Accuracy Test of Software Architecture Compliance Checking Tools – Test Instruction

Version 2

Dr. L.J. Pruijt
C. Köppe MSc.
Dr.ir. J.M.E.M. van der Werf
Prof.dr. S. Brinkkemper

Technical Report UU-CS-2015-020
December 2015

Department of Information and Computing Sciences
Utrecht University, Utrecht, The Netherlands
www.cs.uu.nl

ISSN: 0924-3275

Department of Information and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands

Accuracy Test of Software Architecture Compliance Checking Tools – Test Instruction

Version 2

Leo Pruijt, HU University of Applied Sciences, Utrecht, The Netherlands
Christian Köppe, HAN University of Applied Sciences, Arnhem, The Netherlands
Jan Martijn van der Werf and **Sjaak Brinkkemper**, University Utrecht, Utrecht, The Netherlands

Abstract

Software Architecture Compliance Checking (SACC) is an approach to verify conformance of implemented program code to high-level models of architectural design. Static SACC focuses on the modular software architecture and on the existence of rule violating dependencies between modules. Accurate tool support is essential for effective and efficient SACC. This document describes a test approach that may be used to determine how accurate a tested SACCT-tool is with respect to dependency analysis and violation reporting. This technical report is intended as a test manual and describes how a SACCT-tool can be tested. Two separate tests are described: the Benchmark test, and the FreeMind test.

Table of Contents

1. Introduction

Software Architecture Compliance Checking (SACC) is an approach to verify conformance of implemented program code to high-level models of architectural design. Static SACC focuses on the modular software architecture and on the existence of rule violating dependencies between modules. Accurate tool support is essential for effective and efficient SACC.

The objective of the full test in this report is to determine how accurate a tested SACCT-tool is with respect to dependency analysis and violation reporting, by answering the following questions:

- Does the SACC-tool find all the dependencies between modules in the test software?
- Does the SACC-tool report all the violating dependencies in the test software?
- Does the SACC-tool report non-violating dependencies as violations?
- Does the SACC-tool report the exact type and location of violations and dependencies?

This document is intended as a test manual and describes how a SACCT-tool can be tested.
Two separate tests are described in this document: the Benchmark test, and the FreeMind test.
The following sections introduce these test and provide instructions on how to conduct the tests and how to register the results.

The testware, program code and scoring documents, are downloadable (select “Download ZIP”) from the following address:
https://github.com/SaccToolTests/SacctAccuracyTest

More information on the objectives of the SACC-tool accuracy test, the method, et cetera may be found in the paper below. This paper also describes the test results of ten SACC-tools, which are tested with this second version of the test.

- Pruijt, L., Köppe, C., Brinkkemper, S., and van der Werf, J. M. (2015).
  The Accuracy of Dependency Analysis in Architecture Compliance Checking.
  Submitted for publication.

The first version of this test was used to test seven SACC-tools. The paper describing this test and its results is the following:

- Pruijt, L., Köppe, C., and Brinkkemper, S. (2013).
  On the Accuracy of Architecture Compliance Checking: Accuracy of Dependency Analysis and Violation Reporting.
  In H. Kagdi, D. Poshyvanyk, & M. Di Penta (Eds.), *21st International Conference on Program Comprehension* (pp. 172–181). San Francisco, CA, USA: IEEE Computer Society Press.

2. Benchmark Test – Instruction

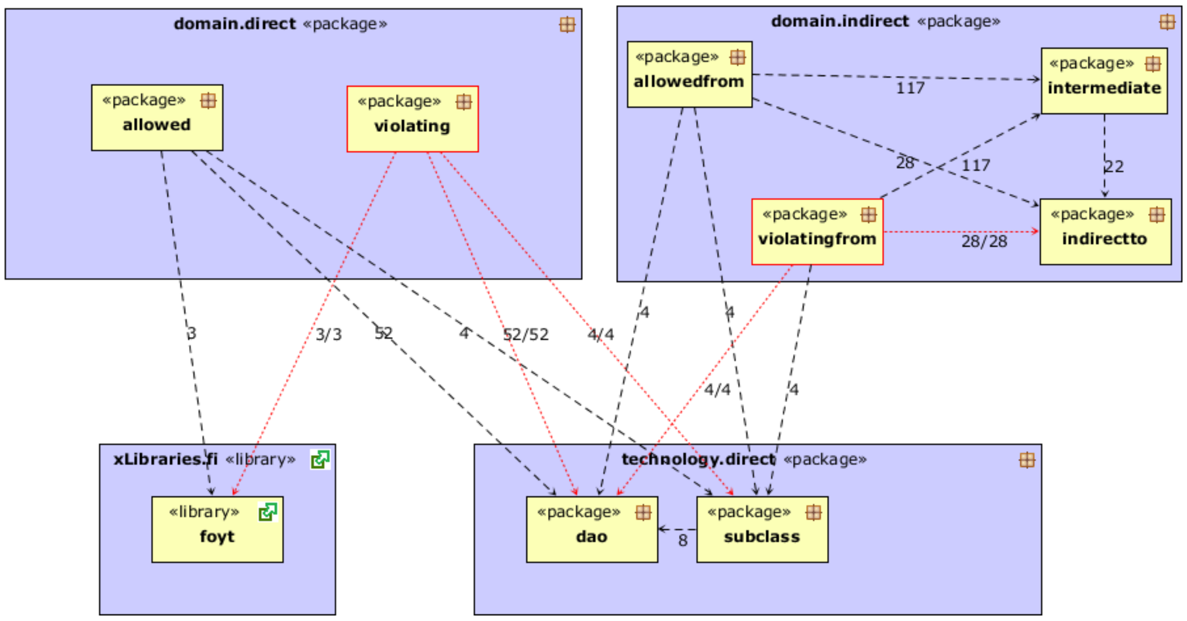
2.1 Introduction

The Benchmark test is used to investigate if a SACC-tool is able to detect 34 different types of dependency, and if it is able to report rule violating dependencies of these 34 types. The Java code of the benchmark testware contains 25 different types of direct dependencies and 9 different types of indirect dependencies.

The figure below shows the structure of the test code: the relevant packages and the dependencies between them (the black, dashed lines; the number represents the number of detected dependencies). The red, dotted lines show the focus of the test. They indicate that a number of dependencies (the first number) of all the detected dependencies (the second number) violate an architectural rule.

To measure the sensitivity (true positive rate) of the ACC tools, 64 cases are aimed at the detection of true positives and false negatives.Package domain.direct.violating contains 34 classes with rule violating direct dependencies on a class in library fi.foyt, or on classes in package technology.direct. Furthermore package domain.indirect.violatingfrom contains 30 classes with rule violating indirect dependencies on classes in package domain.indirect.indirectto or technology.direct.dao. Each class in domain.direct.violating and domain.indirect.violatingfrom represents a test case for a specific type of dependency.

A tested tool scores high on sensitivity in this test, if it is able to report all (or many of) the existing dependencies and, if a rule forbids such a dependency, report these dependencies as violations.



