



# Introduction to Static Analysis & System Test Tool

200611494 원스타  
200810047 김성원  
200811466 허태경

# Index

## 1. Static Analysis

1. JDepend
2. Findbugs
3. PMD
4. Sonar Qube
5. Checkstyle
6. Clang-Analyzer
7. 설치 및 연동 결과

## 2. System Test

1. Requirement Management
  2. Category-Partition Testing
  3. Pairwise Testing
  4. Pairwise Testing Sequence Generating Tools
  5. Requirement Management Tool
  6. JFeature 연동
- ## 3. Trouble Shooting



# 1. Static Analysis

정적 분석 도구



# 1. Static Analysis

- ▶ 프로그램을 직접 실행하지 않고, 소스 코드 혹은 오브젝트 코드를 분석해 잠재적 결함이나 구조상 문제를 찾는 방법.
- ▶ 주로 자동화된 툴을 사용한다.
- ▶ 장점
  - ▶ 테스트만으로 발견하기 힘든, 잠재적인 결함을 발견
  - ▶ 프로그램을 직접 실행해 보기 힘든 경우 유용
    - ▶ ex) safety-critical system: 의료기기 소프트웨어, 원자력 발전소 제어 소프트웨어 등
- ▶ 문제점
  - ▶ 실제 프로그램을 실행해 보지 않기 때문에, False Alarm이 발생할 수 있음
    - ▶ ex) 로직상 실제로 실행되지 않는 플로우에 대해 분석 후 경고

# 1. Static Analysis

## JDepend

- ▶ 자바 패키지간의 의존성을 검사하여 디자인의 품질을 측정해 준다.
  - ▶ 확장성, 재사용성, 유지보수성 등 factor
- ▶ 각 패키지에 대해 Design quality metrics 제공
- ▶ 특히, 순환 의존성 확인 등에 유용
  - ▶ 단, 모든 순환 의존성 사이클을 한 번에 찾아 주지 않는다.
- ▶ 소스 코드의 복잡도를 측정하지는 않는다.

# 1. Static Analysis

JDepend

## Design Quality Metrics

CC, AC	클래스나 추상 클래스의 갯수
CA (Afferent Couplings)	패키지 내의 클래스에 종속성을 가지는 패키지 갯수. 패키지의 책임 (responsibility)을 나타냄
Ce (Efferent Couplings)	패키지 내의 클래스가 종속하는 패키지 갯수. 패키지의 독립성 (independence)를 나타냄
A (Abstractness)	패키지 내에서 AC/Interface의 갯수 비율. 높을수록 추상화된 것
I (Instability)	클래스 구현의 변경 가능성을 나타냄. $Ce / Ce + Ca$
D (Distance)	Main Sequence ( $A + I = 1$ 직선)로부터 거리. 추상화 정도와 불안정성 간의 밸런스를 나타냄
Package Dependency Cycles	패키지간의 순환 종속성을 나타냄

# 1. Static Analysis

## FindBugs

- ▶ Java의 오브젝트 코드를 분석하여
- ▶ 자주 발견되는 버그의 패턴을 찾아냄
- ▶ 플러그인 구조로 새로운 패턴을 추가할 수도 있음.
- ▶ Eclipse에 붙일 수 있고, 별도 GUI도 제공



# 1. Static Analysis

## FindBugs

The screenshot displays the Eclipse IDE interface with FindBugs integrated. On the left, the Bug Explorer shows a list of detected issues, with 'Possible null pointer dereference of bundleGroup' selected. The main editor window shows the source code of `AboutFeaturesPage.java`, where a red bug icon is placed over the line `bundleGroup.getIdentifier()`. The Properties window at the bottom provides details for this bug:

**Bug: Possible null pointer dereference of bundleGroup**  
Pattern id: NP\_NULL\_ON\_SOME\_PATH, type: NP, category: CORRECTNESS

There is a branch of statement that, if executed, guarantees that a null value will be dereferenced, which would generate a `NullPointerException` when the code is executed. Of course, the problem might be that the branch or statement is infeasible and that the null pointer exception can't ever be executed; deciding that is beyond the ability of FindBugs.



# 1. Static Analysis

## FindBugs

### FindBugs Result

#### Warnings Trend

All Warnings	New Warnings	Fixed Warnings
300	0	0

#### Summary

Total	High Priority	Normal Priority	Low Priority
300	28	91	181

#### Module Statistics

Module	Total	Distribution
<a href="#">Action Call Editor</a>	4	
<a href="#">SQL Plus console</a>	28	
<a href="#">UT PLSql Tests</a>	53	
<a href="#">PLDoc</a>	29	
<a href="#">AST View</a>	6	
<a href="#">Action Call Core</a>	31	
<a href="#">Avafactory UI</a>	55	
<a href="#">PLDoc Tests</a>	1	
<a href="#">Action Call Tests</a>	1	
<a href="#">UT PLSql</a>	24	
<a href="#">ENV Win Diff</a>	8	
<a href="#">SQL Plus Tests</a>	10	
<a href="#">Avafactory Core</a>	12	
<a href="#">UT PLSql UI</a>	38	

# 1. Static Analysis

## PMD

- ▶ Java 소스 코드를 분석해 잠재적 문제점을 탐색
  - ▶ 사용되지 않는 코드, 중복 코드, 비효율적인 코드 등
- ▶ 룰 세트 기반으로 동작
- ▶ 사용자 룰을 정의해서 검사 가능



