JUnit and Coverage Assessment
DTP Connectivity
Codign Software

By: Joe Ponczak
www.codign.com

Objectives

- Assess current JUnit test cases
- Build additional JUnit tests cases using CoView
- Educate committers on appropriate static and dynamic metrics
- Measure path and branch coverage (Pre- and Post-CoView implementation)
- Determine value proposition for adoption among DTP committers

Benefits

- Improve unit test results with less resources
- Leverage existing software assets
- Reduce unit test churn
- Improve code design and maintainability
- Reduce code complexity
- Implement an automation program
- Provide a unit testing audit trail

Code Analyzed

Package: org.eclipse.datatools.connectivity
Package: org.eclipse.datatools.connectivity.ui
DTP Version: 1.0
IDE: Eclipse 3.2 and 3.2.1

Definitions

- **Paths (also known as Cyclomatic Complexity)**: Identifies the number of paths within a method. A path is the combination of branch outcomes from the top of the method to the bottom. Ideally used to measure reliability and determine the minimal set of unit tests needed.
- **Path Coverage**: Identifies the number of executed paths within a method. More effective than branch or code coverage.
- **Branches:** Identifies the number of decision outcomes in a method. Each decision has two outcomes (TRUE or FALSE) in addition to the entry of that method. A method with no decisions still has 1 branch. Switch statements result in one branch for each switch.
- **Branch Coverage**: Identifies the number of executed branches within a method.
- **Detailed difference between Path and Branch coverage is available here.**

Results

Pre-CoView

Package	Tested Paths	% Path Cov	Tested Branches	% Branch Cov
Org.eclipse.datatools.connectivity	59	38.31%	145	65.02%
Org.eclipse.datatools.connectivity.ui	6	66.67%	10	83.33%
Overall Coverage	65	39.88%	155	65.96%

Total JUnit Test Cases: 65

Post-CoView

Package	Tested Paths	% Path Cov	Tested Branches	% Branch Cov
Org.eclipse.datatools.connectivity	96	57.83%	195	80.25%
Org.eclipse.datatools.connectivity.ui	6	66.67%	10	83.33%
Overall Coverage	102	58.29%	205	80.39%

Total JUnit Test Cases: 105

Overall Increase Using CoView:

- Path Coverage: 46.2% increase
- Branch Coverage: 21.87% increase
- JUnit: 40 additional JUnit tests created (61% increase)

Cost:

- 6 hours to complete
- $288 (assuming $100,000 salary with 2080 work hours per year)
-

Additional

- **Some of the methods contain unrealizable paths, as identified by CoView:**

TemplateDescriptor.hasVisibleProperties, the templateprops.length value is checked twice -- once in the IF statement and once in the condition of the FOR statement. If length > 0 is TRUE in the IF, it can't be FALSE in the FOR. So it would be desirable to remove the condition from the IF statement, and reduce the cc by one.

- **Test methods have conditional logic, which could lead to false positives:**

ConnectionProfileManagerUITests.testGetWizardCategory When I run this test, keys.size() returns 0, which means that the method under test (getWizardCategory) is never invoked. The test is recorded as a successful run, leading you to think the method has been tested. Using our coverage shows otherwise.

- **Some methods have a moderately high Cyclomatic Complexity:**

DriverManager.getFullJarList() A high Cyclomatic Complexity makes it harder to increase coverage on these methods, since (a) it's less obvious to see how to drive down a particular path and (b) even if you can see what data is needed to drive down a path, it becomes tedious and time-consuming to create the data that will hit each decision just so.

- **Some methods use constructs that make it difficult to set up a test state.**

getFullJarList() loads property sets from a hard-coded file name. It would be easier to test if the file name was a parameter, allowing you to specify a test file instead.

Value to DTP Project Management & Committers

- **Objective measurements for JUnit tests**
- **Increase coverage of existing code**
- **Identify unrealizable paths**
- **Historical coverage analysis**
- **Easily add tests to existing code base**
- **Test logic of new code being designed**
- **Flush out design flaws early in the development process**
- **Develop "testable" code**

DTP Project Management & Committers Implementation Costs

- CoView Product: Free – Codign Software will donate one license per DTP committer, with full support.
- Committer Training: Approximately 2 – 3 hours via webinar
- PM Training: Approximately 2 hours via webinar

Why Codign Software?

