Nusrs Testing #2

200511357 조재연
200212053 한성근
200611486 안정아

테스팅 과정

- 이전 테스트과정 요약
- 테스트 커버리지 사용
- 카테코리 테스트케이스 추출
- 테스트 케이스 선택
- 테스트 결과 도출
- 디버깅

요구사항

① SDT Condition, Action의 Undefined Variable
② SDT Condition, Action에서 연산자 사용 잘못
③ SDT에서 한 개의 Condition에 둘 이상의 Action 할당
④ FSM, TTS Transition의 Undefined Variable
⑤ FSM, TTS Transition의 연산자 사용 잘못
⑥ FOD, FSM, TTS에서 Transition이 없는 노드
⑦ FSM, TTS에서 Initial State로부터 Unreachable 노드
⑧ FOD에서 Output 변수와 연결된 노드 동일 명칭 체크

Test Plan

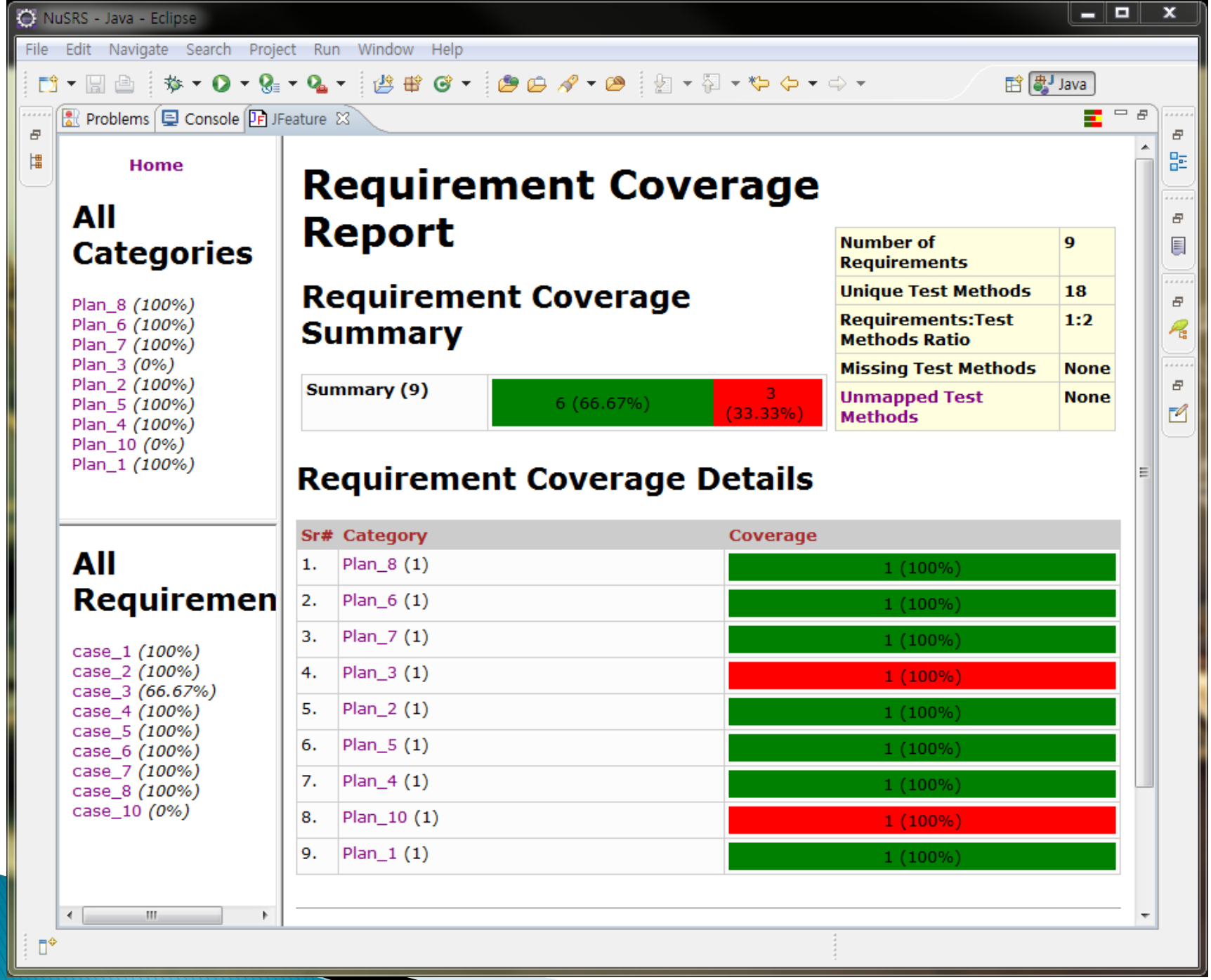
- Requirement에 대응하는 8가지와 2가지의 추가로 총 10가지

topic	Value
Test plan identifier	Plan_1
Introduction	SDT Condition 또는 Action의 Undefined Variable
Test items	Quick Check Module
Features to be tested	SDT의 Condition, Action, Description Window
Features not to be tested	-
Approach	Unit Testing
Item pass/fail criteria	Undefined Variable의 체크여부
Suspension criteria and resumption requirements	해당사항 없음
Test deliverables	Test Documentation Program Source Report in JFeature
Testing tasks	테스트케이스도출 환경설정 테스트케이스입력 결과작성
Environmental needs	Eclipse, JUnit, JFeature 필요
Responsibilities	조재연
Staffing and training needs	해당사항 없음
Schedule	해당사항 없음
Risks and contingencies	해당사항 없음
Approvals	해당사항 없음