# 1. Static Analysis

## PMD

```
<!-- We'll use the entire 'strings' ruleset -->
<rule ref="rulesets/java/strings.xml"/>

<!-- Here's some rules we'll specify one at a time -->
<rule ref="rulesets/java/unusedcode.xml/UnusedLocalVariable"/>
<rule ref="rulesets/java/unusedcode.xml/UnusedPrivateField"/>
<rule ref="rulesets/java/imports.xml/DuplicateImports"/>
<rule ref="rulesets/java/basic.xml/UnnecessaryConversionTemporary"/>

<!-- We want to customize this rule a bit, change the message and raise the priority -->
<rule
  ref="rulesets/java/basic.xml/EmptyCatchBlock"
  message="Must handle exceptions">
  <priority>2</priority>
</rule>

<!-- Now we'll customize a rule's property value -->
<rule ref="rulesets/java/codesize.xml/CyclomaticComplexity">
  <properties>
    <property name="reportLevel" value="5"/>
  </properties>
</rule>

<!-- We want everything from braces.xml except WhileLoopsMustUseBraces -->
<rule ref="rulesets/java/braces.xml">
```

Example Ruleset

# 1. Static Analysis

## Sonar Qube

- ▶ 코드 품질을 관리할 수 있는 오픈 플랫폼
- ▶ 7가지 코드 품질 요소를 커버
  - ▶ 아키텍처와 디자인
  - ▶ 코드 중복
  - ▶ 유닛 테스트
  - ▶ 코드 복잡도
  - ▶ 잠재적 버그
  - ▶ 코딩 룰
  - ▶ 주석
- ▶ 여러가지 프로그래밍 언어를 지원. (Java, C/C++, C# 등)
- ▶ Standalone 형태로 실행하거나, 다른 툴에 연계해 사용 가능

# 1. Static Analysis

## Checkstyle

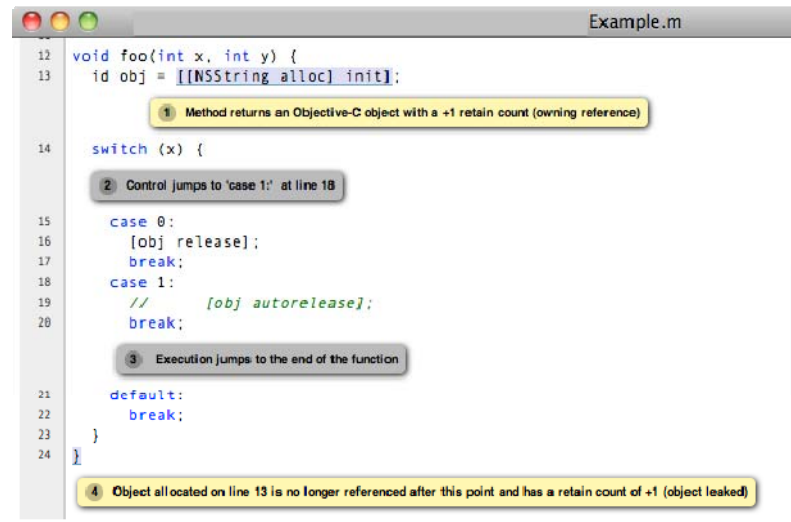
- ▶ 코드 스타일을 준수하도록 도와주는 툴
  - ▶ 코드 레이아웃, 클래스 / 메서드 / 필드 이름 규칙 등
- ▶ 설정 파일을 통해 여러가지 스타일로 맞춤 수 있음
  - ▶ ex) Sun Java Code Conventions, Google Java Style 등

The logo for Checkstyle, featuring the word "checkstyle" in a bold, lowercase, sans-serif font. The letter 'e' is stylized as a yellow pencil with a pink eraser. A red wavy line underlines the text.

# 1. Static Analysis

## Clang-Analyzer

- ▶ C++용 정적 분석 툴
- ▶ C++ 컴파일러 Clang을 그대로 정적 분석도구로 활용



```
12 void foo(int x, int y) {
13     id obj = [[NSString alloc] init];
14     switch (x) {
15         case 0:
16             [obj release];
17             break;
18         case 1:
19             // [obj autorelease];
20             break;
21         default:
22             break;
23     }
24 }
```

1 Method returns an Objective-C object with a +1 retain count (owning reference)

2 Control jumps to 'case 1' at line 18

3 Execution jumps to the end of the function

4 Object allocated on line 13 is no longer referenced after this point and has a retain count of +1 (object leaked)

# 1. Static Analysis

## 설치 및 연동 결과



### FindBugs™ - Find Bugs in Java Programs

This is the web page for FindBugs, a program which uses static analysis to look for bugs in Java code under the terms of the [Lesser GNU Public License](#). The name FindBugs™ and the [FindBugs logo](#) are of [Maryland](#). FindBugs has been downloaded more than a million times.

The current version of FindBugs is 3.0.1.

FindBugs requires JRE (or JDK) 1.7.0 or later to run. However, it can analyze programs compiled for 1.8.

The current version of FindBugs is 3.0.1, released on 13:05:33 EST, 06 March, 2015. [We are very into how to improve FindBugs](#). File bug reports on [our sourceforge bug tracker](#)

[Changes](#) | [Talks](#) | [Papers](#) | [Sponsors](#) | [Support](#)

### FindBugs 3.0.1 Release

- A number of changes described in the [changes document](#), including new bug patterns:
  - `BSHIFT_WRONG_ADD_PRIORITY`.

**Docs and Info**

- FindBugs 2.0
- Demo and data
- Users and supporters
- FindBugs blog
- Fact sheet
- Manual
- Manual(日本語)
- FAQ
- Bug descriptions
- Bug descriptions(日本語)
- Bug descriptions(中文)
- Mailing lists
- Documents and Publications
- Links

**Downloads**




### PMD

DON'T SHOOT THE MESSENGER

PMD is a source code analyzer. It finds common programming flaws like unused variables, empty catch blocks, unnecessary object creation, and so forth. It supports Java, JavaScript, PL/SQL, Apache Velocity, XML, XSL. Additionally it includes CFD, the copy-paste-selector. CFD finds duplicated code in Java, C, C++, C#, PHP, Ruby, Fortran, JavaScript, PL/SQL, Apache Velocity, Ruby, Scala, Objective C, Matlab, Python, Go.

