RÖPERWEISE SYSTEMS

Projektdokumentation

Planung, Implementierung und Ausführung von Softwaretests für das Projekt CvsScanner unter Einsatz von Eclipse mit Hilfe der „Test & Performance Tools Platform“

Dokumentation der betrieblichen Projektarbeit im Rahmen der IHK-Abschlussprüfung zum Fachinformatiker FR Anwendungsentwicklung

---

Name:	Sebastian Just
Geburtsdatum / -ort:	01. Mai 1986 / Krefeld
Wohnort:	Bröhnstr. 26 30952 Ronnenberg
Ausbildungsberuf:	Fachinformatiker FR Anwendungsentwicklung
Ausbildungsbetrieb:	RÖPERWEISE Systems GmbH Robert-Bosch-Straße 12 30989 Gehrden

Inhaltsverzeichnis

1 Allgemeines

1.1 Vorwort

Um die Aufgabenstellung für den Leser zu verdeutlichen, werden die Grundelemente des Projektes am Anfang der Dokumentation beschreiben. Die Fachwörter, welche im Text Verwendung finden sind im Glossar nachzuschlagen. Abkürzungen werden direkt per Fußnote erklärt. Eine längere Erklärung der Abkürzungen befindet sich ebenfalls im Gloassar.

1.2 RöperWeise Systems GmbH

Die RöperWeise Systems GmbH wurde 1996, damals noch unter dem alten Namen Softwork-EDV GmbH, gegründet und ist in der Informationstechnologie-Branche tätig.

Die Tätigkeitsfelder der in Gehrden ansässigen RöperWeise Systems GmbH sind:

- Entwicklung und Anpassung von Individualsoftware
- Projektunterstützung
- Systembetreuung

1.3 CvsScanner

Das diesem IHK-Projekt zugrunde liegende Projekt CvsScanner ist ein Kundenauftrag. Der Kunde wünschte sich eine Software, welche die Produktivität der IT-Abteilung des Unternehmens messen soll. Als Grundlage dieser Messung wurden LOCs[1] gewählt. Die LOCs sollen mittels des Versionsverwaltungsprogrammes CVS für einzelne Projekte und über verschiedene zeitliche Abschnitte (Die kleine Auflösung sollte 1 Tag betragen) ermittelt werden können. Die Zuordnung, für welches Projekt die LOCs pro Datei gewertet werden soll wird über den Kommentar beim Einchecken[2] einer Datei ins CVS-Repository ermittelt. Als einzige Einschränkung brachte der Kunde ein, dass geänderte Leerzeilen (hinzufügen, löschen) sich nicht auf die LOCs auswirken sollten.

[1] LOC steht für „Line of Code“ und gibt die Anzahl an Zeilen in einer (Text-)Datei wieder

[2] Einchecken bedeutet, dass eine neue Version einer Datei in das CVS-Repository übertragen wird

2 Ausgangssituation

Bei diesem IHK-Projekt handelt es sich um ein Teilprojekt eines Kundenauftrages. Da sich sowohl zeitlicher Rahmen wie auch Komplexität und verwendete Technologien mit einem frei gewählten IHK-Projekt überschnitten hatten, lag es nahe, ein Teilprojekt im Rahmen des IHK-Projektes zu bearbeiten. Da der gesamte Kundenauftrag jedoch den gegebenen Bearbeitungszeitraum von 70 Stunden überstiegen hätte, wurde im Vorfeld entschieden, lediglich die umfassenden Softwaretests in diesem IHK-Projekt zu bearbeiten.

Dieses Teilprojekt behandelt ausschließlich umfassende Softwaretests, welche die Entwicklungstests ergänzen. Dies ist nötig, da es sehr unwahrscheinlich ist, dass Entwickler beim Testen ihres eigenen Codes alle Fehler und Unzulänglichkeiten erkennen.

2.1 Ist-Analyse

Das Testen von Software beeinflusst in erheblichem Maße die Qualität der entwickelten Software im Betrieb.

Deshalb muss das Projekt CvsScanner neben der Implementierung und den, von den Entwicklern durchgeführten, Kurztests intensiven Abschlusstests, wie Leistungs- und Funktionstests, vor der Auslieferung unterzogen werden.

Tests sind jedoch sowohl in der Implementierung als auch in der Durchführung sehr aufwändig und daher stellt sich auch die Frage nach einer geeigneten Testentwicklungsumgebung. Durch die Einführung der IDE[3] Eclipse im Unternehmen als Standardtool für die Java- Entwicklung sollten die Tests ebenfalls möglichst mit Eclipse implementiert und durchgeführt werden.

Da TPTP[4] ebenfalls vor kurzer Zeit zum Firmenstandard erhoben wurde lag es nahe dieses auch für dieses Projekt zu verwenden.

2.2 Soll-Konzept

Nach Abschluss dieses Teilprojektes soll das Projekt CvsScanner ein umfassend getestetes Produkt sein, welches sich korrekt und fehlerfrei verhält.

---

[3] Eine IDE vereint die wichtigsten Funktionalitäten zum Erstellen eines Programms unter einen einheitlichen Oberfläche

[4] TPTP ist eine Erweiterung für Eclipse um Softwaretests zu Erstellen und Durchzuführen

Dabei sollen alle Funktions- und Leistungsanforderungen mit den Anforderungen aus dem Pflichtenheft erneut überprüft und bestätigt werden. Wichtig dabei ist, dass es sich möglichst um automatisierte Tests handelt, die auch später, z.B. bei der Erweiterung von CvsScanner, ausgeführt können. Es soll sich also nicht um „per Hand“ ausgeführte Tests handeln sondern die Tests selber sollen auch programmiert sein um sie jeder Zeit wieder ausführen zu können. So kann, z.B. bei einer Erweiterung von CvsScanner, überprüft werden, ob durch die Erweiterung die bestehenden Methoden von CvsScanner nicht mehr so funktionieren wie vorher.