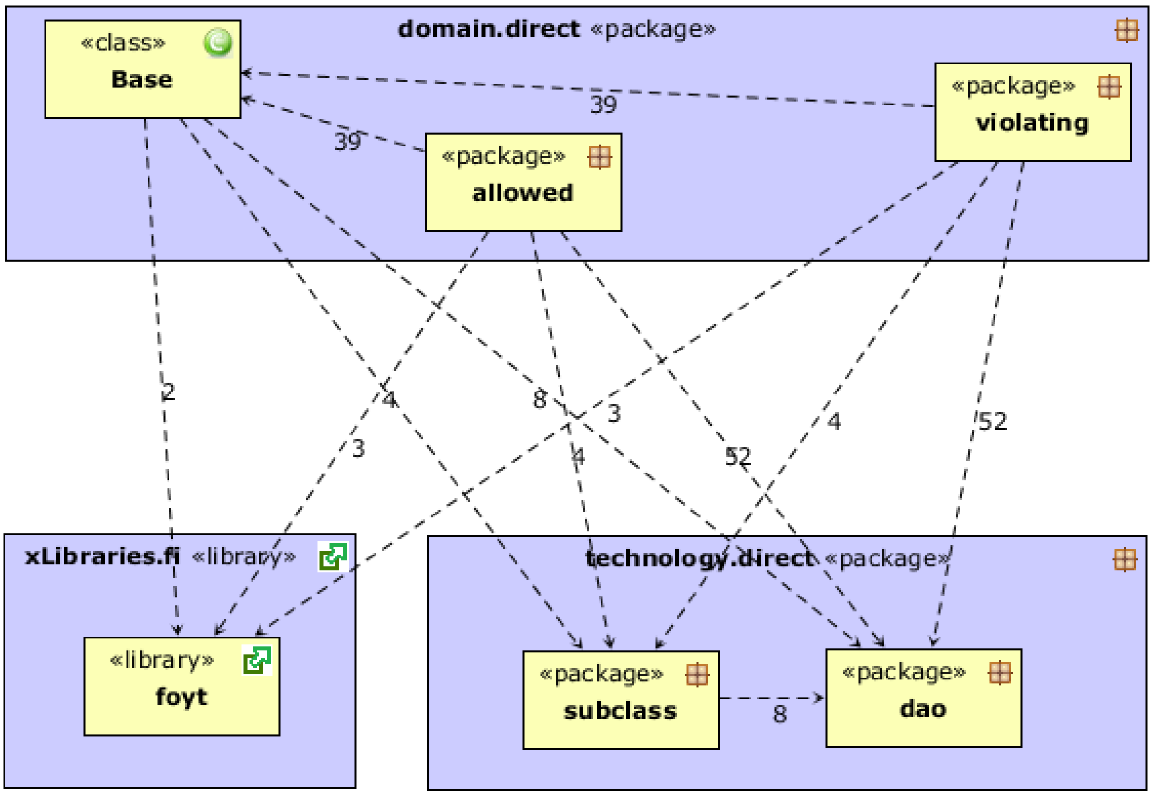
To measure the false positive rate of the ACC tools, 64 cases are aimed at the detection of false positives. Packages domain.allowed and domain.indirect.allowedfrom contain copies of the classes in the corresponding violating packages, so dependencies to the same to-classes are contained. Since no architectural rule constrains the classes in the “allowed” packages, reported violations of these classes are qualified as false positives.

Note 1: Since SACC-tools differ considerably, the instructions on how to define modules, assign code to these modules, define rules, et cetera, are described in general terms, and should be interpreted as intended actions. Knowledge of the tested tool is required to use the available tool-options in order to perform the intended actions.

Note 2: The diagrams are constructed with HUSACCT_4.4. More information on HUSACCT in:
Pruijt, L., Köppe, C., van der Werf, J. M., and Brinkkemper, S. (2014).
HUSACCT: Architecture Compliance Checking with Rich Sets of Module and Rule Types.
In Proceedings of the 29th ACM/IEEE international conference on Automated software engineering - ASE '14 (pp. 851–854). ACM Press.

Note 3: The names of the top-pckages domain and technology are what they are for historical reasons, but currently have no meaning.

Note 4: To reduce the dependencies between a from-class and to-class per test case as much as possible to only the dependency of the type specific for a test case, the Base class has been introduced. The Base class is a super class of many test case specific from-classes. The Base class, visible in the figure below, declares many vatiables needed for the test cases. The resulting dependencies of type Import and Declaration are on the Base class and not on the specific test case class. The variables are available via inheritance to the subclasses. We have verified that this construction does not influence the test results, and it did not; all tested tools in the published papers were able to determine the type of an inherited variable.



Within package domain.indirect, class BaseIndirect has the same function within this package.

2.2 Direct Dependencies

2.2.1 Introduction

Detection of the following direct dependency types is tested in this section.

PRT	Category	Dependency Types
1	Call	Instance method
		instance method, inherited
		Class method
		Constructor
		Inner class method
		Interface method
		Library class method
2	Access	Instance variable (read/write)
		Instance variable, inherited
		Class variable
		Class variable, constant
		Class variable, interface
		Enumeration
		Object reference, ref. variable
		Object reference, var within if
3	Inheritance	Extends class
		Extends abstract class
		Implements interface
4	Declaration (type)	Instance variable
		Class variable
		Local variable
		Parameter
		Return type
		Exception
		Type cast
5	Annotation	Class annotation
6	Import	Class import

2.2.2 Test Procedure

1) Download the test code and score sheet of the benchmark test

In the BenchmarkTest directory a zip with the test code is available, as well as a writeable version of the score form. Extract the test code files. Java files as well as the class files are available in respectively the src and bin subdirectories.

2) Analyse the code

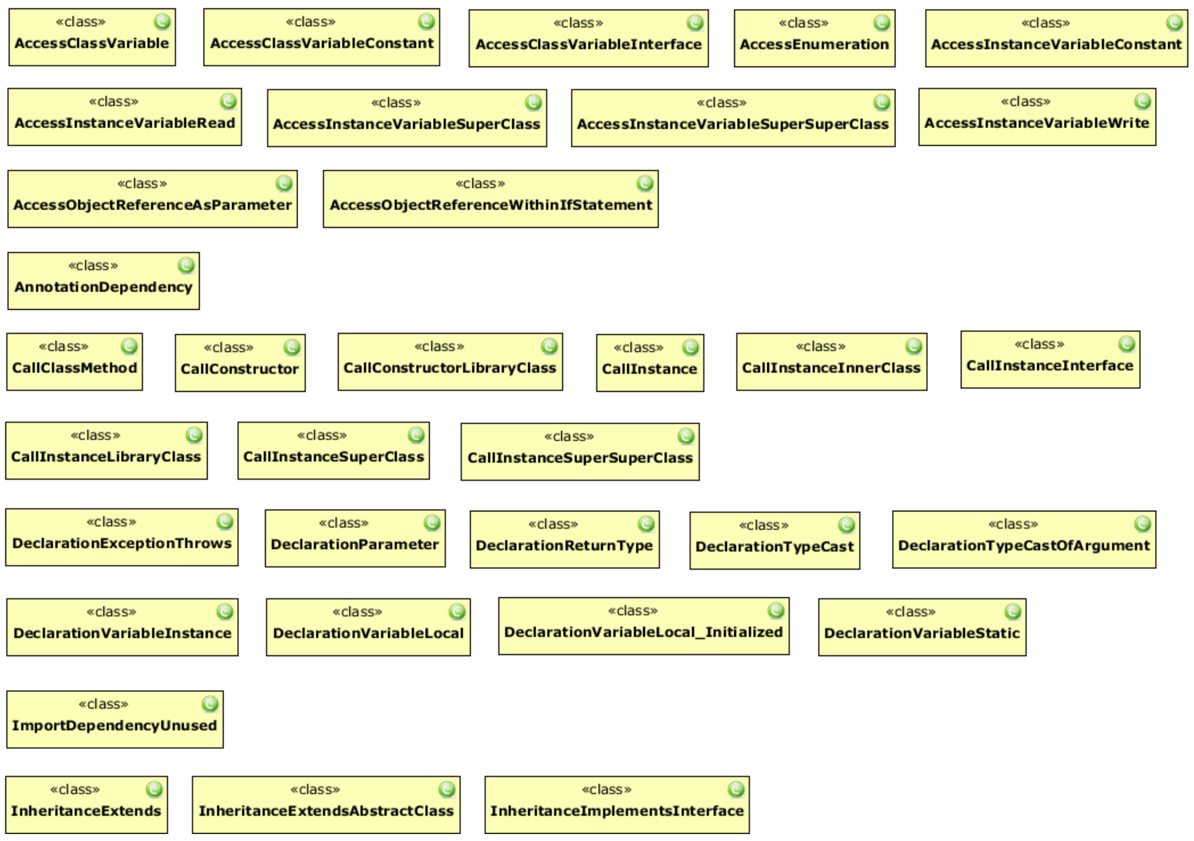
Activate the SACC-tool to analyse the code in the src or bin subdirectories (or both).

3) Define the modules and assign code packages.

Define the following three modules (of the specified module type, if possible).
Next, assign the corresponding packages in the code to the modules.

Module	Module Type	Assigned Code Package
DomainDirectViolating	Subsystem	domain.direct.violating
TechnologyDirect	Subsystem	technology.direct (including both subpackages)
FoursquareAPI	Library	fi.foyt.foursquare.api Or to: fi.foyt.foursquare.api.* (or a tool specific RegEx) The jar can be found in the lib folder.

Package domain.direct.violating contains 34 classes. A class name indicates a specific type of dependency included in the class.