Latest version	Latest version
Get Involved	<b>5.3.0 (1st April 2015)</b>
Plugins	<ul style="list-style-type: none"><li>Release Notes</li><li>Download (Sourcecode, Documentation)</li><li>Online Documentation</li></ul>
Recent Announcements	
Next development version	
Previous versions	

Copyright © PMD. All Rights Reserved. [sourceforge](#)



### checkstyle

Last Published: 2015-03-28 | Version: 6.5

**About**

- Checkstyle
- Release Notes
- Consulting

**Documentation**

- Configurations
- Priority Types
- Banning
- Ant Task
- Command Line
- Checks
  - Annotations
  - Block Checks
  - Class Design
  - Coding
  - Headers
  - Imports
  - JavaDoc Comments
  - Metrics
  - Miscellaneous
  - Modifiers
  - Naming Conventions
  - Regexp
  - Size Violations
  - Whitespace
- Style Configurations
  - Google's Style
  - Sun's Style

### Overview

Checkstyle is a development tool to help programmers write Java code that adheres to a coding standard. It automates the process of checking Java code to spare humans of this boring (but important) task. This makes it ideal for projects that want to enforce a coding standard.

Checkstyle is highly configurable and can be made to support almost any coding standard. An example configuration files are supplied supporting the Sun Code Conventions [is](#), Google Java Style [is](#).

A good example of a report that can be produced using Checkstyle and Maven [is](#) can be seen here [is](#).

### Important Development Changes

As of September 2013, the Checkstyle project is using Github for hosting the following:

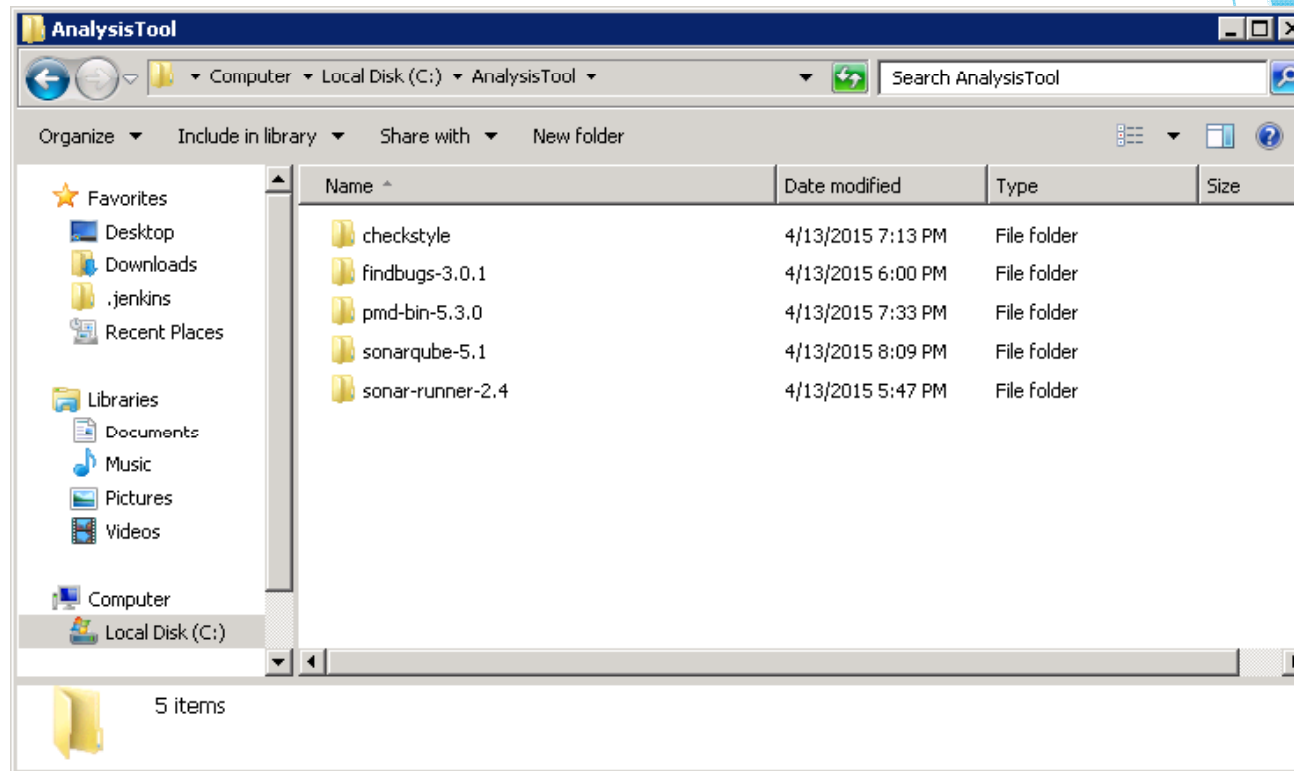
- GitHub Source code repository [is](#) - replacing the Mercurial repository on SourceForge.
- GitHub Issue management [is](#) - replacing the Bugs/Feature/Patches on SourceForge. All new issues should be raised at GitHub, and pull requests are now the preferred way to submit patches.

SourceForge will still be used for website hosting and binary hosting for downloads.

## Download Tools

# 1. Static Analysis

설치 및 연동 결과





# 1. Static Analysis

## 설치 및 연동 결과

**FindBugs Plug-in**  
This plug-in collects the [FindBugs](#) analysis results of the project modules and visualizes the found warnings. [4.60](#)  
 If you like this open source plug-in please consider supporting my work by buying my Android game [Inca Trails](#). Uninstall

**Checkstyle Plug-in**  
This plug-in collects the [Checkstyle](#) analysis results of the project modules and visualizes the found warnings. [3.42](#)  
 If you like this open source plug-in please consider supporting my work by buying my Android game [Inca Trails](#). Uninstall

**JDepend Plugin**  
This is a plugin that runs JDepend reports on builds. [1.2.4](#)  
 Uninstall

**PMD Plug-in**  
This plug-in collects the [PMD](#) analysis results of the project modules and visualizes the found warnings. [3.41](#)  
 If you like this open source plug-in please consider supporting my work by buying my Android game [Inca Trails](#). Uninstall