2.3 Projektumfeld

Die Projektarbeit wird in den Räumlichkeiten der RöperWeise Systems GmbH in Gehrden durchgeführt. Der direkte Auftraggeber ist der Projektleiter des Projektes CvsScanner. Projektbetreuer während der Projektarbeit sind Herr Stefan Röper und Herr Jens Weise.

2.3.1 Hardware

Für die Durchführung des Projektes werden verschiedene Hardwarekomponenten benötigt:

1. Ein Entwicklungsrechner
2. Ein Testserver
3. Ein Datenbankserver (mySQL[5])

2.3.2 Software

Das Projekt wird mit der folgenden Software durchgeführt. Dabei handelt es sich um den Firmenstandard bzw. um Vorgaben vom Kunden die nicht weiter zur Diskussion standen:

1. IDE Eclpise/Java 1.5
2. TPTP-Erweiterung für Eclipse
3. Remote-Access-Controller (RAC[6])
4. JDBC[7]-Treiber für mySQL

[5] mySQL ist eine weit verbreitete SQL-Datenbank

[6] RAC ermöglicht das Ausführen und Protokollieren von Java-Programmen auf anderen PCs

[7] JDBC ist eine Abstraktionsschicht für Java-Programme um auf unterschiedliche Datenbanken zugreifen zu können

3 Kosten-/Nutzenanalyse

3.1 Kosten

Die Kosten dieses Teilprojektes dürften keinesfalls nur für dieses Teilprojekt angerechnet werden. Da das Gesamtprojekt CvsScanner beim Kunden sehr lange im Einsatz sein wird ist ein fehlerfreier Betrieb wichtig. Je später Fehler im Programm gefunden werden, desto teurer ist ihre Behebung. Deshalb sind auch zukünftige Kosten zu beachten, welche vermieden werden, indem das Gesamtprojekt durch dieses Teilprojekt intensiv, und somit auf den ersten Blick kostenintensiv, getestet wird.

Die Zeit für dieses Teilprojekt beläuft sich auf 70 Stunden. Bei einem realistischen (internen) Stundensatz von 70 € (incl. Kosten für die genutzten Geräte und aufkommende Verwaltungskosten o.Ä.).

Als Software kommt bei diesem IHK-Projekt nur freie Software zum Einsatz deren Kosten bei 0,00€ liegen, da die Beschaffungskosten von wenigen Cent für die Internetübertragung kaum berechnet werden können.

Art	Anzahl	Einzelkosten	Gesamtkosten
Lohn	70 h	70€/h	4.900€
Geräte	0	0	0€
Verwaltung etc.	0	0	0€
Gesamt			***4.900€***

3.2 Nutzen

Da durch das Projekt keine Fehler gefunden wurden, kann davon ausgegangen werden, dass es auch beim Einsatz des Projektes beim Kunden zu keinen späteren Korrekturen kommen muss, welche besonders teuer sind:

Zu den Kosten für den Entwickler, der den Fehler aufnehmen, reproduzieren, dokumentieren und beheben muss, kommen die Kosten für ein erneutes Einspielen beim Kunden zzgl. Kosten, die durch eine Korrektur des Datenbestandes entstehen. Dies kann z.B. nötig sein, wenn ein Fehler im CvsScanner falsche LOC-Werte ermittelt und in die Datenbank schreibt.

Die Kosten hierfür sind schwer zu beziffern, ein doppelter Stundenlohn erscheint aber als realistisch. Als Ansatz für die Berechnung wird ein durchschnittlicher Stundenaufwand von 16 Stunden pro Fehler in CvsScanner angenommen. Dabei ist zu beachten, dass in diesen 16 Stunden auch Abstimmungsgespräche mit dem Kunden enthalten sind ebenso wie Anfahrt zum Kunden und Dokumentation. Es handelt sich hierbei nicht um die reine Arbeit am Fehler selber.

Art	Anzahl	Einzelkosten	Gesamtkosten
Lohn	16 h	140€/h	2.240€
Gesamt			***2.240€***

So liegen die Kosten pro Bug bei mindestens 2.240€

3.3 Wirtschaftlichkeit

So sind zwei potentielle Bugs schon ausreichend um die Kosten für das intensive Testen aufzuwiegen.

Hinzu kommt, dass durch ein fehlerfreies Produkt auch das Image der Firma selber keinen Schaden nimmt. Ein Projekt, welches für einen Kunden entwickelt wurde und fehlerhaft ist fügt der entwickelnden Firma einen schwer zu beziffernden Schaden zu, welcher ebenso schwer zu korrigieren ist.

Auch bieten diese automatisierten Tests den Vorteil, dass Erweiterungsarbeiten an CvsScanner wesentlich zügiger Umgesetzt werden können, da die alten Methoden ohne zusätzliche Kosten erneut getestet werden können und nur für die neuen Methoden neue Testfälle nötig sind.

Insgesamt kann von einer positiven Wirtschaftlichkeit gesprochen werden.

4 Projektdurchführung / Realisierung

4.1 Planungsphase

Insgesamt standen für die Projektarbeit siebzig (70) Stunden zur Verfügung. Es wird von einer täglichen Projektarbeitszeit von acht (8) Stunden ausgegangen.

Zu Beginn des Projektes wurde in der Planungsphase eine Ist-Analyse durchgeführt. Auf Basis derer, und in Absprache mit dem Projektleiter des Projektes CvsScanner, wurden die Projektziele entwickelt. Als nächstes stand die Überprüfung der Durchführbarkeit an, indem die geplanten und vorhandenen Ressourcen geprüft wurden.

Aus allen diesen Informationen wurde das Sollkonzept entwickelt und das Projekt durchgeführt.

