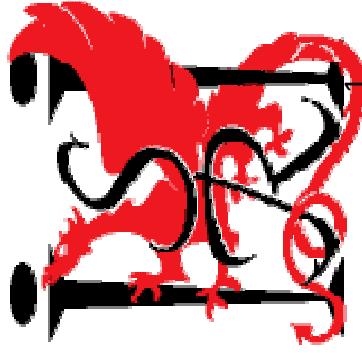


EclipsePro Audit – Evaluation

Team Diversity + Team 13

Adilan Israilov Brian Krausz Majid Alfifi

Mohit Bhonde Raul Véjar



17-654: Analysis of Software Artifacts

Analysis of Software Artifacts - Spring 2009

Agenda

- Introduction – EclipsePro Audit
- Tool description
- Analysis techniques
- Some examples
- Summary



Introduction



- Instantiations
(<http://www.instantiations.com/codenpro index.html>)
- Tool under evaluation EclipsePro Audit 6.0
- 519\$, with 90 days of maintenance
- Eclipse based
- Includes EclipsePro Test (519\$)
- Code analysis functions powered by CodePro engine



What is EclipsePro Audit?

- **Code Analysis**
 - Extensible tool that detect, report and repair deviations or non-compliance with predefined coding standards and style conventions
- **Metrics**
 - Automated tool that measures and report on key quality indicators in a body of Java source code
- **E.g.**
 - Coding styles
 - Dead code
 - Comments
 - Internationalization
 - JUnit & Logging
 - Performance
 - Portability

EclipsePro Setup

The screenshot shows the EclipsePro Instantiations website. At the top, there's a search bar with the placeholder "EclipsePro audit". Below the search bar, there are navigation links for "Home", "Solutions", "New Products", "Alex Products", "Smalltalk Products", "Downloads", "Pricing & Purchase", "Support", "Services", and "Company".

EclipsePro Audit

EclipsePro Test

Tools that make every developer a quality expert

Download EclipsePro Products

Before downloading, be sure to see the [System Requirements](#) and the [License Agreement](#).

After [downloading](#), you'll want to:

- Install the product(s). See [Installation Instructions](#) »
- Activate it (as a free trial or with your purchased Activation Key). See [Product Activation](#) »
- Try it out (for up to 14 days). Start with the [Evaluation Guide](#) »
- Buy it and then put in your permanent Activation Key. See [Pricing & Purchase](#) page »

Download an Installer

Target OS/IDE	Install Type	v 6.6.0 GA 2008.06.26	v 6.6.0 GA 2008.10.23	v 6.6.0 GA 2008.01.25
Windows (any Eclipse version)	full installer	Download Audit	Download Test	Download Audit
Linux (any Eclipse version)	full installer	Download Audit	Download Test	Download Audit
Eclipse 3.2.x		Download Audit	Download Test	Download Audit
Eclipse 3.3.x		Download Audit	Download Test	Download Audit
Eclipse 3.4		Download Audit	Download Test	Download Audit

Installation Tip: Eclipse 3
To avoid common Eclipse 3.x plugin conflicts, delete your Eclipse configuration directory before restarting Eclipse.

