Schnellstartanleitung

Osama El Hosami

29. Juni 2010

Inhaltsverzeichnis

Abkürzungsverzeichnis

RTT Refactoring Tool Tester

SVN Subversion

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Der RTT ist ein Test-Framework für die Entwicklung von Refaktorisierungswerkzeugen in Eclipse, das ein zu testendes Refaktorisierungswerkzeug auf alle möglichen Programmelemente eines Testprojekts anwendet und nach jeder Anwendung mittels vorhandener Testfälle prüft, ob das Programm immer noch dieselbe Bedeutung hat. Dabei wird ein Refaktorisierungswerkzeug sowohl syntaktischen, als auch semantischen, Überprüfung unterzogen. Hierfür stellt das RTT Framework sogenannte Orakel bereit, die diese Überprüfungen übernehmen. Die syntaktische Überprüfung besteht aus einer Überprüfung der Kompilierbarkeit der Testprojekt-Quellen, ggf. vor und nach einer Refaktorisierung und wird durch das 'Java Compile Oracle' übernommen. Die semantische Überprüfung erfolgt durch Ausführung von im Testprojekt enthaltenen JUnit-Testfällen vom 'JUnit Oracle'. Für den RTT lassen sich damit folgende notwendige Eingabeelemente unterscheiden:

- Testprojekte - im Sprachgebrauch des RTT Mock-Projekte. Ein oder mehrere Eclipse Java-Projekte auf denen verschiedene Refaktoring-Testfälle ausgeführt werden können. Für die semantische Überprüfung können JUnit-Testfälle enthalten sein.
- Refaktoring Testprojekt - ein Eclipse Plug-in Projekt, in dem die Testfälle des zu testenden Refaktorisierungswerkzeugs enthalten sind.
- Testkonfiguration - die Konfiguration, die die zu verwendenden Mock- und Testprojekte für einen Testlauf ausweist und zusätzliche Ausführungsparameter beinhaltet.

2 Installation und Deinstallation

2.1 Auslieferung

Zu der Auslieferung des RTT gehören die nachfolgenden Feature, die über die RTT Update-Site installiert werden können.

- RTT Core - beinhaltet das Framework des RTT zur Ausführung von Testadaptern und die Basis Orakel zur syntaktischen und semantischen Überprüfung von Mock-Projekten.
- RTT SVN Connector - stellt die Dienste zur Anbindung an ein SVN Repository.
- RTT UI - beinhaltet das Benutzerinterface des RTT und stellt einen Testadapter zur Ausführung von Refaktoring-Tests mittels Introspektion.
- RTT JQuery Test Adapter (Optional) - spezialisierter Testadapter zur Ausführung von Refaktoring-Tests mittels Introspektion. Bietet dem Tester die Möglichkeit Testeingaben durch Abfragen, in einer Prolog ähnlichen Syntax, zu selektieren.

2.2 Voraussetzungen

Die Installation des RTT setzt folgende Umgebung voraus:

- Eclipse Java Development Tools 3.5 [1]
- Subversive SVN Team Provider (Incubation) 0.7 [1]
- Subversive SVN Connectors 2.2 [4]
- SVNKit 1.3 Implementation [4]
- JQuery Backend Plug-in 4.0.3 [7] (Optional)
- TortoiseSVN for Windows [5] (Optional)

Nach Installation von Eclipse können die SVN Feature über die entsprechenden Update-Sites installiert werden. Um SVN nutzen zu können muss allerdings ein SVN-Server installiert werden. Für ein Windows basiertes Zielsystem wird, zur einfachen Erzeugung und Verwaltung der SVN Repositories, der TortoiseSVN Client vorgeschlagen, mit dessen Installation auch ein SVN-Server bereitsteht. Unter der Ubuntu Linux Distribution steht SVN über das Software Repository bereit [6]. Zur Installation des JQuery Plug-in steht keine Update-Site zur Verfügung, bitte die Installationshinweise des Herstellers beachten.

2.3 Installation

Der RTT kann, in der o.g. Umgebung, einfach über den Eclipse Update-Manager installiert werden. Hierzu ist es nur notwendig dem Update-Manager den Ort der lokalen RTT Update-Site mitzuteilen.

Zur Installation des RTT:

1. Eclipse starten und das Menü 'Help > Install New Software...' auswählen.

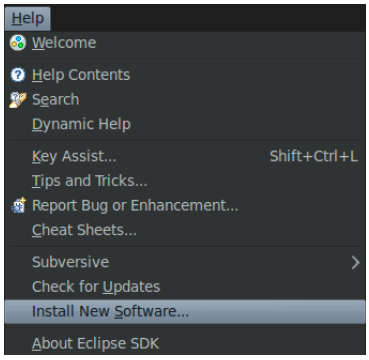

Abbildung 2.1: Software Updates

2. Über 'Add > Local' das Hauptverzeichnis der RTT-Update-Site eintragen und die gewünschten RTT-Feature auswählen.

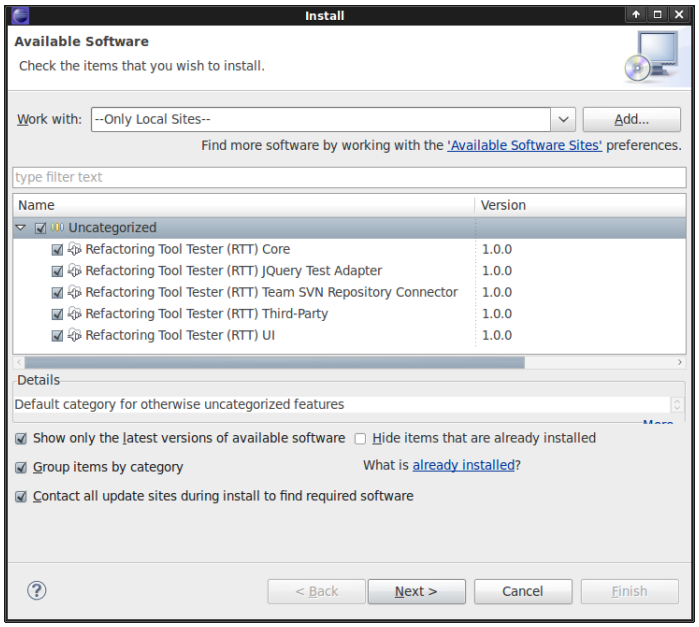

Abbildung 2.2: Installation und Feature-Auswahl

3. Mit 'Next' die Auswahl bestätigen und nach Annahme der Lizenzbedingungen, die Installation einleiten 'Finish'.

4. Nach erfolgreicher Installation Neustart bestätigen 'Yes'.

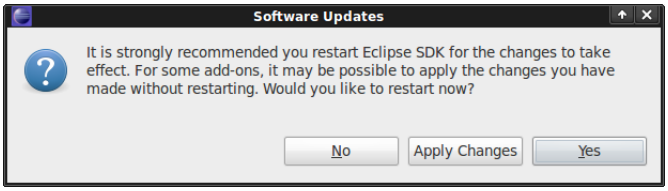

Abbildung 2.3: Installation abschließen

2.4 Deinstallation

Zur Deinstallation der RTT Feature:

1. Eclipse starten und das Menü 'Help > About Eclipse...' auswählen, um zu den Installationsdetails zu gelangen.

Abbildung 2.4: About Eclipse und Installationsdetails

2. Aktion 'Installations Details' bestätigen und in der Liste 'Installed Software' die bereits installierten RTT-Feature selektieren. Eine Mehrfachauswahl ist über STRG möglich.