4.2 Realisierungsphasen

4.2.1 Erstellen der Testfälle

Das Gesamtprojekt soll möglichst umfassend getestet werden, um dem Kunden eine hochwertige und stabile Software zu liefern, die seinen Wünschen und Vorstellungen entspricht und in allen Situationen

die erwarteten Ergebnisse liefert. Dazu sind verschiedene Softwaretest nötig, welche hier beschrieben sind und durchgeführt werden:

4.2.1.1 Codeanalyse

Die Codeanalyse ist ein statischer Test und analysiert den kompletten Quelltext. Dabei wird darauf, geachtet, dass verschiedene stilistische Konventionen im Quelltext eingehalten sind.

Als Basis für die stilistischen Konventionen dient das Regelwerk der „Java Best Practices“.

4.2.1.2 Methodenabdeckung

Dieser Test ist, ebenso wie der Test des Laufzeitverhaltens, ein dynamischer Whitebox-Test.

Der Test der Methodenabdeckung, des Laufzeitverhaltens und des Arbeitsspeicherverbrauches werden in einem einzigen Testdurchlauf durchgeführt.

Als Basis für diesen Test dient das firmeninterne CVS-Repository.

4.2.1.3 Laufzeitverhalten

Der Test, welcher das Laufzeitverhalten der einzelnen Methoden protokolliert, ist ein dynamischer Whitebox-Test.

Als Basis für den Test dient das firmeninterne CVS-Repository.

4.2.1.4 Arbeitsspeicherverbrauch

Dieser Test ist ein dynamischer Test. Es wird dabei überprüft, wie viele Instanzen von jeder Klasse gebildet werden und wie viel Arbeitsspeicher dadurch verbraucht wird.

Als Basis für den Test dient das firmeninterne CVS-Repository.

4.2.1.5 LineMarker

Dieser Test ist ein dynamischer Blackbox-Test. Er prüft, ob sich die sog. LineMarker richtig verhalten.

Ein LineMarker-Objekt ist ein Objekt innerhalb des Projektes CvsScanner, welche den Zustand einzelner Zeilen einer zu analysierenden Textdatei repräsentiert.

Dabei wird nur beachtet, ob es sich um eine leere oder eine nicht-leere (d.h. gefüllte) Zeile handelt. Diese Art der Abstraktion ist notwendig, um später im Algorithmus den Wunsch des Kunden berücksichtigen zu können, dass Leerzeichen bei der LOC-Berechnung nicht berücksichtigt werden sollen.

Als Basis für den Test dienen Testdateien bzw. bereits teilweise abstrahierte Testdateien in einem passenden Objekt innerhalb des Programmes. Diese Form wurde gewählt, damit nur die LineMarker getestet

werden und alle Methoden, welche für das direkte Einlesen der Daten von der Festplatte in den Arbeitsspeicher zuständig sind, umgangen werden.

4.2.1.6 RegionChanger

Dieser Test ist ein dynamischer Blackbox-Test. Er prüft, ob sich die sog. RegionChanger richtig verhalten.

Ein RegionChanger-Objekt ist ein Objekt innerhalb des Projektes CvsScanner, welches die Anzahl an Änderungen einer Datei zu einer anderen Version derselben Datei enthält. Dies ist das wichtigste Objekt im Projekt CvsScanner, da auf Basis dieser Daten die LOCs aufaddiert werden.

Als Basis für den Test dienen Testdateien bzw. bereits teilweise abstrahierte Testdateien in einem passenden Objekt um die Methoden, welche für den Festplattenzugriff zuständig sind, zu umgehen.

4.2.1.7 Filenamefilter

Die LOCs können nur von Textdateien richtig berechnet werden. Daher ist es unabdingbar, dass - bevor die eigentliche LOC-Berechnung beginnt - der Dateityp bestimmt wird und ggf. die Analyse abgebrochen wird, wenn es sich um eine Binärdatei handelt. Dies ist ein dynamischer Blackbox-Test.

Als Basis dient das firmeneigene CVS-Repository.

4.2.2 Implementierung der Testfälle

Durch die Verwendung von Eclipse samt TPTP ist die Implementierung nach dem Einarbeiten in TPTP relativ problemlos. Da TPTP in einigen Bereichen auf JUnit aufbaut und dieses teilweise erweitert ergibt sich eine relativ steile Lernkurve.

So konnten die dynamischen Tests sehr zügig implementiert werden, zumal bis auf den eigentliche Testcode praktisch alles mit wenig Einstellarbeit automatisch erzeugt wird.

4.2.3 Durchführung der Tests

Nach einigen wenigen Testläufen auf dem Entwicklungs-PC selber wird schließlich das Testumfeld genutzt, welches dem Umfeld beim Kunden relativ nahe kommt.

Es wird ein Testserver (Linux) mit dem Remote-Access-Controller ausgestattet und CvsScanner auf ihm installiert.

Ein zweiter Server stellt den Datenbankserver dar. Dazu wird ein bereits existierender Datenbankserver nur um eine Datenbank erweitert und die entsprechenden Parameter in CvsScanner eingestellt. Der verwendete Datenbankserver entspricht nicht dem, der bei dem Kunden zukünftig zum Einsatz kommen wird. Dies ist aber unkritisch, da der Datenbankserver nur über eine Abstraktionsschicht (JDBC) angesprochen wird und so prinzipiell mit jedem Datenbankserver kommunizieren kann.

Über TPTP und den Remote-Access-Controller werden dann alle Tests auf dem Testserver ausgeführt, wobei das Protokollieren und Aufzeichnen der anfallenden Test-Daten auf dem Entwicklungs-PC ausgeführt wird.