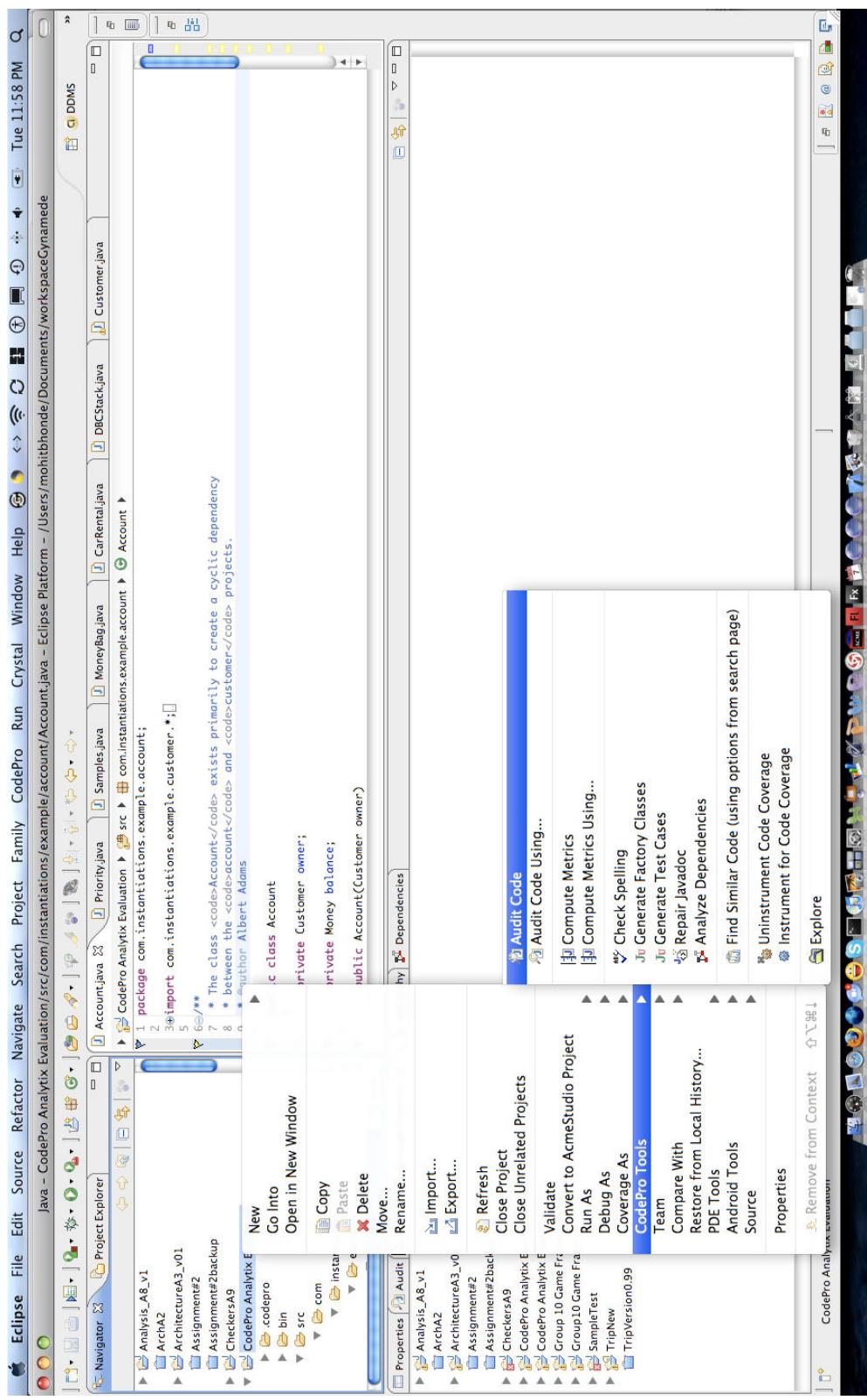
If you are installing into Eclipse 3.0, follow the instructions for installing into Eclipse 3.1, then everything will work fine.

[See full Installation Instructions](#) »

