

# CTIP 환경 구축 & Unit Test “NuSRS”

200511305 김성규  
200511306 김성훈  
200614164 김효석  
200611124 유성배  
200518036 곡진화

# CTIP

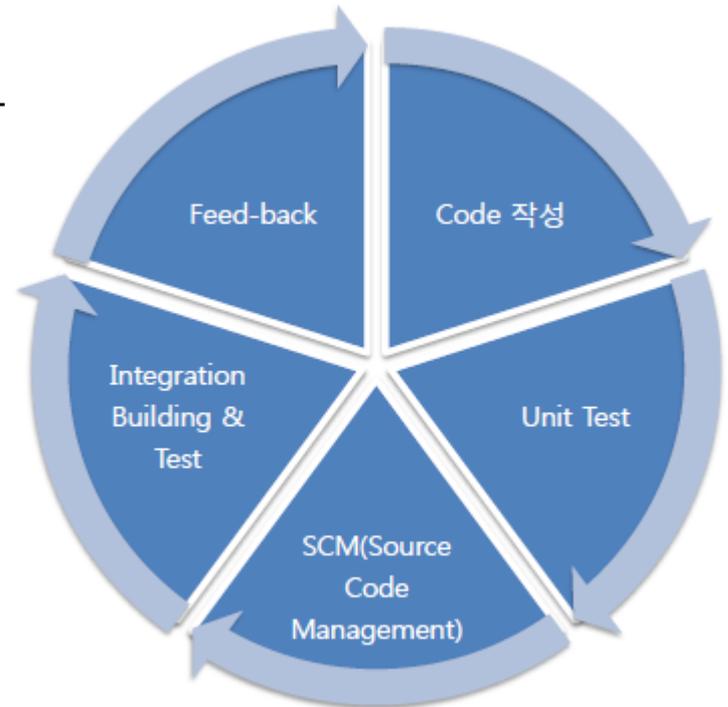
## ❖ CTIP

### ■ Continuous Test & Integration Platform

- CI 서버를 통한 지속적 통합 및 빌드
- 품질 도구를 통한 코드 품질 검토 (테스트 및 정적 분석)
- 빌드 결과의 배포 및 관련자에게 통보

### ■ 구축 환경

- 코드 작성 : Eclipse
- SCM : Subclipse
- Unit Test : JUnit
- Build 자동화 : Ant
- CI 도구 : Hudson



# Ant

---

## ❖ ANT

- Java 기반의 빌드 프로세스 자동화 도구
- Eclipse는 Ant 플러그인 기본 내장
- 플랫폼 독립적

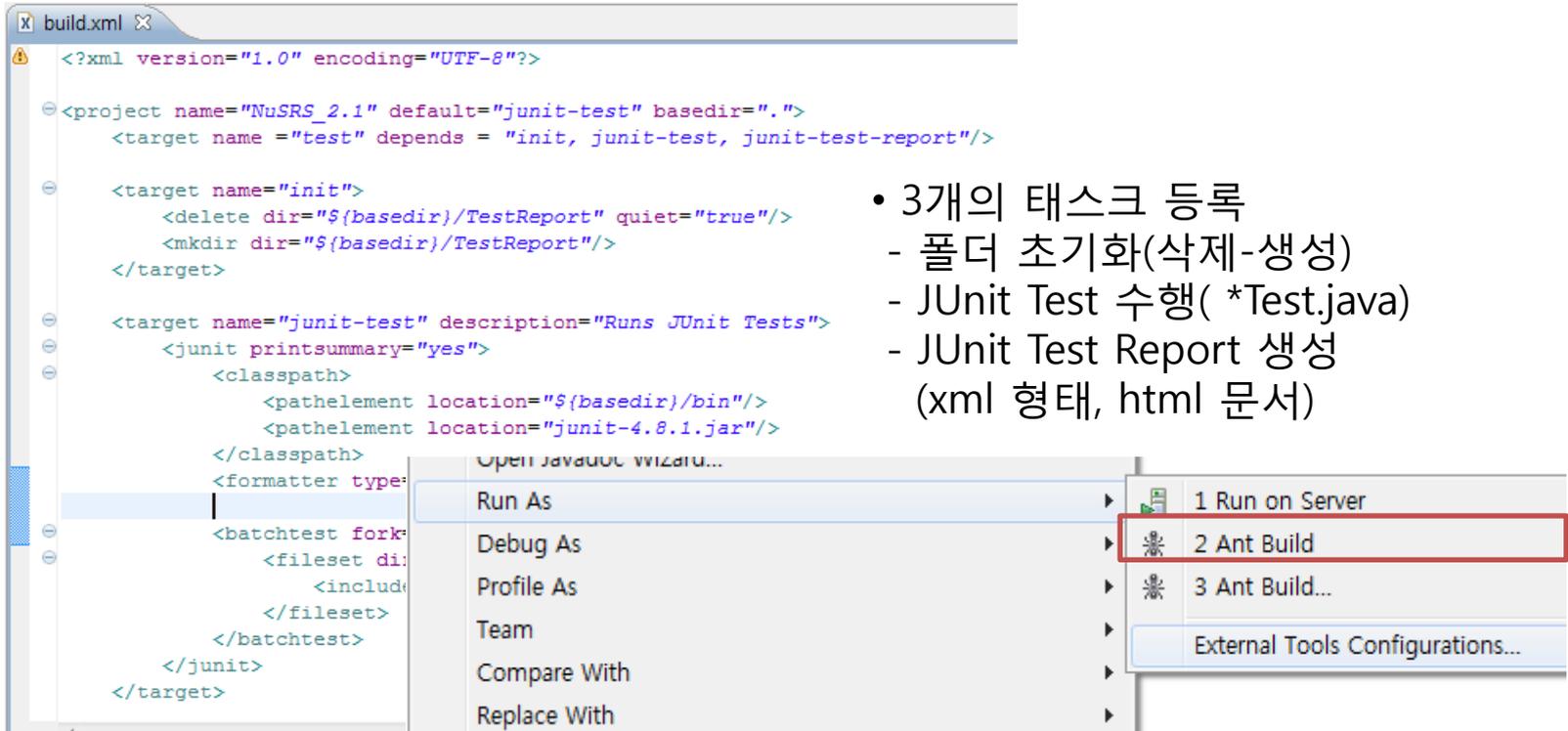
## ❖ Ant 요소

- project : 루트 태그, 프로젝트 설명
- target : 수행할 태스크를 지정
- property : 사용하게 될 프로퍼티 지정
- 수행 작업 : 태스크
  - mkdir : 디렉토리 생성
  - Copy, Delete : 파일/ 디렉토리 복사, 삭제
  - Javac : 자바 파일 컴파일
  - jar : jar 파일 생성
  - Javadoc : javadoc을 생성

# Ant

## ❖ Ant 작성 & 수행

- File>New>File
- build.xml 파일 생성 후 스크립트 작성
- xml 파일 선택 후 Run As>Ant Build로 수행



The screenshot shows an IDE window titled 'build.xml' with the following XML content:

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="NuSRS_2.1" default="junit-test" basedir=".">
  <target name="test" depends="init, junit-test, junit-test-report"/>
  <target name="init">
    <delete dir="${basedir}/TestReport" quiet="true"/>
    <mkdir dir="${basedir}/TestReport"/>
  </target>
  <target name="junit-test" description="Runs JUnit Tests">
    <junit printsummary="yes">
      <classpath>
        <pathelement location="${basedir}/bin"/>
        <pathelement location="junit-4.8.1.jar"/>
      </classpath>
      <formatter type="xml" usefile="true"/>
    </junit>
  </target>
  <batchtest fork="true" failure="error" maxmemory="1024m">
    <fileset dir="src">
      <include name="**/*Test.java"/>
    </fileset>
  </batchtest>
</project>
```

On the right side, the 'Run As' context menu is open, showing the following options:

- 1 Run on Server
- 2 Ant Build
- 3 Ant Build...
- External Tools Configurations...

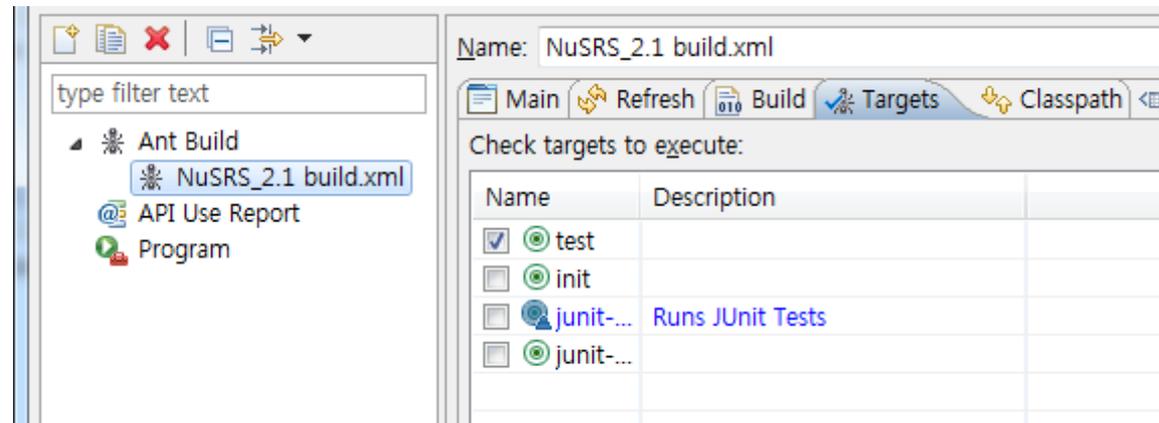
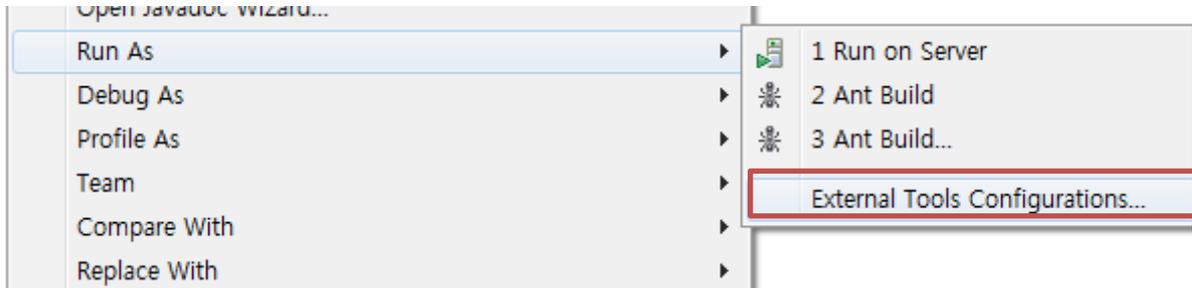
The '2 Ant Build' option is highlighted with a red box.

- 3개의 태스크 등록
  - 폴더 초기화(삭제-생성)
  - JUnit Test 수행( \*Test.java)
  - JUnit Test Report 생성 (xml 형태, html 문서)

# Ant

## ❖ Ant 설정

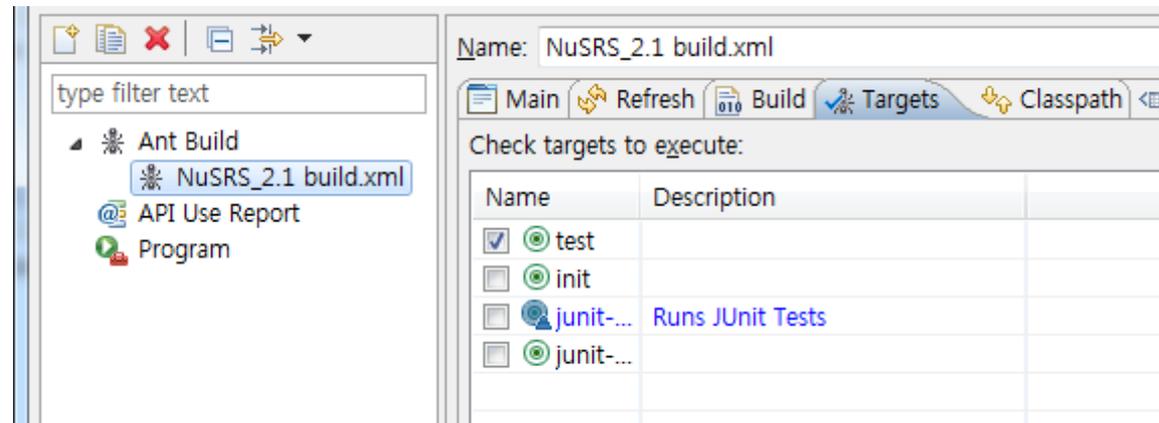
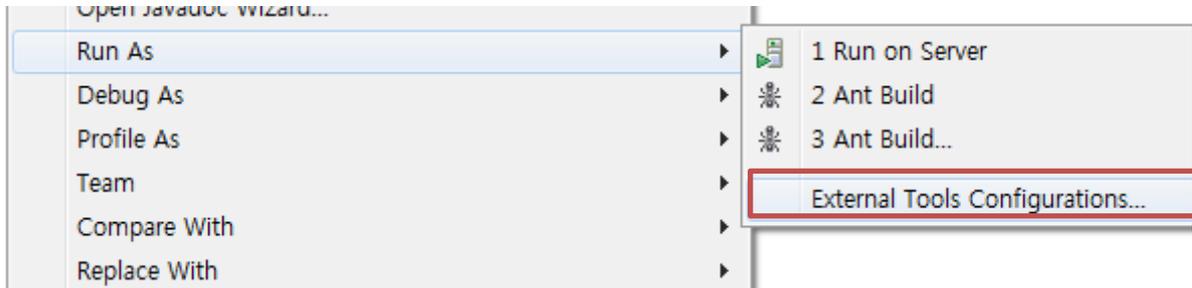
- 실행시킬 target 선택 가능



# Ant

## ❖ Ant 설정

- 실행시킬 target 선택 가능



# Ant

## ❖ JUnit Report - Html 생성

file:///D:/workspace/NuSRS\_2.1/TestReport/html/index.html

### Unit Test Results.

Designed for use with [JUnit](#) and [Ant](#).

#### Summary

| Tests | Failures | Errors | Success rate | Time  |
|-------|----------|--------|--------------|-------|
| 33    | 16       | 1      | 48.48%       | 0.328 |

Note: *failures* are anticipated and checked for with assertions while *errors* are unanticipated.

#### Packages

| Name  | Tests | Errors | Failures | Time(s) | Time Stamp          | Host |
|-------|-------|--------|----------|---------|---------------------|------|
| logic | 18    | 0      | 10       | 0.133   | 2011-05-05T20:36:38 | Wind |
| NuFTA | 15    | 1      | 6        | 0.195   | 2011-05-05T20:36:37 | Wind |

#### Class logic.stringToLogicTest

| Name              | Tests | Errors | Failures | Time(s) | Time Stamp          | Host |
|-------------------|-------|--------|----------|---------|---------------------|------|
| stringToLogicTest | 18    | 0      | 10       | 0.133   | 2011-05-05T20:36:38 | Wind |

#### Tests

| Name              | Status  | Type   | Time(s) |
|-------------------|---------|--|---------|
| testStringToLogic | Failure | Not yet implemented<br>junit.framework.AssertionFailedError: Not yet implemented<br>at logic.stringToLogicTest.testStringToLogic (stringToLogicTest.java:31) | 0.010   |
| testSetSubElement | Success |  | 0.016   |
| testDivideSub     | Failure | Not yet implemented<br>junit.framework.AssertionFailedError: Not yet implemented<br>at logic.stringToLogicTest.testDivideSub (stringToLogicTest.java:76)     | 0.002   |
| testConvertTruth  | Success |  | 0.001   |
| testRemoveNot     | Success |  | 0.001   |
| testRemoveBracket | Success |  | 0.000   |
| testRemoveEnter   | Success |  | 0.000   |
| testMakeMember    | Success |  | 0.001   |
| testSetTruth      | Success |  | 0.000   |
| testCopyLM        | Success |  | 0.001   |

JFeature보다 그래픽적이지 않지만  
html은 배포하기 쉬움

# Hudson

---

## ❖ Hudson

- Continuous Integration을 위한 Java build 지원 도구
- 특징
  - 쉬운 설치
  - 웹 기반 UI로 Client에서 별도의 소프트웨어 설치 불필요
  - 동시 여러 프로젝트 관리 가능
- 기능
  - Source 관리 프로그램을 이용하여 일관성 유지
  - 자동 build (commit / 시간 간격)
  - 자동 테스트
  - 일일 Check out과 build

# Hudson

## ❖ 설치 & 실행

- hudson-ci.org 에서 hudson.war 파일 download
- java -jar hudson.war 명령어로 실행
- 8080 port번호로 접속 가능

The screenshot shows the Hudson web interface. At the top, there is a search bar and a link to '자동 재실행 켜기'. Below the search bar, there are links for '소개 내용 입력', '새 작업', 'Hudson 관리', '개발자', and '빌드 기록'. The main content area displays a table of jobs. The table has columns for 'S' (Status), 'W' (Weather icon), '작업' (Job name), '최근 성공' (Last Success), '최근 실패' (Last Failure), and '최근 소요 시간' (Last Build Time). The 'test' job is shown with a success status, a weather icon, and a build time of 2.3 seconds. Below the table, there is a section for '빌드 대기 목록' (Build Queue) which is currently empty, and a section for '빌드 실행 상태' (Build Execution Status) which shows two builds in progress.

| S | W | 작업 ↓                 | 최근 성공             | 최근 실패             | 최근 소요 시간 |
|---|---|----------------------|-------------------|-------------------|----------|