4.2.4 Ergebnisse der Tests

4.2.4.1 Codeanalyse

Das Ergebnis des Tests ist positiv. Es sind keine weiteren Änderungen am Quelltext notwendig. Zwar gab es einige Anmerkungen zu dem Quelltext von CvsScanner, diese sind jedoch sehr speziell und die Richtigkeit dieser Verbesserungsvorschläge wird bis heute kontrovers diskutiert.[8]

4.2.4.2 Methodenabdeckung

Das Ergebnis des Tests ist positiv. Es sind keine weiteren Änderungen am Quelltext notwendig. Es gab keine „vergessenen" Methoden im Projekt CvsScanner und keine Methode stach durch eine besonders häufige Verwendung hervor (Dies kann ein Hinweis auf das Vermischen von objektorientierter und prozeduraler Programmierung sein).[9]

4.2.4.3 Laufzeitverhalten

Das Ergebnis des Tests ist positiv. Es sind keine weiteren Änderungen am Quelltext notwendig. Insgesamt ist auch der Projektleiter des Projektes CvsScanner von der schnellen Arbeitsweise des Programmes CvsScanner sehr zufrieden.

Ungefähr 5 Minuten für das Verarbeiten von Rund 5.000 Dateien die zu mehr als 17.000 Einträge in der Datenbank führen sind ein gutes Ergebnis, wobei von einem relativ linearen Wachstum der Laufzeit ausgegangen werden kann, da der begrenzende Faktor lediglich die Festplattengeschwindigkeit ist.[10]

4.2.4.4 Arbeitsspeicherverbrauch

Das Ergebnis des Tests ist positiv. Es sind keine weiteren Änderungen am Quelltext notwendig. Die Anzahl der einzelnen Instanzen decken sich mit den Überlegungen, die auf Grundlage der Größe des analysierten CVS-Repositories angestellt wurden.

[8] Abb. 01 (Codeanalyse)

[9] Abb. 02 (Methodenabdeckung)

[10] Abb. 02 (Methodenabdeckung)

So sind z.B. etwas mehr LineMarker- als ChangeRegion-Instanzen erstellt worden. Dies ist in sofern sinnvoll, als dass es mindestens so viele LineMarker wie ChangeRegions geben muss. Gibt es mehr, so bedeutet dies, dass es sich überschneidende Bereiche gibt. Gibt es weniger LineMarker als ChangeRegions, so existiert ein Fehler im Programm.[11]

4.2.4.5 LineMarker

Das Ergebnis des Tests ist positiv. Es sind keine weiteren Änderungen am Quelltext notwendig. Die Ergebnisse der einzelnen LineMarker decken sich mit den, in Absprache mit dem Projektleiter, von Hand errechneten Werten.[12]

4.2.4.6 RegionChanger

Das Ergebnis des Tests ist positiv. Es sind keine weiteren Änderungen am Quelltext notwendig. Die Ergebnisse der RegionChanger decken sich mit den vorher per Hand errechneten Werten.[13]

4.2.4.7 Filenamefilter

Das Ergebnis des Tests ist positiv. Es sind keine weiteren Änderungen am Quelltext notwendig. Der Filter hat nur die Dateiendungen akzeptiert, welche im Produktivumfeld auftreten und wovon ausgegangen werden kann, dass es sich hierbei um Testdateien handelt, die zur LOC-Berechnung herangezogen werden können. Dabei ist zu beachten, dass es sich um einen Whitelist-Filter handelt. Es können also durchaus Dateien an den CvsScanner übermittelt werden, die zwar reinen Text enthalten aber vom Filter abgewiesen werden, da dem Filter die Endung unbekannt ist.[14]

4.2.5 Weitere Testfälle

Prinzipiell lassen sich natürlich noch beliebig viele weitere Testfälle ausdenken, welche das Programm noch intensiver testen. Hauptsächlich aus Zeitgründen wurde davon jedoch von vorne herein Abstand genommen.

So wäre z.B. das Testen der I/O-Schicht des Programms denkbar jedoch kaum sinnvoll:

[11] Abb. 04 (Arbeitsspeicherverbrauch)

[12] Abb. 05 (LineMarker, RegionChanger und Filenamefilter)

[13] Abb. 05 (LineMarker, RegionChanger und Filenamefilter)

[14] Abb. 05 (LineMarker, RegionChanger und Filenamefilter)

Das Einlesen der Dateien in den Arbeitsspeicher basiert auf (bereits vom Hersteller SUN) getesteten Methoden, welche Java selbst mitbringt. Ein Fehler der in diesen (auch von anderen häufig benutzen) Java-Routinen stecken sollte wäre mit Sicherheit schon lange gefunden worden.

5 Projektbewertung

Der Projektverlauf entsprach im Großen und Ganzen der Planung. Es entstanden zwar kleinere Abweichungen, welche sich doch untereinander ausgeglichen haben und so den gesamten Zeitplan nicht beeinflussten:

Arbeit	Zeit (geplant)	Zeit (ausgeführt)	Abweichung
Einarbeiten in CvsScanner und in TPTP	5h	7h	+2h
Design der Testfälle	22h	22h	±0h
Implementierung	16h	14h	-2h
Durchführung der Tests	9h	8h	-1h
Dokumentation	18h	19h	+1h
Summe	***70 h***	***70h***	***±0h***

5.1 Soll-Ist-Vergleich

Der Soll-Zustand wurde wie geplant erreicht. Es wurden zwar keine weiteren Fehler in dem Programm gefunden, was jedoch nicht bedeutet, dass das Testen unnötig war. Nur so ließ sich bestätigen, dass die Software einwandfrei funktioniert.

Auch die Betrachtung der Wirtschaftlichkeit in Kapitel 5 zeigt, dass das Projekt aus wirtschaftlichen Gesichtspunkten positiv zu bewerten ist.

5.2 Ausblick

Das Ziel des Projektes wurde erreicht und der Kunde erhält ein qualitativ hochwertiges Produkt, welches ihn bei seinen täglichen Prozessen unterstützt.

Ebenso sind die ausgeführten Tests jederzeit auch von anderen Mitarbeitern ausführbar und die Funktionalität von CvsScanner kann jederzeit erneut überprüft werden.