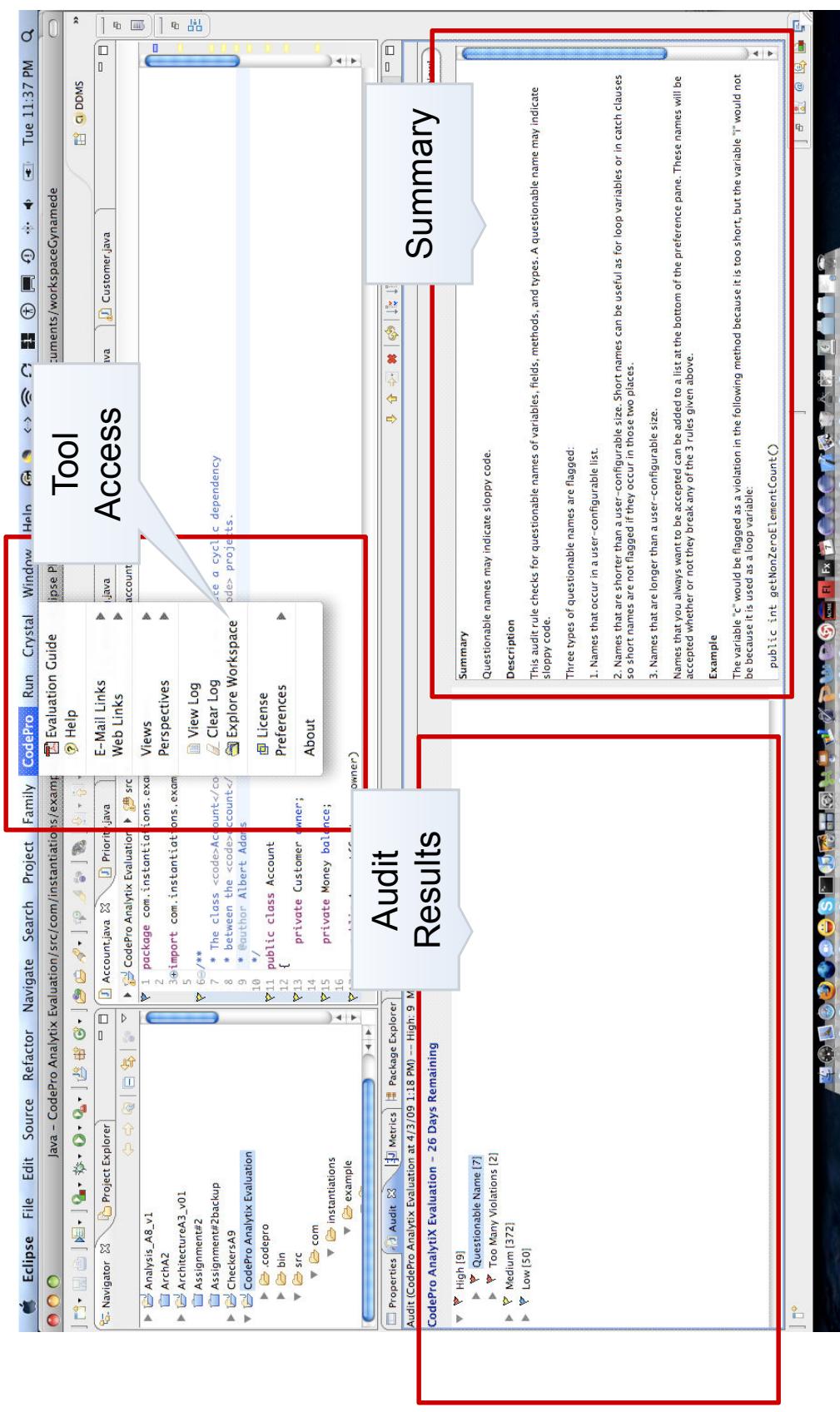
Evaluation Guide
Detailed exercises that you can follow to use EclipsePro to code audit, analyze, and JUnit test case generation. [Download the Zipped Project File](#) (at right to use in the exercises).