|   |   | <a href="#">test</a> | 3 days 4 hr (#27) | 3 days 4 hr (#23) | 2.3 sec  |

아이콘: [S](#) [M](#) [L](#)

범례 [모든 것에 대해](#) [실패에 대해](#) [마지막 빌드에 대해](#)

**빌드 대기 목록**  
빌드 대기 항목이 없습니다.

**빌드 실행 상태**

| # | 상태   |
|---|------|
| 1 | 대기 중 |
| 2 | 대기 중 |

# Hudson

## ❖ Hudson 설정



새 작업

**Hudson 관리**

개발자

빌드 기록

**빌드 대기 목록**  
빌드 대기 항목이 없습니다.

**빌드 실행 상태**

| # | 상태   |
|---|------|
| 1 | 대기 중 |
| 2 | 대기 중 |

### Manage Hudson



[Configure System](#)  
Configure global settings and paths.



[Select Configuration from Disk](#)  
**JDK**



[System Information](#)  
Displays various en



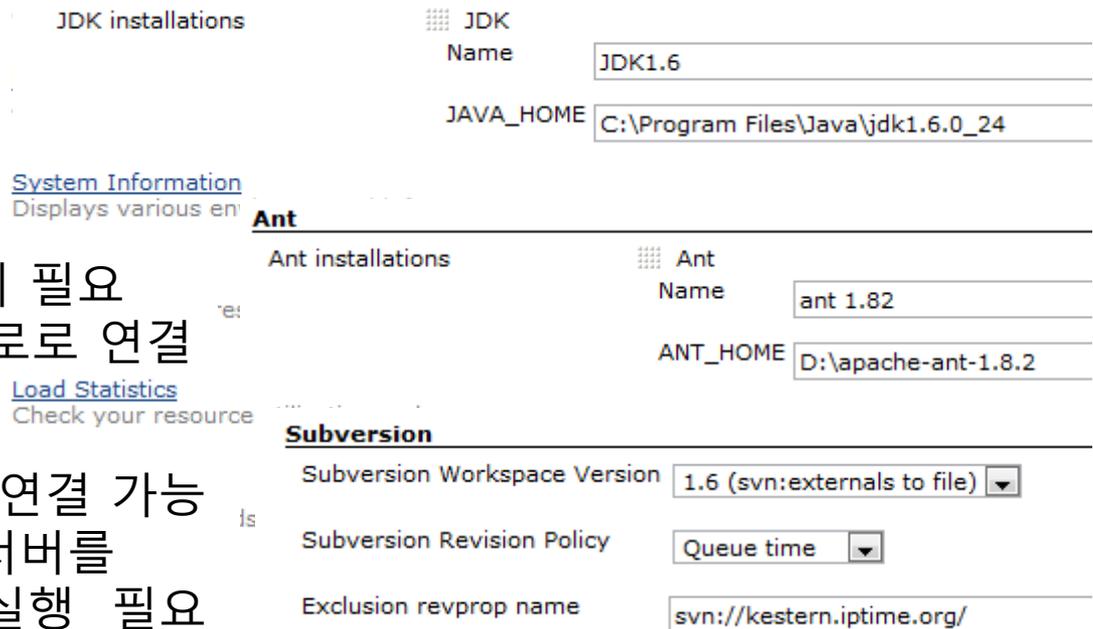
[Load Statistics](#)  
Check your resource

- Ant 설치 필요
- 설치 경로로 연결



- 외부 저장소 연결 가능
- subversion 서버를 독립적으로 실행 필요

### JDK, Ant, Subversion 설정



**JDK**

JDK installations

| JDK | Name   | JAVA_HOME                         |
|-----|--------|-----------------------------------|
|     | JDK1.6 | C:\Program Files\Java\jdk1.6.0_24 |

**Ant**

Ant installations

| Ant | Name     | ANT_HOME            |
|-----|----------|---------------------|
|     | ant 1.82 | D:\apache-ant-1.8.2 |

**Subversion**

Subversion Workspace Version: 1.6 (svn:externals to file)

Subversion Revision Policy: Queue time

Exclusion revprop name: svn://kester.n.iptime.org/

# Hudson

## ❖ 작업 생성 & 설정

**새 작업**

[Hudson 관리](#)

[개발자](#)

[빌드 기록](#)

작업명

- Build a free-style software project**  
이것은 Hudson의 주요 기능입니다. Hudson은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.
- Build a maven2/3 project**  
Maven2/3 프로젝트를 빌드합니다. Hudson은 POM 파일의 이점을 가지고 있고 급격히 설정을 줄입니다.
- Monitor an external job**  
이 유형의 작업은 원격 장비처럼 Hudson 외부에서 동작하는 프로세스의 실행을 기록하는 것을 허용합니다. 그렇게 설계되어서, 기존의 자동 시스템의 대시보드로서 Hudson을 사용할 수 있습니다. 자세한 설명은 [여기\(영문\)](#)를 보세요.
- Build multi-configuration project**  
다양한 환경에서의 테스트, 플래폼 특성 빌드, 기타 등등처럼 다수의 서로다른 환경설정이 필요한 프로젝트에 적합함.
- 기존 작업 복사**  
Copy from

**빌드 대기 목록**

빌드 대기 항목이 없습니다.

**빌드 실행 상태**

| # | 상태   |
|---|------|
| 1 | 대기 중 |
| 2 | 대기 중 |

Hudson » [Software](#)

- [Back to Dashboard](#)
- [Status](#)
- [Changes](#)
- [Workspace](#)
- [Build Now](#)
- [Delete Project](#)
- [Configure](#)

**Build Triggers**

- Build after other projects are built
- Poll SCM
- Build periodically

**Build**

- Invoke Ant**

Ant Version

Targets

Build File

**Post-build Actions**

- Publish Javadoc
- Aggregate downstream test results
- Publish JUnit test result report
- Archive the artifacts
- Build other projects
- Publish Cobertura Coverage Report
- Publish testing tools result report
- Email Notification
- Perform Subversion tagging on successful build

- 다른 작업 build 후
- 주기별 소스의
- 변경이 있는 경우
- 일정한 시간 주기

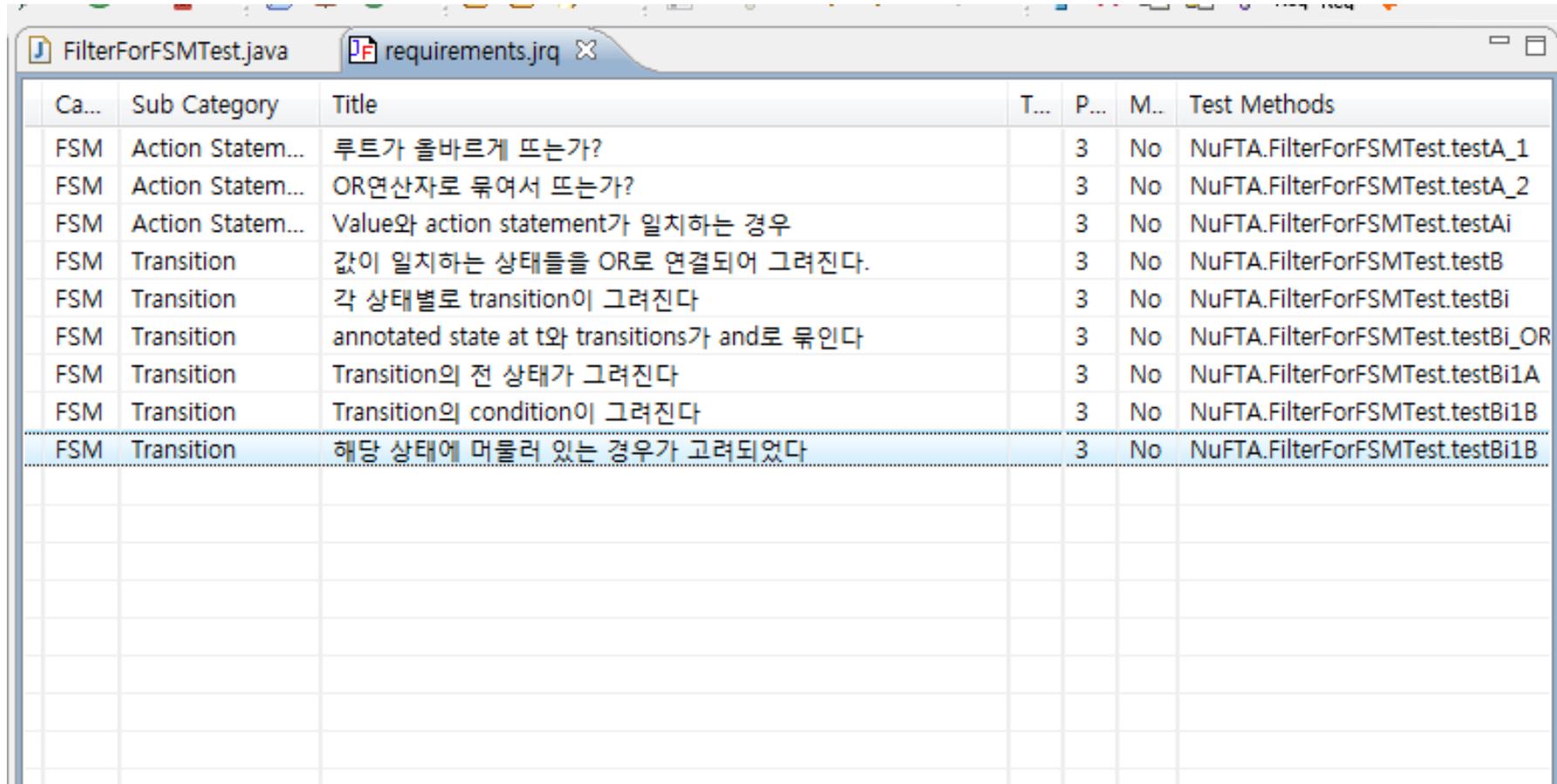
# Unit Test “NuSRS”

Requirement 1.  
Generated Annotated FSM

---

# Generate Fault Tree(FSM)

## ❖ 요구사항 작성

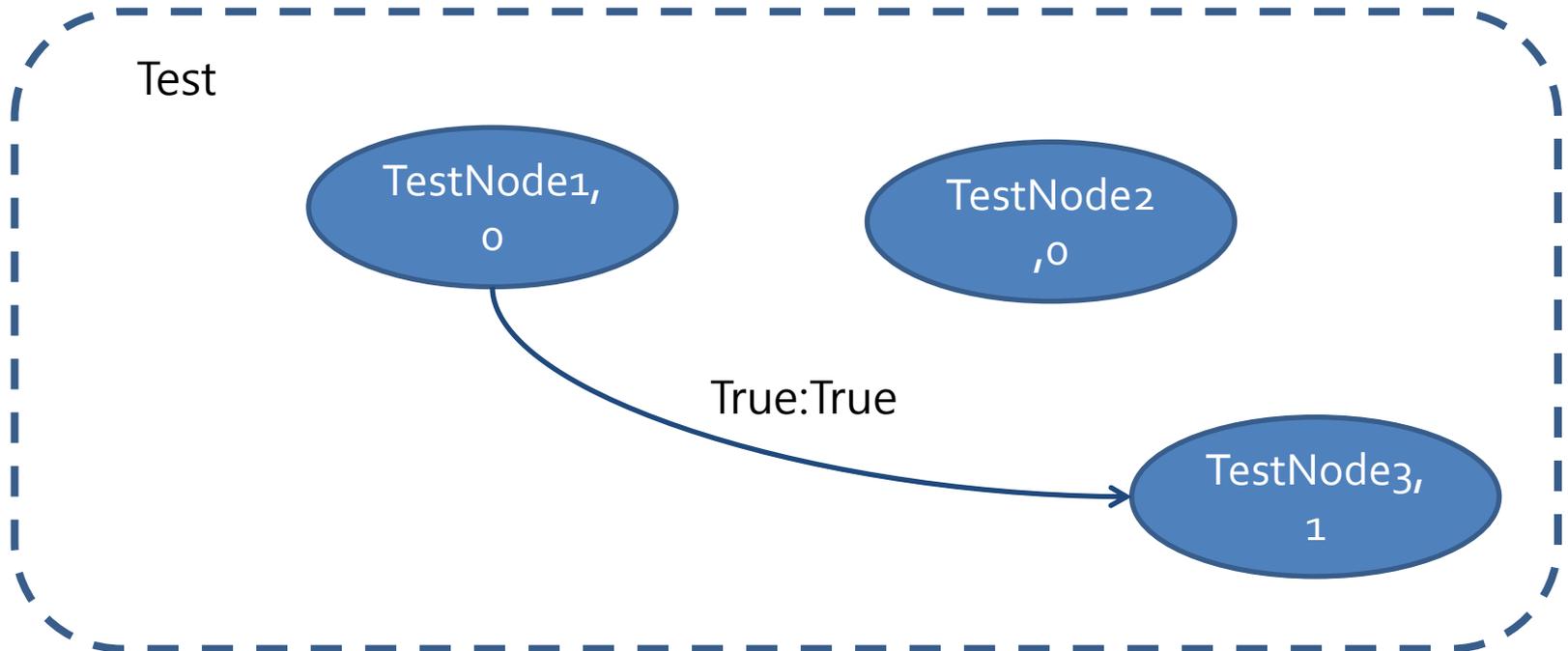


| Ca... | Sub Category     | Title                                       | T... | P... | M... | Test Methods                     |
|-------|------------------|---|------|------|------|----------------------------------|
| FSM   | Action Statem... | 루트가 올바르게 뜨는가?                               |      | 3    | No   | NuFTA.FilterForFSMTest.testA_1   |
| FSM   | Action Statem... | OR연산자로 묶어서 뜨는가?                             |      | 3    | No   | NuFTA.FilterForFSMTest.testA_2   |
| FSM   | Action Statem... | Value와 action statement가 일치하는 경우            |      | 3    | No   | NuFTA.FilterForFSMTest.testAi    |
| FSM   | Transition       | 값이 일치하는 상태들을 OR로 연결되어 그려진다.                 |      | 3    | No   | NuFTA.FilterForFSMTest.testB     |
| FSM   | Transition       | 각 상태별로 transition이 그려진다                     |      | 3    | No   | NuFTA.FilterForFSMTest.testBi    |
| FSM   | Transition       | annotated state at t와 transitions가 and로 묶인다 |      | 3    | No   | NuFTA.FilterForFSMTest.testBi_OR |
| FSM   | Transition       | Transition의 전 상태가 그려진다                      |      | 3    | No   | NuFTA.FilterForFSMTest.testBi1A  |
| FSM   | Transition       | Transition의 condition이 그려진다                 |      | 3    | No   | NuFTA.FilterForFSMTest.testBi1B  |
| FSM   | Transition       | 해당 상태에 머물러 있는 경우가 고려되었다                     |      | 3    | No   | NuFTA.FilterForFSMTest.testBi1B  |

# Generate Fault Tree(FSM)

## ❖ Set Up 코드

- 테스트에 사용하기 위한 간단한 형태의 FSM
- FSM : Node 3, Transition 1



# Generate Fault Tree(FSM)

## ❖ Set Up 코드

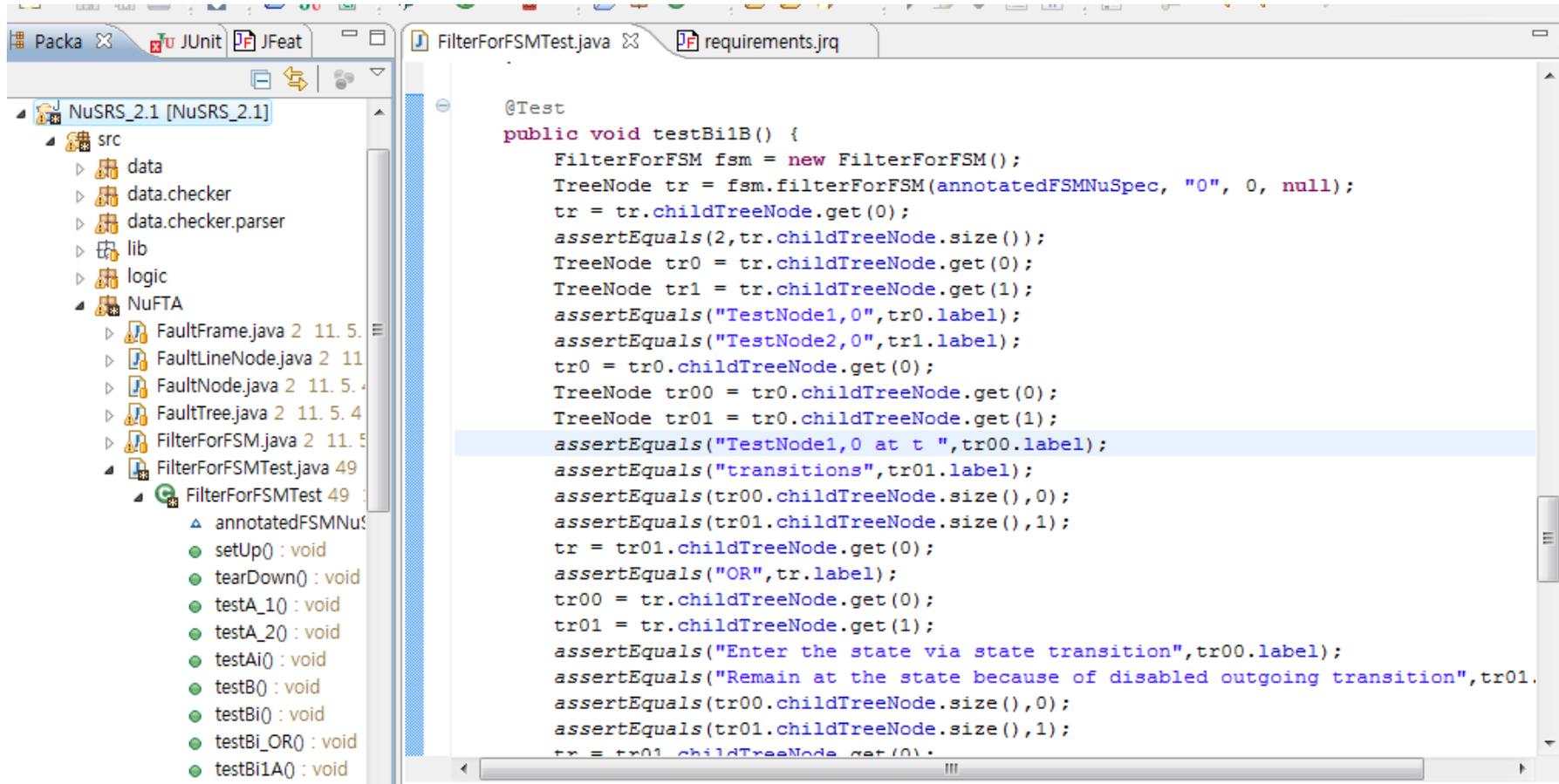
- setUp 함수는 매 Test 시작전에 호출되는 함수
- 테스트 전에, 다음의 소스코드로 FSM 생성