«class» AccessClassVariable
«class» AccessClassVariableConstant
«class» AccessClassVariableInterface
«class» AccessEnumeration
«class» AccessInstanceVariableConstant
«class» AccessInstanceVariableRead
«class» AccessInstanceVariableSuperClass
«class» AccessInstanceVariableSuperSuperClass
«class» AccessInstanceVariableWrite
«class» AccessObjectReferenceAsParameter
«class» AccessObjectReferenceWithinIfStatement
«class» AnnotationDependency
«class» CallClassMethod
«class» CallConstructor
«class» CallConstructorLibraryClass
«class» CallInstance
«class» CallInstanceInnerClass
«class» CallInstanceInterface
«class» CallInstanceLibraryClass
«class» CallInstanceSuperClass
«class» CallInstanceSuperSuperClass
«class» DeclarationExceptionThrows
«class» DeclarationParameter
«class» DeclarationReturnType
«class» DeclarationTypeCast
«class» DeclarationTypeCastOfArgument
«class» DeclarationVariableInstance
«class» DeclarationVariableLocal
«class» DeclarationVariableLocal_Initialized
«class» DeclarationVariableStatic
«class» ImportDependencyUnused
«class» InheritanceExtends
«class» InheritanceExtendsAbstractClass
«class» InheritanceImplementsInterface

4) Inspect the detected dependencies between the packages.

Open the writeable score form of the benchmark test and make use of section “Direct Dependencies: Test Results - Expected Violations”. (Note: A read-only version is available in the next section of this document).
In the table, 34 test cases are specified and for each test case is described which dependencies are present between a from-class (in package domain.direct.violating) and a to-class. The name of the

from-class includes the specific dependency type of the test case.
Check if the (specific) dependency is reported by the tool.
For each test case:

1. Register the result in the column “Dependency detected?”. Check the detection of all the dependencies per test case. Mark a “+” if the type specific to the test case is detected, or a “-“ if not. Make a note (in Comment), when a dependency is not detected.
2. Gather evidence, e.g.: 1) Generate or create a report with all the detected dependencies between domain and technology; 2) Screenshots; or 3) diagrams.

5) Define the rules.

Architectural rule
domain.direct.violating is not allowed to use technology.direct.
domain.direct.violating is not allowed to use foursquareapi.jar.

Note: The rules are based on the Non-Restricting Principle (dependencies are allowed, except when explicitly forbidden). Consequently, when a tool is based on the Restricting Principle (no dependencies are allowed, except when explicitly specified), the rules have to be specified differently, to gain the intended result.
Otherwise, more violations are reported than expected by the test cases, since each dependency which is not explicitly allowed is reported as a violation.

6) Activate the Architecture Compliance Check.

7) Inspect the reported violations.

Re-iterate and inspect as described under “Inspect the detected dependencies between the packages”, but now for dependencies reported as violations.

For each test case:

1. Register the result in the column “Violation reported?”.
2. Gather evidence.

8) Inspect and register not-expected violations.

If violations are be reported from classes in domain.direct.allowed, register them in the table under “Direct Dependencies: Test Results - Unexpected Violations”. These violations should be regarded as false positives. Gather evidence.

9) Summarize the results to performance per dependency type

Fill-in the table under “Direct Dependencies: Summary - Dependency Detection and Violation Reporting”.
For each dependency type, determine if the tool is able to detect/report dependencies and violations, based on the results from the previous steps (+ = detected; ± = partially detected (explain in comment); - = not detected). The class names indicate the related dependency types. Multiple test cases (classes) may be used to test a dependency type.

2.3 Indirect Dependencies

2.3.1 Test Procedure

1) Define the modules and assign code packages.

Define the following three modules.
Next, assign the corresponding packages in the code to the modules.

Module	Module Type	Map to Code
DomainIndirectViolatingFrom	Subsystem	domain.indirect.violatingfrom
DomainIndirectIndirectTo	Subsystem	domain.indirect.indirectto
TechnologyDirectDao	Subsystem	technology.direct. dao

2) Inspect the detected dependencies between the packages specified in the architectural rules.

Open the score form of the benchmark test and make use of section “Indirect Dependencies: Test Results - Expected Violations”.
In the table, 30 test cases are specified and for each test case is described which dependencies are present between a from-class (in package domain.indirect.violatingfrom) and a to-class. The name of the from-class includes the specific dependency type of the test case.
Check if the (specific) dependency is reported by the tool.
For each test case:

1. Register the result in the column “Dependency detected?”. Check the detection of all the dependencies per test case. Mark a “+” if the type specific to the test case is detected, or a “-“ if not. Make a note (in Comment), when a dependency is not detected.
   Dependencies to classes in package domain.indirect.intermediate or technology.direct.subclass do not count as indirect dependencies.
2. Gather evidence, e.g.: 1) Generate or create a report with all the detected dependencies between domain and technology; 2) Screenshots; or 3) diagrams.

3) Define the rules.

Architectural rule
domain.indirect.violatingfrom is not allowed to use domain.indirect.indirectto
domain.indirect.violatingfrom is not allowed to use technology.direct.dao

4) Activate the Architecture Compliance Check.

5) Inspect the reported violations.

The test cases include dependencies from a class in package domain.indirect.violatingfrom to a class in package domain.indirect.violatingto. Except when specified differently: four test cases in domain.indirect.violatingfrom make violating use of a class in technology.direct.dao.
All these dependencies should be reported as violations.

For each test case:

1.1. Register the result in the column “Violation reported?”.
1.2. Gather evidence.

6) Inspect and register not-expected violations.

If violations are be reported from classes in domain.direct.allowed, register them in the table under “Indirect Dependencies: Test Results - Unexpected Violations”. These violations should be regarded as false positives. Gather evidence.

7) Summarize the results

Fill-in the table under “Indirect Dependencies: Summary - Dependency Detection and Violation Reporting”.
For each dependency type, determine if the tool is able to detect/report dependencies and violations, based on the results from the previous steps (+ = detected; ± = partially detected (explain in comment); - = not detected). The class names indicate the related dependency types. Multiple test cases (classes) may be used to test a dependency type.

3. Benchmark Test – Score Form – Read-only Version

SACCT Benchmark Test: <Tool Name>

Accuracy of Dependency Detection

Direct dependencies
Indirect dependencies

Name of the tool:	**x**
Version:	**y**
Website:	**z**
End date test:	**d**

3.1 Direct Dependencies: Summary - Dependency Detection and Violation Reporting

Summary of the findings from the test: + = detected; ± = partially detected (explanation in comment); - = not detected.

#	Category	Dependency Types	Dependency detected	Violation reported	Comment
1	Call	Instance method			
		instance method, inherited			
		Class method			
		Constructor			
		Inner class method			
		Interface method			
		Library class method			
2	Access	Instance variable			
		Instance variable, constant			
		Instance variable, inherited			
		Class variable			
		Class variable, constant			
		Class variable, interface			
		Enumeration			
		Object reference, ref. variable			
		Object reference, var within if			
3	Inheritance	Extends class			
		Extends abstract class			
		Implements interface			
4	Declaration (type)	Instance variable			
		Class variable			
		Local variable			
		Parameter			
		Return type			
		Exception			
		Type cast			
5	Annotation	Class annotation			
6	Import	Class import			

3.2 Direct Dependencies: Test Results - Expected Violations

The following test cases all include dependencies:

From a class in: domain.direct.violating

To a class in: technology.direct.dao

Except when specified differently: four test cases make use of technology.direct.subclass.

Test cases per Dependency type	**Dependencies**	**Dependency detected? +(Yes), -(No)**	**Violation reported? +(Yes), -(No)**	**Comment**
Access				
Type: Access – Class variable From: AccessClassVariable To: CheckInDAO	1-Import 2-Access- class variable			
Type: Access – Class variable - Constant From: AccessClassVariableConstant To: UserDAO	1-Import 2-Access- class -constant			
Type: Access – Class variable - Interface From: AccessClassVariableInterface To: ISierraDAO	1-Import 2-Access-class variable-interface			
Type: Access – Enumeration From: AccessEnumeration To: TipDAO	1-Import 2-Access-Enumeration			
Type: Access – Instance variable – Read From: AccessInstanceVariableRead To: ProfileDAO	1-Access-instance var.			
Type: Access – Instance variable – Write From: AccessInstanceVariableWrite To: ProfileDAO	1-Access-instance var.			
Type: Access – Instance variable - Constant From: AccessInstanceVariableConstant To: UserDAO	1-Access-instance-constant			
Type: Access – Instance variable – Inherited From: AccessInstanceVariableSuperClass To: technology.direct.subclass.CallInstanceSubClassDOA. VariableOnSuperClass **Note**: Also OK if access to technology.direct.dao.CallInstanceSuperClassDAO is reported. Write as comment the reported class(es).	1- Access-instance - inherited			