Glossar

Wort	Erklärung
Blackbox-Test	Bei einem Blackbox-Test hat der Ersteller des Tests kein Wissen über den inneren Aufbau des zu testenden Objektes. Er besitzt nur Dokumentation darüber, wie ein Objekt auf bestimmte Anfragen reagieren soll.
CVS	CSV steht für „*Concurrent Version System*“. Es handelt sich hierbei um ein Softwaresystem, mit Hilfe dessen sich alle Entwicklungsstände einer Datei speichern und jederzeit unabhängig von danach getätigten Änderungen wiederherstellen lassen. Die Nutzung von CVS ist besonders im Rahmen von Softwareentwicklung sehr verbreitet.
CVS-Repository	Ein CVS-Repository bildeten den Datenbestand aller ein einem CVS vorhandenen Dateien und Verzeichnisse.
IDE	IDE steht für „*Integrated Development Environment*“. Dabei handelt es sich um ein Programm, welches möglichst viele Einzelkomponenten in einer Oberfläche integriert, die für die Softwareentwicklung von Bedeutung sind. Als Hauptbestandteile sind ein Editor und ein Compiler (Übersetzt eine Programmiersprachen in eine für den Computer verständliche Sprache) zu nennen.
Java	Java ist ein objektorientierte Programmiersprache der Firma SUN Microsystems
Java Best Practices	„Java Best Practices“ ist eine Zusammenstellung von Stilkonventionen für Programmierern. Diese Stilkonventionen beschreiben z.B. wie Programmcode kommentiert und eingerückt werden soll.
JDBC	JDBC steht für „*Java Database Connectivity*“. Dabei handelt es sich um eine Abstraktionsschicht um von Java-Applikationen aus auf unterschiedliche Datenbank zugreifen zu können ohne die Applikation ändern zu müssen.

JUnit	Bei JUnit handelt es sich um ein bekanntes und weit verbreitetes Framework um in Java geschriebene Methoden zu testen.
LOC, **Lines of Code**	LOC, auch LoC oder „*Lines of Code*“ geschrieben ist eine Maßeinheit in der Programmierung. Unter LOC wird die Anzahl von Codezeilen verstanden, die sich beispielweise innerhalb einer Datei befinden. Mit Hilfe von LOC- Zahlen können also bedingt die Komplexität und der Umfang einer Softwareentwicklung definiert werden.
mySQL	mySQL ist ein relationales Datenbank-Management-System welches vor allem in der OpenSource- und Web-Gemeine eine große Verbreitung hat.
RAC,	RAC (*Remote-Access-Controll*) ist ein Programm, welches andere (Java-)Programme startet und überwacht. Die so gewonnen Informationen können über ein Netzwerk an einen beliebigen anderen PC gesendet und dort aufgezeichnet und ausgewertet werden.
TPTP	TPTP, auch „*Test and Performance Tools Platform*“ (ehemals Hyades) oder scherzhaft „*Thousands of Problems for Theorem Provers*“ ist eine Erweiterung (Plugin) für Eclipse, um direkt in Eclipse Software zu testen und während der Ausführung zu überwachen. So können Fehler entdeckt und identifiziert werden.
Whitebox-Test	Bei einem Whitebox-Test ist der Ersteller des Tests sich darüber im Klaren, wie das zu testende Objekt im inneren Aufgebaut ist. So kann der Tester z.B. alle Möglichkeiten des Objektes ausschöpfen, übersieht aber ggf. auch Möglichkeiten, da er zu sehr mit den Details vertraut ist.

Abbildungen

Die hier dargestellten Abbildungen zeigen jeweils das Ergebnis der ausgeführten Testläufe in der TPTP-typischen Ansicht innerhalb von Eclipse.

Abb. 01 (Codeanalyse)