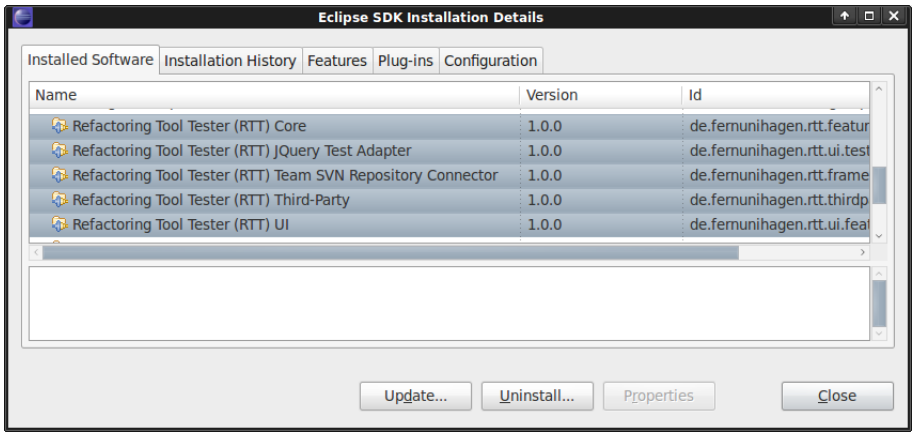

Abbildung 2.5: Installationsdetails

3. Über 'Uninstall' die Auswahl bestätigen.

4. Die Deinstallation über die Aktion 'Finish' abschließen.

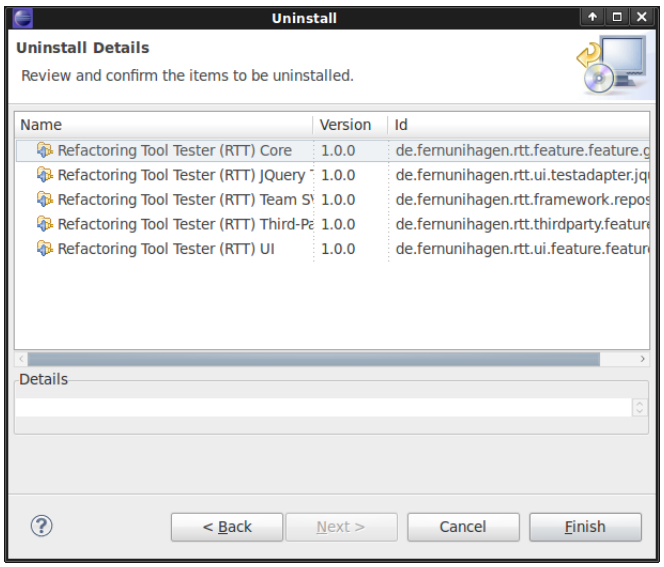

Abbildung 2.6: Deinstallation

5. Nach erfolgreicher Deinstallation Neustart bestätigen 'Yes'.

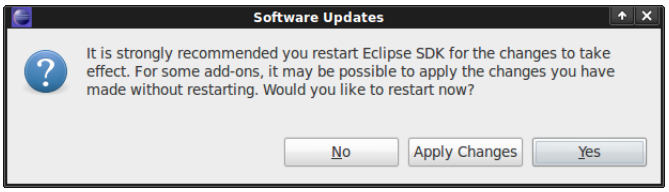

Abbildung 2.7: Deinstallation abschließen

3 Erstellung eines Testprojekts

Testfälle werden im RTT Framework durch Testadapter bereitgestellt. Das Konzept der Testadapter ist dabei so gewählt, dass sowohl ein einzelner Testfall, als auch eine Menge von Testfällen, durch einen Testadapter repräsentiert werden kann. Zwei Implementierungen von Testadaptern sind bereits Teil des RTT Frameworks, beide nutzen die Java Introspektion um annotierte Testfälle auszuführen. Die Deklaration von Testfällen vereinfacht sich somit auf die Annotation der entsprechenden Testmethoden. Ähnlich zu der Deklaration von Testfällen im JUnit 4 Testing Framework, existieren im RTT Framework entsprechende Annotationen. Die bereitgestellten Testadapter sind:

- der 'Java Refactoring Testadapter'
- und der 'JQuery Refactoring Testadapter'

Der JQuery Refaktoring Testadapter unterscheidet sich zum Java Refaktoring Testadapter lediglich in der Art der Bereitstellung von Testeingaben - der Tester kann Abfragen, in einer Prolog ähnlichen Syntax, zur Selektion von Testeingaben spezifizieren. Die Tupel der aus der Abfrage resultierenden Ergebnismenge, werden über die deklarierten Testmethoden-Parameter bereitgestellt. Es folgt eine Auflistung der im Framework definierten Annotationen und der Programmelemente, auf denen diese anwendbar sind:

Annotation	Programmelement	Bedeutung
@Test	Öffentliche Methode	Markiert einen Testfall und spezifiziert ggf. die Abfrage zur Selektion der Testeingaben
@TestParam	Methoden-Parameter	Markiert einen Platzhalter für eine Variable aus der spezifizierten Abfrage
@TestSession	Feld	Markiert einen Platzhalter zur Anforderung von Sitzungsinformationen

Tabelle 3.1: Übersicht der Annotationen

Um das Bild zu vervollständigen folgt eine Darstellungen möglicher Testfall-Signaturen, die die Anwendung der o.g. Annotationen verdeutlicht - aber keineswegs vollständig ist. Weitere Details und Hintergründe zur Implementierung von Testadaptern und Testfällen, werden in den Masterarbeiten [2][3] beschrieben. Abbildung 3.1 zeigt exemplarisch die Definition verschiedener Testfälle, zur Ausführung mit dem Java Refaktoring Testadapter. Analog zeigt Abbildung 3.2 die Definition verschiedener Testfälle für den JQuery Refaktoring Testadapter und deutet die Möglichkeiten der Abfragen zur Selektion von

```
public class TestsForUseWithJavaRefactoringTestAdapter {
   @TestSession
   private TestClientSession session;

   @Test(description="A test case without any parameters")
   public TestProcedureStep testCaseWithOutParams() {
      IProject[] projects = session.getScope();
      Refactoring refactoring = ...
      ...
      return TestProcedureStep.create(refactoring);
   }

   @Test()
   public TestProcedureStep testCaseWithJavaProjectParam(IJavaProject[] projects) {
      Refactoring refactoring = ...
      ...
      return TestProcedureStep.create(refactoring);
   }

   @Test()
   public TestProcedureStep testCaseWithProjectParam(IProject[] projects) {
      Refactoring refactoring = ...
      ...
      return TestProcedureStep.create(refactoring);
   }

   @Test()
   public TestProcedureStep testCaseWithAdaptableParam(IAdaptable[] projects) {
      Refactoring refactoring = ...
      ...
      return TestProcedureStep.create(refactoring);
   }
}
```

