2]++ / dpp[0].means=Metro (Metro)
PTS_UTC_000_005	CARD *dpp / count[] / tag==6	dpp[0].ID=3 / count[2]++ / dpp[0].means=Metro (Metro)
PTS_UTC_000_006	CARD *dpp / count[] / tag==7	dpp[0].ID=4 / count[3]++ / dpp[0].means=Metro (Metro)
PTS_UTC_000_007	CARD *dpp / count[] / tag==8	dpp[0].ID=4 / count[3]++ / dpp[0].means=Metro (Metro)
PTS_UTC_000_008	CARD *dpp / count[] / tag==9	dpp[0].ID=5 / count[4]++ / dpp[0].means=Metro (Metro)
PTS_UTC_000_009	CARD *dpp / count[] / tag==10	dpp[0].ID=5 / count[4]++ / dpp[0].means=Metro (Metro)
PTS_UTC_001_000	dpp[1].inout==in	unsettledCheckResult=1 / dpp[0].inout=in
PTS_UTC_001_001	dpp[1].inout==out	unsettledCheckResult=2 / dpp[0].inout=in
PTS_UTC_002_000	dpp[1].inout==in	dpp[0].inout=out
PTS_UTC_002_001	dpp[1].inout==iout	dpp[0].inout=out

PTS_UTC_003_000	usc==0 && dpp[1].means==Bus	transferCheckResult=2
PTS_UTC_003_001	usc==0 && dpp[2].means==Bus && timeCalculate(dpp, 1)<=120	transferCheckResult=1
PTS_UTC_003_002	usc==1 && dpp[1].means==Metro && dpp[2].means==Bus && timeCalculate(dpp, 1)<=120	transferCheckResult=1
PTS_UTC_003_003	usc==1 && dpp[1].means==Bus && dpp[2].means==Metro && timeCalculate(dpp, 1)<=120	transferCheckResult=2
PTS_UTC_003_004	usc==1 && dpp[1].means==Metro	transferCheckResult=3
PTS_UTC_003_005	usc==2 && dpp[1].means==Bus && timeCalculate(dpp, 2)<=120	transferCheckResult=1
PTS_UTC_003_006	usc==1 && tcr==1 && dpp[1].means==Metro && dpp[2].means==Bus (Bus)	transferCheckResult=3
PTS_UTC_003_007	usc==2 && dpp[1].means==Metro && timeCalculate(dpp, 2)<=120 (Bus)	transferCheckResult=1
PTS_UTC_004_000	ucr==0 && tcr==0 && |dpp[0].ID-dpp[1].ID|>=2	fare=200

PTS_UTC_004_001	ucr==0 && tcr==1 && |dpp[0].ID-dpp[1].ID|*300<600	fare=abs(atoi(dpp[0].ID)-atoi(dpp[1].ID))*300
PTS_UTC_004_002	ucr==0 && tcr==1 && |dpp[0].ID-dpp[1].ID|*300>600	fare=600
PTS_UTC_004_003	ucr==1 && tcr==1	fare=1650
PTS_UTC_004_004	ucr==1 && tcr==2	fare=1750
PTS_UTC_004_005	ucr==1 && tcr==3	fare=1250
PTS_UTC_004_006	ucr==1 && (tcr!=1 && tcr!=2 && tcr!=3)	fare=1050
PTS_UTC_004_007	ucr==2 && tcr==1	fare=0
PTS_UTC_004_008	ucr==2 && tcr!=1	fare=1050
PTS_UTC_004_009	ucr==0 && tcr==1 && (timeCalculate(dpp,2)/30)*100>700 (Bus)	fare=700
PTS_UTC_004_010	ucr==0 && tcr==1 && (timeCalculate(dpp,2)/30)*100<=700 (Bus)	fare=(timeCalculate(dpp,2)/30)*100
PTS_UTC_005_000	ucr==1 && tcr==1 && dpp[1].money<1650	balanceCheckResult=0
PTS_UTC_005_001	ucr==1 && tcr==1 && dpp[1].money>=1650	balanceCheckResult=1
PTS_UTC_005_002	ucr==1 && tcr==2 && dpp[1].money<1750	balanceCheckResult=0

PTS_UTC_005_003	ucr==1 && tcr==2 && dpp[1].money>=1750	balanceCheckResult=1
PTS_UTC_005_004	ucr==1 && tcr==3 && dpp[1].money<1250	balanceCheckResult=0
PTS_UTC_005_005	ucr==1 && tcr==3 && dpp[1].money>=1250	balanceCheckResult=1
PTS_UTC_005_006	ucr==1 && tcr!=1 && tcr!=2 && tcr!=3 &&dpp[1].money<1050	balanceCheckResult=0
PTS_UTC_005_007	ucr==1 && tcr!=1 && tcr!=2 && tcr!=3 &&dpp[1].money>=1050	balanceCheckResult=1
PTS_UTC_005_008	ucr==2 && tcr==1 && dpp[1].money<600	balanceCheckResult=0
PTS_UTC_005_009	ucr==2 && tcr==1 && dpp[1].money>=600	balanceCheckResult=1
PTS_UTC_005_010	ucr!=1 && ucr!=2 && dpp[1].money<1050	balanceCheckResult=0
PTS_UTC_005_011	ucr!=1 && ucr!=2 && dpp[1].money>=1050	balanceCheckResult=1
PTS_UTC_006_000	bcr==1 / fare	dpp[0].money=dpp[1].money - fare
PTS_UTC_006_001	bcr!=1 / fare	LED Command
PTS_UTC_007_000	dpp[0].inout==in || (dpp[0].inout==out && dpp[0].time[6]==dpp[1].time[6] &&	DataControl Command