Detailed exercises formatted

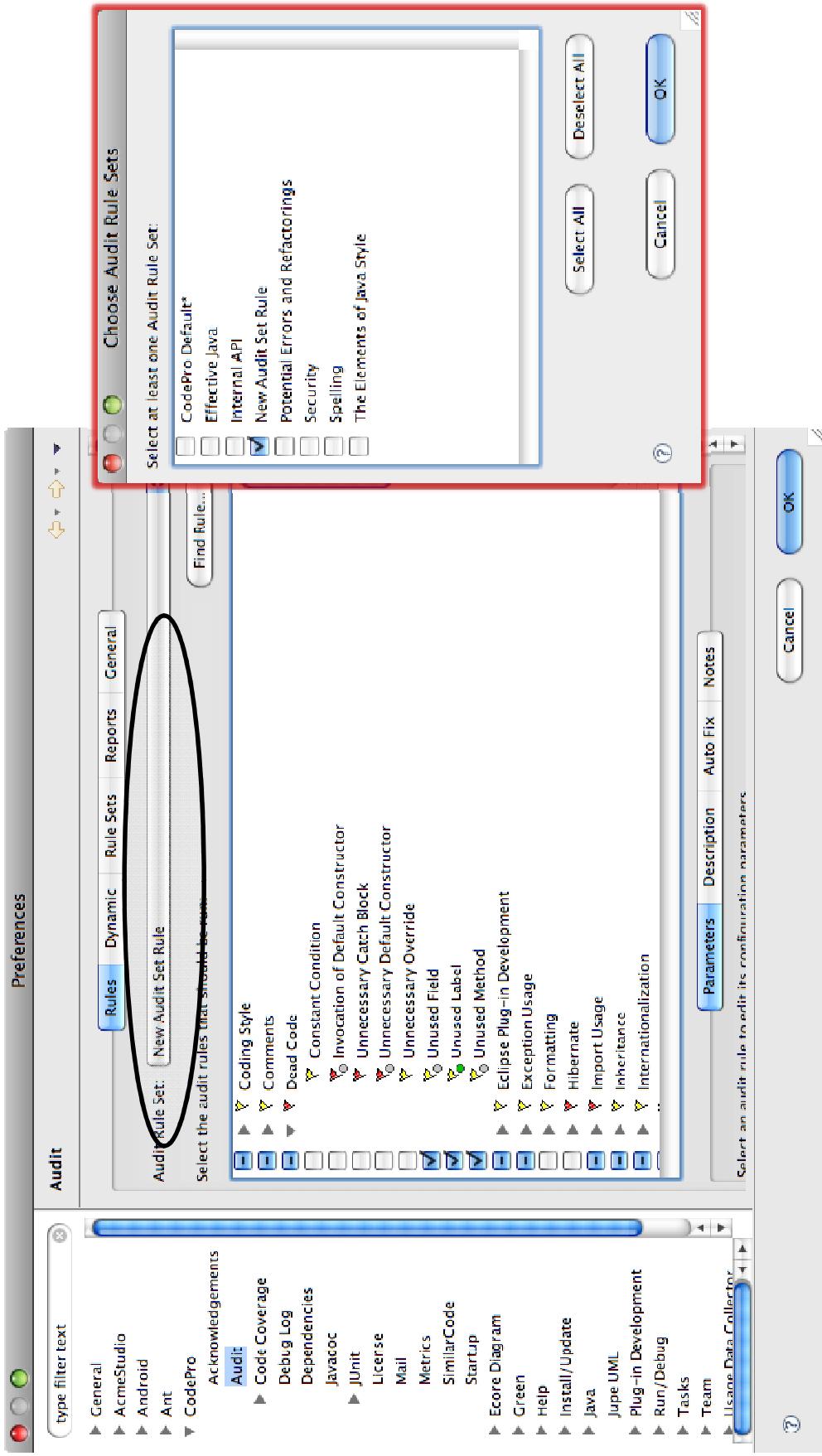
EclipsePro Usage



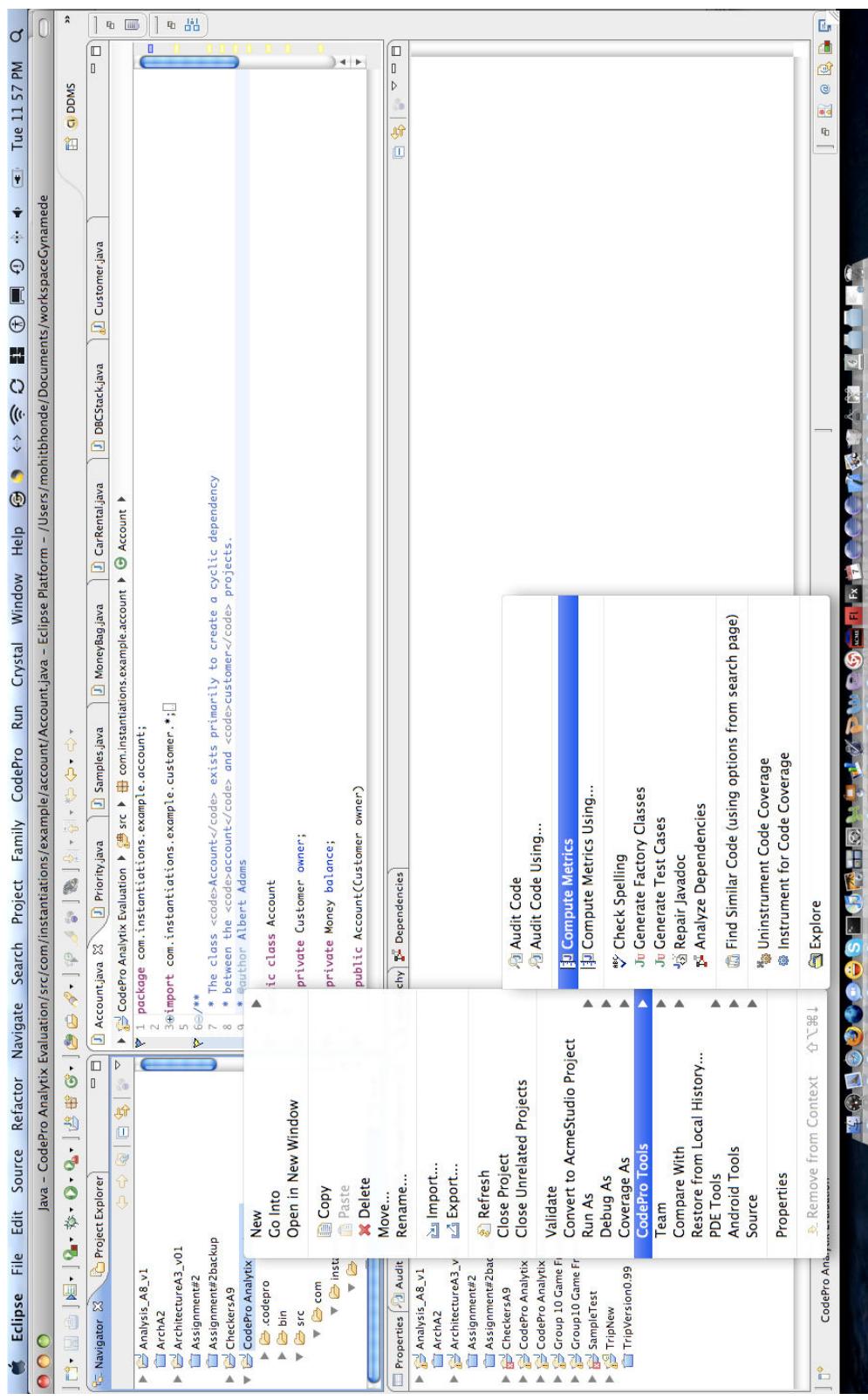
EclipsePro Usage



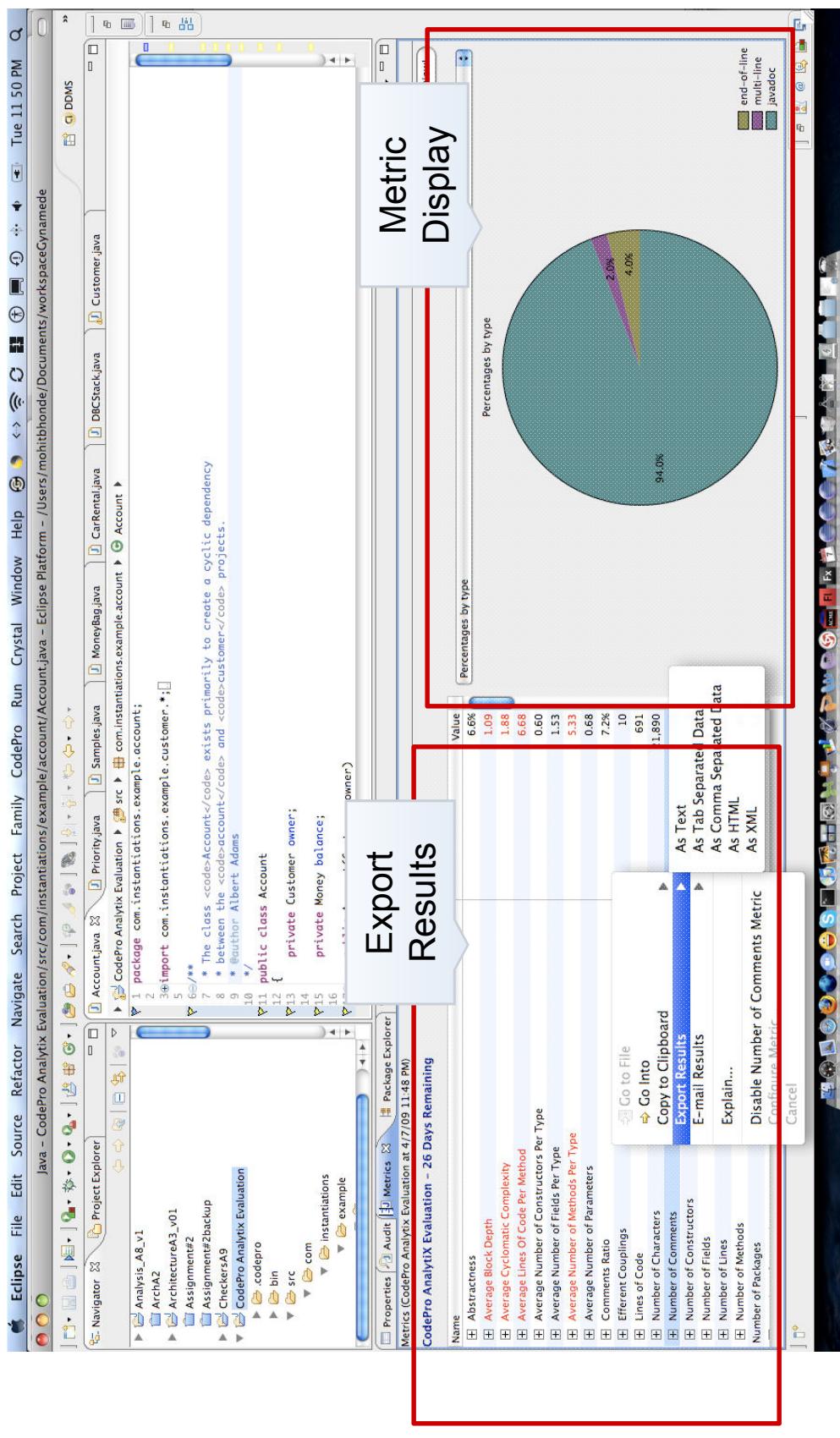
EclipsePro Audit



EclipsePro Audit



EclipsePro Audit



Eclipse Pro Audit

The screenshot shows the Eclipse Pro Audit interface with the following sections:

- CodePro Code Audit Results**: A summary table showing violations by severity and rule group.
- Violation Counts by Severity**: A table showing the count of violations for High, Medium, and Low severities.
- Coding Style**: A table showing violations by audit rule group.
- Avoid Nested Assignments (2)**: A section with two recommendations for nested assignments.
- Conditional Operator Use (1)**: A section with one recommendation for conditional operator use.
- Constants in Comparison (42)**: A section with two recommendations for comparison operators.

Each section includes a "Violation" column, a "Recommendation" column, and a "Resource" column with line numbers.

Violation	Recommendation	Resource	Line
Nested assignment	Move the nested assignment to a separate statement; before the surrounding assignment.	Medium	RPCStack.java 45