Abbildung 3.1: Signaturen von Testfällen, Java Refaktoring Testadapter

Testeingaben an. Wie in beiden Abbildungen gezeigt, macht das RTT Framework keinerlei Vorgaben bezüglich des Aufbaus bzw. der Struktur von Testfällen. Sowohl die Namensgebung, Vererbungshierarchie als auch die Beziehung zwischen Testobjekten sind frei wählbar.

```
public class TestsForUseWithJQueryRefactoringTestAdapter {
   @TestSession
   private TestClientSession session;

   @Test(description="A test case without any parameters")
   public TestProcedureStep testCaseWithoutParams() {
      IProject[] projects = session.getScope();
      Refactoring refactoring = ...
      ...
      return TestProcedureStep.create(refactoring);
   }
   @Test("class(?C)")
   public TestProcedureStep testCaseWithClassParam(@TestParam("?C") IType type) {
      Refactoring refactoring = ...
      ...
      return TestProcedureStep.create(refactoring);
   }
   @Test("class(?C),field(?C,?F)")
   public TestProcedureStep testCaseWithFieldParam(@TestParam("?F") IField field) {
      Refactoring refactoring = ...
      ...
      return TestProcedureStep.create(refactoring);
   }
   @Test("class(?C),method(?C,?M)")
   public TestProcedureStep testCaseWithMemberParam(@TestParam("?C") IType type, @TestParam("?M") IMember member) {
      Refactoring refactoring = ...
      ...
      return TestProcedureStep.create(refactoring);
   }
}
```

Abbildung 3.2: Signaturen von Testfällen, JQuery Refaktoring Testadapter

Nachfolgend wird die Vorgehensweise zum Erzeugen eines Testprojekts für die o.g. Refaktoring Testadapter, am Beispiel eines Testfalls für das Rename Refaktoringwerkzeug, vorgestellt. Die vollständige Implementierung des Beispiels ist dem beigefügten Testprojekt-Archiv zu entnehmen. Die Vorgehensweise zum Import eines existierenden Projektes aus einem Projekt-Archiv ist im Abschnitt 4.1 in den Schritte 1-3 beschrieben.

Zur Erstellung eines Testprojekts:

1. Eclipse starten und das Menü 'File > New > Project ...' auswählen, um zum Dialog 'New Project' zu gelangen.

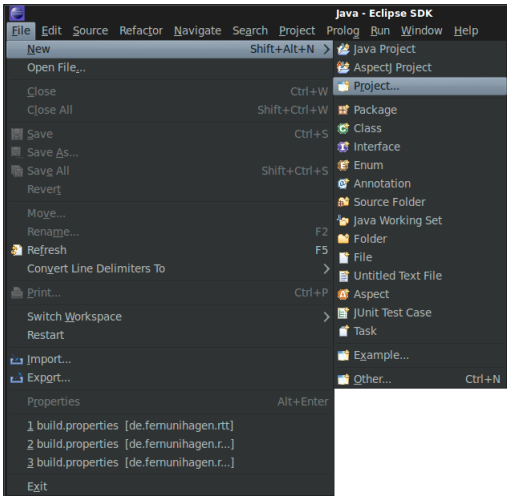

Abbildung 3.3: Erzeugen eines Projektes

2. Den Eintrag 'Plug-in Project' auswählen und mit 'Next' bestätigen.

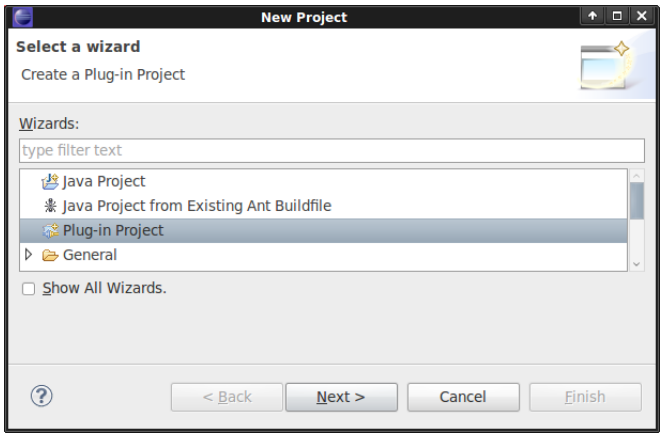

Abbildung 3.4: Erzeugen eines Projektes

3. Im Dialog 'New Plug-in Project' den geeigneten Projektnamen vorgeben und mit 'Next' bestätigen.

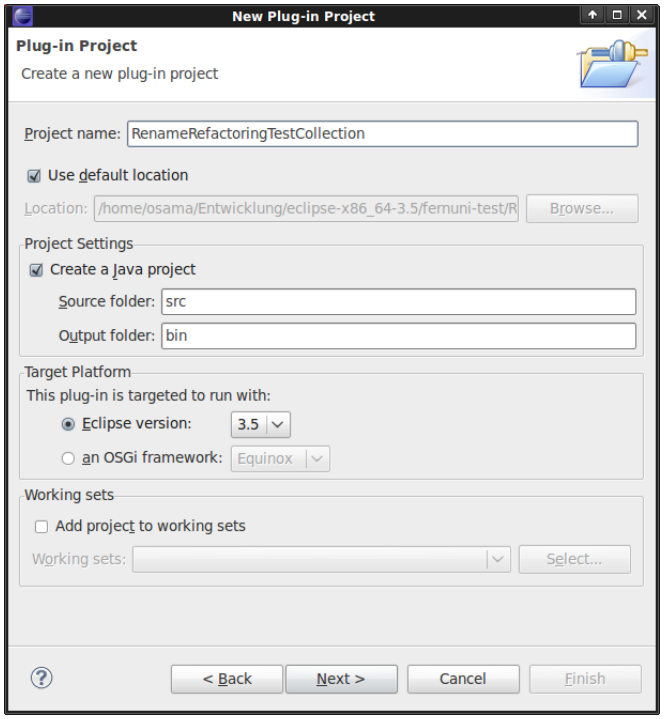

Abbildung 3.5: Erzeugen eines Plug-in Projektes

4. Die Optionen 'Generate an activator ...' und 'This plug-in will make contributions to the UI' können deaktiviert werden. Anschließend das Erzeugen des Projektes mit 'Finish' einleiten.