Type: Access – Instance variable – Inherited of 2nd super cl. From: AccessInstanceVariableSuperSuperClass To: technology.direct.subclass.CallInstanceSubSubClassDOA **Note**: Also OK if access to technology.direct.subclass.CallInstanceSubClassDOA or technology.direct.dao.CallInstanceSuperClassDAO is reported. Write a comment!	1- Access-instance - inherited			
Type: Access – Object reference – Ref.Variable - Parameter From: AccessObjectReferenceAsParameter To: ProfileDAO	1-Access-object ref-param			
Type: Access – Object reference – Ref.Variable – If exists From: AccessObjectReferenceWithinIfStatement To: ProfileDAO	1-Access-object ref-in if			
Annotation				
Type: Class annotation From: AnnotationDependency To: SettingsAnnotation	1-Import 2-Annotation			
Call				
Type: Call – Class method From: CallClassMethod To: BadgesDAO	1-Import 2- Call-class method			
Type: Call – Constructor From: CallConstructor To: AccountDAO	1-Import 2- Call-constructor			
Type: Call – Constructor – Library class From: CallConstructorLibraryClass To: fi.foyt.foursquare.api.FoursquareApi	1-Import 2x 2- Call-constructor-library			
Type: Call – Instance method From: CallInstance To: ProfileDAO	1- Call-instance			
Type: Call – Instance method – Inner class From: CallInstanceInnerClass To: CallInstanceInnerClassDAO within CallInstanceOuterClassDAO	1-Call-instance-inner class			

Type: Call – Instance method - Interface From: CallInstanceInterface To: CallInstanceInterfaceDAO	1-Call-instance-interface			
Type: Call – Instance method – Library class From: CallInstanceLibraryClass To: fi.foyt.foursquare.api.FoursquareApi	1- Call-instance-library			
Type: Call – Instance method – Inherited (Virtual call) From: CallInstanceSuperClass To: technology.direct.subclass.CallInstanceSubClassDOA **Note**: Also OK if a call to technology.direct.dao.CallInstanceSuperClassDAO is reported. Write as comment the reported class(es).	1- Call-instance-inherited			
Type: Call – Instance method – Inherited of 2nd super class From: CallInstanceSuperSuperClass To: technology.direct.subclass.CallInstanceSubSubClassDOA **Note**: Also OK if a call to technology.direct.subclass.CallInstanceSubClassDOA or technology.direct.dao.CallInstanceSuperClassDAO is reported. Write a comment!	1- Call-instance- inherited			
Declaration (of type)				
Type: Declaration – Exception (throws) From: DeclarationExceptionThrows To: StaticsException	1-Import 2-Declaration-exception 3- Call-constructor (throw new)			
Type: Declaration -Parameter From: DeclarationParameter To: ProfileDAO	1-Import 2-Declaration-param.			
Type: Declaration – Return type From: DeclarationReturnType To: VenueDAO	1-Import 2-Declaration-return type			
Type: Declaration – Type cast From: DeclarationTypeCast To: ProfileDAO	1-Import 2-Declaration- type cast			
Type: Declaration – Type cast – Within argument section From: DeclarationTypeCastOfArgument To: ProfileDAO	1-Import 2-Declaration- type cast			

Type: Declaration – Variable – Instance From: DeclarationVariableInstance To: ProfileDAO	1-Import 2-Declaration-variable			
Type: Declaration – Variable – Local From: DeclarationVariableLocal To: ProfileDAO	1-Import 2-Declaration-variable			
Type: Declaration – Variable – Local - Initialized From: DeclarationVariableLocal_Initialized To: ProfileDAO	1-Import 2-Declaration-variable			
Type: Declaration – Variable – Static From: DeclarationVariableStatic To: ProfileDAO	1-Import 2-Declaration-variable			
Import				
Type: Class import - Unused From: ImportDependencyUnused To: AccountDAO	1-Import			
Inheritance				
Type: Inheritance – Extends class From: InheritanceExtends To: HistoryDAO	1-Import 2- Inheritance-extends			
Type: Inheritance – Extends class – Abstract class From: InheritanceExtendsAbstractClass To: FriendsDAO	1-Import 2- Inheritance-extends-abstract			
Type: Inheritance – Implements interface From: InheritanceImplementsInterface To: IMapDAO	1-Import 2-Inheritance-interface			

3.2.1 Gathered Evidence Direct Dependencies: Expected

…

3.3 Direct Dependencies: Test Results - Unexpected Violations

No violating dependencies should be reported between:

From: domain.direct.allowed

To: technology.direct.dao

If an unexpected violation is reported, make a note in the table below.

Test cases	**Dependencies**	**Dependency detected? +(Yes), -(No)**	**Violation reported? +(Yes), -(No)**	**Comment**
Type: From: To:				

3.3.1 Gathered Evidence Direct Dependencies: Unexpected

…

3.4 Indirect Dependencies: Summary - Dependency Detection and Violation Reporting

Summary of the findings from the test: + = detected; ± = partially detected (explanation in comment); - = not detected.

#	Category	Dependency Types	Dependency detected	Violation reported	Comment
1	Call	Instance method			
		Instance method, inherited			
		Class method			
2	Access	Instance variable			
		Instance variable, inherited			
		Class variable			
		Object reference – Reference variable			
		Object reference – Return value			
3	Inheritance	Extends - extends			
		Extends - implements			
		Implements - extends			

3.5 Indirect Dependencies: Test Results - Expected Violations

The following test cases all include dependencies:

From a class in: domain.indirect.violatingfrom

Via a class in: domain.indirect.intermediate

To a class in: domain.indirect.violatingto

Except when specified differently: four test cases in domain.indirect.violatingfrom make use of technology.direct.dao.

Test cases per Dependency type	**Dependencies**	**Dependency detected? +(Yes), -(No)**	**Violation reported? +(Yes), -(No)**	**Comment**
Access				
Type: Access – Instance variable From: AccessInstanceVariableIndirect_MethodVar Via: BackgroundService.getServiceOne() To: ServiceOne.name	1- Access - Instance			
Type: Access – Instance variable From: AccessInstanceVariableIndirect_VarVar Via: BackgroundService.serviceOne To: ServiceOne.name	1- Access- Instance			
Type: Access – Instance variable From: AccessInstanceVariableIndirect_VarVarToString Via: BackgroundService.serviceOne To: ServiceOne.day	1- Access - Instance			
Type: Access – Instance variable – Double indirect From: AccessInstanceVariableIndirectIndirect_MethodVarVar Via: BackgroundService.getServiceTwo() Via: ServiceTwo.serviceOne To: ServiceOne.name	1- Access - Instance			
Type: Access – Instance variable – Double indirect From: AccessInstanceVariableIndirectIndirect_VarVarVar Via: BackgroundService.serviceTwo Via: ServiceTwo.serviceOne To: ServiceOne.name	1- Access - Instance			
Type: Access – Instance variable – Inherited From: AccessInstanceVariableIndirect_SuperClass Via: technology.direct.subclass.CallInstanceSubClassDOA To: CallInstanceSuperClassDAO.VariableOnSuperClass **Note:** Only OK if access to the super class is reported.	1-Access-instance-inherited			To: Technology.Direct.Dao contains ServiceTwo

Type: Access – Instance variable – Inherited of 2nd super class From: AccessInstanceVariableIndirect_SuperSuperClass Via: technology.direct.subclass.CallInstanceSubSubClassDOA and technology.direct.subclass.CallInstanceSubClassDOA To: CallInstanceSuperClassDAO.VariableOnSuperClass **Note:** Only OK if access to the super class is reported.	1-Access-instance-inherited			To: Technology.Direct.Dao contains ServiceTwo
Type: Access – Object reference – Reference Variable From: AccessObjectReferenceIndirect_AsParameter_POI Via: ServiceTwo.getServiceOne() and ServiceOne.poi To: POI	1- Access –Object reference			
Type: Access – Object reference – As return value From: AccessObjectReferenceIndirect_AsParameter Via: ServiceTwo.getServiceOne() To: ServiceOne	1- Access –Object reference			
Type: Access – Object reference – Reference Variable From: AccessObjectReferenceIndirect_WithinIfStament_POI Via: ServiceTwo.getServiceOne() and ServiceOne.poi To: POI	1- Access –Object reference			
Type: Access – Object reference – As return value From: AccessObjectReferenceIndirect_WithinIfStament Via: ServiceTwo.getServiceOne() To: ServiceOne	1- Access –Object reference			
Type: Access – Static variable From: AccessStaticVariableIndirect_MethodVar Via: BackgroundService.getServiceOne() To: ServiceOne.*sName*	1- Access -Static			
Type: Access – Static variable From: AccessStaticVariableIndirect_VarVar Via: BackgroundService.serviceOne To: ServiceOne.*sName*	1- Access -Static			
Type: Access – Static variable From: AccessStaticVariableIndirect_VarVarToString Via: BackgroundService.serviceOne To: ServiceOne.*sName*	1- Access -Static			
Type: Access – Static variable – Double indirect From: AccessStaticVariableIndirectIndirect_MethodVarVar Via: BackgroundService.getServiceTwo()	1- Access -Static			