```
@Before
public void setUp() throws Exception {
    annotatedFSMNuSpec = new FSM("Test");
    NuNode node1 = new NuNode("TestNode1,0");
    ((FSM) annotatedFSMNuSpec).addNode(node1);
    ((FSM) annotatedFSMNuSpec).addInput(node1);
    NuNode node2 = new NuNode("TestNode2,0");
    ((FSM) annotatedFSMNuSpec).addNode(node2);
    NuNode node3 = new NuNode("TestNode3,1");
    ((FSM) annotatedFSMNuSpec).addNode(node3);
    FSMTransition transition1 = new FSMTransition(node1,node3);

    transition1.setConditions("true := true");
    ((FSM) annotatedFSMNuSpec).addTransition(transition1);
}
@After
public void tearDown() throws Exception {
    //System.out.println("@After");
}
```

# Generate Fault Tree(FSM)

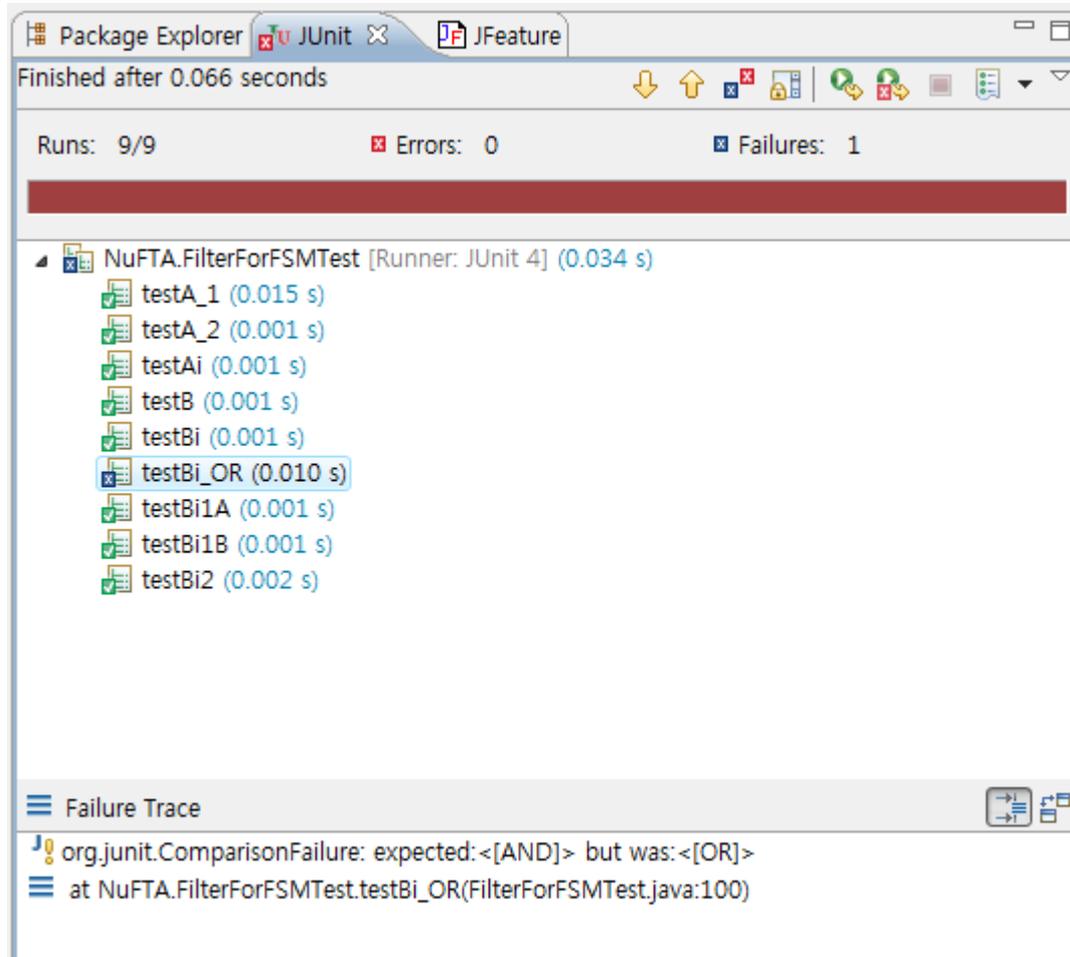
## ❖ 테스트케이스 작성



