Unit Testing Plan

for Point Of Sale System

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

Project Team

Team 1

Date

2017-11-03

Team Information
201211337 김재현
201112052 방민석
201312259 백만일
201211383 조영래

Table of Contents

1 Introduction

1.1 Objectives

본 문서는 2017 학년도 2학기 소프트웨어 공학 개론 수업의 Team1이 개발한 POS system(이하 POS) Unit Test 하기 위한 계획 문서이다. Team 1이 정의한 Unit Testing 을 수행하기 위하여 Testing Pass/Fail Criteria를 정의하고 이를 수행하기 위한 Test Design & Test Cases 를 제작한다.

1.2 Background

POS system 은 사용자의 요구에 따라 판매와 관련된 데이터를 일관적으로 관리하고, 고객정보를 수집하여 부가가치를 향상시키는 시스템이다.
Unit Test는 시스템을 구성하는 최소 단위 모듈들을 대상으로 하는 Test이며, 시스템의 성 능을 좌우하는 요소들이 요구사항을 만족하는지를 확인할 수 있는 기본적인 Test approach이다

1.3 Scope

POS system 대한 Unit Test를 수행하기 위한 자원과 절차, Test approach와 Technique와 필요로 하는 환경 및 도구 등을 정의한다. Unit Test는 시스템 핵심 기능 관련 프로세스에 중점을 두며 전달 역할 등 단순 프로세스는 Test에서 제외한다.