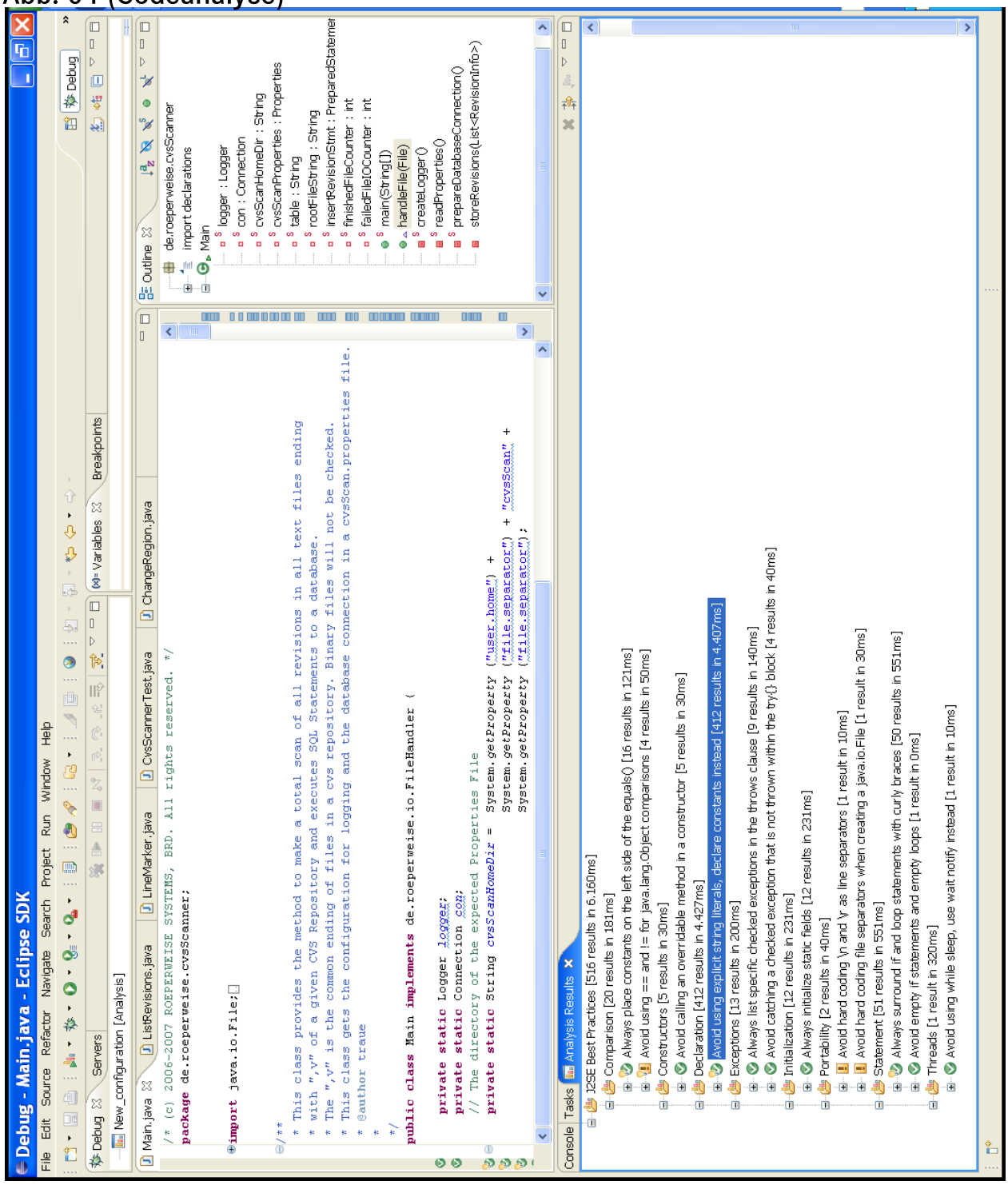


Abb. 02 (Methodenabdeckung)

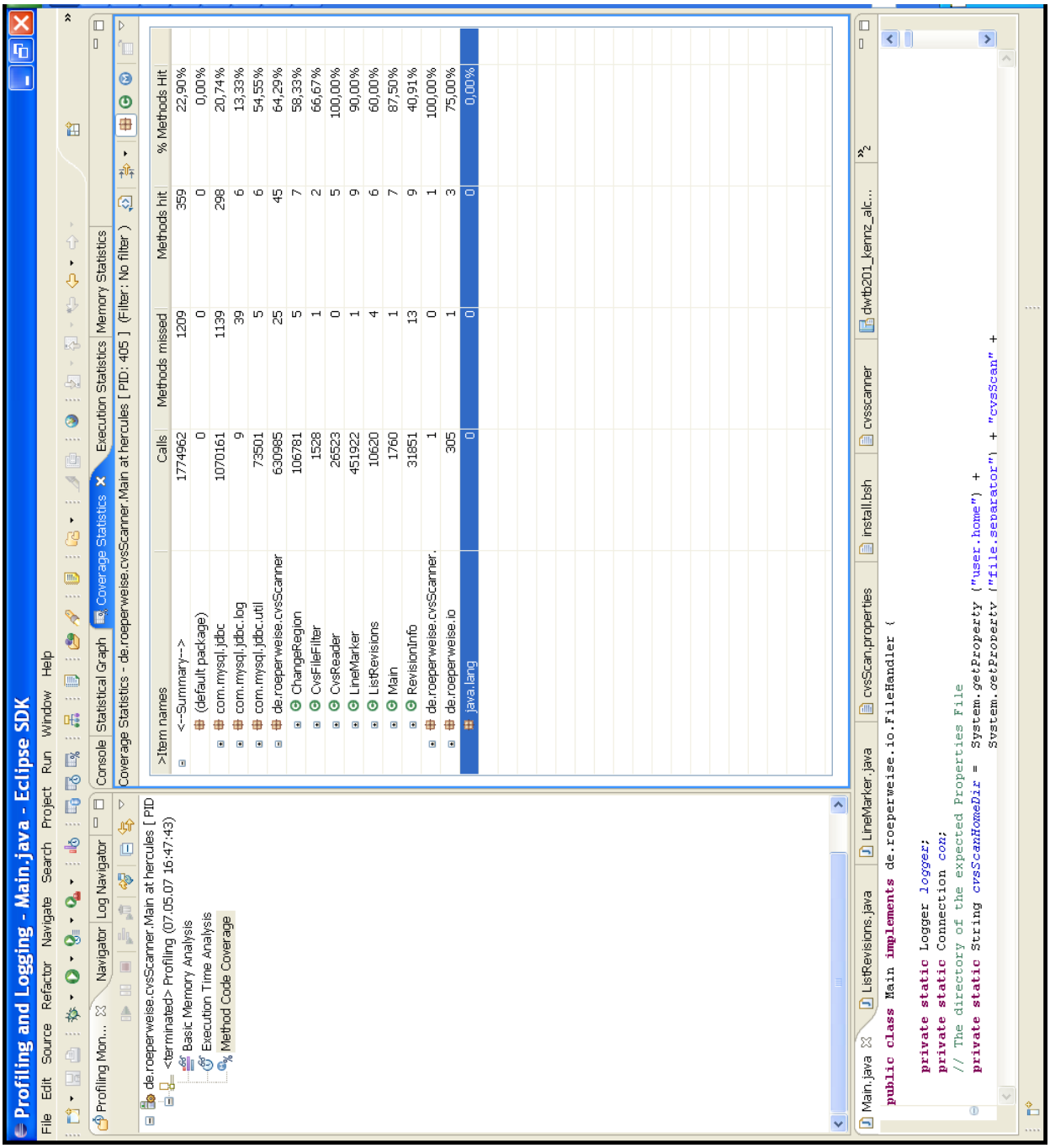


Abb. 03 (Laufzeitverhalten)