Test Cases

- Case 1.1 부터 Case 10.1까지 총 18개

Topic	Value
Test case specification identifier	Case1.1
Test items	Plan_1, Design_1.1
Input specifications	Open Filename : case1_1-f_Refund.xml SDT name : f_Refund SDT의 테이블 변수명이 Description window에 있는 변수명과 일치하도록 함
Output specifications	Pass : No Error Exists Fail : Error At f_Refund, Action Row : 1 Col : 0 - Undefined Variable : 변수명을 출력
Environmental needs	Eclipse및 NuSRS프로그램이 필요 그리고 Test File이 필요하다.
Special procedural requirements	f_Refund SDT를 만든뒤, 아이콘을 더블클릭하여 Table을 오픈한다.
Intercase dependencies	Case1.2와 상호의존함.


NuSRS - Java - Eclipse
File Edit Navigate Search Project Run Window Help
Java
Problems Console JFeature
Home
All Categories
Plan_8 (100%)
Plan_6 (100%)
Plan_7 (100%)
Plan_3 (0%)
Plan_2 (100%)
Plan_5 (100%)
Plan_4 (100%)
Plan_10 (0%)
Plan_1 (100%)
All Requiremen
case_1 (100%)
case_2 (100%)
case_3 (66.67%)
case_4 (100%)
case_5 (100%)
case_6 (100%)
case_7 (100%)
case_8 (100%)
case_10 (0%)
Requirement Coverage Report
Requirement Coverage Summary
Summary (9)
6 (66.67%)
3 (33.33%)
Number of Requirements 9
Unique Test Methods 18
Requirements:Test Methods Ratio 1:2
Missing Test Methods None
Unmapped Test Methods None
Requirement Coverage Details
Sr# Category Coverage
1. Plan_8 (1) 1 (100%)
2. Plan_6 (1) 1 (100%)
3. Plan_7 (1) 1 (100%)
4. Plan_3 (1) 1 (100%)
5. Plan_2 (1) 1 (100%)
6. Plan_5 (1) 1 (100%)
7. Plan_4 (1) 1 (100%)
8. Plan_10 (1) 1 (100%)
9. Plan_1 (1) 1 (100%)

Eclemma

- http://eclemma.org/
- 테스팅 커버리지 툴이다.
- Eclipse Plugin으로 사용한다.
- Junit과 맞물려 돌아간다.
- 개발/테스트 싸이클이 빨라진다.
- 강력한 커버리지 분석이 가능하다.
- 프로젝트를 변경하지 않아도 된다.

이전 프로젝트의 테스팅 결과

Problems | Javadoc | Declaration | Coverage

TestSuite (2010. 6. 5 오전 9:57:28)

Element	Coverage	Covered Instructio...	Total Instructions
HierarchyWindowPanel.java	47.1 %	144	306
FSMTransitionDialog.java	0.0 %	0	530
ExtFilter.java	0.0 %	0	27
EditorPanel.java	72.2 %	385	533
DescriptionWindowPanel.java	40.2 %	552	1374
Console.java	39.3 %	24	61
data	33.5 %	6252	18673
XMLLoader.java	82.4 %	1202	1459
Variable.java	55.4 %	158	285
Transition.java	23.4 %	64	274
TokenMgrError.java	0.0 %	0	169
Token.java	100.0 %	10	10
SimpleCharStream.java	28.5 %	232	813
SDTPanel.java	95.9 %	141	147
SDTConActPanel.java	59.7 %	649	1087
SDTableModel.java	9.8 %	6	61
SDTableData.java	26.5 %	79	298
SDT.java	9.0 %	48	535
RefNode.java	35.5 %	11	31
QuickCheckPanel.java	78.1 %	964	1234
ParseException.java	63.0 %	199	316
NuSpec.java	28.9 %	181	626
NuNode.java	22.6 %	45	199
IONode.java	33.8 %	44	130
FSMViewer.java	2.3 %	20	881
FSMTransition.java	45.2 %	109	241
FSM.java	2.6 %	46	1736

SDT.java	9.0 %	48	535
RefNode.java	35.5 %	11	31
QuickCheckPanel.java	78.1 %	964	1234
ParseException.java	63.0 %	199	316
NuSpec.java	28.9 %	181	626

이전 프로젝트의 테스팅 결과

```
                    row = conData.getRowCount();
                    col = conData.getColCount();
                    for (i = 0; i < row; i++) {
                        data = conData.getValueAt(i, 0);
                        data = data + ";";
                        while (data.charAt(0) == ' ') {
                            data = data.substring(1);
                        }
                        input = new ByteArrayInputStream(data.getBytes());
                        parser.ReInit(input);
                        output = parser.hello(table, checking.getVars());
                        if (output.compareTo("OK") != 0) {
                            output = "Error At " + sdt.getName()
                                    + ", Condition Row : " + i + " Col : 0 - "
                                    + output;
                            System.out.println(output);
                            errors.addElement(output);
                            error++;
                            errorList.add(checking);
                        }
                    }

                    row = actData.getRowCount();
                    col = actData.getColCount();
                    for (i = 0; i < row; i++) {
                        data = actData.getValueAt(i, 0);
                        data = data + ";";
                        while (data.charAt(0) == ' ') {
                            data = data.substring(1);
                        }
                        input = new ByteArrayInputStream(data.getBytes());
                        parser.ReInit(input);
```