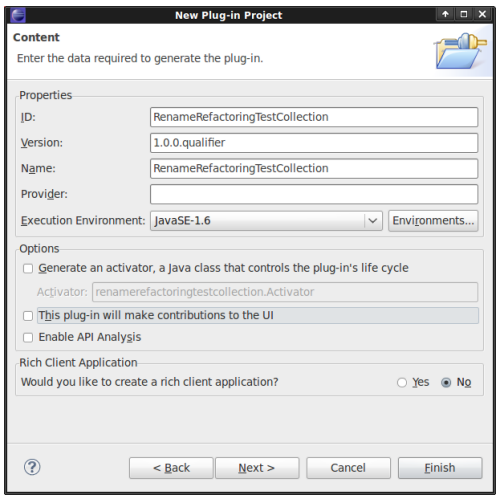

Abbildung 3.6: Inhalt des Plug-in Projektes festlegen

5. Im Manifest Editor das Tab 'Dependencies' aktivieren und über die Aktion 'Add' die Abhängigkeit auf das RTT Core Plug-in (de.fernunihagen.rtt) einfügen.

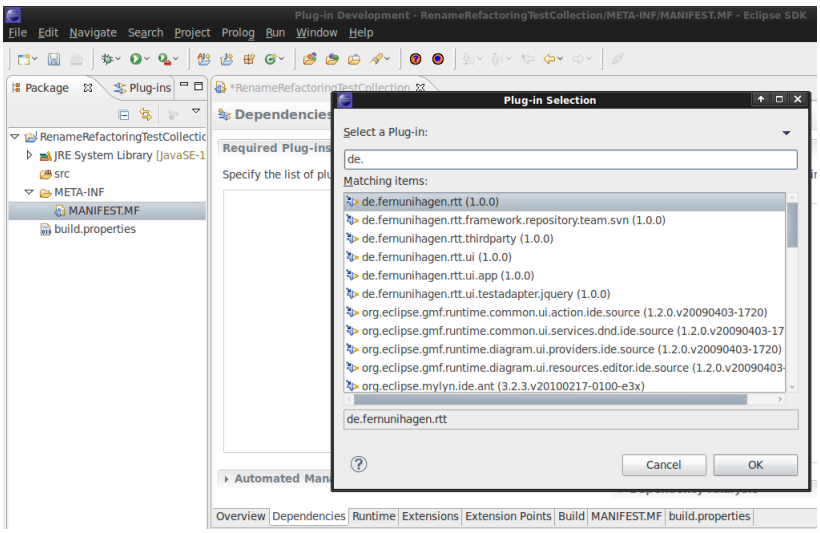

Abbildung 3.7: RTT Core Plug-in Abhängigkeit hinzufügen

6. Über das Menü 'File > New > Class' einen Test Klasse für die zu implementierenden Testfälle erzeugen.

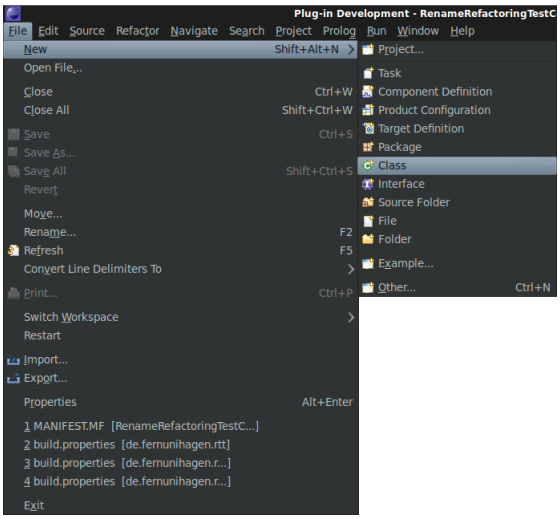

Abbildung 3.8: Test Klasse erzeugen

7. Im Dialog 'New Java Class' einen geeigneten Namen für die Test Klasse vorgeben und mit 'Finish' die Erzeugung einleiten.

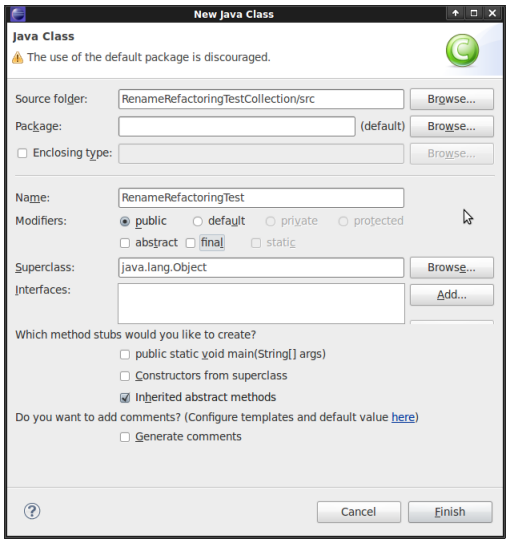

Abbildung 3.9: Informationen der Test Klasse festlegen

8. In den Editor wechseln, um mit der Codierung der Testfälle zu beginnen.

4 Erzeugen einer Konfiguration

4.1 Vorbereitungen

Nach der erfolgreichen Installation des RTT und der Einrichtung eines SVN Repositories, sind vor einem Testlauf noch Mock-Projekte bereitzustellen. Als Mock-Projekt kommt jedes beliebige Eclipse Java-Projekt für den RTT in Frage, zu präferieren sind jedoch Projekte mit einer hohen Testabdeckung, um auch semantische Fehler in einem Refaktorisierungswerkzeug aufdecken zu können. Eclipse Java-Projekte, die bereits in einem SVN Repository verwaltet sind, benötigen keine Vorbereitung um als Mock-Projekte zu fungieren - für diese Projekte können die nachfolgenden Schritte übersprungen werden.

Vorbereiten eines neuen Mock-Projekts

Im nachfolgenden Beispiel dient ein, speziell für das Testprojekt des Rename Refaktoring-werkzeug, vorbereitetes Java Projekt als Testeingabe. Die Schritte 1-3 beschreiben den Vorgang des Imports dieses Java-Projektes aus dem beigefügten Projekt-Archiv. Falls das Projekt bereits im Workspace vorliegt können diese Schritte übersprungen werden.