Via: ServiceTwo.serviceOne To: ServiceOne.*sName*				
Type: Access – Static variable – Double indirect From: AccessStaticVariableIndirectIndirect_VarVarVar Via: BackgroundService.serviceTwo Via: ServiceTwo.serviceOne To: ServiceOne.*sName*	1- Access -Static			
Call				
Type: Call – Instance method From: CallInstanceMethodIndirect_MethodMethod Via: BackgroundService.serviceOne To: ServiceOne.getName()	1- Call-Instance			
Type: Call – Instance method From: CallInstanceMethodIndirect_MethodMethodToString Via: BackgroundService.getServiceOne() To: ServiceOne.getDay()	1- Call-Instance			
Type: Call – Instance method From: CallInstanceMethodIndirect_MethodMethod_ViaConstructor Via: new BackgroundService.getServiceOne() To: ServiceOne.getName()	1- Call-Instance			
Type: Call – Static method From: CallInstanceMethodIndirect_StaticMethodInstanceMethod Via: BackgroundService.*getServiceOneviaStaticAttribute*() To: ServiceOne.getName()	1-Call-Instance			
Type: Call – Instance method – Inherited (Virtual call) From: CallInstanceMethodIndirect_SuperClass Via: CallInstanceSubClassDOA To: CallInstanceSuperClassDAO.MethodOnSuperClass() **Note:** Only OK if a call to the super class is reported.	1-Call-instance-inherited			To: Technology.Direct.Dao contains ServiceTwo
Type: Call – Instance method – Inherited of 2nd super class From: CallInstanceMethodIndirect_SuperSuperClass Via: CallInstanceSubSubClassDOA and CallInstanceSubClassDOA To: CallInstanceSuperClassDAO.MethodOnSuperClass() **Note:** Only OK if a call to the super class is reported.	1-Call-instance-inherited			To: Technology.Direct.Dao contains ServiceTwo
Type: Call – Instance method From: CallInstanceMethodIndirect_VarMethod Via: BackgroundService.serviceOne To: ServiceOne.getName()	1- Call-Instance			

Type: Call – Instance method – Double indirect From: CallInstanceMethodIndirectIndirect_MethodVarMethod Via: BackgroundService.getServiceTwo() Via: ServiceTwo.serviceOne To: ServiceOne.getName()	1- Call-Instance			
Type: Call – Instance method – Double indirect From: CallInstanceMethodIndirectIndirect_VarVarMethod Via: BackgroundService.serviceTwo Via: ServiceTwo.serviceOne To: ServiceOne.getName()	1- Call-Instance			
Type: Call – Static method From: CallStaticMethodIndirect_MethodStaticMethod Via: BackgroundService.getServiceOne() To: ServiceOne.*getsName*()	1- Call-Instance			
Type: Call – Static method From: CallStaticMethodIndirect_VarStaticMethod Via: BackgroundService.serviceOne To: ServiceOne.*getsName*()	1- Call-Instance			
Inheritance				
Type: Inheritance – Extends -extends From: InheritanceExtendsExtendsIndirect Via: MapsService (extends POI) To: POI	1- Inheritance-extends			
Type: Inheritance – Extends - implements From: InheritanceExtendsImplementsIndirect Via: Whrrl (implements IPreferences) To: IPreferences	1-Inheritance-implements			
Type: Inheritance – Implements - extends From: InheritanceImplementsExtendsIndirect Via: IWhrrl (extends IPreferences) To: IPreferences	1- Inheritance-extends			

3.5.1 Gathered Evidence Indirect Dependencies: Expected

…

3.6 Indirect Dependencies: Test Results - Unexpected Violations

No violations are expected from domain.indirect.allowedfrom to:

- domain.indirect.indirectto
- technology.direct.dao

If an unexpected violation is reported, make a note in the table below.

Test cases	**Dependencies**	**Dependency detected? +(Yes), -(No)**	**Violation reported? +(Yes), -(No)**	**Comment**
Type: From: To:				

3.6.1 Gathered Evidence Indirect Dependencies: Unexpected

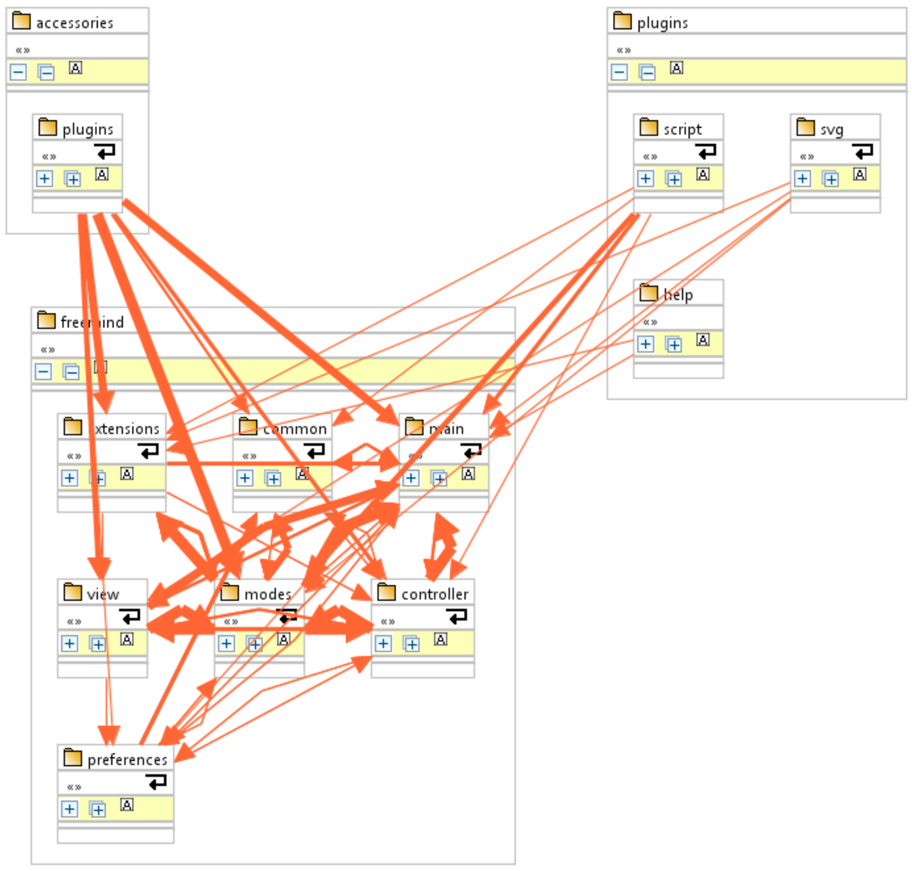
…

4. Freemind Test – Instruction

4.1 Introduction

The Freemind test complements the custom made benchmark test. Freemind, a mind-mapping tool, is a freely available open source system, developed in Java. The Freemind test is aimed at quantitative and qualitative tool evaluation with respect to the accuracy of dependency detection and violation reporting.

The figure below shows the three top-level packages in Freemind: accessories, plugins and freemind. The figure shows also subpackages with their dependency relations, as depicted by the SAVE tool. Thick lines represent more dependency relations than thin lines.



4.1.1 Code Files and Excluded Packages

Freemind version 0.9.0 is used in the test, which we retrieved on 23-08-2012 from http://freemind.sourceforge.net/wiki/index.php/Download.

Since some packages are only available in the source code and not in the compiled version, they are excluded from the test. These packages are: plugins.latex.*, plugins.collaboration.*, and tests.*.

4.1.2 Two Types of Tests

Two types of tests may be executed: 1) a quantitative test to determine the number of reported dependencies and violations at top-package level; 2) a qualitative test to determine the capability of a tool to report the (violating) dependencies within the large class plugins.script.ScriptingEngine to package freemind. This class was selected for the test, since it contains a rich variety of dependency types.

4.2 Quantitative Test at Top-Package Level

1) Download the test code and score sheet of the Freemind test

Extract the zip file with the code: FreemindTest_TestCodeFiles_bin_en_src.zip. The java files as well as the class files are available in the zip.
The relevant Java class files are in: freemind-bin.
The relevant Java source files are in: freemind-src.freemind (accessoires, freemind, plugins).