Install Plugins

# 1. Static Analysis

## 설치 및 연동 결과

```
▼<project basedir="." default="run" name="TestProj">
  <property environment="env" />
  <property name="checkstyle.home.dir" location="C:/AnalysisTool/checkstyle"/>
  <property name="findbugs.home.dir" value="C:/AnalysisTool/findbugs-3.0.1"/>
  <property name="pmd.home.dir" value="C:/AnalysisTool/pmd-bin-5.3.0"/>
  <property name="lib.dir" location="${basedir}/lib"/>
  <property name="src.dir" location="${basedir}/src"/>
  <property name="bin.dir" location="${basedir}/bin"/>
  <property name="htm.dir" location="${basedir}/html"/>
  <property name="report.dir" location="${basedir}/report"/>
  <property name="report.checkstyle.dir" location="${report.dir}/checkstyle"/>
  <property name="report.junit.dir" location="${report.dir}/junit"/>
  <property name="report.findbugs.dir" location="${report.dir}/findbugs"/>
  <property name="report.pmd.dir" location="${report.dir}/pmd"/>
  <property name="instrumented.dir" location="${basedir}/instrumented"/>
  <property name="report.temp.dir" location="${report.dir}/temp"/>
  <path id="run.classpath">...</path>
  <path id="pmd2.classpath">...</path>
  <taskdef name="pmd" classname="net.sourceforge.pmd.ant.PMDTask" classpathref="pmd2.classpath"/>
  <target name="report.pmd">...</target>
  <target name="prepare.report.dir" description="Prepares the reports folder">...</target>
  <target name="clean">...</target>
  <target name="compile" depends="clean" description="Compile the entire project.">...</target>
  <target name="findbugs" depends="compile" description="Run code analysis over code to check for problems.">...</target>
  <target name="report.checkstyle" description="Generate a report of code convention violations.">...</target>
  <target name="junit" depends="clean, compile" description="Run all junit test cases.">...</target>
  <target name="report.junit" depends="junit" description="Create a report for the test result.">...</target>
  <!-- -->
  <target name="report.findbugs" description="Generate a report on error analysis.">...</target>
  <target name="clean.temp" description="Delete all temporary files and folders.">...</target>
  <target name="run" description="Run the build" depends="clean, report.checkstyle, report.pmd, compile, junit, report.junit, findbugs, report.findbugs, clean.temp"></target>
</project>
```

Tool 경로 변수  
참조 및 출력 경로 변수  
Report 경로 변수

Target 지정

Build.xml에 Tool 경로 지정 및 작업 내역 지정  
(Tool 경로는 시스템 환경변수 설정으로도 가능)

# 1. Static Analysis

## 설치 및 연동 결과

### Findbugs

```
▼ <target name="findbugs" depends="compile" description="Run code analysis over code to check for problems.">
  <!-- Fail this target if FindBugs is not installed. -->
  <available file="${findbugs.home.dir}/lib/findbugs.jar" property="findbugs.available"/>
  <fail unless="findbugs.available" message="Error: FINDBUGS_HOME not set or findbugs.jar not found."/>
  <taskdef name="findbugs" classname="edu.umd.cs.findbugs.anttask.FindBugsTask" classpath="${findbugs.home.dir}/lib/findbugs-ant.jar"/>
  <!-- Run FindBugs. -->
  <mkdir dir="${report.findbugs.dir}"/>
  ▼ <findbugs home="${findbugs.home.dir}" workHard="true" output="xml:withMessages" outputFile="${report.findbugs.dir}/findbugs.xml">
    <class location="${bin.dir}"/>
    ▼ <auxClasspath>
      <fileset file="${basedir}/lib/junit-4.12.jar"/>
    </auxClasspath>
  </findbugs>
</target>
▼ <target name="report.findbugs" description="Generate a report on error analysis.">
  <xslt in="${report.findbugs.dir}/findbugs.xml" style="${findbugs.home.dir}/src/xsl/fancy.xsl" out="${report.findbugs.dir}/findbugs-default.html"/>
</target>
```

### 출력 경로와 스타일 지정

# 1. Static Analysis

## 설치 및 연동 결과

### PMD

```
▼<target name="report.pmd">
  <mkdir dir="${report.pmd.dir}"/>
  ▼<pmd rulesetfiles="${pmd.home.dir}/pmd-rulesets.xml">
    <formatter type="xml" toFile="${report.pmd.dir}/pmd_report.xml"/>
    ▼<fileset dir="${src.dir}">
      <include name="**/*.java"/>
    </fileset>
    </pmd>
    <xslt in="${report.pmd.dir}/pmd_report.xml" style="${pmd.home.dir}/pmd-report-per-class.xslt" out="${report.pmd.dir}/pmd_report.html"/>
  </target>
```

rulesets, 출력 스타일, 출력 html과 xml 경로 지정  
(rulesets과 style은 스스로 작성하거나 작성된 품을 구해야 함)

# 1. Static Analysis

## 설치 및 연동 결과

### Checkstyle

```
▼<target name="report.checkstyle" description="Generate a report of code convention violations.">
  <taskdef resource="checkstyletask.properties" classpath="${checkstyle.home.dir}/checkstyle-6.5-all.jar"/>
  <!-- run verification of installation -->
  <available file="${checkstyle.home.dir}/checkstyle-6.5-all.jar" property="checkstyle.available"/>
  <fail unless="checkstyle.available" message="Error: CHECKSTYLE_HOME not set or checkstyle-6.5-all.jar not found."/>
  <mkdir dir="${report.checkstyle.dir}"/>
  <!-- run analysis -->
  ▼<checkstyle config="${checkstyle.home.dir}/sun_checks.xml" failureProperty="checkstyle.failure" failOnViolation="false">
    <formatter type="xml" tofile="${report.checkstyle.dir}/checkstyle_report.xml"/>
    <fileset dir="src" includes="**/*.java"/>
  </checkstyle>
  <style in="${report.checkstyle.dir}/checkstyle_report.xml" out="${report.checkstyle.dir}/checkstyle_report.html" style="${checkstyle.home.dir}/checkstyle-noframes-sorted.xsl"/>
</target>
```

검사 룰과 출력 스타일, 출력 경로를 지정  
(PMD와 마찬가지로 검사 및 출력 스타일을 직접 지정 해야함)  
(구글 스타일로 지정함)

# 1. Static Analysis

## 설치 및 연동 결과

☰ Publish Checkstyle analysis results ?