Unit testing is an important part of any development effort, yet current approaches are often manual and time consuming with little regards to design and consistency. Poor unit testing techniques and tools result in uncaught defects, overly complex code and increased maintenance efforts. Codign Software's approach to unit testing is based on a combination of proven techniques and tools that provide meaning and consistency to developers and management.

About Codign Software

Codign Software offers training, consulting and tools for unit testing professionals. Our staff has extensive experience in software development, unit testing, and parsing technologies and methodologies.

Codign Software is committed to providing companies with excellent solutions comprised of consultants, processes and tools that foster better development, repeatable processes and audit results.

Contact Information

Joe Ponczak, CEO and Co-Founder
Joe.ponczak@codign.com
410.908.5201

Codign Software
www.codign.com
516 N. Charles St. Suite 405
Baltimore, MD 21201

Report Details

Package	Class	Method	# Tested Paths	% Path Cov	# Tested Branches	% Branch Cov
org.eclipse.datatools.connectivity						
	ProfileManager					
		getRootCategories()	1	100.00%	1	100.00%
		getInstance()	1	50.00%	2	66.67%
		getProfiles()	1	100.00%	1	100.00%
		getCategory(java.lang.String)	1	100.00%	1	100.00%
		getProfilesByCategory(java.lang.String)	1	100.00%	1	100.00%
		getProfileByName(java.lang.String)	1	100.00%	1	100.00%
		getProfileByInstanceID(java.lang.String)	1	100.00%	1	100.00%
		getProfileByProviderID(java.lang.String)	1	100.00%	1	100.00%
		createProfile(java.lang.String,java.lang.String,java.lang.String,java.util.Properties,java.lang.String)	1	100.00%	1	100.00%
		createProfile(java.lang.String,java.lang.String,java.lang.String,java.util.Properties,java.lang.String,boolean)	1	100.00%	1	100.00%
		duplicateProfile(org.eclipse.datatools.connectivity.IConnectionProfile)	1	100.00%	1	100.00%
		addProfile(org.eclipse.datatools.connectivity.IConnectionProfile)	1	100.00%	1	100.00%
		addProfile(org.eclipse.datatools.connectivity.IConnectionProfile,boolean)	1	100.00%	1	100.00%
		deleteProfile(org.eclipse.datatools.connectivity.IConnectionProfile)	1	100.00%	1	100.00%
		modifyProfile(org.eclipse.datatools.connectivity.IConnectionProfile)	1	100.00%	1	100.00%
		modifyProfile(org.eclipse.datatools.connectivity.IConnectionProfile,java.lang.String,java.lang.String)	1	100.00%	1	100.00%

Package	Class	Method	# Tested Paths	% Path Cov	# Tested Branches	% Branch Cov
		modifyProfile(org.eclipse.datatools.connectivity.IConnectionProfile,java.lang.String,java.lang.String,java.lang.Boolean)	1	100.00%	1	100.00%
		addProfileListener(org.eclipse.datatools.connectivity.IProfileListener)	1	100.00%	1	100.00%
		removeProfileListener(org.eclipse.datatools.connectivity.IProfileListener)	1	100.00%	1	100.00%
		getAdapter(java.lang.Class)	1	100.00%	1	100.00%
	Class Coverage		**20**	**95.24%**	**21**	**95.45%**
Package Coverage			**20**	**95.24%**	**21**	**95.45%**

Package	Class	Method	# Tested Paths	% Path Cov	# Tested Branches	% Branch Cov
org.eclipse.datatools.connectivity.drivers						
	DriverInstance					
		getJarListAsArray()	1	25.00%	4	57.14%
	Class Coverage		**1**	**25.00%**	**4**	**57.14%**
	DriverManager					
		getInstance()	1	50.00%	2	66.67%
		createNewDriverInstance(java.lang.String,java.lang.String,java.lang.String)	4	100.00%	7	100.00%
		getDriverInstanceByID(java.lang.String)	1	20.00%	5	71.43%
		getFullJarList()	0	0.00%	13	61.90%
		removeDriverInstance(java.lang.String)	0	0.00%	8	72.73%
		resetDefaultInstances()	0	0.00%	1	5.26%
	Class Coverage		**6**	**12.50%**	**36**	**52.94%**
Package Coverage			**7**	**13.46%**	**36**	**53.33%**