2) Determine the number of reported dependencies

Analyse the Java source files or class files.
Thereafter:

1. Determine the number of reported dependencies between the following packages:
    a. plugins -> freemind
    b. accessories -> freemind
2. Register the number of dependencies in Table 1.

Note 1: Table 1-3 below are available in a writeable file: Freemind Test – Summary Tables.docx.
Note 2: When options are available to analyze the code at different granularity levels, choose the most detailed option.

3) Define the rules

Specify the following architectural rules:

1. Package accessories is not allowed to use package freemind.
2. Package plugins is not allowed to use package freemind.

Note: Not allowed to use rules can be checked by all the tools we have studied so far, but the way how the rule should be specified may be very tool specific.

4) Activate the Conformance Check

Note: When options are available to analyse the code at different granularity levels, or to report violations to the defined rules at different granularity levels, choose the most detailed option.

5) Register the Number of Violations

Determine the number of reported violations against the defined rules and register them in table 1.

Table 1: Reported dependencies and violations at top-package level

	<insert Tool name here>
Dep: plugins -> freemind	
Dep: accessories -> freemind	
Dep: Total	
Viol: plugins->freemind	
Viol: accessoriess->freemind	
Viol: Total	

4.3 Qualitative Test with Class ScriptingEngine

Objective: Determine the type and number of the reported violating dependencies from class plugins.script.ScriptingEngine to package Freemind.

1) Define the Rules

1. Delete the rules from the previous test.
2. Add the rule: Class plugins.script.ScriptingEngine is not allowed to use package freemind.

Note: This rule, and the rules from the previous test, are no “real” architectural rules (e.g. as defined in a software architecture document of Freemind), but are defined by ourselves in line with the objectives of the tests.

2) Activate the Conformance Check

Run the compliance check and store the results.

3) Analyze the results

1. Open a writeable version of “Freemind Test - Score Form - ScriptingEngine-Violating Dependencies-Version 2.xlsx”. (Note: A read-only version is available in the next section of this document).
2. Map the reported violations to the dependencies as described in the score form.
   - 2.1. Determine which dependencies are covered by the reported violations. Mark them with a “+” (or with “1”) in the column “Violation Reported”. Furthermore, make a note in the column “Reported as” of the type as reported by the tool.
   - 2.2. Determine which dependencies are not reported by the tool. Mark them with a “-” (or with a “0”) in the column “Violation Reported”.

Note: If the violations are reported only at a high level of abstraction (e.g., only at the level of from-class, to-class), while the SACC-tool provides other reports, browsers, et cetera, that show dependency messages at a lower level of abstraction, than make use of these facilities to score.

4) Summarize the results

1. Determine which dependency types are reported and register the results in table 2. If some dependencies of the same type are detected, while others are not, study the differences and determine the cause. We refer to our published studies for examples of differences and causes.
2. Determine the classes in the freemind top package reported as used by class ScriptingEngine. Register them in table 3.

Table 2: Reported dependencies per dependency type

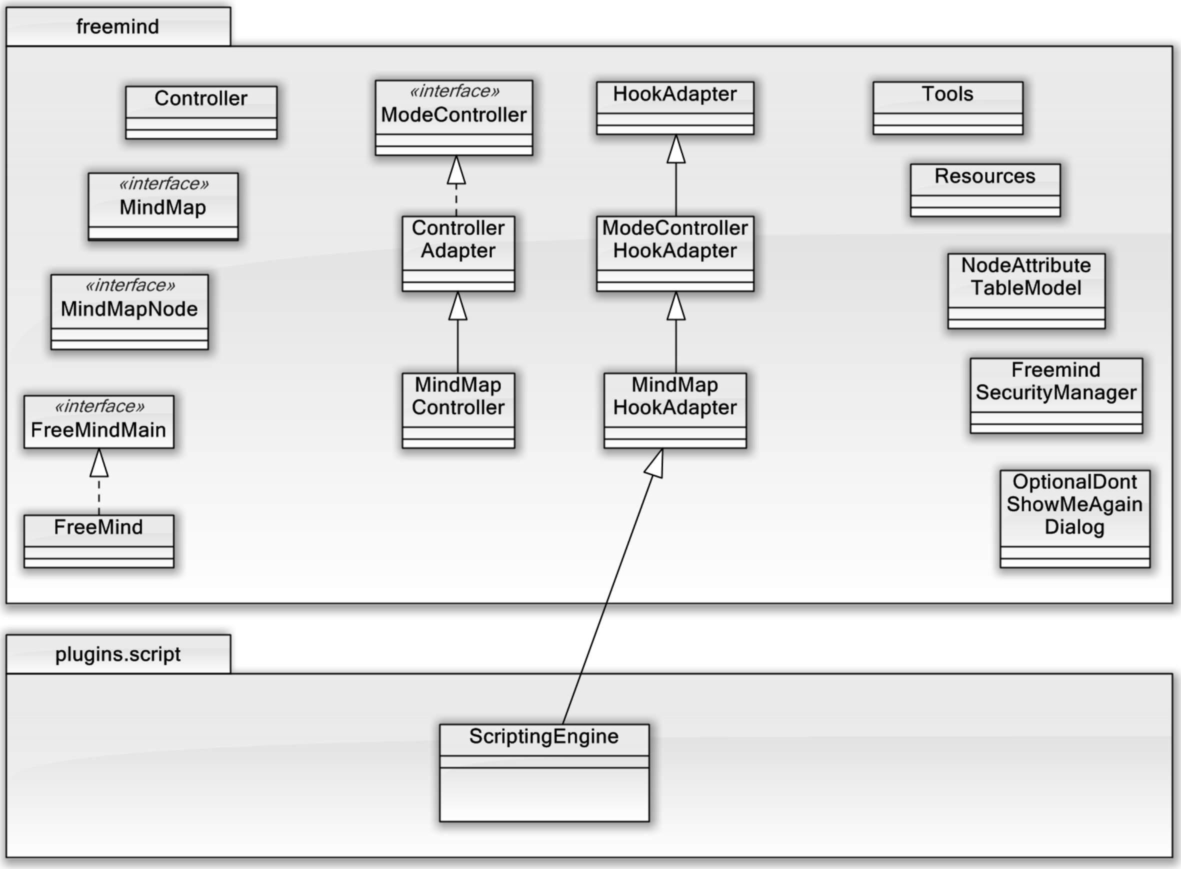
Dependency type (number of constructs)		**Number of Dependencies Reported by Tool X**
Import		
Class import	(10)	
Declaration		
Local variable	(6)	
Parameter	(7)	
Type cast	(2)	
Call		
Instance method	(11)	
Instance method-inherited	(14)	
Class method	(6)	
Constructor	(3)	
Inner class method (instance)	(2)	
Interface method	(19)	
Access		
Constant variable	(12)	
Object reference	(16)	
Inheritance		
Extends class	(1)	
Detected	(109)	
Sensitivity (in %)	(average = 72)	

Table 3: Detected and not-detected depended-upon classes

Depended-Upon Classes	**Number of Dependencies**	**Number of Dependencies Reported by Tool X**
Classes		
freemind.common.OptionalDontShowMeAgainDialog	5	
freemind.controller.Controller	1	
freemind.extensions.HookAdapter	6	
freemind.main.FreeMind	12	
freemind.main.FreeMindMain	16	
freemind.main.FreeMindSecurityManager	5	
freemind.main.Resources	2	
freemind.main.Tools	6	
freemind.modes.attributes.NodeAttributeTableModel	6	
freemind.modes.ControllerAdapter	5	
freemind.modes.MindMap	1	
freemind.modes.mindmapmode.hooks.MindMapHookAdapter	5	
freemind.modes.mindmapmode.MindMapController	6	
freemind.modes.MindMapNode	17	
freemind.modes.ModeController	2	
Inner Classes		
OptionalDontShowMeAgainDialog$StandardPropertyHandler	1	
freemind.main.Tools.BooleanHolder	13	
Total	109	

4.3.1 Scoring Notes

1. ScriptingEngine depends-upon the classes drawn in the freemind package in the diagram below, and on inner classes of these classes. This diagram helps to interpret the reported dependencies or violations, especially in case of a call of an inherited method or an access of an inherited attribute. Two classes contain inner classes, which are also used by ScriptingEngine, namely OptionalDontShowMeAgainDialogue and Tools. Please note that the figure provides a simplified view. There are many more classes in package freemind, and the shown classes are in reality included in different subpackages. Furtermore, for reasons of readability, we have no dependency arrows drawn in the diagram, only UML inheritance relations (generalizations and implementations).