Checkstyle results

[Fileset includes](#) setting that specifies the generated raw CheckStyle XML report files, such as `**/checkstyle-result.xml`. Basedir of the fileset is [the workspace root](#). If no value is set, then the default `**/checkstyle-result.xml` is used. Be sure not to include any non-report files into this pattern.

고급...

삭제

☰ Publish FindBugs analysis results ?

FindBugs results

[Fileset includes](#) setting that specifies the generated raw FindBugs XML report files, such as `**/findbugs.xml` or `**/findbugsXml.xml`. Basedir of the fileset is [the workspace root](#). If no value is set, then the default `**/findbugsXml.xml` or `**/findbugs.xml` are used for maven or ant builds, respectively. Be sure not to include any non-report files into this pattern.

Use rank as priority

Uses the bug rank when evaluating the priority of the warnings (otherwise the FindBugs priority is used).

고급...

삭제

☰ Publish PMD analysis results ?

PMD results

[Fileset includes](#) setting that specifies the generated raw PMD XML report files, such as `**/pmd.xml`. Basedir of the fileset is [the workspace root](#). If no value is set, then the default `**/pmd.xml` is used. Be sure not to include any non-report files into this pattern.

고급...

삭제

☰ Report JDepend

Pre-generated JDepend File

Provide a path to a JDepend file created during the build.  
Use a preceding `"/` to specify an absolute path, leave off the `"/` to specify a path within the workspace.  
Leave blank to have the plugin generate its own file.