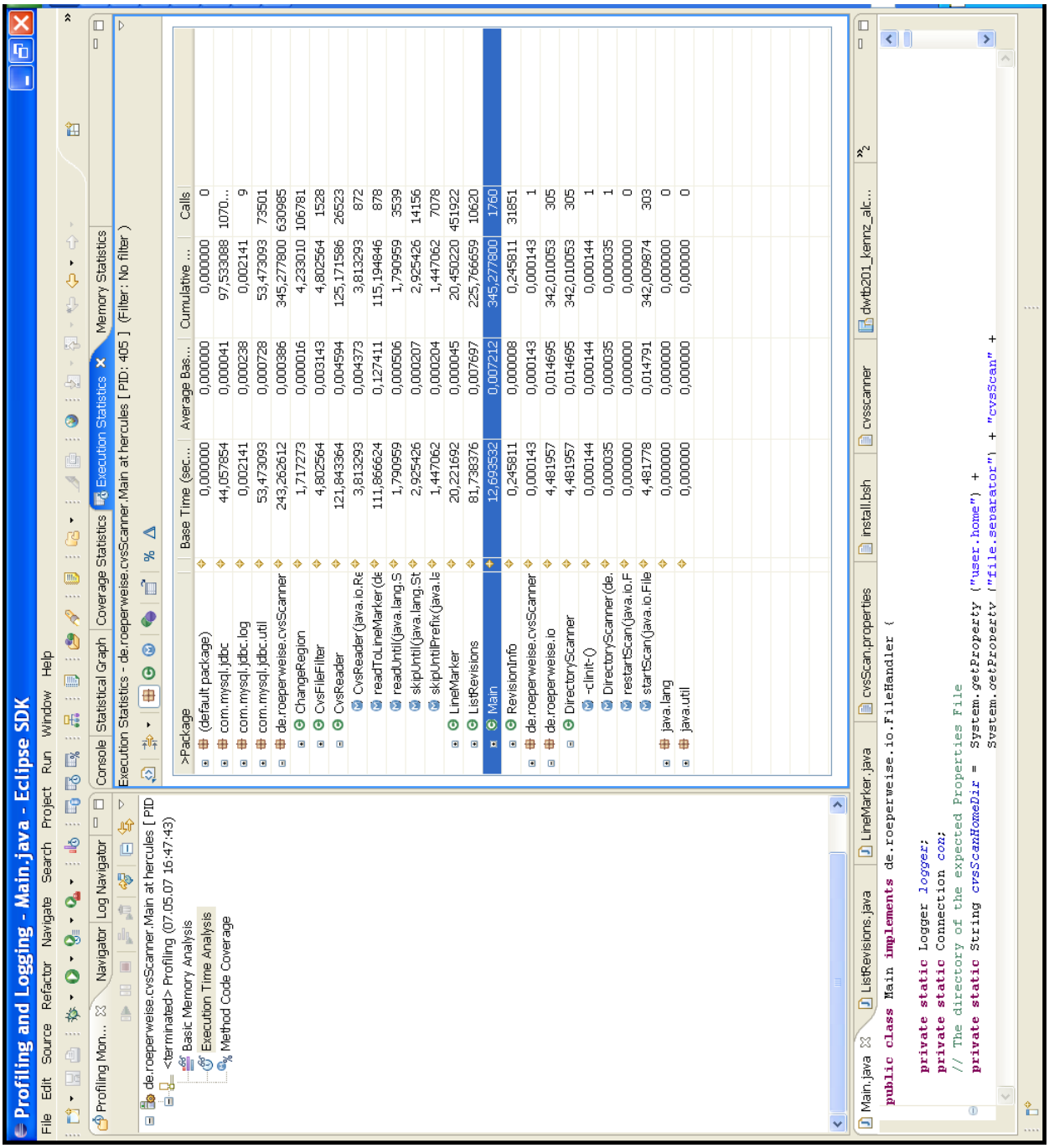


Abb. 04 (Arbeitsspeicherverbrauch)