2. We included a method call as one dependency in the score form, but more dependencies will be reported by some tools, e.g. if a tool reports the class reference and the method (or new) as two dependencies.
3. How hard it is to relate violation/dependency messages to the dependencies at code-level depends on the accuracy of the tool-output. When tools report dependencies at from-class, to-class level only, it is not possible to perform this test.
4. We scored mildly in our studies (published in the papers mentioned in the introduction section), meaning that we marked a dependency as detected, if one of the reported dependency messages could be related to the dependency-causing code construct. With a strict accuracy level in mind, the number of missed dependencies would have been higher.
   - In case of inner class related dependencies we scored a dependency also to be detected, if it was reported as a dependency to the outer class instead of to the inner class.
   - In case of inheritance related dependencies we scored a dependency also to be detected, if it was reported as a dependency to a sub class instead of the super class that actually implemented a depended-upon variable or method.
   - In case of dependency messages with a non-optimal accuracy, we scored all dependencies to be detected that could be related to the dependency message. For instance, if a tool reported one dependency to class X of type declaration or access at line Y, while in the source code a declaration construct and a type cast construct were present, both were scored to be detected. Similarly, if a tool reported one dependency to class X of type access in method Z, while in the source code of the method five of these access construct were present, all five were scored to be detected.

5. Freemind Test – Score Form – Read-only Version

plugins.script.ScriptingEngine							
					Violation		
Id	Line	Type	Direct	Target	Reported	Reported as	Comment
1	39	Import		freemind.common.OptionalDontShowMeAgainDialog			freemind.common.OptionalDontShowMeAgainDialog
2	40	Import		freemind.main.FreeMind			freemind.main.FreeMind
3	41	Import		freemind.main.FreeMindMain			freemind.main.FreeMindMain
4	42	Import		freemind.main.FreeMindSecurityManager			freemind.main.FreeMindSecurityManager
5	43	Import		freemind.main.Tools			freemind.main.Tools
6	44	Import		freemind.main.Tools.BooleanHolder			freemind.main.Tools.BooleanHolder (inner class, static)
7	45	Import		freemind.modes.MindMapNode			freemind.modes.MindMapNode
8	46	Import		freemind.modes.attributes.NodeAttributeTableModel			freemind.modes.attributes.NodeAttributeTableModel
9	47	Import		freemind.modes.mindmapmode.MindMapController			freemind.modes.mindmapmode.MindMapController
10	48	Import		freemind.modes.mindmapmode.hooks.MindMapHookAdapter			freemind.modes.mindmapmode.hooks.MindMapHookAdapter
11	58	Inheritance-Extends	Direct	freemind.modes.mindmapmode.hooks.MindMapHookAdapter			extends MindMapHookAdapter
12	68	Call-Instance-Inherited	Indirect	freemind.extensions.HookAdapter			calls super.startupMapHook() in freemind.extensions.HookAdapter.startupMapHook()
13	69	Declaration-Local Variable		freemind.modes.MindMapNode			MindMapNode node = ...
14	69	Call-Instance-Inherited	Indirect	freemind.modes.mindmapmode.hooks.MindMapHookAdapter			getMindMapController() in freemind.modes.mindmapmode.hooks.MindMapHookAdapter (su
15	69	Call-Instance-Inherited	Indirect	freemind.modes.ControllerAdapter			getMindMapController().**getMap()**.getRootNode() in freemind.modes.ControllerAdapter
16	69	Call-Interface	Indirect	freemind.modes.MindMap			getMindMapController().getMap().**getRootNode()** in freemind.modes.MindMap
17	70	Declaration-Local Variable		freemind.main.Tools.BooleanHolder			BooleanHolder booleanHolder = ... from freemind.main.Tools (static class)
18	70	Call-Constructor-Inner class	Direct	freemind.main.Tools.BooleanHolder			... = new BooleanHolder(FALSE)
19	72	Call-Instance-Inherited	Indirect	freemind.extensions.HookAdapter			freemind.extensions.MindMapHook
20	74	Access-Object Reference-Parameter	Direct	freemind.modes.MindMapNode			node from MindMapNode
21	74	Access-Object Reference-Parameter	Direct	freemind.main.Tools.BooleanHolder			booleanHolder from BooleanHolder
22	78	Access-Object Reference-Parameter	Direct	freemind.modes.MindMapNode			node from MindMapNode
23	78	Access-Object Reference-Parameter	Direct	freemind.main.Tools.BooleanHolder			booleanHolder from BooleanHolder
24	82	Declaration-Parameter		freemind.modes.MindMapNode			pNode from MindMapNode
25	82	Declaration-Parameter		freemind.main.Tools.BooleanHolder			pBooleanHolder from BooleanHolder
26	84	Call-Instance-Inherited	Indirect	freemind.extensions.HookAdapter			getPluginBaseClass() from freemind.extensions.HookAdapter
27	85	Call-Class	Direct	freemind.main.Tools			Tools.getFile(...) from freemind.main.Tools (static class)
28	89	Access-Object Reference-Parameter	Direct	freemind.modes.MindMapNode			pNode from MindMapNode in executeScript(**pNode**, pBooleanHolder, ...
29	89	Access-Object Reference-Parameter	Direct	freemind.main.Tools.BooleanHolder			pBooleanHolder from BooleanHolder in executeScript(pNode, **pBooleanHolder**,...
30	90	Call-Instance-Inherited	Indirect	freemind.modes.mindmapmode.hooks.MindMapHookAdapter			getMindMapController() in freemind.modes.mindmapmode.hooks.MindMapHookAdapter

31	96	Declaration-Parameter		freemind.modes.MindMapNode			node from MindMapNode
32	97	Declaration-Parameter		freemind.main.Tools.BooleanHolder			pAlreadyAScriptExecuted from BooleanHolder
33	98	Call-Instance-Inherited	Indirect	freemind.extensions.HookAdapter			**getController()** in freemind.extensions.HookAdapter
34	98	Call-Interface	Indirect	freemind.modes.ModeController			getController().**getFrame()**.setWaitingCursor(true) in freemind.modes.ModeController
35	98	Call-Interface	Indirect	freemind.main.FreeMindMain			getController().getFrame().**setWaitingCursor(true)** in freemind.main.FreeMindMain
36	100	Call-Interface	Direct	freemind.modes.MindMapNode			node.childrenUnfolded()
37	101	Declaration-Local Variable		freemind.modes.MindMapNode			MindMapNode element =
38	101	Declaration-Type cast		freemind.modes.MindMapNode			= (MindMapNode) iter.next();
39	102	Access-Object Reference-Parameter	Direct	freemind.modes.MindMapNode			performScriptOperation(element, pAlreadyAScriptExecuted)
40	102	Access-Object Reference-Parameter	Direct	freemind.main.Tools.BooleanHolder			performScriptOperation(element, pAlreadyAScriptExecuted)
41	104	Declaration-Local Variable		freemind.modes.attributes.NodeAttributeTableModel			NodeAttributeTableModel attributes = from freemind.modes.attributes
42	104	Call-Interface	Direct	freemind.modes.MindMapNode			= node.getAttributes()
43	105	Access-Object Reference	Direct	freemind.modes.attributes.NodeAttributeTableModel			if (attributes == null)
44	107	Call-Instance	Direct	freemind.modes.attributes.NodeAttributeTableModel			attributes.getRowCount()
45	108	Call-Instance	Direct	freemind.modes.attributes.NodeAttributeTableModel			attributes.getName(row)
46	109	Call-Instance	Direct	freemind.modes.attributes.NodeAttributeTableModel			attributes.getValue(row)
47	113	Call-Instance-Inherited	Indirect	freemind.extensions.HookAdapter			getPluginBaseClass() in freemind.extensions.HookAdapter
48	115	Access-Object Reference-Parameter	Direct	freemind.modes.MindMapNode			node from MindMapNode in executeScript(**node**, pAlreadyAScriptExecuted, ...
49	115	Access-Object Reference-Parameter	Direct	freemind.main.Tools.BooleanHolder			pAlreadyAScriptExecuted from BooleanHolder in executeScript(node, **pAlreadyAScriptExecute**
50	116	Call-Instance-Inherited	Direct	freemind.modes.mindmapmode.hooks.MindMapHookAdapter			getMindMapController() in freemind.modes.mindmapmode.hooks.MindMapHookAdapter
51	125	Call-Instance-Inherited	Indirect	freemind.extensions.HookAdapter			getController() in freemind.extensions.HookAdapter
52	125	Call-Interface	Indirect	freemind.modes.ModeController			getController().**getFrame()**.setWaitingCursor(false) in freemind.modes.ModeController
53	125	Call-Interface	Indirect	freemind.main.FreeMindMain			getController().getFrame().**setWaitingCursor(true)** in freemind.main.FreeMindMain
54	148	Declaration-Parameter		freemind.modes.MindMapNode			node from MindMapNode
55	149	Declaration-Parameter		freemind.main.Tools.BooleanHolder			pAlreadyAScriptExecuted from BooleanHolder
56	150	Declaration-Parameter		freemind.modes.mindmapmode.MindMapController			pMindMapController from freemind.modes.mindmapmode.MindMapController
57	153	Declaration-Local Variable		freemind.main.FreeMindMain			FreeMindMain frame = ...
58	153	Call-Instance-Inherited	Indirect	freemind.modes.ControllerAdapter			pMindMapController.getFrame() calls freemind.modes.ControllerAdapter.getFrame()
59	154	Call-Instance-Inner class	Direct	freemind.main.Tools.BooleanHolder			pAlreadyAScriptExecuted.getValue(); from BooleanHolder
60	155	Call-Constructor	Direct	freemind.common.OptionalDontShowMeAgainDialog			new OptionalDontShowMeAgainDialog(frame
61	156	Call-Interface	Direct	freemind.main.FreeMindMain			frame.getJFrame() from FreeMindMain
62	156	Call-Instance-Inherited	Indirect	freemind.modes.ControllerAdapter			pMindMapController.getSelectedView() calls freemind.modes.ControllerAdapter.getSelectedV
63	158	Access-Object Reference-Parameter	Direct	freemind.modes.mindmapmode.MindMapController			pMindMapController
64	159	Call-Constructor-Inner class	Direct	freemind.common.OptionalDontShowMeAgainDialog.StandardPr opertyHandler			new OptionalDontShowMeAgainDialog.StandardPropertyHandler (constructor of inner class)