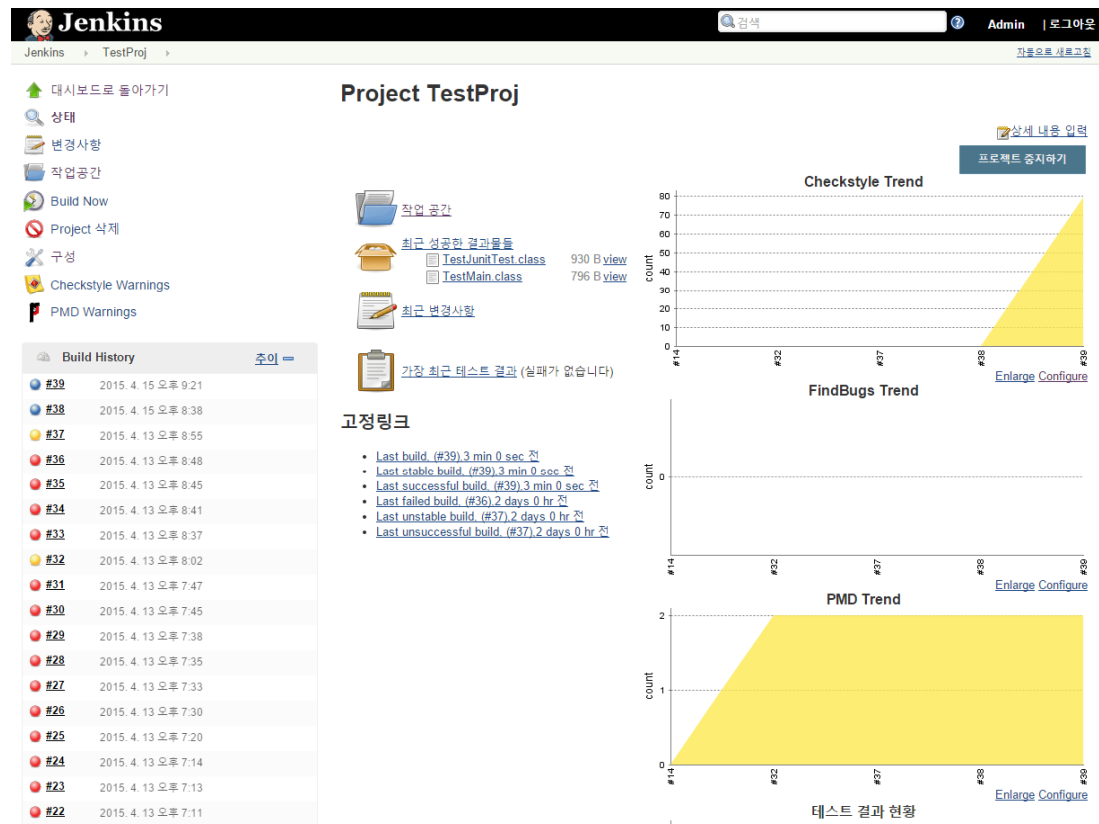
삭제

출력된 레포트를 굵어올 경로와  
파일형식을 지정

JDepend는 plugin 지정만으로도  
돌아감

# 1. Static Analysis

설치 및 연동 결과



Jenkins 빌드 결과

# 1. Static Analysis

## 설치 및 연동 결과

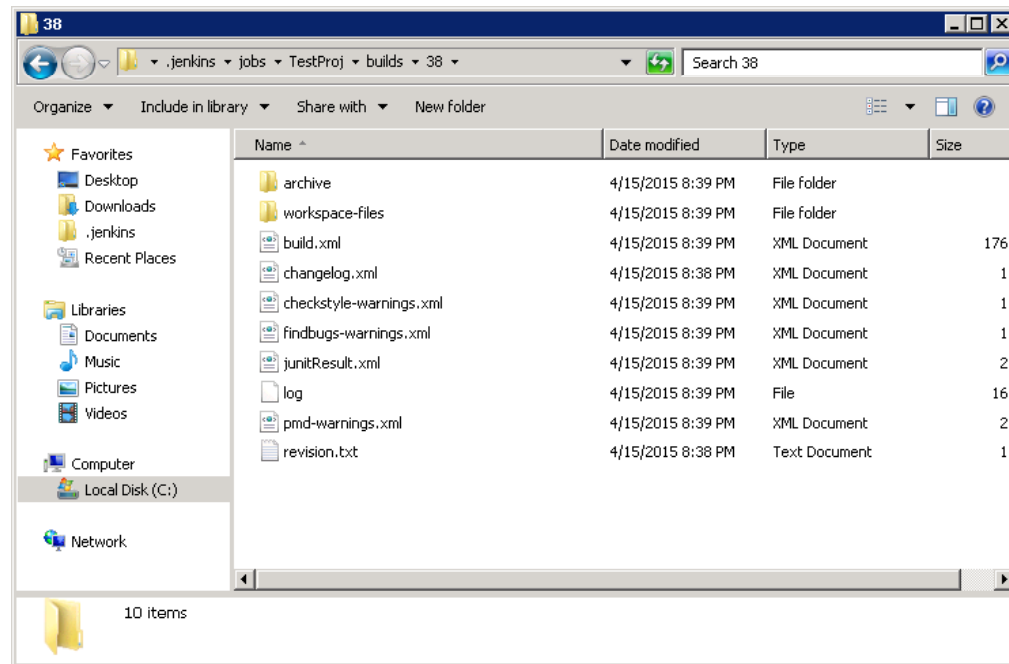
```
BUILD SUCCESSFUL
Total time: 17 seconds
[CHECKSTYLE] Collecting checkstyle analysis files...
[CHECKSTYLE] Finding all files that match the pattern **/report/checkstyle/checkstyle_report.xml
[CHECKSTYLE] Parsing 1 file in C:\Users\Administrator#.jenkins#workspace#TestProj
[CHECKSTYLE] Successfully parsed file C:\Users\Administrator#.jenkins#workspace#TestProj#report#checkstyle#checkstyle_report.xml
with 79 unique warnings and 0 duplicates.
[CHECKSTYLE] Computing warning deltas based on reference build #38
[FINDBUGS] Collecting findbugs analysis files...
[FINDBUGS] Finding all files that match the pattern **/report/findbugs/findbugs.xml
[FINDBUGS] Parsing 1 file in C:\Users\Administrator#.jenkins#workspace#TestProj
[FINDBUGS] Successfully parsed file C:\Users\Administrator#.jenkins#workspace#TestProj#report#findbugs#findbugs.xml with 0
unique warnings and 0 duplicates.
[FINDBUGS] Computing warning deltas based on reference build #38
[PMD] Collecting PMD analysis files...
[PMD] Finding all files that match the pattern **/report/pmd/pmd_report.xml
[PMD] Parsing 1 file in C:\Users\Administrator#.jenkins#workspace#TestProj
[PMD] Successfully parsed file C:\Users\Administrator#.jenkins#workspace#TestProj#report#pmd#pmd_report.xml with 2 unique
warnings and 0 duplicates.
[PMD] Computing warning deltas based on reference build #38
Archiving artifacts
Recording test results
[JDepend] JDepend plugin is ready
[JDepend] Starting JDepend file, outputting to C:\Users\ADMINI~1\AppData\Local\Temp#2#jdepend1853488994928696931.xml
[JDepend] Found 333 classes in 34 packages
[JDepend] Unable to remove temp JDepend file in C:\Users\ADMINI~1\AppData\Local\Temp#2#jdepend1853488994928696931.xml
Finished: SUCCESS
```

Jenkins 빌드 콘솔 결과



# 1. Static Analysis

설치 및 연동 결과



#38의 출력 결과

# 1. Static Analysis

설치 및 연동 결과

The image displays three screenshots of static analysis tool results in a web browser. The leftmost window shows the FindBugs (3.0.1) Analysis for a project, with a summary table indicating 28 code size units, 0 bugs, and 0 high priority bugs. The middle window shows the PMD 5.3.0 Report for the same project, dated 2015-04-15, listing two violations for the TestMain class. The rightmost window shows the CheckStyle Audit results, listing 16 violations across various files.

Package	Code Size	Bugs	High Pri. Bugs	Medium Pri. Bugs	Low Pri. Bugs	Err. Bugs	Ratio
Overall (1 package), (2 classes)	28	0					

File	Total	Priority 1	Priority 2	Priority 3	Priority 4	Priority 5
C:\Users\Administrator\jenkins\workspace\TestProj\src\TestMainTest	3	1	2	0	0	0

File	Errors
C:\Users\Administrator\jenkins\workspace\TestProj\src\TestMain.java	16
C:\Users\Administrator\jenkins\workspace\TestProj\src\TestMainTest.java	0

Html 형태의 분석 결과 보고

# 1. Static Analysis

## 설치 및 연동 결과

### CheckStyle Result

#### Warnings Trend

All Warnings	New Warnings	Fixed Warnings
79	79	0

#### Summary

Total	High Priority	Normal Priority	Low Priority
79	0	79	0

#### Details

Files	Categories	Types	Warnings	Details	New
<b>Warnings - File TestMain.java</b>					
File	Total	Distribution			
<a href="#">TestJUnitTest.java</a>	39				
<a href="#">TestMain.java</a>	40				
Total	79				

Total	High Priority	Normal Priority	Low Priority
40	0	40	0

#### Details

Categories	Types	Warnings	Details
<b>Warnings - File TestMain.java</b>			
Category	Total	Distribution	
<a href="#">Indentation</a>	12		
<a href="#">Naming</a>	4		
<a href="#">Whitespace</a>	24		
Total	40		

CheckStyle 상세 화면  
(18줄 코드에 40개 경고...)