이전 프로젝트의 테스팅 결과

```
                output = parser.hello(table, checking.getVars());
                if (output.compareTo("OK") != 0) {
                    output = "Error At " + checking.getName()
                            + ", Transition " + tr.toString2()
                            + ", Action - " + output;
                    System.out.println(output);
                    errors.addElement(output);
                    error++;
                    errorList.add(checking);
                }

            }

            // check whether transition exists or not, reachable from
            // initial State

            Iterator node = ((FSM) checking).getNodeIterator();
            while (node.hasNext()) {
                nodes.add(node.next());
            }
            transitions = ((FSM) checking).getTransitionIterator();
            while (transitions.hasNext()) {
                FSMTransition tr = (FSMTransition) transitions.next();
                nodes.remove(tr.getSource());
                nodes.remove(tr.getTarget());
            }
            while (nodes.size() != 0) {
                output = "Error At " + checking.getName() + ", Node "
                        + nodes.removeFirst() + " has no transition.";
                System.out.println(output);
                errors.addElement(output);
                error++;
                errorList.add(checking);
            }
```

QuickCheckPanel 테스팅 결과 분석

- 에러나 성공을 출력하는 8곳은 테스트가 되었고, 에러를 출력하는 6곳이 테스트가 되지 않았다.
- 예외를 출력하는 곳은 테스트 되지 않았고 마우스를 더블클릭 하는 곳도 테스트 되지 않았으나 무시한다.

Pesticide paradox

- 동일한 테스트 케이스로 동일한 테스트를 반복적으로 수행한다면, 더 이상 새로운 버그를 찾아내지 못할 것이다.
- 이러한 Pesticide paradox를 극복하기 위해서는 테스트 케이스를 정기적으로 리뷰하고 개선할 필요가 있다.

SDT – Category partition test

Contradict Conditions Rows	Operation Not Corrected Rows In	Condition rows Have Duplication Result	Used Undefined Variable In
0	Not	0	Not
2	Condition	2	Condition
many	Action	many	Action
	Both		Both

FOD – Category partition test

Number of not transition in nodes	Number of different names in node and output node	Number of Unused Variable
0	1	0
1	2	1
many	many	many

FSM – Category partition test

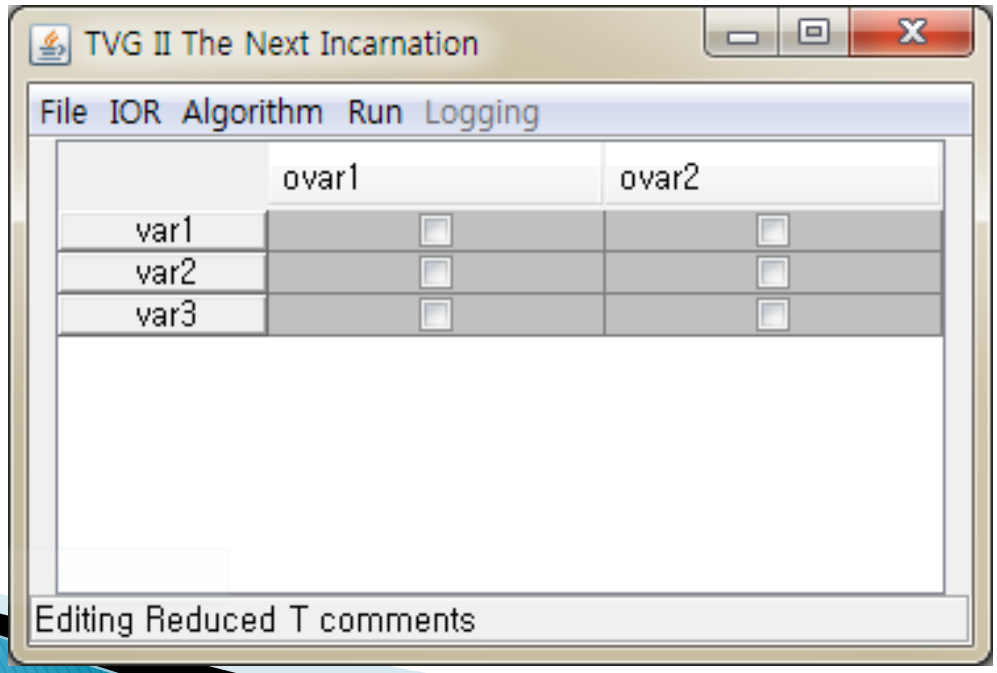
Number of node using undefined variable	Number of node having no transition	Number of unreachable node from Initial State	Number of using incorrect operator
0	0	0	0
1	1	1	1
many	many	many	many