65	160	Call-Instance-Inherited	Indirect	freemind.modes.ControllerAdapter			pMindMapController.getController() calls freemind.modes.ControllerAdapter.getController()
66	161	Access-Class-Constant	Direct	freemind.main.FreeMind			FreeMind.RESOURCES_EXECUTE_SCRIPTS_WITHOUT_ASKING
67	162	Access-Class-Constant	Direct	freemind.common.OptionalDontShowMeAgainDialog			OptionalDontShowMeAgainDialog.ONLY_OK_SELECTION_IS_STORED
68	163	Call-Instance	Direct	freemind.common.OptionalDontShowMeAgainDialog			**.show()**.getResult(); from OptionalDontShowMeAgainDialog
69	163	Call-Instance	Indirect	freemind.common.OptionalDontShowMeAgainDialog			.show().**getResult()**; from OptionalDontShowMeAgainDialog
70	168	Call-Instance-Inner class	Direct	freemind.main.Tools.BooleanHolder			pAlreadyAScriptExecuted.setValue(true); from BooleanHolder
71	170	Access-Object Reference-Parameter	Direct	freemind.modes.mindmapmode.MindMapController			pMindMapController
72	171	Access-Object Reference-Parameter	Direct	freemind.modes.MindMapNode			node
73	195	Call-Interface	Direct	freemind.main.FreeMindMain			getProperty(...) in freemind.main.FreeMindMain
74	195	Access-Class-Constant	Direct	freemind.main.FreeMind			FreeMind.RESOURCES_EXECUTE_SCRIPTS_WITHOUT_ASKING
75	197	Call-Interface	Direct	freemind.main.FreeMindMain			getProperty(...) in freemind.main.FreeMindMain
76	197	Access-Class-Constant	Direct	freemind.main.FreeMind			FreeMind.RESOURCES_EXECUTE_SCRIPTS_WITHOUT_FILE_RESTRICTION
77	199	Call-Interface	Direct	freemind.main.FreeMindMain			getProperty(...) in freemind.main.FreeMindMain
78	199	Access-Class-Constant	Direct	freemind.main.FreeMind			FreeMind.RESOURCES_EXECUTE_SCRIPTS_WITHOUT_NETWORK_RESTRICTION
79	201	Call-Interface	Direct	freemind.main.FreeMindMain			getProperty(...) in freemind.main.FreeMindMain
80	201	Access-Class-Constant	Direct	freemind.main.FreeMind			FreeMind.RESOURCES_EXECUTE_SCRIPTS_WITHOUT_EXEC_RESTRICTION
81	203	Call-Interface	Direct	freemind.main.FreeMindMain			getProperty(...) in freemind.main.FreeMindMain
82	203	Access-Class-Constant	Direct	freemind.main.FreeMind			FreeMind.RESOURCES_SIGNED_SCRIPT_ARE_TRUSTED
83	212	Call-Class	Direct	freemind.main.Tools			Tools.isPreferenceTrue(...) static method!
84	214	Call-Class	Direct	freemind.main.Tools			Tools.isPreferenceTrue(...) static method!
85	216	Call-Class	Direct	freemind.main.Tools			Tools.isPreferenceTrue(...) static method!
86	217	Call-Class	Direct	freemind.main.Tools			Tools.isPreferenceTrue(...) static method!
87	228	Declaration-Local Variable		freemind.main.FreeMindSecurityManager			FreeMindSecurityManager securityManager from freemind.main.FreeMindSecurityManager
88	228	Declaration-Type cast		freemind.main.FreeMindSecurityManager			(FreeMindSecurityManager) System.getSecurityManager()
89	244	Call-Instance	Direct	freemind.main.FreeMindSecurityManager			securityManager.setFinalSecurityManager(...)
90	252	Call-Instance	Direct	freemind.main.FreeMindSecurityManager			securityManager.setFinalSecurityManager(...)
91	267	Call-Interface	Direct	freemind.main.FreeMindMain			setProperty(...) in freemind.main.FreeMindMain
92	268	Access-Class-Constant	Direct	freemind.main.FreeMind			FreeMind.RESOURCES_EXECUTE_SCRIPTS_WITHOUT_ASKING
93	271	Call-Interface	Direct	freemind.main.FreeMindMain			setProperty(...) in freemind.main.FreeMindMain
94	272	Access-Class-Constant	Direct	freemind.main.FreeMind			FreeMind.RESOURCES_EXECUTE_SCRIPTS_WITHOUT_FILE_RESTRICTION
95	275	Call-Interface	Direct	freemind.main.FreeMindMain			setProperty(...) in freemind.main.FreeMindMain
96	276	Access-Class-Constant	Direct	freemind.main.FreeMind			FreeMind.RESOURCES_EXECUTE_SCRIPTS_WITHOUT_NETWORK_RESTRICTION
97	279	Call-Interface	Direct	freemind.main.FreeMindMain			setProperty(...) in freemind.main.FreeMindMain
98	280	Access-Class-Constant	Direct	freemind.main.FreeMind			FreeMind.RESOURCES_EXECUTE_SCRIPTS_WITHOUT_EXEC_RESTRICTION

99	282	Call-Interface	Direct	freemind.main.FreeMindMain			setProperty(...) in freemind.main.FreeMindMain
100	282	Access-Class-Constant	Direct	freemind.main.FreeMind			FreeMind.RESOURCES_SIGNED_SCRIPT_ARE_TRUSTED
101	308	Call-Class	Direct	freemind.main.Resources			freemind.main.Resources.getInstance()
102	308	Call-Instance	Indirect	freemind.main.Resources			freemind.main.Resources.getInstance().**logException(e2)**;
103	314	Call-Instance-Inherited	Indirect	freemind.modes.ControllerAdapter			pMindMapController**.getController()** calls freemind.modes.ControllerAdapter.getController()
104	315	Call-Instance	Indirect	freemind.controller.Controller			pMindMapController.getController().**errorMessage(...)** in freemind.controller.Controller
105	324	Call-Interface	Direct	freemind.main.FreeMindMain			getResourceString(...) in freemind.main.FreeMindMain
106	328	Call-Instance	Direct	freemind.modes.mindmapmode.MindMapController			pMindMapController.setNodeText(...)
107	328	Access-Object Reference-Parameter	Direct	freemind.modes.MindMapNode			setNodeText(**node**, value.toString()
108	330	Call-Instance	Direct	freemind.modes.mindmapmode.MindMapController			pMindMapController.editAttribute(...)
109	330	Access-Object Reference-Parameter	Direct	freemind.modes.MindMapNode			editAttribute(**node**, ...)
				Total			