# 1. Static Analysis

## 설치 및 연동 결과

### PMD Result

#### Warnings Trend

All Warnings	New Warnings	Fixed Warnings
2	0	0

#### Summary

Total	High Priority	Normal Priority	Low Priority
2	0	2	0

#### Details

File	Total	Distribution
<a href="#">TestJUnitTest.java</a>	1	
<a href="#">TestMain.java</a>	1	
Total	2	

#### Warnings - File TestMain.java

Priority	Normal Priority	Low Priority
	1	0

#### Details

##### Details

[TestMain.java:2](#), NoPackage, Priority: Normal

All classes and interfaces must belong to a named package.

Detects when a class or interface does not have a package definition.

```
// no package declaration
public class ClassInDefaultPackage {
}
```

PMD 결과화면  
(Package 이름 지정하라고 경고)

# 1. Static Analysis

## 설치 및 연동 결과

### Metric Results

[ [summary](#) ] [ [packages](#) ] [ [cycles](#) ] [ [explanations](#) ]

The following document contains the results of a JDepend metric analysis. The various metrics are defined at the bottom of this document.

### Summary

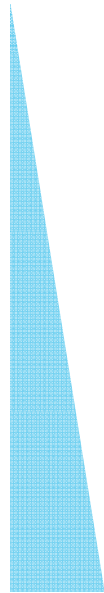
[ [summary](#) ] [ [packages](#) ] [ [cycles](#) ] [ [explanations](#) ]

Package	TC	CC	AC	Ca	Ce	A	I	D	V
<a href="#">Default</a>	2	2	0	0	3	0.0%	100.0%	0.0%	1
<a href="#">junit.extensions</a>	6	6	0	1	2	0.0%	67.0%	33.0%	1
<a href="#">junit.framework</a>	17	13	4	7	9	24.0%	56.0%	20.0%	1
<a href="#">junit.runner</a>	3	1	2	2	6	67.0%	75.0%	42.0%	1
<a href="#">junit.textui</a>	2	2	0	0	6	0.0%	100.0%	0.0%	1
<a href="#">org.hamcrest</a>	21	7	14	8	6	67.0%	43.0%	10.0%	1
<a href="#">org.hamcrest.core</a>	20	18	2	2	5	10.0%	71.0%	19.0%	1
<a href="#">org.hamcrest.internal</a>	4	4	0	1	4	0.0%	80.0%	20.0%	1
<a href="#">org.junit</a>	17	8	9	12	6	52.999996%	33.0%	14.0%	1

JDepend 결과 화면

# 1. System Test

시스템 테스트 도구



## 2. System Test

### Requirement management

- ▶ 요구사항을 기록해 관리하고, 코드와 요구사항을 맵핑해 테스트를 수행한다.
  - ▶ JFeature, OSRMT 등

## 2. System Test

### Requirement Management Tool

#### ▶ JFeature

- ▶ 요구사항들을 기록하고, JUnit의 테스트 케이스와 매칭, 자동화된 요구사항 커버리지 테스트 지원
- ▶ 코드로부터 요구사항을 추적할 수 있도록 함. (Traceability)
- ▶ 사용 방법
  1. 요구사항 리스트를 작성하거나 импорт
  2. 요구사항에 유닛테스트를 맵핑
  3. 유닛테스트 수행시 자동으로 요구사항 커버리지 레포트 생성



# 2. System Test Requirement Management Tool

The screenshot displays the 'Requirement Coverage Report' interface. It features a sidebar on the left with sections for 'All Categories' (listing 'Advanced (66.67%)' and 'Basic (75%)') and 'All Requirements' (listing various mathematical operations with their coverage percentages). The main area contains a 'Requirement Coverage Summary' with a progress bar for 'Summary (8)' showing 6 (75%) in green, 1 (12.5%) in red, and 1 (12.5%) in yellow. Below this is a 'Requirement Coverage Details' table. To the right, an 'Overview Statistics' table provides summary metrics. At the bottom, it states 'Report generated on Mon, 09 Apr 2007 22:40:53 GMT+05:30' and includes a 'Powered By JF JFeature' logo.

Summary (8)	6 (75%)	1 (12.5%)	1 (12.5%)
-------------	---------	-----------	-----------

Sr#	Category	Coverage
1.	Advanced (3)	2 (66.67%) 1 (33.33%)
2.	Basic (4)	3 (75%) 1 (25%)

Number of Requirements	8
Unique Test Methods	14
Requirements:Test Methods Ratio	1:1
Missing Test Methods	None
Unmapped Test Methods	2

## 2. System Test

### Category-Partition Testing

- ▶ 테스트하는 기능의 입력 도메인을 카테고리 단위로 분할해서, 각 단위에 대해 테스트 데이터를 하나씩 선택하는 방법.
  
- ▶ 테스트 생성 방법
  1. 요구사항을 독립적으로 테스트 가능한 기능 유닛으로 쪼갬다.
  2. 각 기능 유닛에서 "파라미터 (Parameter)"와 "환경 상태 (Environment Conditions)"를 구분한다.
    - 파라미터: 기능 유닛에 들어가는 명시적 입력 값
    - 환경 상태: 현재 시스템의 상태에 대한 특성
  3. 각 파라미터와 환경 상태를 카테고리로 분류한다.
  4. 각 카테고리에 대해 선택 가능한 값들의 집합을 만든다.
  5. 테스트 케이스를 만든다.

## 2. System Test

### Pairwise Testing

- ▶ 취급해야 하는 테스트 케이스 갯수가 너무 많은 상황에서 테스트 케이스를 줄이는 방법
- ▶ 2개 파라미터에 대한 모든 페어를 포함하도록 테스트 시퀀스 생성.
- ▶ 모든 케이스를 취급하지 않으므로 한계점을 가진다.

