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](#_Toc166233778)

[1.1 Vorwort 1](#_Toc166233779)

[1.2 RöperWeise Systems GmbH 1](#_Toc166233780)

[1.3 CvsScanner 1](#_Toc166233781)

[2 Ausgangssituation 2](#_Toc166233782)

[2.1 Ist-Analyse 2](#_Toc166233783)

[2.2 Soll-Konzept 2](#_Toc166233784)

[2.3 Projektumfeld 3](#_Toc166233785)

[2.3.1 Hardware 3](#_Toc166233786)

[2.3.2 Software 3](#_Toc166233787)

[3 Kosten-/Nutzenanalyse 3](#_Toc166233788)

[3.1 Kosten 3](#_Toc166233789)

[3.2 Nutzen 4](#_Toc166233790)

[3.3 Wirtschaftlichkeit 5](#_Toc166233791)

[4 Projektdurchführung / Realisierung 5](#_Toc166233792)

[4.1 Planungsphase 5](#_Toc166233793)

[4.2 Realisierungsphasen 5](#_Toc166233794)

[4.2.1 Erstellen der Testfälle 5](#_Toc166233795)

[4.2.2 Implementation der Testfälle 8](#_Toc166233796)

[4.2.3 Durchführung der Tests 8](#_Toc166233797)

[4.2.4 Weitere Testfälle 9](#_Toc166233798)

[5 Projektbewertung 9](#_Toc166233799)

[5.1 Soll-Ist-Vergleich 10](#_Toc166233800)

[5.2 Ausblick 10](#_Toc166233801)

[Glossar A](#_Toc166233802)

[Abbildungen C](#_Toc166233803)

[Abb. 01 C](#_Toc166233804)

[Verweise E](#_Toc166233805)

[Selbsterklärung F](#_Toc166233806)

[Erklärung der/des Auszubildenden F](#_Toc166233807)

# Allgemeines

## 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.

## 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

## CvsScanner

Das diesem Projekt zugrunde liegenden Projekt CvsScanner ist ein Kundenauftrag zur Bestimmung der Produktivität einzelner Mitarbeiter.

Der Kunde wünschte sich eine Software, welche die Produktivität der Mitarbeiter der IT-Abteilung des Unternehmens messen können sollte. Als Grundlage dieser Messung wurden LOC gewählt. Dieses LOCs sollte mittels des Versionsverwaltungsprogrammes CVS Mitarbeiter-genau und über verschiedene zeitliche Abschnitte (Die kleine Auflösung sollte 1 Tag betragen). Als einzige Einschränkung brachte der Kunde ein, dass geänderte Leerzeilen (hinzufügen, löschen) sich nicht auf die LOCs auswirken sollten.

Dieses Projekt war bei beginn dieser IHK-Projektarbeit beendet und von den Entwicklern selbstständig getestet worden.

# 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, dieses 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 abschließenden Softwaretests in diesem IHK-Projekt zu bearbeiten. Der Entwurf und die Implementation des Projektes wurden schon abgeschlossen.

Dieses Teilprojekt behandelt ausschließlich umfassende Softwaretests, die von den Entwicklern des Projektes sowohl aus zeitlich wie auch aus mentalen Gründen nicht durchgeführt wurden.

## Ist-Analyse

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

Daher soll das Projekt CvsScanner nach den abgeschlossenen Entwicklungsarbeiten nun 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 Eclipse im Unternehmen als Standardtool für die Java- Entwicklung sollten die Tests ebenfalls möglichst mit Eclipse implementiert und durchgeführt werden.

## Soll-Konzept

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

Dabei sollen alle Funktions- und Leistungsanforderungen mit den Anforderungen aus dem Pflichtenheft erneut überprüft und bestätigt werden.

Auch soll eine Testmöglichkeit ausgesucht und dokumentiert werden, um zukünftige Softwaretests effizienter und effektiver zu gestalten.

## Projektumfeld

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

### Hardware

Für die Durchführung des Projektes werden verschiedene Hardwarekomponenten benötigt:

1. Ein Entwicklungsrechner
2. Ein Testserver
3. Ein Datenbankserver

### Software

Das Projekt wird mit der folgenden Software durchgeführt. Dabei handelt es sich um den Firmenstandard der nicht weiter zur Diskussion stand:

1. IDE Eclpise
2. TPTP-Erweiterung für Eclipse
3. Remote-Access-Controller (RAC)

# Kosten-/Nutzenanalyse

## Kosten

Die Kosten dieses Teilprojektes dürften keines Falls nur für dieses Teilprojekt angerechnet werden. Da das Gesamtprojekt CvsScanner beim Kunden sehr lange im Einsatz sein wird ist ein fehlerfreier Betrieb wichtig. Desto später Fehler im Programm gefunden werden, desto teurer ist ihre Behebung. Von daher 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 €70,00 pro Stunde.