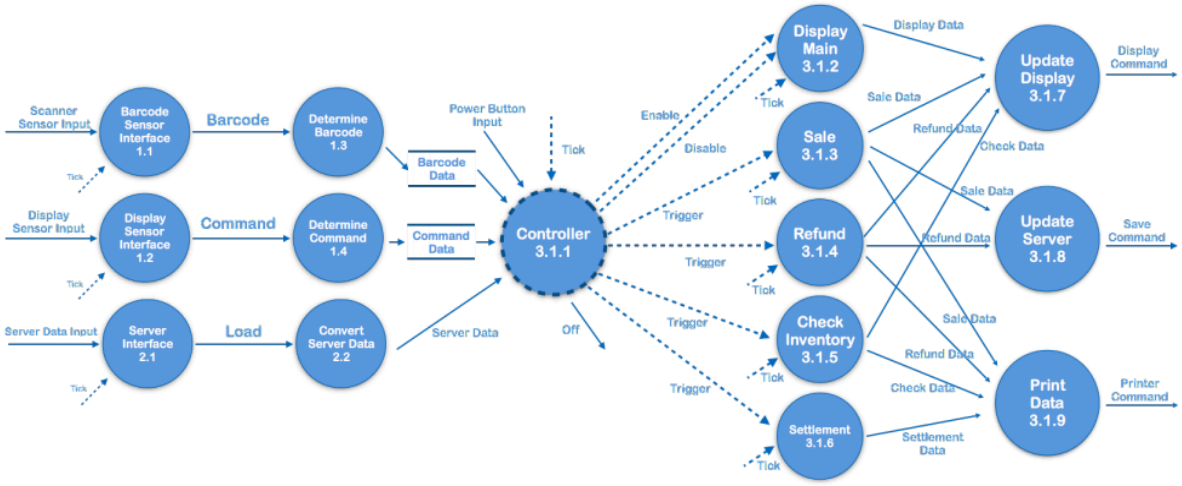
1.4 Project plan

1.5 Configuration management plan

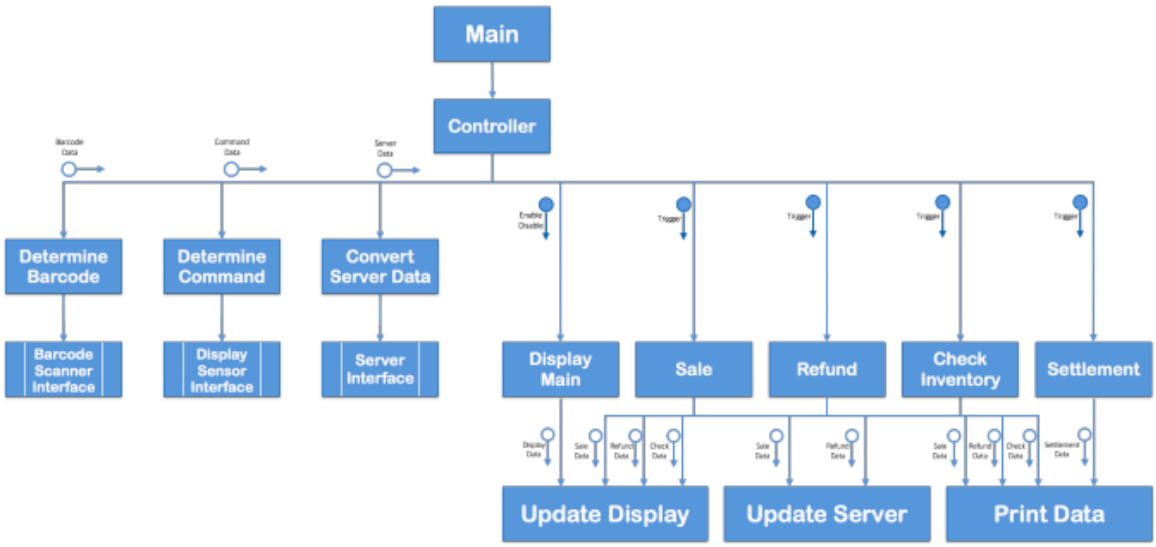
1.6 References

2 Test items

Team1 이 SASD 기법을 이용하여 개발한 POS 를 Testing한다. SA에서 최소단위의 각 Process별로 요구사항을 만족하는지, 정상적인 결과가 나오는지, 잘못된 값 입력 시 예외 처리가 동작하는지 Testing을 수행한다. <Figure 1 Overall DFD>는 SA를 이용하여 요구사항을 분석 한 결과를 DFD를 이용해 나타낸 그림이고, <Figure 2 Structural Chart>는 SD의 Basic Structural Chart를 나타낸 그림이다. 각 그림을 참조하여 Unit을 지정하고, 지정한 Unit을 SRA에 명세 된 내용과 같은 동작을 하는지 확인한다.



<Figure 1 Overall DFD>



<Figure 2 Structural Chart>

3 Features to be tested

시스템 핵심 기능 관련 모듈 중 입력 담당 모듈 부분과 처리 담당 모듈 부분을 거쳐 요구사항 만족, 정상적인 동작과 잘못된 값 입력시 동작에 중점을 두어 테스트 한다.

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

2) Modules in SDS : 각 모듈이 가지고 있는 데이터 인터페이스를 테스트 한다.

<Table1 Testing Process(DFD) 리스트>

ID	Name	Description
1.3	Determine Barcode	입력받은 데이터를 Barcode Data형태로 변환한다.
1.4	Determine Command	디스플레이를 통해 입력 받은 데이터를 Command Data형태로 변환한다.
2.2	Convert Server Data	서버로부터 받은 데이터를 Server Data형태로 변환한다.
3.1.1	Controller	입력된 Data를 Command Data를 바탕으로 프로세스들을 트리거 한다.
3.1.2	Display Main	입력 데이터를 Display Data형태로 만든다.
3.1.3	Sale	판매 될 수량을 서버와 비교해서 판매 가능하면 Sale Data형태로 만든다.
3.1.4	Refund	환불 물품을 서버와 비교해서 환불 가능하면 Refund Data형태로 만든다.
3.1.5	Check Inventory	서버에서 재고 목록을 받아온다.
3.1.6	Settlement	서버에서 판매, 환불 영수증을 읽어와 Settlement Data형태로 만든다.
3.1.7	Update Display	계산된 Data를 Display Data형태로 만들어 Display Interface에 전달한다.
3.1.8	Update Server	계산된 Data를 Server Data형태로 만들어 Server Interface에 전달한다
3.1.9	Print Data	계산된 Data를 Printer Interface에 전달한다

4 Features not to be tested

전달 역할, 단순한 프로세스, UI 구성 프로세스, 특수한 상황에서만 작동하는 프로세스는 테스트에서 제외한다.