Nested assignment	Move the nested assignment to a separate statement; before the surrounding assignment.	Medium	SimpleStack.java 35

Violation	Recommendation	Resource	Line
Use of conditional operator	Avoid using the conditional operator as it makes the code hard to understand.	Low	Priory.java 104

Violation	Recommendation	Resource	Line
Conditional right side of comparison	Reverse the order of the operands.	Medium	CustomerUtil.java 21
Conditional left side of comparison	Reverse the order of the operands.	Medium	BranchingCode.java 13

Eclipse Pro Audit



Eclipse Pro Audit

The screenshot shows the Eclipse Pro Analytix interface. At the top, there's a toolbar with various icons like file, copy, paste, and search. Below the toolbar is a menu bar with options like Inbound, Eclipse Plugin Links, My Music, Router, Google News India, Cricket Live India, Retro Hindi Music Video, Adobe Flex 3.5, OA Staff, CMU, and RT Research.

The main window has tabs for "Violations" and "Source". The "Violations" tab is currently selected and displays 12 violations across 3 categories: 0 high, 12 medium, and 3 low. The violations listed include:

- Missing file comment
- Missing documentation for type CustomerUsage
- Method @Override is missing
- Missing Javadoc comment for method "makeCustomer"
- The string "foo" is not a word
- Missing Javadoc comment for method "setFoo"
- The string "foo" is not a word
- Invalid string literal: "John Doe"
- Invalid string literal: "Sam"
- Invalid string literal: "Sam"
- Invalid string literal: "Hello"

The "Source" tab shows the Java code for CustomerUsage.java:

```
1 package com.instantiations.example.customer;
2
3 /**
4 * The class <code>CustomerUsage</code> exists to show how the test code
5 * generator makes use of factory classes to create instances of objects.
6 * @author Donna Devon
7 */
8 public class CustomerUsage {
9
10    public static String getFoo(Customer customer)
11    {
12        if (customer.getName().equals("John Doe"))
13        {
14            return "bar";
15        }
16        if (customer.getName().equals("Jill"))
17        {
18            return "Hello";
19        }
20    }
21    public static Customer makeCustomer(int a, String b)
22    {
23        if (a == 1)
24        {
25            return null;
26        }
27    }
28}
```