Package	Class	Method	# Tested Paths	% Path Cov	# Tested Branches	% Branch Cov
org.eclipse.datatools.connectivity.drivers.models						
	CategoryDescriptor					
		getCategoryDescriptors()	2	100.00%	3	100.00%
		getCategoryDescriptor(java.lang.String)	3	75.00%	7	100.00%
		getRootCategories()	2	66.67%	5	100.00%
		getParent()	2	50.00%	6	85.71%
		getChildCategories()	2	50.00%	7	100.00%
		getAssociatedDriverTypes()	1	25.00%	6	85.71%
		getId()	1	100.00%	1	100.00%
		getParentCategory()	1	100.00%	1	100.00%
		getElement()	1	100.00%	1	100.00%
		getName()	3	100.00%	5	100.00%
		getDescription()	2	100.00%	3	100.00%
		compareTo(java.lang.Object)	2	100.00%	3	100.00%
		toString()	1	100.00%	1	100.00%
		CategoryDescriptor(org.eclipse.core.runtime.IConfigurationElement)	1	100.00%	1	100.00%
		createCategoryDescriptors(org.eclipse.core.runtime.IConfigurationElement[])	3	60.00%	7	77.78%
	Class Coverage		**27**	**71.05%**	**57**	**93.44%**
	TemplateDescriptor					
		getParent()	2	66.67%	5	100.00%
		getName()	3	100.00%	5	100.00%
		compareTo(java.lang.Object)	2	100.00%	3	100.00%
		getDriverTemplateDescriptors()	1	50.00%	2	66.67%

Package	Class	Method	# Tested Paths	% Path Cov	# Tested Branches	% Branch Cov
		getDriverTemplateDescriptor(java.lang.String)	2	50.00%	7	100.00%
		getJarList()	2	25.00%	4	44.44%
		getCreateDefaultFlag()	2	100.00%	3	100.00%
		getEmptyJarListIsOKFlag()	2	100.00%	3	100.00%
		getProperties()	3	100.00%	5	100.00%
		getPropertyValue(java.lang.String)	4	100.00%	7	100.00%
		getPropertyIDFromName(java.lang.String)	4	100.00%	7	100.00%
		hasVisibleProperties()	5	83.33%	11	100.00%
		TemplateDescriptor(org.eclipse.core.runtime.IConfigurationElement)	1	100.00%	1	100.00%
		createDriverTemplateDescriptors(org.eclipse.core.runtime.IConfigurationElement[])	3	60.00%	7	77.78%
		getId()	1	100.00%	1	100.00%
		getParentCategory()	1	100.00%	1	100.00%
		getElement()	1	100.00%	1	100.00%
		getDescription()	2	100.00%	3	100.00%
	Class Coverage		**41**	**75.93%**	**76**	**90.48%**
Package Coverage			**68**	**73.91%**	**76**	**91.72%**

Package	Class	Method	# Tested Paths	% Path Cov	# Tested Branches	% Branch Cov
org.eclipse.datatools.connectivity.internal						
	ConnectivityPlugin					
		getDefault()	1	100.00%	1	100.00%
	Class Coverage		**1**	**100.00%**	**1**	**100.00%**
Package Coverage			**1**	**100.00%**	**1**	**100.00%**
org.eclipse.datatools.connectivity			**96**	**57.83%**	**195**	**80.25%**

Package	Class	Method	# Tested Paths	% Path Cov	# Tested Branches	% Branch Cov
org.eclipse.datatools.connectivity.internal.ui						
	ConnectionProfileManagerUI					
		getInstance()	1	100.00%	1	100.00%
		getWizardCategories()	1	50.00%	2	66.67%
		getNewWizards()	1	50.00%	2	66.67%
		getWizardCategory(java.lang.String)	0	0.00%	1	100.00%
		getNewWizard(java.lang.String)	2	100.00%	3	100.00%
	Class Coverage		**5**	**62.50%**	**9**	**81.82%**
	ConnectivityUIPlugin					
		getDefault()	1	100.00%	1	100.00%
	Class Coverage		**1**	**100.00%**	**1**	**100.00%**
Package Coverage			**6**	**66.67%**	**1**	**83.33%**
org.eclipse.datatools.connectivity.ui			**6**	**66.67%**	**10**	**83.33%**
Total			**102**	**58.29%**	**205**	**80.39%**