```
@Test
public void testBi1B() {
    FilterForFSM fsm = new FilterForFSM();
    TreeNode tr = fsm.filterForFSM(annotatedFSMNuSpec, "0", 0, null);
    tr = tr.childTreeNode.get(0);
    assertEquals(2, tr.childTreeNode.size());
    TreeNode tr0 = tr.childTreeNode.get(0);
    TreeNode tr1 = tr.childTreeNode.get(1);
    assertEquals("TestNode1,0", tr0.label);
    assertEquals("TestNode2,0", tr1.label);
    tr0 = tr0.childTreeNode.get(0);
    TreeNode tr00 = tr0.childTreeNode.get(0);
    TreeNode tr01 = tr0.childTreeNode.get(1);
    assertEquals("TestNode1,0 at t ", tr00.label);
    assertEquals("transitions", tr01.label);
    assertEquals(tr00.childTreeNode.size(), 0);
    assertEquals(tr01.childTreeNode.size(), 1);
    tr = tr01.childTreeNode.get(0);
    assertEquals("OR", tr.label);
    tr00 = tr.childTreeNode.get(0);
    tr01 = tr.childTreeNode.get(1);
    assertEquals("Enter the state via state transition", tr00.label);
    assertEquals("Remain at the state because of disabled outgoing transition", tr01.label);
    assertEquals(tr00.childTreeNode.size(), 0);
    assertEquals(tr01.childTreeNode.size(), 1);
    tr = tr01.childTreeNode.get(0);
}
```

# Generate Fault Tree(FSM)

## ❖ Junit 실행



Package Explorer JUnit JFeature

Finished after 0.066 seconds

Runs: 9/9 Errors: 0 Failures: 1

- NuFTA.FilterForFSMTest [Runner: JUnit 4] (0.034 s)
  - testA\_1 (0.015 s)
  - testA\_2 (0.001 s)
  - testAi (0.001 s)
  - testB (0.001 s)
  - testBi (0.001 s)
  - testBi\_OR (0.010 s)
  - testBi1A (0.001 s)
  - testBi1B (0.001 s)
  - testBi2 (0.002 s)

Failure Trace