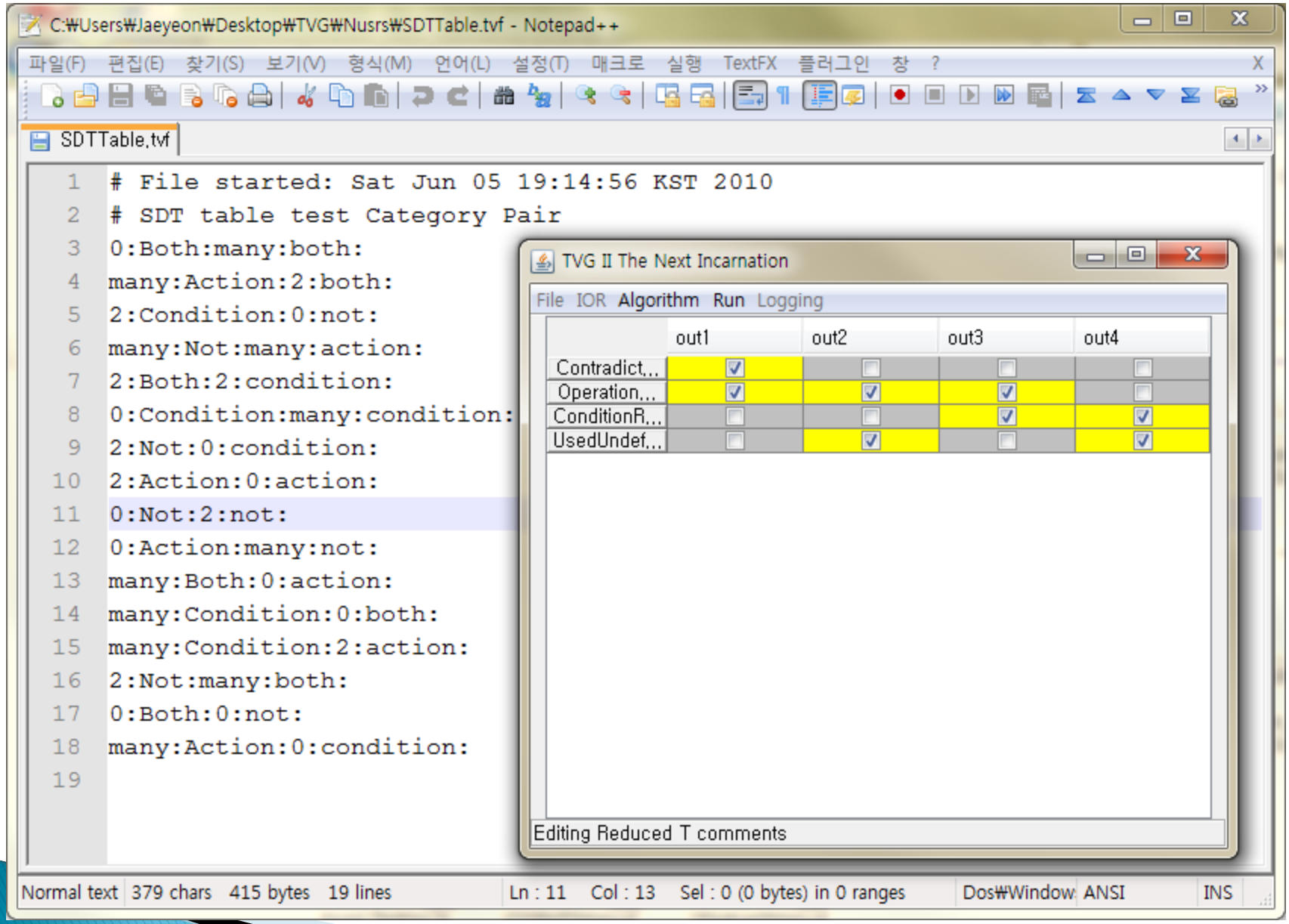
TTS – Category partition test

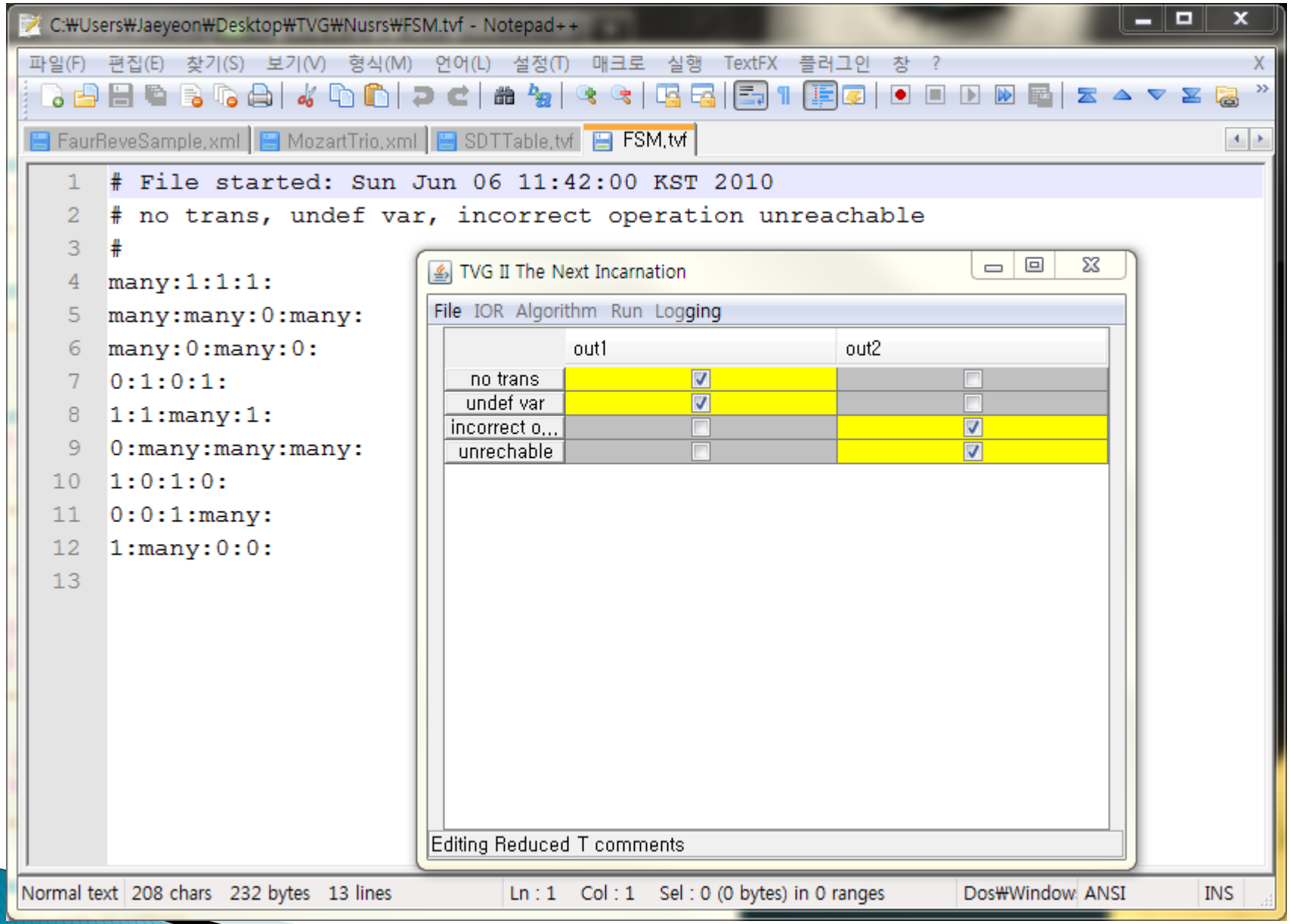
Number of node using undefined variable	Number of node having no transition	Number of unreachable node from Initial State	Number of using incorrect operator
0	0	0	0
1	1	1	1
many	many	many	many