Kosten wir die genutzten Geräte sind in den 70,00€ Stundensatz schon verrechnet ebenso wie die Kosten für Verwaltung etc.

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€*** |

## Nutzen

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

Zu den Kosten für den Entwickler, der den Fehler aufnehmen, reproduzieren, dokumentieren und behoben 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. Wenn nun pro Fehler in CvsScanner von 16 Stunden Korrekturarbeit ausgegangen wird (was sehr optimistisch ist).

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

So liegen die Kosten pro Bug bei mindestens (16\*140€)=2.240€

## 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 Firma einen schwer zu beziffernden Schaden zu, welcher ebenso schwer nur korrigiert werden kann.

Insgesamt kann von einer positiven Wirtschaftlichkeit gesprochen werden.

# Projektdurchführung / Realisierung

## Planungsphase

Die Projektarbeit wurde im Durchführungszeitrum vom 05. März 2007 bis zum 16. März 2007 durchgeführt. Insgesamt standen für die Projektarbeit siebzig (70) Stunden zur Verfügung, wobei drei (3) Stunden dafür gedacht sind als Puffer zu fungieren, um etwaige Verzögerungen aufzufangen. 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 CvsScanners, wurden die Projektziele entwickelt. Als nächstes stand die Überprüfung der Durchführbarkeit an, indem die geplanten und vorhandenen Resourcen geprüft wurden.

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

## Realisierungsphasen

### 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:

#### 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 dienen die „Java Best Practices“.

Das Ergebnis des Tests ist positiv. Es sind keine weiteren Änderungen am Quelltext notwendig. Zwar gab es einige Anmerkungen im Testergebnis die jedoch sehr speziell sind und deren Richtigkeit bis heute kontrovers diskutiert wird. (Abb. 01)

#### Laufzeitverhalten

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

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

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. (Abb. 02)

#### Methodenabdeckung

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

Der Test der Methodenabdeckung und des Laufzeitverhaltens wurden in einem einzigen Testdurchlauf durchgeführt.

D.h. als Basis für den Test dient das firmeninterne CVS-Repository.

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). (Abb. 02)

#### Arbeitsspeicherverbraucht

BLABLA

#### Linemarker

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

Ein Linemarker-Obkeit 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 dienten Testdateien bzw. bereits teilweise abstrahierte Testdateien in einem passenden Objekt innerhalb des Programmes. Diese Form wurde gewählt, damit man nur die Linemarker testen kann und alle. Methoden, welche für das direkte einlesen der Daten von der Festplatte in den Arbeitsspeicher zuständig sind, zu umgehen.

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

#### Regionchanger

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

Ein Regionchanger-Obkeit ist ein Objekt innerhalb des Projektes CvsScanner, welche 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 dann die LOCs aufaddiert werden.

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

Das Ergebnis des Tests ist positiv. Es sind keine weiteren Änderungen am Quelltext notwendig. Die Ergebnisse der Regionchanger dekcen sich mit den, vorher per Hand errechneten, Werten.

#### 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.

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 von 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. (Abb. 03)

### Implementation der Testfälle

Durch die Verwendung von Eclipse samt TPTP ist die Implementation nach dem Einarbeiten in TPTP relativ problemlos. Da TPTP auf 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 der eigentliche Testcode praktisch alles mit wenig Einstellarbeit automatisch erzeugt wird.

### Durchführung der Tests

Nach einigen wenigen Testlaufen auf dem Entwicklungs-PC selber wurde schließlich das Testumfeld genutzt, welches dem Einsatz beim Kunden relativ nahe kommt.

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

Ein zweiter Server stellte den Datenbankserver dar. Dazu wurde ein bereits existierender Datenbankserver nur um eine Datenbank erweitert und die entsprechenden Parameter in CvsScanner eingestellt.

Über TPTP und dem 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 (Abb. 01).

### Weitere Testfälle

Prinzipiell lassen sich natürlich noch beliebig viele weitere Testfälle ausdenken, welche des Programmes 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:

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.

# 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 Test | 9h | 8h | -1h |
| Dokumentation | 18h | 19h | +1h |
| Summe | 70 h | 70h | ±0h |

## 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.

Es ist aber auch keinesfalls davon auszugehe, dass die Software absolut perfekt ist. Dieser Zustand ist praktisch nicht zu erreichen und kann deshalb nicht bewiesen werden. Jedoch hat die Betrachtung der Wirtschaftlichkeit in Kapitel 5 gezeigt, dass das Projekt aus wirtschaftlichen Gesichtspunkten durchaus positiv zu bewerten ist.

Es zeigt sich, dass die Software sowohl in ihrem normalen Rahmen als auch unter erschwerten bzw. ungewöhnlichen Bedingungen fehlerfrei arbeitet.