EclipsePro Audit - Analysis

- 3 projects
 - Personal size
 - “Hnefatafl” code base
 - Medium size
 - “AETool” 2 eclipse plugins tested
 - Enterprise size
 - SeisPM



EclipsePro Audit evaluation - AETool

- **Purpose:** To study legacy code for Maintenance, Performance, Correctness
- **Process:** Updated the rules set before audit
- **Artifact Analyzed:**
 - *edu.cmu.archevol.edit*
 - *edu.cmu.archevol.diagram*
- **Results:**

Plug-in #	Execution time (seconds)	LOC	Number of comments	Number of Lines	Number of Methods
1	6	1860	273	3340	140
2	10	12104	1782	18605	907

EclipsePro Audit – Modified RuleSet



Maintainability	Description/Rationale
File comment	Compilation units (class files) should have at least a header comment explaining the purpose of that unit.
String concatenation	Correctness Enforce Singleton property with private constructor Multiple return statement
	Ensure that a class defined as a singleton follows specific rules that disallow multiple instances to be created. Singleton classes should have a single, private constructor and static access to the instance.
	Illegal main method
	Main method should not occur in non-application classes. (might be forgotten to remove testing code)
Float and String comparison	Because of the type specifics should be compared with equality and inequality operators
	clared in a loop oked in a loop
 r condition.
Methods should be static	Methods that do not access any instance state or instance should be static.
Append string	Appending strings with single characters to buffers or streams is slower than appending just the single character.

AETool Evaluation Results



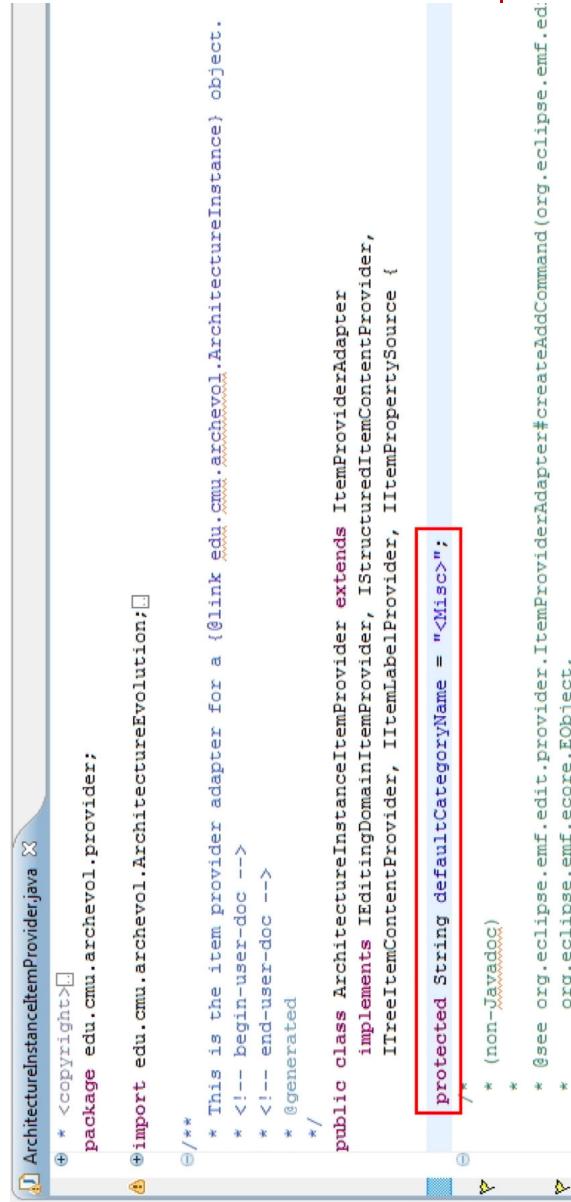
Maintainability	Number of violations detected	
	Plug-in #1	Plug-in #2
File comment	1	107
String concatenation	29	52
Multiple return statements	28	216
Use “for” loop instead of “while” loop	6	
Variable usage	1	1
Code in comments	34	84
Comment local variables	74	752
Unused field, label, method	3	
Missing image file (plug-ins development)		
Undefined property (plug-ins development)		
Exception creation		13
Convert class to Interface		1
Use of instanceof should be minimized	4	175
Add methods to interface		
Close where created		1
Empty catch clauses	4	1
Variable should be final		34
String concatenation in a loop		
Performance		
Variable declared within a loop	7	106
Method invocation in loop condition	3	12
Methods should be static	13	101
Append string		
Define initial capacity	1	52
Define load factor		14
Index Arrays with integers		
Prefers interfaces to reflection		