5 Approach

CMS의 Program source code와 Unit Tests는 Cygwin 환경에서 이루어지며,

Program code의 변경 및 수정사항은 지속적으로 통합되고 테스트 된다.

6 Item pass/fail criteria

각 Unit별 Pass/Fail Criteria는 <Test Case Identification>을 참조한다.

7 Unit test design specification

7.1 Test design specification identifier

POS_[Category]_000

7.2 Features to be tested

<Table 1 Testing Process(DFD) 리스트> 참조

7.3 Approach refinements

각 Process Specification에 명시된 내용을 기반으로 Test Design 및 Test Cases를 생성해 낸다.

7.4 Test identification

<Table2 Test Design Identification>

Identifier	Feature (Process DFD)	Valid / Invalid value
POS_BAR_DATA_01	1.3 Determine Barcode	determine_barcode(LIST, SERVER, COMMAND, barcode, state) : {&list, &server, "b", "001", -1}
POS_BAR_DATA_02	1.3 Determine Barcode	determine_barcode(LIST, SERVER, COMMAND, barcode, state) : {&list, &server, "b", "010", -1}
POS_BAR_DATA_03	1.3 Determine Barcode	determine_barcode(LIST, SERVER, COMMAND, barcode, state) : {&list, &server, "b", "011", -1}
POS_BAR_DATA_04	1.3 Determine Barcode	determine_barcode(LIST, SERVER, COMMAND, barcode, state) : {&list, &server, "b", "100", -1}
POS_BAR_DATA_05	1.3 Determine Barcode	determine_barcode(LIST, SERVER,

		COMMAND, barcode, state) : {&list, &server, "b", "101", -1}
POS_BAR_DATA_06	1.3 Determine Barcode	determine_barcode(LIST, SERVER, COMMAND, barcode, state) : {&list, &server, "b", "110", -1}
POS_BAR_DATA_07	1.3 Determine Barcode	determine_barcode(LIST, SERVER, COMMAND, barcode, state) : {&list, &server, "b", "111", -1}
POS_BAR_DATA_08	1.3 Determine Barcode	determine_barcode(LIST, SERVER, COMMAND, barcode, state) : {&list, &server, "b", "000", -1}
POS_BAR_DATA_09	1.3 Determine Barcode	determine_barcode(LIST, SERVER, COMMAND, barcode, state) : {&list, &server, "r", barcode, -1}
POS_COMMAND_DATA_01	1.4 Determine Command	determine_command(COMMAND, SERVER, LIST, state) : {'b', &server, &list, -1}
POS_COMMAND_DATA_02	1.4 Determine Command	determine_command(COMMAND, SERVER, LIST, state) : {'s', &server, &list, 1}
POS_COMMAND_DATA_03	1.4 Determine Command	determine_command(COMMAND, SERVER, LIST, state) : {'r', &server, &list, 2}
POS_COMMAND_DATA_04	1.4 Determine Command	determine_command(COMMAND, SERVER, LIST, state) : {'C', &server, &list, 3}
POS_COMMAND_DATA_05	1.4 Determine Command	determine_command(COMMAND, SERVER, LIST, state) : {'i', &server, &list, -1}
POS_COMMAND_DATA_06	1.4 Determine Command	determine_command(COMMAND, SERVER, LIST, state) : {'c', &server, &list, -1}
POS_COMMAND_DATA_07	1.4 Determine Command	determine_command(COMMAND, SERVER, LIST, state) : {'e', &server, &list, 0}
POS_COMMAND_DATA_08	1.4 Determine Command	determine_command(COMMAND, SERVER, LIST, state) : {'x', &server, &list, -1}