Zur Vorbereitung eines neuen Mock-Projekts:

1. Eclipse starten und 'File > Import' auswählen.

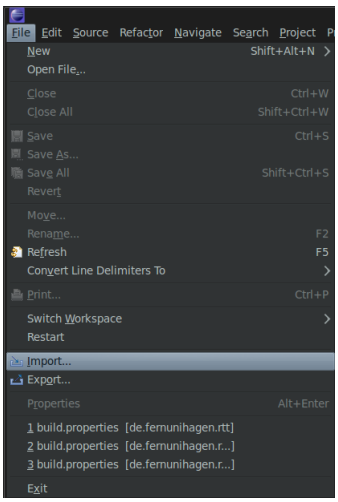

Abbildung 4.1: Import Aktion

2. Die Import Aktion in der Kategorie 'General > Existing Projekt into Workspace' auswählen und mit 'Next' bestätigen.

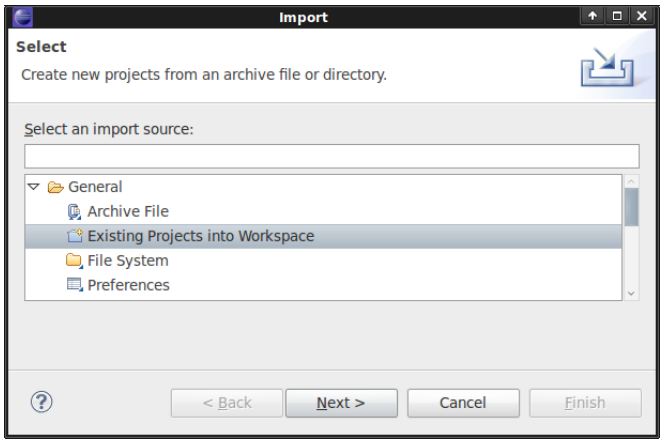

Abbildung 4.2: Import eines bestehenden Projektes

3. Im Feld 'Select archive file' den Pfad zum Eclipse Java-Projekt-Archiv angeben und nach Selektion der zu importierenden Projekte im Archiv, den Import über 'Finish' anstoßen.

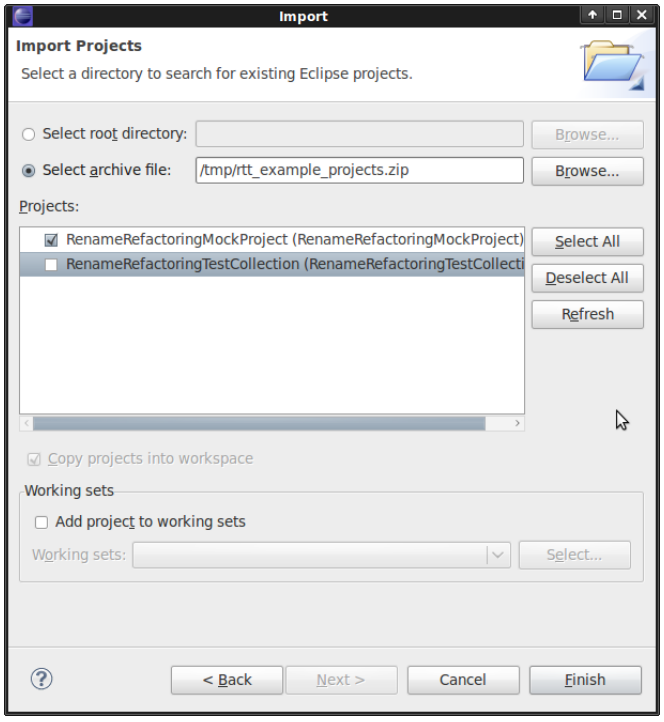

Abbildung 4.3: Import aus einem Projekt-Archiv

4. Das importierte Projekt im Workspace auswählen und über das Kontextmenü 'Team > Share Project ...' die Verknüpfung zu einem Team Repository aufbauen.