	dpp[0].time[7]==dpp[1].time[7])	
PTS_UTC_008_000	CARD *dpp	Accumulated.txt dpp저장
PTS_UTC_009_000	olddate!=00000000 && (date[6]!=olddate[6] || date[7]!=oddate[7])	AccountTrigger.txt 1저장
PTS_UTC_010_000	CARD *dpp / int count[]	card.txt dpp저장 count.txt count저장
PTS_UTC_011_000	bcr==1 / CARD *dpp	LED Command
PTS_UTC_012_000	bcr!=1 / CARD *dpp	LED Command
PTS_UTC_013_000	trigger==1	Account Command
PTS_UTC_013_001	trgger!=1	LED Command
PTS_UTC_014_000	dpp[i].means==Bus && dpp[i].fare>=100 && dpp[i].fare<=700 / int* BusMoney / int* MetroMoney	BusMoney += dpp[i].fare
PTS_UTC_014_001	dpp[i].means==Bus && dpp[i].fare==1650/ int* BusMoney / int* MetroMoney	BusMoney += (1050+218) / MetroMoney += 382
PTS_UTC_014_002	dpp[i].means==Bus && dpp[i].fare==1750/ int* BusMoney / int* MetroMoney	BusMoney += (1050+420) / MetroMoney += 280
PTS_UTC_014_003	dpp[i].means==Bus && dpp[i].fare==1250/ int*	BusMoney += 1050 / MetroMoney += 200

	BusMoney / int* MetroMoney	
PTS_UTC_014_004	dpp[i].means==Bus && dpp[i].fare==1050/ int* BusMoney / int* MetroMoney	BusMoney += 1050
PTS_UTC_014_005	dpp[i].means==Metro && dpp[i].fare==200/ int* BusMoney / int* MetroMoney	MetroMoney += dpp[i].fare
PTS_UTC_014_006	dpp[i].means==Metro && dpp[i].fare==300/ int* BusMoney / int* MetroMoney	BusMoney -= 459 / MetroMoney += 759
PTS_UTC_014_007	dpp[i].means==Metro && dpp[i].fare==600/ int* BusMoney / int* MetroMoney	MetroMoney += 1008 / BusMoney -= 408
PTS_UTC_014_008	dpp[i].means==Metro && dpp[i].fare==1650/ int* BusMoney / int* MetroMoney	MetroMoney += (1050+218) / BusMoney += 382
PTS_UTC_014_009	dpp[i].means==Metro && dpp[i].fare==1750/ int* BusMoney / int* MetroMoney	MetroMoney += (1050+420) / BusMoney += 280
PTS_UTC_014_010	dpp[i].means==Metro && dpp[i].fare==1250/ int* BusMoney / int*	MetroMoney += 1050 / BusMoney += 200

	MetroMoney	
PTS_UTC_014_011	dpp[i].means==Metro && dpp[i].fare==1050/ int* BusMoney / int* MetroMoney	MetroMoney += 1050
PTS_UTC_015_000	trigger==1	finsihSignal Command

8.2 Test items

<Table 3 Test Design Identification> 참조

8.3 Input specifications

<Table 4 Test Case Identification> 참조

8.4 Output specifications

<Table 4 Test Case Identification> 참조

9 Testing tasks

<Table 5 Testing tasks & Schedule>

Task	Predecessor task	Special skills	Effort	Finish date
(1) Unit Test Plan 작성	T2.2014.PTS.SRS작성 T2.2014.PTS.SRA작성 T2.2014.PTS.SDA작성 PTS구현		4	
(2) Test design specification	Task1	PTS에 대한 이해	5	
(3) Test case specification	Task2	PTS에 대한 이해	5	

(4) Test Execution	Task3	Test code 작성 Test tools에 대한 이해	4	
(5) Test result report	Task4		1	

10 Environmental needs

PTS의 unit test를 위한 환경적 요구사항은 다음과 같다.

(1) Hardware & Platform

GCC compiler/linker

(2) CTIP(Continuous Testing & Integrated Platform) Environment

Cygwin

11 Unit Test deliverables

11.1 Unit test plan

11.2 Unit test design specification

11.3 Unit test case specification

11.4 Unit test summary report

12 Schedules

<Table 5 Testing tasks & Schedule> 참조