POS_SERVER_DATA_01	2.2 Convert Server Data	convert_server(SERVER, LIST, receipt_num, type) : {&server, &list, receipt_num,0}
POS_SERVER_DATA_02	2.2 Convert Server Data	convert_server(SERVER, LIST, receipt_num, type) : {&server, &list, receipt_num,1}
POS_SERVER_DATA_03	2.2 Convert Server Data	convert_server(SERVER, LIST, receipt_num, type) : {&server, &list, receipt_num,2}
POS_Controller_01	3.1.1 Controller	selected_state {-1}
POS_Controller_02	3.1.1 Controller	selected_state {'0}
POS_Controller_03	3.1.1 Controller	selected_state {1}
POS_Controller_04	3.1.1 Controller	selected_state {2}
POS_Controller_05	3.1.1 Controller	selected_state {3}
POS_Controller_06	3.1.1 Controller	selected_state {4}
POS_Controller_07	3.1.1 Controller	selected_state {int_max}
POS_Display_Main_01	3.1.2 Display Main	display_main(DISPLAY) : {&display}
POS_SALE_DATA_01	3.1.3 Sale	sale(r_price, LIST, SERVER, SALE, state, DISPLAY) : {r_price, list, server, sale, 1, display}
POS_SALE_DATA_02	3.1.3 Sale	sale(r_price, LIST, SERVER, SALE, state, DISPLAY) : {0, 0, server, sale, 1, display}
POS_SALE_DATA_03	3.1.3 Sale	sale(r_price, LIST, SERVER, SALE, state, DISPLAY) : {r_price, 0, server, sale, 1, display}
POS_SALE_DATA_04	3.1.3 Sale	sale(r_price, LIST, SERVER, SALE, state, DISPLAY) : {r_price, 0, 0, sale, 1, display}
POS_REFUND_DATA_01	3.1.4 Refund	Refund(LIST, SERVER, REFUND, state, DISPLAY) : {&list, &server, ,&refund, 2, display}
POS_CHECK_DATA_01	3.1.5 Check Inventory	Check(SERVER, state, DISPLAY) : {&server, 3 , &display}
POS_SETTLEMENT_01	3.1.6 Settlement	settlement(LIST, SERVER, SETTLEMENT, state) : {&list, &server, &settlement, 4}
POS_SETTLEMENT_02	3.1.6 Settlement	settlement(LIST, SERVER, SETTLEMENT, state) : {0, &server, &settlement, 4}

POS_Update_Display_01	3.1.7 Update Display	update_display(SALE,REFUND, SERVER , DISPLAY) : {1, 0, 0, 0}
POS_Update_Display_02	3.1.7 Update Display	update_display(SALE,REFUND, SERVER , DISPLAY) : {0, 1, 0, 0}
POS_Update_Display_03	3.1.7 Update Display	update_display(SALE,REFUND, SERVER , DISPLAY) : {0, 0, 1, 0}
POS_Update_Server_01	3.1.8 Update Server	update_server(SERVER, LIST) : {&server, &list}
POS_Update_Server_02	3.1.8 Update Server	update_server(SERVER, LIST) : {&server, 0}
POS_PRINT_DATA_01	3.1.9 Print Data	print_data(SALE, REFUND, SERVER, SETTLEMENT) : {&sale, 0, 0, 0}
POS_PRINT_DATA_02	3.1.9 Print Data	print_data(SALE, REFUND, SERVER, SETTLEMENT) : {0, &refund, 0, 0}
POS_PRINT_DATA_03	3.1.9 Print Data	print_data(SALE, REFUND, SERVER, SETTLEMENT) : {0, 0, &server, 0}
POS_PRINT_DATA_04	3.1.9 Print Data	print_data(SALE, REFUND, SERVER, SETTLEMENT) : {0, 0, 0, &settlemenet}

7.5 Feature pass/fail criteria

POS의 각 모듈은 SRA에 정의되어 있는 요구사항을 만족해야 한다. 각 모듈의 입/출력 및 동작은 SRA의 Process description 항목 및 STD를 참조한다.