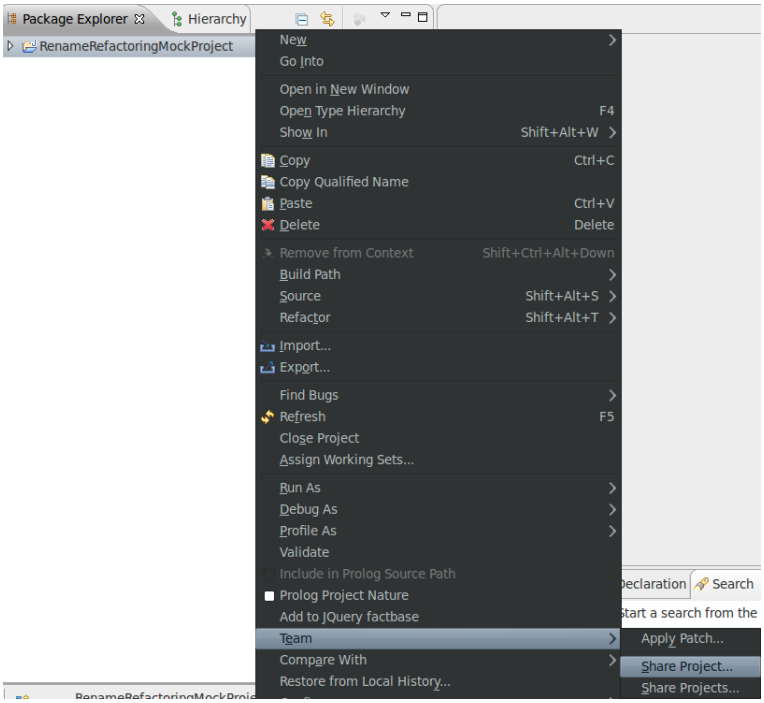

Abbildung 4.4: Projekt mit Repository verknüpfen

5. Im Dialog 'Share Project' den SVN Connector auswählen und die Selektion mit 'Next' bestätigen.

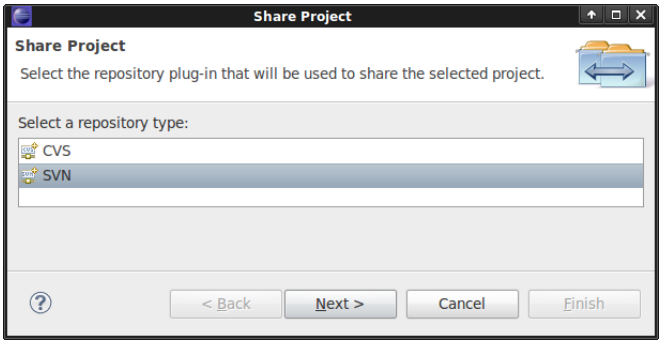

Abbildung 4.5: SVN Connector Auswahl

6. Im Dialog 'Share Project Wizard' die URL des SVN Repository ausweisen und mit 'Next' bestätigen.

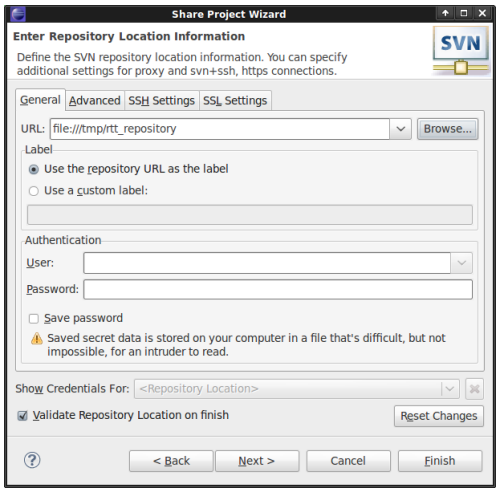

Abbildung 4.6: Repository Informationen festlegen

7. Die Aktion 'Share Project' mit 'Finish' abschließen.

8. Die Übergabe des Projekt Dialog 'Commit' bestätigen.

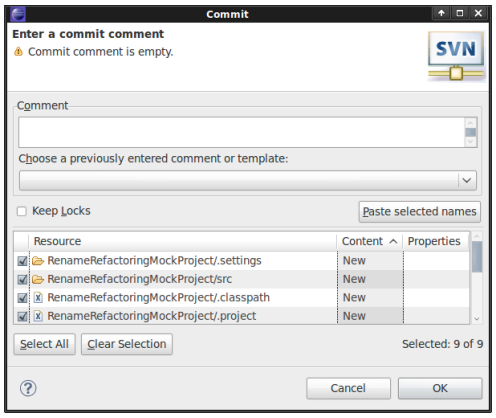

Abbildung 4.7: Initialer Commit

9. Abschließend steht das Projekt zur Selektion innerhalb einer RTT-Testkonfiguration zur Verfügung - womit die Vorbereitungen erfolgreich abgeschlossen sind.

10. Die Aktionen 1-9 zur Vorbereitung weitere Mock-Projekte wiederholen.

4.2 Erzeugen der Launch-Konfiguration

Über die Launch-Konfiguration des RTT, wird die notwendige Testkonfiguration für einen Testlauf erzeugt. Notwendige Bestandteile für eine Testkonfiguration sind:

- Das Testprojekt für das zu testende Refaktoringwerkzeug
- und die vorbereiteten Mock-Projekte.

Zum erzeugen einer neuen Lauch-Konfiguration:

1. Eclipse starten und in das Workspace wechseln, in dem die vorbereiteten Projekte vorliegen.
2. Das Testprojekt selektieren und über das Kontextmenü 'Run as > Run Configurations...' auswählen.

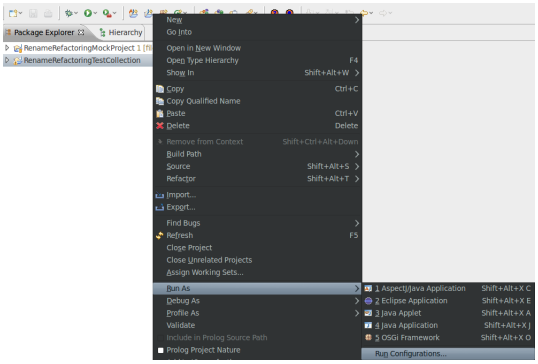

Abbildung 4.8: Launch-Konfiguration öffnen

3. Aus der Liste der verfügbaren Konfigurationstypen 'RTT Java' auswählen und über das Kontextmenü 'RTT Java > New' eine neue Konfiguration erzeugen.

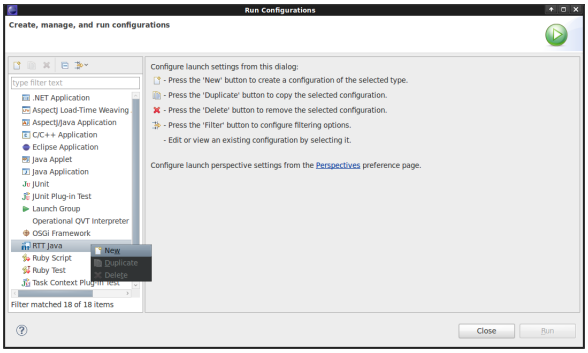

Abbildung 4.9: Testkonfiguration erzeugen

4. Im Tab 'Test' der Konfiguration das Test-Projekt auswählen.

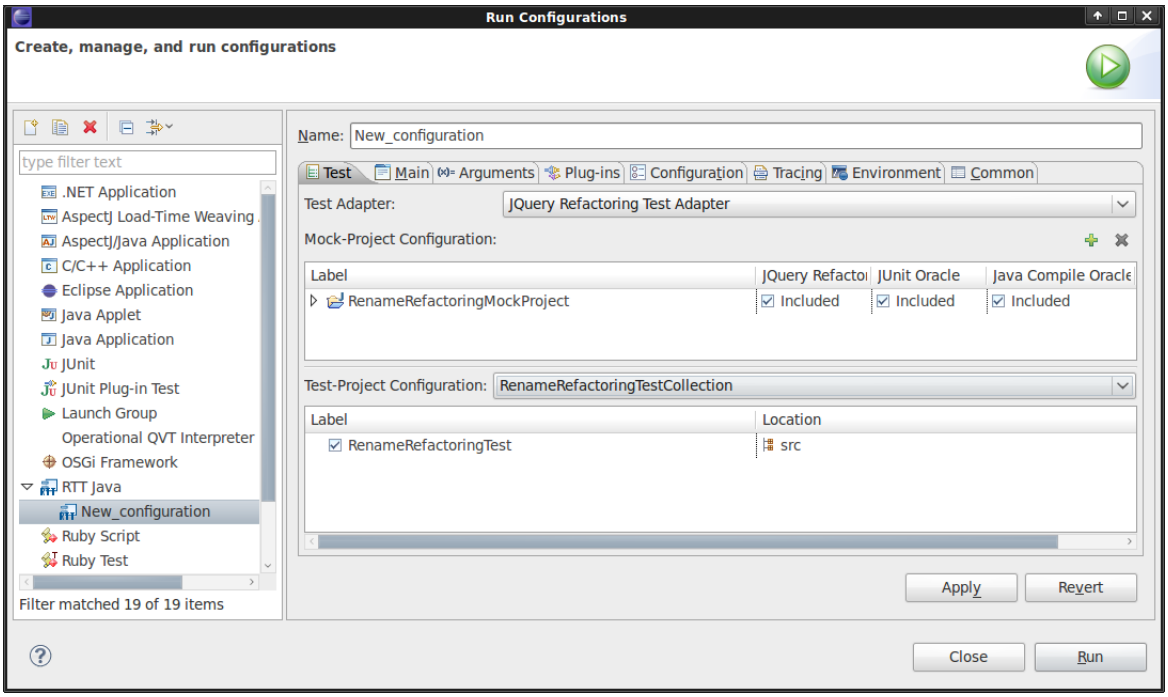

Abbildung 4.10: Auswahl des Testprojekts

5. Im Baum der Test-Projekt Konfiguration erscheinen die verfügbaren Test-Klassen, in denen Testfälle enthalten sind. Durch Selektion bzw. Deselektion von Test-Klassen können enthaltene Testfällen ausgeschlossen werden.

6. Zum Ausschluss von Ressourcen aus dem Mock-Projekt, für den jeweiligen Test Aktor, in der Spalte der Testadapter oder Orakel, entsprechende Ressourcen selektieren. In der Standard Auswahl sind alle Ressourcen enthalten. Zu beachten ist, das im Baum der Mock-Projekt Ressource-Konfiguration jeweils nur die, für den Test Aktor, anwendbare Ressourcen auswählbar sind.