Test Vecter Generator(TVG)

- Pairwise Testing을 돕는 도구
- Input-Output relationship에 기반해서 조합을 만드는 것을 제공한다.




C:₩Users₩Jaeyeon₩Desktop₩TVG₩Nusrs₩SDTTable.tvf - Notepad++
파일(F) 편집(E) 찾기(S) 보기(V) 형식(M) 언어(L) 설정(T) 매크로 실행 TextFX 플러그인 창 ?
SDTTable.tvf
1 # File started: Sat Jun 05 19:14:56 KST 2010
2 # SDT table test Category Pair
3 0:Both:many:both:
4 many:Action:2:both:
5 2:Condition:0:not:
6 many:Not:many:action:
7 2:Both:2:condition:
8 0:Condition:many:condition:
9 2:Not:0:condition:
10 2:Action:0:action:
11 0:Not:2:not:
12 0:Action:many:not:
13 many:Both:0:action:
14 many:Condition:0:both:
15 many:Condition:2:action:
16 2:Not:many:both:
17 0:Both:0:not:
18 many:Action:0:condition:
19
TVG II The Next Incarnation
File IOR Algorithm Run Logging
out1 out2 out3 out4
Contradict...
Operation...
ConditionR...
UsedUndef...
Editing Reduced T comments
Normal text 379 chars 415 bytes 19 lines Ln : 11 Col : 13 Sel : 0 (0 bytes) in 0 ranges Dos₩Window ANSI INS


C:₩Users₩Jaeyeon₩Desktop₩TVG₩Nusrs₩FSM.tvf - Notepad++
파일(F) 편집(E) 찾기(S) 보기(V) 형식(M) 언어(L) 설정(T) 매크로 실행 TextFX 플러그인 창 ?
FaurReveSample.xml
MozartTrio.xml
SDTTable.tvf
FSM.tvf
1 # File started: Sun Jun 06 11:42:00 KST 2010
2 # no trans, undef var, incorrect operation unreachable
3 #
4 many:1:1:1:
5 many:many:0:many:
6 many:0:many:0:
7 0:1:0:1:
8 1:1:many:1:
9 0:many:many:many:
10 1:0:1:0:
11 0:0:1:many:
12 1:many:0:0:
13
TVG II The Next Incarnation
File IOR Algorithm Run Logging
out1
out2
no trans
undef var
incorrect o...
unrechable
Editing Reduced T comments
Normal text 208 chars 232 bytes 13 lines
Ln : 1 Col : 1 Sel : 0 (0 bytes) in 0 ranges
Dos₩Window ANSI INS

Home

All Categories

Plan_8 *(100%)*
Plan_6 *(100%)*
Plan_7 *(100%)*
Plan_3 *(0%)*
Plan_2 *(100%)*
Plan_5 *(100%)*
Plan_4 *(100%)*
Plan_11 *(0%)*
Plan_10 *(0%)*
Plan_1 *(100%)*

All Requirements

case_1 *(100%)*
case_2 *(100%)*
case_3 *(66.67%)*
case_4 *(100%)*
case_5 *(100%)*
case_6 *(100%)*
case_7 *(100%)*
case_8 *(100%)*
case_10 *(0%)*
case_11 *(66.67%)*

Requirement Coverage Report

Requirement Coverage Summary

Summary (10)	6 (60%)	4 (40%)

Number of Requirements	10
Unique Test Methods	24
Requirements:Test Methods Ratio	1:2
Missing Test Methods	None
Unmapped Test Methods	None