8 Unit test case specification

8.1 Test case specification identifier

<Table3 Test Case Identification>

Identifier	Input Specification	Output Specification
POS_BAR_DATA_01	determine_barcode(LIST, SERVER, COMMAND, barcode, state) : {&list, &server, "b", "001", -1}	Barcode Data(product)
POS_BAR_DATA_02	determine_barcode(LIST, SERVER, COMMAND, barcode, state) : {&list, &server, "b", "010", -1}	Barcode Data(product)
POS_BAR_DATA_03	determine_barcode(LIST, SERVER,	Barcode Data(product)

	COMMAND, barcode, state) : {&list, &server, "b", "011", -1}	
POS_BAR_DATA_04	determine_barcode(LIST, SERVER, COMMAND, barcode, state) : {&list, &server, "b", "100", -1}	Barcode Data(product)
POS_BAR_DATA_05	determine_barcode(LIST, SERVER, COMMAND, barcode, state) : {&list, &server, "b", "101", -1}	Barcode Data(product)
POS_BAR_DATA_06	determine_barcode(LIST, SERVER, COMMAND, barcode, state) : {&list, &server, "b", "110", -1}	Barcode Data(product)
POS_BAR_DATA_07	determine_barcode(LIST, SERVER, COMMAND, barcode, state) : {&list, &server, "b", "111", -1}	Barcode Data(product)
POS_BAR_DATA_08	determine_barcode(LIST, SERVER, COMMAND, barcode, state) : {&list, &server, "b", "000", -1}	Error(Exception Handling)
POS_BAR_DATA_09	determine_barcode(LIST, SERVER, COMMAND, barcode, state) : {&list, &server, "r", barcode, -1}	Barcode Data(receipt)
POS_COMMAND_DATA_01	determine_command(COMMAND, SERVER, LIST, state) : {'b', &server, &list, -1}	Command = Input Barcode
POS_COMMAND_DATA_02	determine_command(COMMAND, SERVER, LIST, state) : {'s', &server, &list, 1}	Command = Sale
POS_COMMAND_DATA_03	determine_command(COMMAND, SERVER, LIST, state) : {'r', &server, &list, 2}	Command = Refund
POS_COMMAND_DATA_04	determine_command(COMMAND, SERVER, LIST, state) : {'C', &server, &list, 3}	Command = Check
POS_COMMAND_DATA_05	determine_command(COMMAND, SERVER, LIST, state) : {'i', &server, &list, -1}	Command = Input Cash
POS_COMMAND_DATA_06	determine_command(COMMAND, SERVER, LIST, state) : {'c', &server, &list, -1}	Command = product_amt change