## 2. System Test

### Pairwise Testing

- ▶ 예시: {a, b}, {c, d, e, f}, {g, h, i}, {j, k}
- ▶ 이 경우 모든 케이스를 테스트하기 위해서는  $2 * 4 * 3 * 2 = 48$ 개 테스트 케이스가 필요하다.
- ▶ 지정된 페어가 다음과 같다면
  - ▶ (a, c), (a, d), (a, e), (a, f), (a, g), (a, h), (a, i), (a, j), (a, k)
  - ▶ (b, c), (b, d), (b, e), (b, f), (b, g), (b, h), (b, i), (b, j), (b, k)
  - ▶ (c, g), (c, h), (c, i), (c, j), (c, k)
  - ▶ (d, g), (d, h), (d, i), (d, j), (d, k)
  - ▶ (e, g), (e, h), (e, i), (e, j), (e, k)
  - ▶ (f, g), (f, h), (f, i), (f, j), (f, k)
  - ▶ (g, j), (g, k), (h, j), (h, k), (i, j), (i, k)

## 2. System Test

### Pairwise Testing

- ▶ 도출된 테스트케이스
  - ▶ a c g j
  - ▶ b d g k
  - ▶ a e h k
  - ▶ b f i j
  - ▶ b c h j
  - ▶ a d i j
  - ▶ a f g k
  - ▶ b e i j
  - ▶ a d h j
  - ▶ a c i k
  - ▶ a e g j
  - ▶ a f h j
- ▶ 모든 페어가 포함되며, 테스트 케이스가 12개로 줄었다!

## 2. System Test

### Pairwise Test Sequence Generating Tool

- ▶ PICT (Pairwise Independent Combinatorial Testing tool)
  - ▶ MS에서 개발한 Pairwise 테스트 시퀀스 생성 툴
  - ▶ 커맨드라인에서 동작
- ▶ QICT
  - ▶ PICT와 비슷한 기능을 갖춘, 소스가 공개된 툴 (C#)
- ▶ 기타 툴
  - ▶ All Pairs (무료)
  - ▶ Jenny (무료)
  - ▶ Hexawise (유료)
  - ▶ Pairwiser (유료)

## 2. System Test

### JFeature 연동

```
▼<target name="report.jfeature" description="Generates the requirement coverage report." depends="junit">
  <mkdir dir="${report.jfeature.dir}"/>
  ▼<!--
    Load JFeature task definition
    Note: The directory net.technobuff.jfeature below is the one created after expanding the
    JFeature distribution. You may want to copy it to the directory from which your build script
    is fired.
  -->
  ▼<taskdef name="jfeaturecoveragereport" classname="net.technobuff.jfeature.ant.task.JFeatureCoverageReportTask">
    ▼<classpath>
      <path element="location" location="${jfeature.home.dir}/net.technobuff.jfeature"/>
      ▼<fileset dir="${jfeature.home.dir}/net.technobuff.jfeature">
        <include name="**/*.jar"/>
      </fileset>
    </classpath>
  </taskdef>
  ▼<!--
    Generate the requirement coverage report
    * testresultsdir: The directory in which the "junit" task generated the XML files for the unit test results.
    * todir: The directory in which to store the requirement coverage reports.
    * format: (Optional) The format of the requirement coverage report (html|xml|rss).
    * The default is "html" format. format="html"
  -->
  ▼<jfeaturecoveragereport testresultsdir="${report.temp.dir}" todir="${report.jfeature.dir}" format="html">
    ▼<!--
      Provide the requirements file set
      * dir: The directory containing requirements files.
    -->
    -->
    ▼<requirementsfileset dir="${basedir}/requirements">
      <!-- Specify the pattern(s) for the requirements files -->
      <include name="**/*.jrq"/>
    </requirementsfileset>
  </jfeaturecoveragereport>
</target>
▼<target name="zip.report.jfeature" depends="report.jfeature">
  <zip destfile="${bin.dir}/JFeature-Report.zip" basedir="${report.jfeature.dir}"/>
</target>
```

Static Analysis Tool 설정과 같이 build.xml을 통하여 Jfeature의 Ant Task를 붙인다.

## 2. System Test

JFeature 연동



The screenshot shows a file explorer interface with the following items:

- 작업 공간** (Workspace) - Folder icon
- 최근 성공한 결과물들** (Recent successful results) - Folder icon
  - [JFeature-Report.zip](#) 10.31 KB [view](#)
  - [TestJUnitTest.class](#) 930 B [view](#)
  - [TestMain.class](#) 796 B [view](#)
- 최근 변경사항** (Recent changes) - Notepad icon
- 가장 최근 테스트 결과 (실패가 없습니다)** (Latest test results (no failures)) - Clipboard icon

Jfeature Report 결과물을 Zip으로 Export  
(Jenkins plugin이 없으므로 Jenkins에서 볼 수 없다. 만들면 볼 수 있지만...)



# 2. System Test

JFeature 연동

Home

All Categories  
Main (75%)

All Requirements  
add (100%)  
sub (100%)  
fail Test (100%)  
no Requirement (0%)

### Requirement Coverage Report

#### Requirement Coverage Summary

Summary (4)	3 (75%)	1 (25%)
-------------	---------	---------

#### Requirement Coverage Details

Sr#	Category	Coverage
1.	Main (4)	3 (75%) 1 (25%)

Number of Requirements	4
Unique Test Methods	3
Requirements:Test Methods Ratio	1:1
Missing Test Methods	None
Unmapped Test Methods	None

Report generated on Thu, 16 Apr 2015 01:45:50 KST

Powered By  
**JFeature**

HTML 형식의 출력 결과



# Trouble Shooting

연동 시 발생한 문제점들

# Trouble Shooting

- ▶ Checkstyle, PMD의 룰셋이나 출력 스타일을 어딘가에서 따로 구하거나 직접 작성해야 한다.
  - ▶ 어렵지 않게 구할 수 있으나 그렇게 쉽지도 않다.
- ▶ Jfeature는 Ecilpse 4.2 이하에서 작동한다.
  - ▶ 딱히 4.4에도 돌리더라도 괜찮을 것 같은데 안 된다.
  - ▶ 마켓 플레이스에는 3.2~3.4에서 볼 수 있다.
  - ▶ Ant Tast는 관계없다.
- ▶ Jenkins는 출력 산물을 디렉토리로 묶지 않고 전부 나열한다.
  - ▶ 그래서 Jfeature html report를 zip으로 묶어냈다.



Thank you