```
org.junit.ComparisonFailure: expected:<[AND]> but was:<[OR]>  
at NuFTA.FilterForFSMTest.testBi_OR(FilterForFSMTest.java:100)
```

# Generate Fault Tree(FSM)

## ❖ JFeature 실행

Home

### All Categories

FSM (50%)

### All Requirements

루트가 올바르게 뜨는가? (100%)  
 OR연산자로 묶여서 뜨는가? (100%)  
 Value와 action st... (100%)  
 Transition의 cond... (100%)  
 값이 일치하는 상태들을 OR로... (100%)  
 Transition의 전 상태... (100%)  
 각 상태별로 transitio... (100%)  
 해당 상태에 머물러 있는 경우... (100%)  
 annotated state ... (0%)

## Requirement Coverage Report

### Requirement Coverage Summary



|                                 |      |
|---------------------------------|------|
| Number of Requirements          | 9    |
| Unique Test Methods             | 8    |
| Requirements:Test Methods Ratio | 1:1  |
| Missing Test Methods            | None |

### Requirement Coverage Details

| Sr# | Coverage Item        | Coverage                 |
|-----|----------------------|--------------------------|
| 1.  | Action Statement (3) | 3 (100%)                 |
| 2.  | Transition (6)       | 5 (83.33%)<br>1 (16.67%) |

Report generated on 금, 06 5월 2011 01:48:06 KST



# Generate Fault Tree(FSM)

## ❖ Commit

The screenshot shows the Eclipse IDE interface. The Package Explorer on the left shows a project named 'NuSRS\_2.1' with a 'src' folder containing subfolders like 'data', 'lib', and 'logic'. The main editor window displays the code for 'FilterForFSMTest.java', which includes several assertions and tree node manipulations. The Console window at the bottom shows the output of an SVN commit command, including a conflict resolution message and the commit details for revision 48.