Requirement Coverage Details

Sr#	Category	Coverage
1.	Plan_8 (1)	1 (100%)
2.	Plan_6 (1)	1 (100%)
3.	Plan_7 (1)	1 (100%)
4.	Plan_3 (1)	1 (100%)
5.	Plan_2 (1)	1 (100%)
6.	Plan_5 (1)	1 (100%)
7.	Plan_4 (1)	1 (100%)
8.	Plan_11 (1)	1 (100%)
9.	Plan_10 (1)	1 (100%)
10.	Plan_1 (1)	1 (100%)

Report generated on 일, 06 6월 2010 23:08:38 KST

Powered By JFeature

TestSuite (2010. 6. 6 오후 11:22:26)

Element	Coverage	Covered Instructio...	Total Instructions
FODViewer.java	1.7 %	51	3059
FSM.java	2.8 %	49	1736
FSMTransition.java	45.2 %	109	241
FSMViewer.java	2.3 %	20	881
IONode.java	33.8 %	44	130
NuNode.java	22.6 %	45	199
NuSpec.java	28.9 %	181	626
ParseException.java	63.0 %	199	316
QuickCheckPanel.java	90.3 %	1114	1234
QuickCheckPanel	90.3 %	1114	1234
RefNode.java	35.5 %	11	31
SDT.java	9.0 %	48	535
SDTableData.java	26.5 %	79	298
SDTableModel.java	9.8 %	6	61
SDTConActPanel.java	59.7 %	649	1087
SDTPanel.java	95.9 %	141	147
SimpleCharStream.java	30.0 %	244	813
Token.java	100.0 %	10	10
TokenMgrError.java	56.2 %	95	169
Transition.java	23.4 %	64	274
Variable.java	55.4 %	158	285

결함 보고

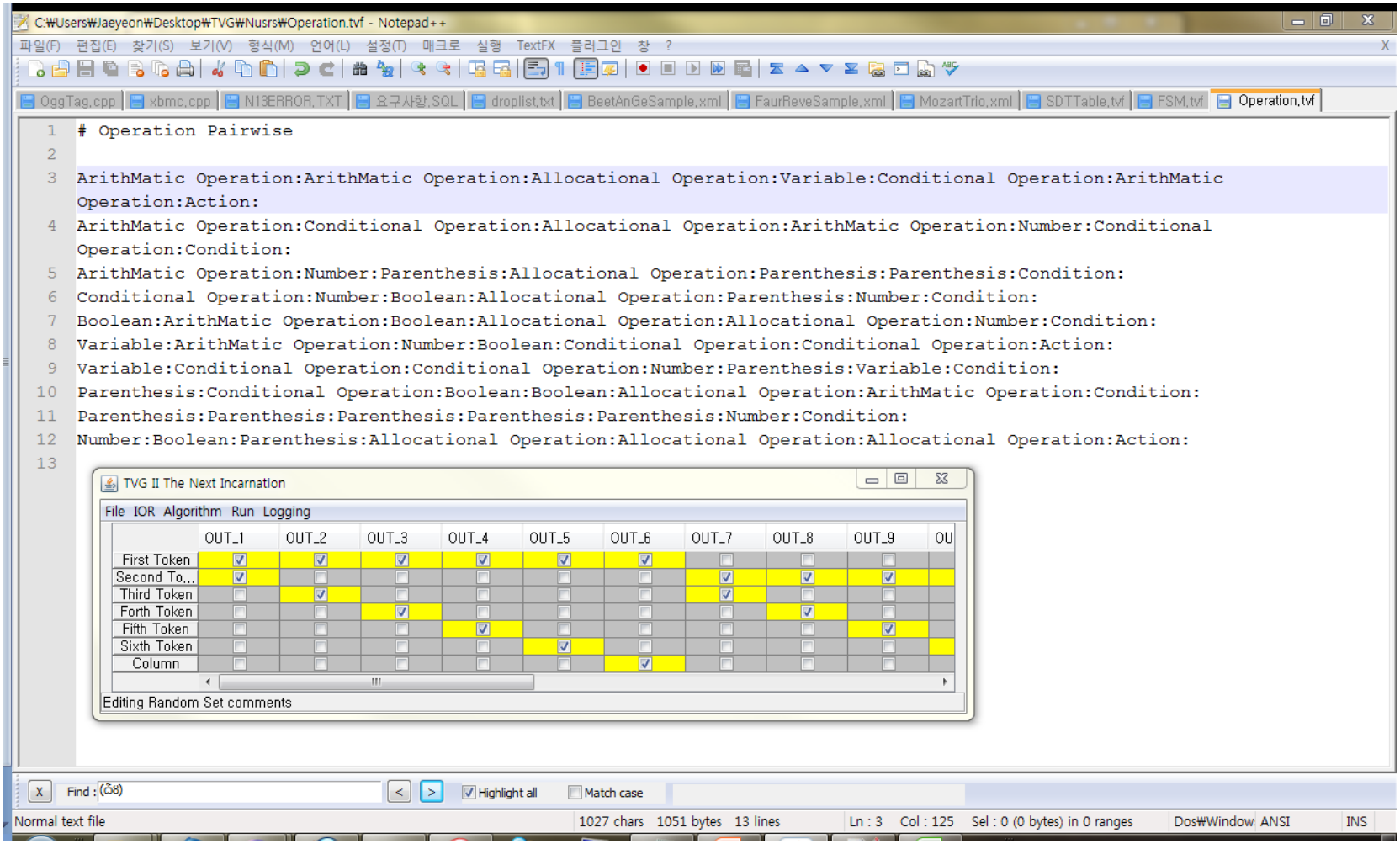
- 한 컨디션에 true가 2곳일 때 에러를 잡지 못한다.(case 3_3, case 11_1)
- 두 컨디션 이상에서 서로 모순인 식이 동시에 참, 거짓으로 나오는 것을 잡지 못한다. testcase10_1, case 11_3
- 조건이 같은 컨디션에서 다른 액션들이 할당 되는 것을 잡지 못한다 testcase10_3