POS_COMMAND_DATA_07	determine_command(COMMAND, SERVER, LIST, state) : {'e', &server, &list, 0}	Command = Exit
POS_COMMAND_DATA_08	determine_command(COMMAND, SERVER, LIST, state) : {'x', &server, &list, -1}	Error(Exception Handling)
POS_SERVER_DATA_01	convert_server(SERVER, LIST, receipt_num, type) : {&server, &list, receipt_num,0}	Server Data
POS_SERVER_DATA_02	convert_server(SERVER, LIST, receipt_num, type) : {&server, &list, receipt_num,1}	Server Data
POS_SERVER_DATA_03	convert_server(SERVER, LIST, receipt_num, type) : {&server, &list, receipt_num,2}	Server Data
POS_Controller_01	selected_state {-1}	Input Barcode
POS_Controller_02	selected_state {'0}	Wait
POS_Controller_03	selected_state {1}	Sale
POS_Controller_04	selected_state {2}	Refund
POS_Controller_05	selected_state {3}	Check
POS_Controller_06	selected_state {4}	Settlement
POS_Controller_07	selected_state {int_max}	Error(Exception Handling)
POS_Display_Main_01	display_main(DISPLAY): {&display}	Display Data
POS_SALE_DATA_01	sale(r_price, LIST, SERVER, SALE, state, DISPLAY) : {r_price, list, server, sale, 1, display}	Sale Data
POS_SALE_DATA_02	sale(r_price, LIST, SERVER, SALE, state, DISPLAY) : {0, list, server, sale, 1, display}	Error(Exception Handling) r_price < total_price
POS_SALE_DATA_03	sale(r_price, LIST, SERVER, SALE, state, DISPLAY) : {r_price, 0, server, sale, 1, display}	Error(Exception Handling) List x
POS_SALE_DATA_04	sale(r_price, LIST, SERVER, SALE, state, DISPLAY) : {r_price, list, 0, sale, 1, display}	Error(Exception Handling) p_amt > server_stock
POS_REFUND_DATA_01	Refund(LIST, SERVER, REFUND, state, DISPLAY) : {&list, &server, ,&refund, 2, display}	Refund Data

POS_CHECK_DATA_01	Check(SERVER, state, DISPLAY) : {&server, 3 , &display}	Chech Data
POS_SETTLEMENT_01	settlement(LIST, SERVER, SETTLEMENT, state) : {list, &server, &settlement, 4}	Settlement Data
POS_SETTLEMENT_02	settlement(LIST, SERVER, SETTLEMENT, state) : {0, &server, &settlement, 4}	Error(Exception Handling) List x
POS_Update_Display_01	update_display(SALE,REFUND, SERVER , DISPLAY) : {1, 0, 0, 0}	Display Command
POS_Update_Display_02	update_display(SALE,REFUND, SERVER , DISPLAY) : {0, 1, 0, 0}	Display Command
POS_Update_Display_03	update_display(SALE,REFUND, SERVER , DISPLAY) : {0, 0, 1, 0}	Display Command
POS_Update_Server_01	update_server(SERVER, LIST) : {&server, &list}	Save
POS_Update_Server_02	update_server(SERVER, LIST) : {&server, 0}	Error(Exception Handling) list x
POS_PRINT_DATA_01	print_data(SALE, REFUND, SERVER, SETTLEMENT) : {&sale, 0, 0, 0}	Printer Command
POS_PRINT_DATA_02	print_data(SALE, REFUND, SERVER, SETTLEMENT) : {0, &refund, 0, 0}	Printer Command
POS_PRINT_DATA_03	print_data(SALE, REFUND, SERVER, SETTLEMENT) : {0, 0, &server, 0}	Printer Command
POS_PRINT_DATA_04	print_data(SALE, REFUND, SERVER, SETTLEMENT) : {0, 0, 0, &settlemenet}	Printer Command

8.2 Test items

<Table 2 Test Design Identification>참조

8.3 Input specifications

<Table3 Test Case Identification>참조

8.4 Output specifications

<Table3 Test Case Identification>참조

9 Testing tasks

< Table4 Testing tasks & Schedule >

Task	Predecessor task	Special skills	Effort	Finish data
Unit Test Plan 작성	SRA 작성 SDS 작성 POS 구현			2017-11-01
Test Design Specification	Task1	POS 에 대한 이해		2017-11-02
Test Case Specification	Task 2			2017-11-03
Test Execution	Task 3	Test Code 작성 Unit test tool 활용		2017-11-04
Test result report	Task 4			2017-11-06

10 Environmental needs

POS system Unit Test를 위한 환경적 요구사항은 다음과 같다.

(1) Hardware &platform

GCC compiler/linker

(2) Unit test framework

CUni

(3) Integrated Platform Cygwin

Cygwin

11 Unit Test deliverables

12 Schedules

< Table4 Testing taskes & Schedule > 참조