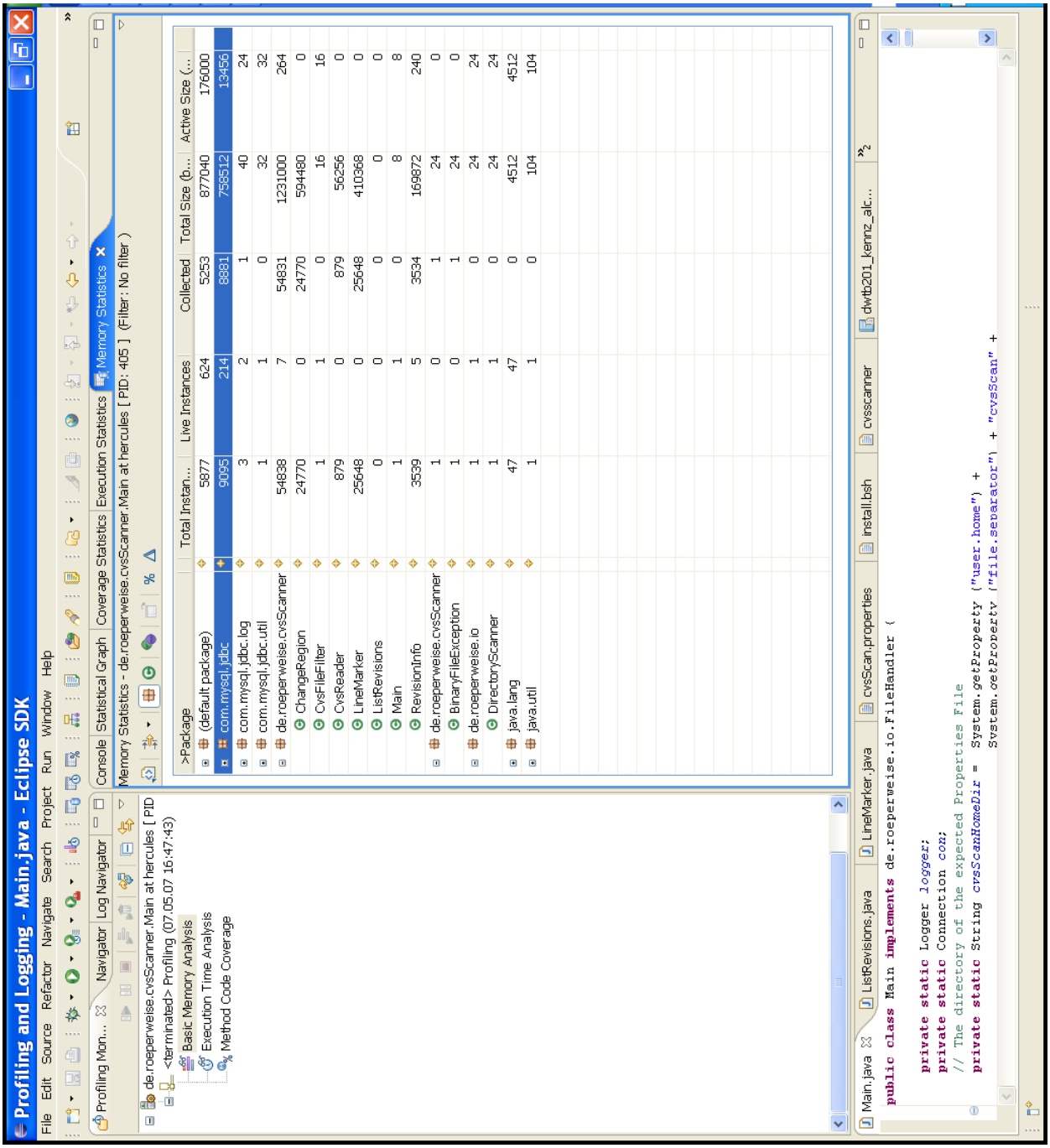
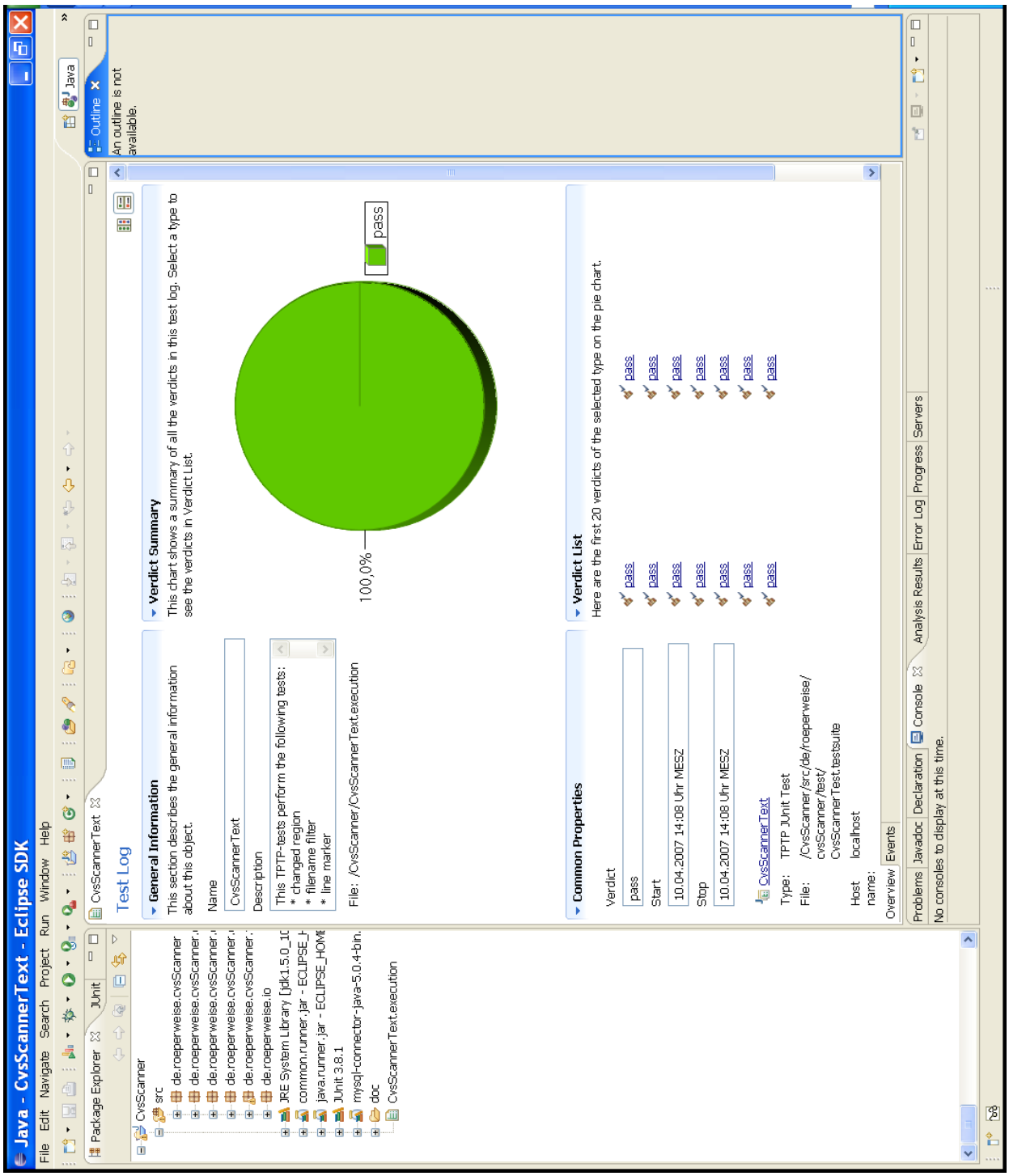


Abb. 05 (LineMarker, RegionChanger und Filenamefilter)



Verweise

Die Internetseite der Programmiersprache Java:

http://java.sun.com

Die Internetseite der IDE Eclipse:

http://www.eclipse.org

Das Modul TPTP für Eclipse:

http://www.eclipse.org/tptp

Die Internetseite des RDMS mySQL

http://www.mysql.com