```
TreeNode tr0 = tr.childTreeNode.get(0);
TreeNode tr1 = tr.childTreeNode.get(1);
assertEquals("TestNode1,0", tr0.label);
assertEquals("TestNode2,0", tr1.label);
tr0 = tr1.childTreeNode.get(0);
//assertEquals("AND", tr.label);
TreeNode tr00 = tr0.childTreeNode.get(0);
TreeNode tr01 = tr0.childTreeNode.get(1);
assertEquals("TestNode2,0 at t", tr00.label);
assertEquals("transitions", tr01.label);
assertEquals(tr00.childTreeNode.size(), 0);
assertEquals(tr01.childTreeNode.size(), 1);
tr = tr01.childTreeNode.get(0);
assertEquals("OR", tr.label);
tr00 = tr.childTreeNode.get(0);
```

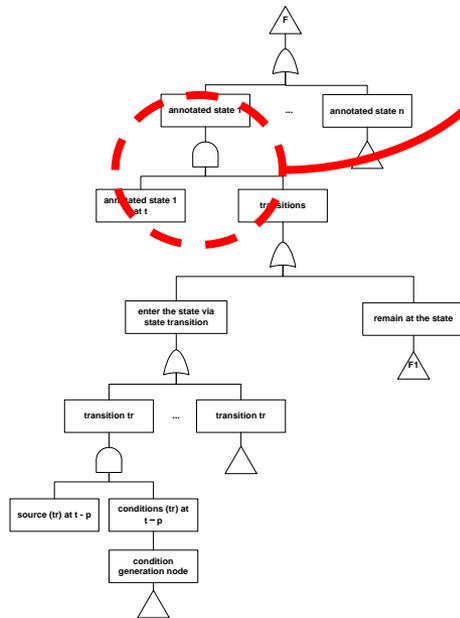
```
SVN
adding      D:/data/workspace/NuSRS_2.1/TEST-logic.stringToLogicTest.xml
Transmitting file data ...
Transaction is out of date
svn: Commit failed (details follow):
svn: 파일 '/NuSRS_2.1/TEST-data.checker.parser.TokenMgrErrorTest.xml' (이)가 오래되었습니다

update D:/data/workspace/NuSRS_2.1/TEST-data.checker.parser.TokenMgrErrorTest.xml -r HEAD --force
C      D:/data/workspace/NuSRS_2.1/TEST-data.checker.parser.TokenMgrErrorTest.xml
Updated to revision 47.
==== Conflict Statistics: ====
File conflicts: 1
commit -m "" D:/data/workspace/NuSRS_2.1/TEST-NuFTA.FilterForFSMTest.xml D:/data/workspace/NuSRS_2.1/TEST-data.checker.pars
Adding      D:/data/workspace/NuSRS_2.1/TEST-NuFTA.FilterForFSMTest.xml
Sending     D:/data/workspace/NuSRS_2.1/TEST-data.checker.parser.TokenMgrErrorTest.xml
Adding      D:/data/workspace/NuSRS_2.1/TEST-logic.stringToLogicTest.xml
Transmitting file data ...
Committed revision 48.
```

# Generate Fault Tree(FSM)

## ❖ Requirements

### ■ 테스트 실패



실패한 테스트 케이스 : annotated state at t와 transitions가 and로 묶인다 (이부분이 실제로는 OR로 묶이는걸 확인함.)

### ■ 그 외의 모든 테스트 케이스는 통과

# Unit Test “NuSRS”

Requirement 2.  
Analyze Logic by expression

---

# Analyze Logic by expression

- ❖ 입력받은 수식을 논리적 구조로 분석
- ❖ 수식에 사용되는 기호
  - |, &, +, -, =, !, <, >, <=, >=, :=, (, )
- ❖ 수식
  - 연산자와 변수로 나누어짐 ( A=ture 형태)
  - 불값을 나타냄
  - 수식 내 '(, )' 가 발생한다면 우선 확인

# Analyze Logic by expression

## ❖ logic.stringToLogic.java

- 입력받은 수식 문자열 s를 트리 형태로 Parsing하는 파일

```
public class stringToLogic {  
  
    public String fullString;  
    public ArrayList<logicMember> lm = new ArrayList<logicMember>();  
    public ArrayList<treeMember> tm = new ArrayList<treeMember>();  
    // protected ArrayList<subList> sl = new ArrayList<subList>();  
    protected int numNot=0;  
    protected int maxLevel=0;
```

```
    public stringToLogic(String s)
```

```
        //logicMember setting  
        fullString = s;  
        s = s.trim();
```

```
        makeMember(s, 0, 0);  
        setTruth();  
        removeNot();  
        removeBracket();  
        removeEnter();  
        convertTruth();  
        setSubElement();
```

```
        //treeMember setting  
        copyLM();  
        setLevelwithPriority();  
        setLevelwithGate();  
        getMaxLevel();  
        setChild(0);  
        removeDuplication();
```

```
    }
```

생성자에서 변환 과정 수행

| Require... | Category | Sub Category | Title                | T... | Priority | Must Have | Test Methods                                     | Depx |
|------------|----------|--------------|----------------------|------|----------|-----------|--|------|
| R2_001     | Parse    | logicMember  | makeMember           |      | 3        | No        | logic.stringToLogicTest.testMakeMember           |      |
| R2_002     | Parse    | logicMember  | setTruth             |      | 3        | No        | logic.stringToLogicTest.testSetTruth             |      |
| R2_003     | Parse    | logicMember  | removeNot            |      | 3        | No        | logic.stringToLogicTest.testRemoveNot            |      |
| R2_004     | Parse    | logicMember  | removeBracket        |      | 3        | No        | logic.stringToLogicTest.testRemoveBracket        |      |
| R2_005     | Parse    | logicMember  | removeEnter          |      | 3        | No        | logic.stringToLogicTest.testRemoveEnter          |      |
| R2_006     | Parse    | logicMember  | convertTruth         |      | 3        | No        | logic.stringToLogicTest.testConvertTruth         |      |
| R2_007     | Parse    | logicMember  | setSubElement        |      | 3        | No        | logic.stringToLogicTest.testSetSubElement        |      |
| R2_008     | Parse    | logicMember  | copyLM               |      | 3        | No        | logic.stringToLogicTest.testCopyLM               |      |
| R2_009     | Parse    | treeMember   | setLevelwithPriority |      | 3        | No        | logic.stringToLogicTest.testSetLevelwithPriority |      |
| R2_010     | Parse    | treeMember   | setLevelwithGate     |      | 3        | No        | logic.stringToLogicTest.testSetLevelwithGate     |      |
| R2_011     | Parse    | treeMember   | getMaxLevel          |      | 3        | No        | logic.stringToLogicTest.testGetMaxLevel          |      |
| R2_012     | Parse    | treeMember   | setChild             |      | 3        | No        | logic.stringToLogicTest.testSetChild             |      |
| R2_013     | Parse    | treeMember   | removeDuplication    |      | 3        | No        | logic.stringToLogicTest.testRemoveDuplication    |      |

# Analyze Logic by expression

예) (f\_SG1\_LO\_FLOW\_Val\_Out\_t1 -f\_SG1\_LO\_FLOW\_Val\_Out <=k\_SG1\_LO\_FLOW\_100ms\_Rate) &  
(f\_SG1\_LO\_FLOW\_Val\_Out -k\_SG1\_LO\_FLOW\_Trip\_Step >=k\_SG1\_LO\_FLOW\_SP\_Lo\_Lim) &  
(f\_SG1\_LO\_FLOW\_Val\_Out -k\_SG1\_LO\_FLOW\_Trip\_Step <=k\_SG1\_LO\_FLOW\_SP\_Hi\_Lim)

## 문자열 변환 과정

- Step 1. 복합문을 쪼갬다 : &, !, (, ), | 로 나누기 - makeMember
- Step 2. !가 미치는 구문을 찾아내어 상쇄시켜 중복 !연산을 최소화 - setTruth
- Step 3. Step2에서 ! 영향을 변수로 저장했기 때문에 구문에서 ! 제거 - removeNot
- Step 4. 괄호, 개행문제 제거 - removeBracket, removeEnter
- Step 5. 연산자에 대해 !을 수행 - convertTruth
- Step 6. 나누어진 구문들을 비교 연산자를 기준으로 나눔 - setSubElement
- Step 7. +, - 연산자를 기준으로 나눔 - divideSub

|                          |   |    |                           |      |
|--------------------------|---|----|---------------------------|------|
| f_SG1_LO_FLOW_Val_Out_t1 | / | -  | /f_SG1_LO_FLOW_Val_Out    | / <= |
| k_SG1_LO_FLOW_100ms_Rate | / | &  | /f_SG1_LO_FLOW_Val_Out    | / -  |
| k_SG1_LO_FLOW_Trip_Step  | / | >= | /k_SG1_LO_FLOW_SP_Lo_Lim) | / &  |
| f_SG1_LO_FLOW_Val_Out    | / | -  | /k_SG1_LO_FLOW_Trip_Step  | / <= |
| k_SG1_LO_FLOW_SP_Hi_Lim) |   |    |                           |      |

# Analyze Logic by expression

❖ void makeMember(String input, int not, int level)

■ 입력받은 수식을 구분자 &, !, (, ), | 로 나누어 Array에 저장