	Correctness
Enforce Singleton property with private constructor	1
Illegal main method	
Float and String comparison	
Invalid loop construction	65
Empty methods, statements, classes	50
Possible null pointer	52
Recursively call with no check	534
Use == to compare with Null	
Dangling else	1

Some Issues

- Couldn't recognize code comments from text comments.
- Example: No difference between `// Text` and `// System.out.println();`
- Couldn't recognize variables, which should have been declared as constants (final).

Example#1: From the table we can see that under Plug-in#1 column 'Variable should be final' category equals 0, which means EclipsePro Audit was not able to recognize the following code example:



```
ArchitectureInstanceItemProvider.java X
+ * <copyright>
package edu.cmu.archevol.provider;

import edu.cmu.archevol.ArchitectureEvolution;

/*
 * This is the item provider adapter for a {@link edu.cmu.archevol.ArchitectureInstance} object.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated
 */
public class ArchitectureInstanceItemProvider extends ItemProviderAdapter
implements IErasingDomainItemProvider, IStructuredItemContentProvider,
ITreeItemContentProvider, IItemLabelProvider, IItemPropertySource {

protected String defaultCategoryName = "<None>";

/*
 * (non-Javadoc)
 * @see org.eclipse.emf.edit.provider.ItemProviderAdapter#createAddCommand(org.eclipse.emf.ed.
 *      * org.eclipse.emf.ecore.EObject,
```

Some False Positives

- Wrong suggestion for making a field final

The screenshot shows the Eclipse IDE interface. In the center is a code editor window displaying Java code. A specific line of code, `private WrapLabel wrapLabel;`, is highlighted with a red rectangular box. This line is part of a class definition:`static private class TextCellEditorLocator implements CellEditorLocator {
 /** * @generated */
 private WrapLabel wrapLabel;
 /** * @generated */
}`

Below the code editor is the Eclipse Problems view, which lists audit findings. One finding is highlighted with a red rectangular box and points to the same line of code in the editor. The finding reads:

Variable Should Be Final [34]
Private field should be final: container (TransitionCreateCommand.java - Line 3)
Private field should be final: wrapLabel (ArchevolEditPartFactory.java - Line 76)
Private field should be final: label (ArchevolEditPartFactory.java - Line 124)

On the right side of the interface, there is a vertical bar labeled "EclipsePro Audit Evaluation - 14 Days Remaining".



Irrelevant positives (SeisPM)

- Project deals with legacy systems where commands are sent to those systems
- The tool reported this as a cross site scripting security issue
- The team was already aware

Violation	Recommendation	Severity	Resource	Line
Cross-Site Scripting	User data should never directly be put onto a web site, the path should be eliminated.	High	ExecRemoteCmd.java	60



EclipsePro Audit - Advantages

- Simple to use “out of the box”, developers & managers
- The “Explain” feature helps understand the audit rule and color-coded flags for severities and categories.
- Metrics for projects (much welcomed in Eclipse Setting)
- Customization of rules
- Use the tool in dynamic setting
- Rules set helps sharing knowledge, enforcing standards
- 350 quick fix options



EclipsePro Audit - Disadvantages

- Some rules are partially redundant, which can get bothersome. Example: Dangling Else and Missing Block often go hand-in-hand
- Autofix function is on by default, which might cause unexpected behavior of analyzed project in after the first run of EclipsePro Audit.
- Not very often, but it makes a mistakes. Usually in case when you need to predict usage of entity, like “Variable should be declared as final” audit rule.

In Summary

- Useful tool for encouraging teams to use a uniform coding style
 - Discretion advised
 - For actually identifying errors
 - Not a free tool
- Can be adopted as a developers tool in the small / large scale development
 - Audit
 - Enforcing standards





Thank You!

Questions?