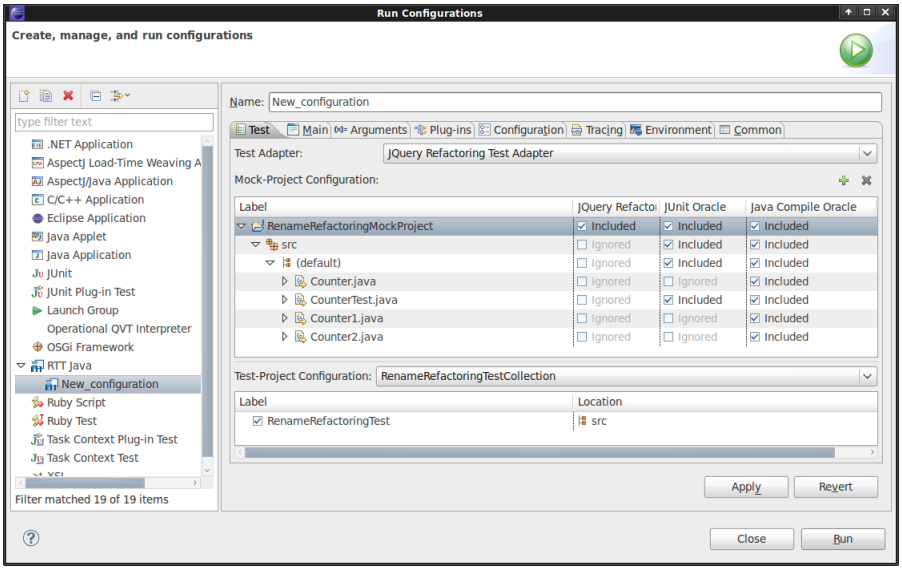

Abbildung 4.11: Konfiguration der Mock-Projekte

7. Über das Kontextmenü im Baum der Mock-Projekt Ressource-Konfiguration, oder der Tool-bar, können Projekte hinzugefügt oder entfernt werden. In der Projekt Auswahl erscheinen im Workspace enthaltene Java-Projekte, die nicht bereits in der Konfiguration enthalten sind und erfolgreich vorbereite wurden. [1]

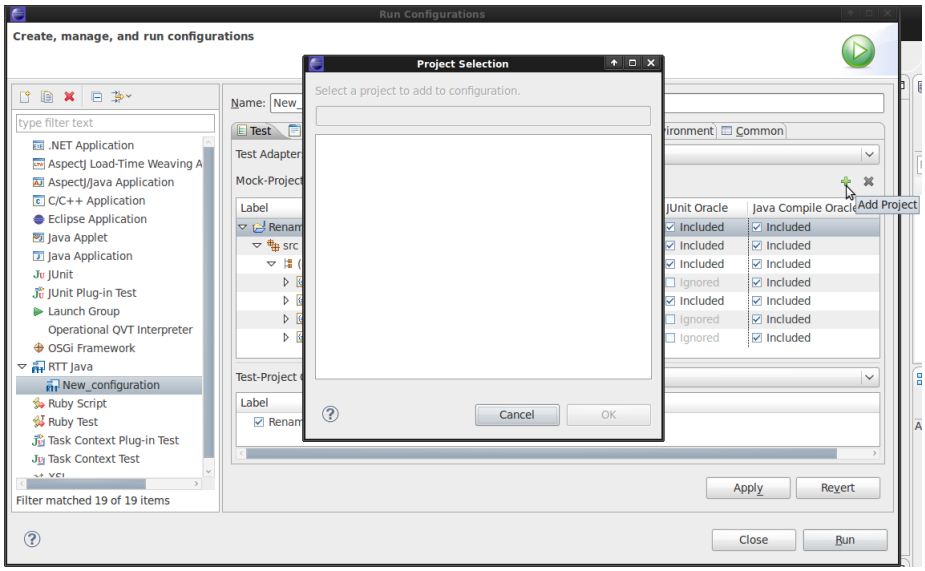

Abbildung 4.12: Hinzufügen von Mock-Projekten

[1] in der aktuellen Version sind Plug-in Projekte ausgeschlossen

8. Die Standard Launch-Konfiguration Einstellungen wie z.B. Main, Arguments oder Common können über die gleichnamigen Tab's verändert werden, so z.B. die Auswahl des Workspace oder der Speicherort der Launch-Konfiguration.

9. Nach erfolgreicher Konfiguration kann über 'Run' ein Testlauf gestartet werden, der in einer neuen Eclipse Instanz ausgeführt wird.

Anmerkung Nach Erzeugung einer Launch-Konfiguration dürfen die Mock-Projekte aus dem Workspace entfernt werden. Vor einer erneuten Veränderung der Launch-Konfiguration, werden diese dann ggf. automatisch abgerufen.

5 Debuggen

Um ein Testfall zu Debuggen, muss eine Launch-Konfiguration im Debug-Mode gestartet werden. Die Schritte zur Erzeugung der Launch-Konfiguration sind im Abschnitt 4.2 beschrieben. Zum starten eines Testfalls im Debug-Mode:

1. Eclipse starten und in das Workspace wechseln, in dem die vorbereiteten Projekte vorliegen.
2. Das Testprojekt selektieren und über das Kontextmenü 'Debug as > Debug Configurations...' auswählen.
3. Die zuvor erzeugte Launch-Konfiguration auswählen.
4. Über 'Debug' den Testlauf im Debug-Mode aktivieren.

Um in das RTT-Framework zu Debuggen müssen die Quell-Projekte des RTT vor dem Launch ins Workspace importiert werden, aus dem der Launch gestartet wird. Dort können dann etwaige Brechpunkte gesetzt werden.

6 Auswertung der Ergebnisse

Wie bereits beschrieben startet ein Testlauf in einer neuen Eclipse Instanz. Während des Laufes sind Benutzerinteraktionen durch das modale Fortschrittsfenster gesperrt. Der Gesamte Testlauf kann jeder Zeit durch den Benutzer abgebrochen werden und erneut gestartet werden. Nach Abschluss des Testlauf stehen die Ergebnisse in der Ansicht des 'RTT Result Navigators' zur Verfügung. In der ersten Zeile der Anzeige wird der Status des Gesamt Testlauf dargestellt, der sich aus den einzelnen Einträgen der Anzeige zusammensetzt. Die einzelnen Einträge eines Testlauf beschreiben:

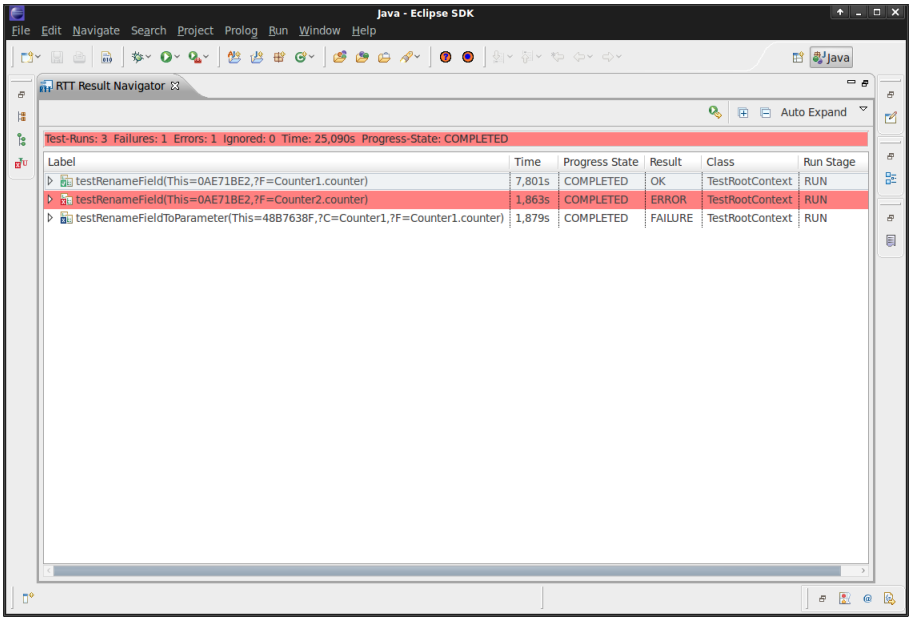

Abbildung 6.1: Testergebnis Ansicht

- Label - den Testfall des Refaktoringwerkzeug ggf. mit seinen Eingaben
- Time - die Durchlaufzeit der Ausführung
- Progress State - den Ausführungsstatus, wie z.B. STOPPED, RUNNING oder COMPLETED
- Result - das Ausführungsergebnis wie z.B. OK, ERROR oder FAILURE. Das Ausführungsergebnis wird am Anfang der Zeile symbolisch repräsentiert.

- Class - das Modelelement im RTT Framework
- Runstage - die Ausführungsebene, wie z.B. Initialisierung oder Testlauf

Des Weiteren stellt die Ansicht zur Analyse, in der Tool-bar, Filter zur Verfügungen, über die die Anzeige eingeschränkt oder erweitert werden kann. Die Anzeige der Testergebnisse

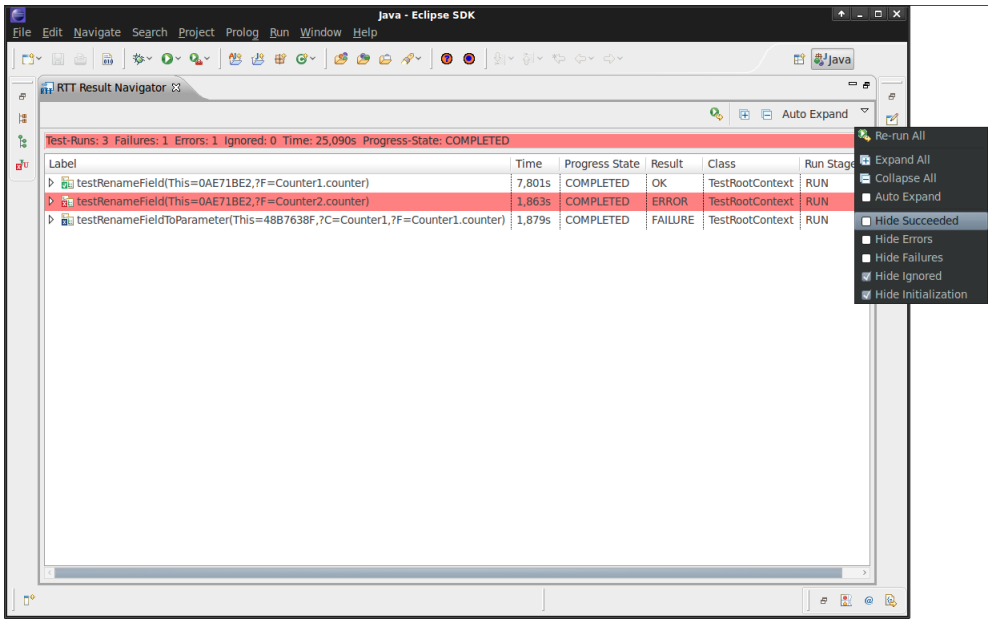

Abbildung 6.2: Filtern der Anzeige

ist strukturiert gewählt, durch Aufklappen eines Testlauf-Eintrags können die Details begutachtet werden, wie z.B. der Status der Orakel-Überprüfungen, die Eingaben und Ausgaben des Testfalls oder die modifizierten Ressourcen. Einträge die Fehler aufweisen werden beim Expandieren automatisch bis zur Fehlerquelle erweitert. Über die Option 'Auto Expand', im Tool-bar Menü, können Einträge grundsätzlich vollständig erweitert werden. Jedes der Elemente in der Anzeige, ist durch ein Modelelement repräsentiert. Einigen Modelelementen sind, über das Kontextmenü, Aktionen zugeordnet:

- TestRootContext oder RefactoringTestElement - erneutes Anwenden des im Testfall erzeugten Refaktoringwerkzeugs, falls dieses ausgeführt wurde. Dazu werden die beteiligten Mock-Projekte in ihren Ursprungszustand versetzt.
- JavaTestElement - Öffnen des Java-Elementes im Editor
- DiffTestElement - Öffnen der Ressource im Editor, die Änderungen werden nur sichtbar, falls das Refaktoring erneut ausgeführt wurde.
- JUnitTestCaseElement - Öffnen des JUnit-Testfalls und Exception Stack Trace in der Console ausgeben

- JavaCompileProblem - Öffnen der Fehler-Quelle im Editor
- ExceptionTestElement - Exception Stack Trace in der Console ausgeben

Literaturverzeichnis

[1] eclipse.org, http://download.eclipse.org/releases/galileo. *Eclipse IDE for Java Developers, Release Galileo.*

[2] Osama El Hosami. Implementierung eines Eclipse-Plugins zum automatisierten Testen von Refaktorisierungswerkzeugen. Master's thesis, Fernuni-Hagen, 2010.

[3] Sergei Ikkert. Untersuchung der Eclipse-JDT-Refaktorisierungen mit Hilfe des Refactoring Tool Testers. Master's thesis, Fernuni-Hagen, 2010.

[4] Polarion, http://www.polarion.com/products/svn/subversive/download.php. *Subversive, Subversion Team Provider for Eclipse.*

[5] Tigris.org, http://tortoisesvn.net/. *TortoiseSVN is an easy-to-use SCM / source control software for Microsoft Windows.*

[6] ubuntu.com, https://help.ubuntu.com/community/Subversion. *Ubuntu Subversion Installationhinweise.*

[7] University of British Columbia, http://jquery.cs.ubc.ca/documentation/installation.html. *JQuery, a query-based code browser Installationshinweise.*