```
@Test
public void testMakeMember() {
    STL.lm.clear();
    STL.makeMember(STL.fullString, 0, 0);
    int count = STL.fullString.length();
    int index=0, level=0;
    char ch;
    boolean exp = false;
    for(int i=0; i<count ; i++ )
    {
        ch = STL.fullString.charAt(i);
        if(ch== ' ')
            continue;
        if( (ch=='&') || (ch=='!') ||(ch=='(') ||(ch==')') || (ch=='\\') )
        {
            if(ch == '(')    level++;
            if(ch==')')    level--;

            assertEquals( ch+"", STL.lm.get(index).element );
            assertTrue( STL.lm.get(index).truth);
            assertEquals( level, STL.lm.get(index).level);
            index++;
            exp=false;
        }else{
            if(exp==false){
                exp = true;
                index++;
            }
        }
    }
}
```

# Analyze Logic by expression

## ❖ void setTruth()

- '!'가 영향을 미치는 요소를 연산
- 영향받는 '!'의 수가 홀수라면 요소는 false
- assertFalse, assertTrue 로 검사

```
for(logicMember lm : this.lm)
{
    if((lm.numNot%2)==1)
        lm.truth = false;
}
```



```
@Test
public void testSetTruth() {
    for(logicMember lm : STL.lm)
    {
        if((lm.numNot%2)==1)
            assertFalse(lm.truth);
        else
            assertTrue(lm.truth);
    }
}
```

# Analyze Logic by expression

## ❖ void removeNot()

- 수식 요소 중 '!'를 제거
- 수식 요소 중 '!' 존재 여부를 검사

```
@Test
public void testRemoveNot() {
    int count = STL.lm.size();

    for(int i=0; i<count; i++)
        assertFalse(STL.lm.get(i).element.contains("!"));
}
```

## ❖ void removeBracket()

## ❖ void removeEnter()

- removeNot과 동일한 방식으로 검사

# Analyze Logic by expression

## ❖ void convertTruth()

- 앞에서 setTruth 함수로 false로 바꾼 요소(!를 갖던 요소)를 true로 변경 & 내부 연산자를 ! 연산
- 여전히 남은 false가 존재하는 검사

```
@Test
public void testConvertTruth() {
    for (logicMember lm : STL.lm)
        assertTrue(lm.truth);
}
```

## ❖ void setSubElement()

- 요소 중 비교 연산자(>=, >, <=, <, :=, !=, =)가 있다면 연산자 기준으로 좌 / 우로 나누어 divideSub 함수 수행

# Analyze Logic by expression

## ❖ void setSubElement()

- 요소 중 비교 연산자(>=, >, <=, <, :=, !=, =)가 있다면 연산자 기준으로 좌 / 우로 나누어 divideSub 함수 수행 및 연산자 위치에 sign 할당

```
@Test
public void testSetSubElement() {
```

```
    stringToLogic STL2 = new stringToLogic( STL.fullString);
    STL2.makeMember(STL.fullString, 0, 0);
    STL2.setTruth();
    STL2.removeNot();
    STL2.removeBracket();
    STL2.removeEnter();
    STL2.convertTruth();
```

```
    int count = STL2.fullString.length();
    int index=0;
    char ch;
    String sign;
    boolean exp = false;
    for(int i=0; i<count ; i++ )
    {
        ch = STL2.lm.toString().charAt(i);
        if(ch== ' ')
            continue;
        if( (ch=='>') || (ch=='<') ||(ch==':') ||(ch=='!') || (ch=='=' ) )
        {
            if(STL2.fullString.charAt(i+1)=='=')
                sign = ch+"=";
            else
                sign = ch+"";

            assertEquals( sign, STL.lm.get(++index).sign);
            exp=false;
        }else{
            if(exp==false){
                exp = true;
                index++;
            }
        }
    }
}
```

# Analyze Logic by expression

## ❖ void copyLM()

- 앞에서 수식을 Parsing 하여 만든 logic 타입을 child를 갖는 tree 타입으로 복사

```
@Test
public void testCopyLM() {
    STL.tm.clear();
    STL.copyLM();
    int count = STL.lm.size();
    for(int i=0; i<count; i++)
    {
        assertEquals(STL.lm.get(i).element, STL.tm.get(i).element);
        assertEquals(STL.lm.get(i).level, STL.tm.get(i).level);
        assertTrue(STL.lm.get(i).truth);
        assertEquals(STL.lm.get(i).leftSubElement, STL.tm.get(i).leftSubElement);
        assertEquals(STL.lm.get(i).sign, STL.tm.get(i).sign);
        assertEquals(STL.lm.get(i).rightSubElement, STL.tm.get(i).rightSubElement);
    }
}
```

# Analyze Logic by expression

## ❖ JFeature

Home

### All Categories

Parse (50%)

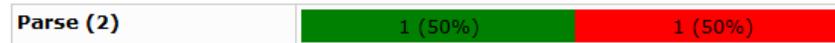
---

### All Requirements

makeMember (100%)  
setTruth (100%)  
removeNot (100%)  
removeBracket (100%)  
removeEnter (100%)  
convertTruth (100%)  
setSubElement (100%)

## Requirement Coverage Report

### Requirement Coverage Summary



|                                 |      |
|---------------------------------|------|
| Number of Requirements          | 13   |
| Unique Test Methods             | 13   |
| Requirements:Test Methods Ratio | 1:1  |
| Missing Test Methods            | None |

### Requirement Coverage Details

| Sr# | Coverage Item   | Coverage |
|-----|-----------------|----------|
| 1.  | logicMember (8) | 8 (100%) |
| 2.  | treeMember (5)  | 5 (100%) |

## 테스트 결과 & 소감

- 문자열을 Parsing하여 의미 있는 단위로 나눈 단계 테스트 완료(검증 완료)
- 계층적 Tree 형태로 변환하는 테스트 케이스 작성은 실패(미검증)
- 입력값을 무조건 받는 생성자에서 모든 연산을 한번에 연산하며 void 형태의 함수로 인해 테스트케이스 작성을 위한 어려움이 발생
- 테스트를 고려한 코드 작성법도 필요



**END**