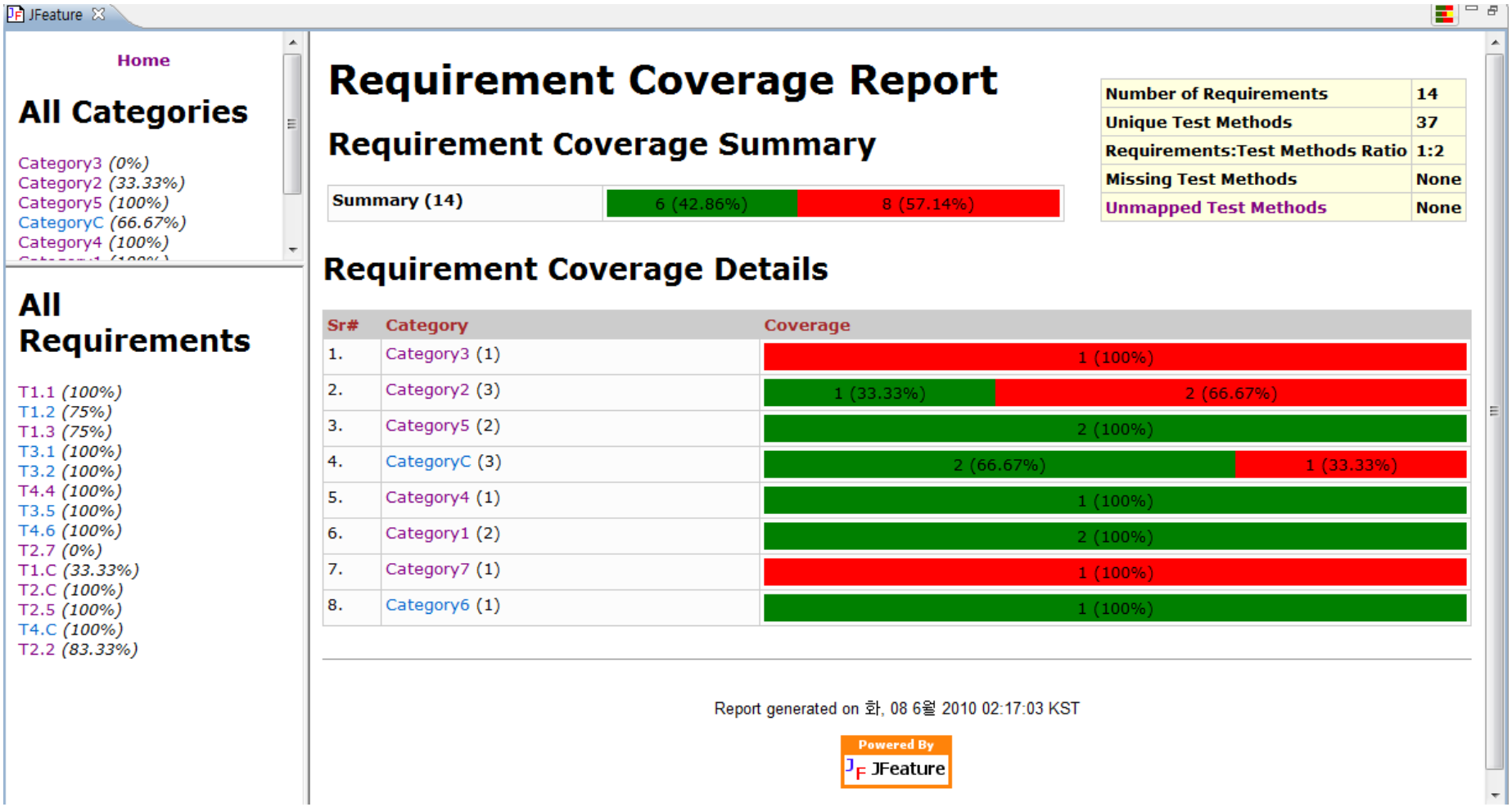
세부 테스트

- 이전에 했던 Pairwise 테스트는 Category들이 추상적이어서 Tester가 임의로 조작할 수밖에 없었다. 따라서 신뢰도가 떨어질 수 있다.
- Failure가 많이 날 수 있다고 판단되는 곳은 연산자검사다. 따라서 집중적으로 검사하였다.

Generate table

Column	FirstToken	SecondToken	ThirdToken	FourthToken	FifthToken	SixthToken	Node
Condition	ArithMatic Operation	ArithMatic Operation	ArithMatic Operation	ArithMatic Operation	ArithMatic Operation	ArithMatic Operation	SDT
Action	Conditional Operation	Conditional Operation	Conditional Operation	Conditional Operation	Conditional Operation	Conditional Operation	FSM
	Boolean	Boolean	Boolean	Boolean	Boolean	Boolean	TTS
	Variable	Variable	Variable	Variable	Variable	Variable	
	Allocational Operation	Allocational Operation	Allocational Operation	Allocational Operation	Allocational Operation	Allocational Operation	
	Parenthesis(괄호)	Parenthesis(괄호)	Parenthesis(괄호)	Parenthesis(괄호)	Parenthesis(괄호)	Parenthesis(괄호)	
	Number	Number	Number	Number	Number	Number	


C:₩Users₩Jaeyeon₩Desktop₩TVG₩Nusrs₩Operation.tvf - Notepad++
파일(F) 편집(E) 찾기(S) 보기(V) 형식(M) 언어(L) 설정(T) 매크로 실행 TextFX 플러그인 창 ?
OggTag.cpp xbmc.cpp N13ERROR.TXT 요구사항.SQL droplist.txt BeetAnGeSample.xml FaurReveSample.xml MozartTrio.xml SDTTable.tvf FSM.tvf Operation.tvf
1 # Operation Pairwise
2
3 ArithMatic Operation:ArithMatic Operation:Allocational Operation:Variable:Conditional Operation:ArithMatic Operation:Action:
4 ArithMatic Operation:Conditional Operation:Allocational Operation:ArithMatic Operation:Number:Conditional Operation:Condition:
5 ArithMatic Operation:Number:Parenthesis:Allocational Operation:Parenthesis:Parenthesis:Condition:
6 Conditional Operation:Number:Boolean:Allocational Operation:Parenthesis:Number:Condition:
7 Boolean:ArithMatic Operation:Boolean:Allocational Operation:Allocational Operation:Number:Condition:
8 Variable:ArithMatic Operation:Number:Boolean:Conditional Operation:Conditional Operation:Action:
9 Variable:Conditional Operation:Conditional Operation:Number:Parenthesis:Variable:Condition:
10 Parenthesis:Conditional Operation:Boolean:Boolean:Allocational Operation:ArithMatic Operation:Condition:
11 Parenthesis:Parenthesis:Parenthesis:Parenthesis:Parenthesis:Number:Condition:
12 Number:Boolean:Parenthesis:Allocational Operation:Allocational Operation:Allocational Operation:Action:
13
TVG II The Next Incarnation
File IOR Algorithm Run Logging
OUT_1 OUT_2 OUT_3 OUT_4 OUT_5 OUT_6 OUT_7 OUT_8 OUT_9 OU
First Token
Second To...
Third Token
Forth Token
Fifth Token
Sixth Token
Column
Editing Random Set comments
Find :
Highlight all
Match case
Normal text file
1027 chars 1051 bytes 13 lines
Ln : 3 Col : 125 Sel : 0 (0 bytes) in 0 ranges
Dos₩Window ANSI INS


JFeature
Home
All Categories
Category3 (0%)
Category2 (33.33%)
Category5 (100%)
CategoryC (66.67%)
Category4 (100%)
All Requirements
T1.1 (100%)
T1.2 (75%)
T1.3 (75%)
T3.1 (100%)
T3.2 (100%)
T4.4 (100%)
T3.5 (100%)
T4.6 (100%)
T2.7 (0%)
T1.C (33.33%)
T2.C (100%)
T2.5 (100%)
T4.C (100%)
T2.2 (83.33%)
Requirement Coverage Report
Requirement Coverage Summary
Summary (14)
6 (42.86%)
8 (57.14%)
Number of Requirements 14
Unique Test Methods 37
Requirements:Test Methods Ratio 1:2
Missing Test Methods None
Unmapped Test Methods None
Requirement Coverage Details
Sr# Category Coverage
1. Category3 (1) 1 (100%)
2. Category2 (3) 1 (33.33%) 2 (66.67%)
3. Category5 (2) 2 (100%)
4. CategoryC (3) 2 (66.67%) 1 (33.33%)
5. Category4 (1) 1 (100%)
6. Category1 (2) 2 (100%)
7. Category7 (1) 1 (100%)
8. Category6 (1) 1 (100%)
Report generated on 화, 08 6월 2010 02:17:03 KST
Powered By
JFeature

추가 결함 발견

- Pairwise로 뽑은 테스트케이스는 테스트를 통과했다.
- Condition에 할당연산자를 넣은경우와 Action에 비교연산자를 넣은 경우 에러를 잡지 못하였다.

Quick Check Panel Testing Issue

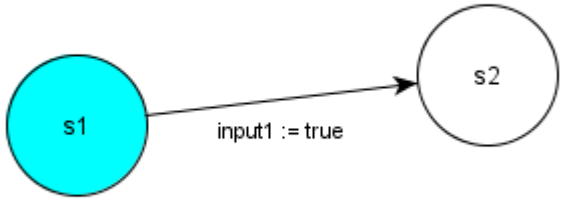
- Input에서 바로 Output으로 갈 경우
- Input으로 Transition이 갈 경우
- Output에서 Transition이 갈 경우
- 한 Node 에서 Output이 2개 이상
- Time에서 start보다 end가 작은 경우

Structured Decision Table:

Conditions	1	2
f_Button_Refund = true	T	T
Action	**1**	**2**
f_Refund := th_Current_Money	O	
f_Refund := 0		O

Conditions	1	2	3	4
th_Current_Money >= k_Cost_Item1	T	F	T	F
th_Current_Money >= k_Cost_Item2	F	T	T	F
th_Current_Money < k_Cost_Item1	T	F	F	F
Action	**1**	**2**	**3**	**4**
f_Button_LED := 1	O			
f_Button_LED := 2		O		
f_Button_LED := 3			O	
f_Button_LED := 0				O

Condition에 할당 연산자 사용



Action에 비교연산자 사용

Structured Decision Table:

Conditions	1	2	3
true	T		
Action	**1**	**2**	**3**
f_case12_13 < 3	O		