## Ausblick

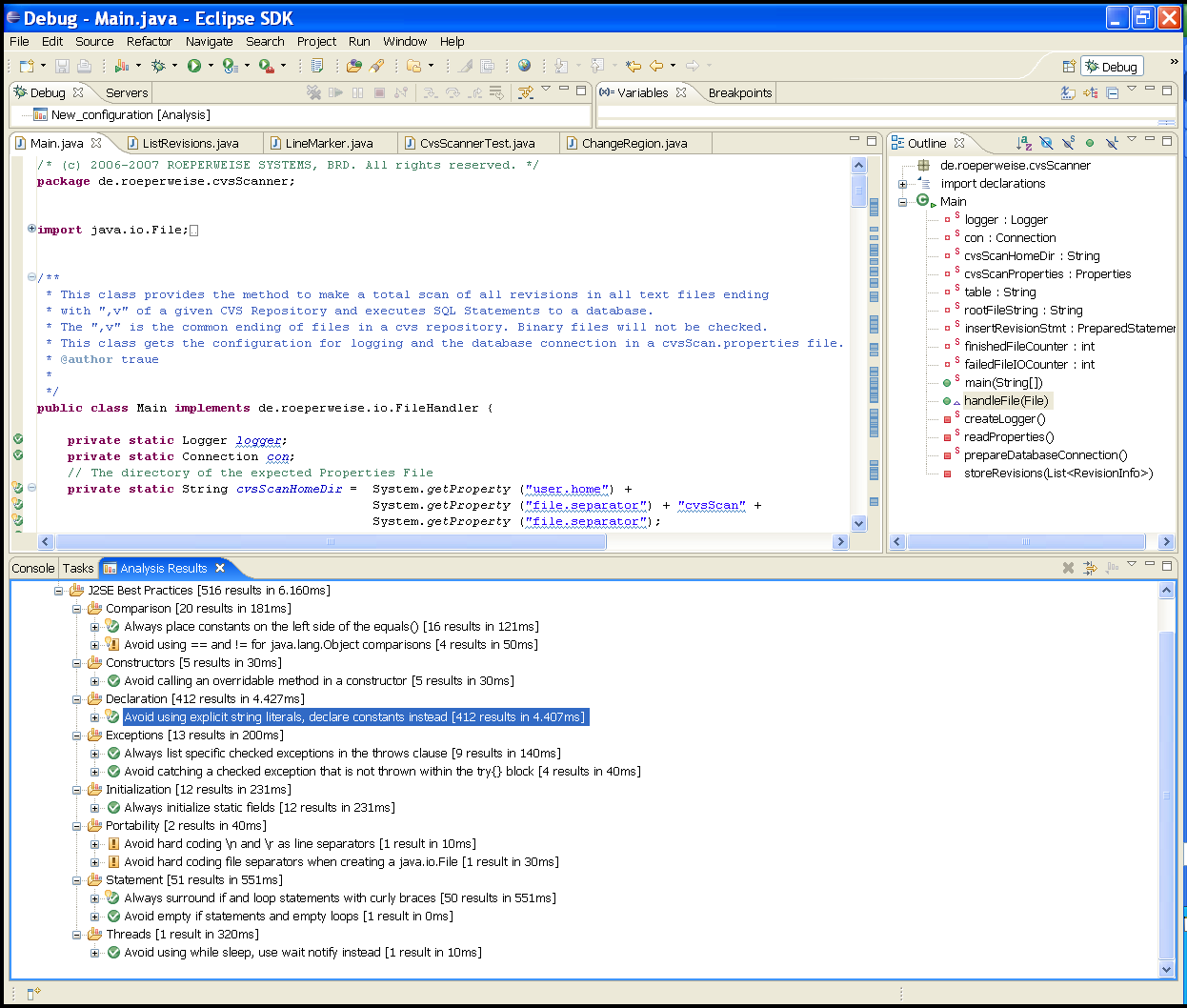
Das Ziel des Projektes wurde erreicht und der Kunde erhält ein qualitativ hochwertiges Produkt, welches ihn bei seinen täglichen Prozessen unterstützt.

# Glossar

|  |  |
| --- | --- |
| Wort | Erklärung |
| Blackbox-Test | Bei einem Blackbox-Test hat der Ersteller des Tests kein Wissen über den inneren Aufbaue des zu testenden Objektes.  Er besitzt nur Dokumentation darüber, wie ein Objekt auf bestimmte Anfragen reagieren soll. |
| CVS- Repository | 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, welche 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 von Programmieren. Sie beschreiben z.B. wie Programmcode kommentiert und eingerückt werden soll. |
| 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 kann also bedingt die Komplexität und der Umfang einer Softwareentwicklung definiert werden. |
| 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. |
|  |  |

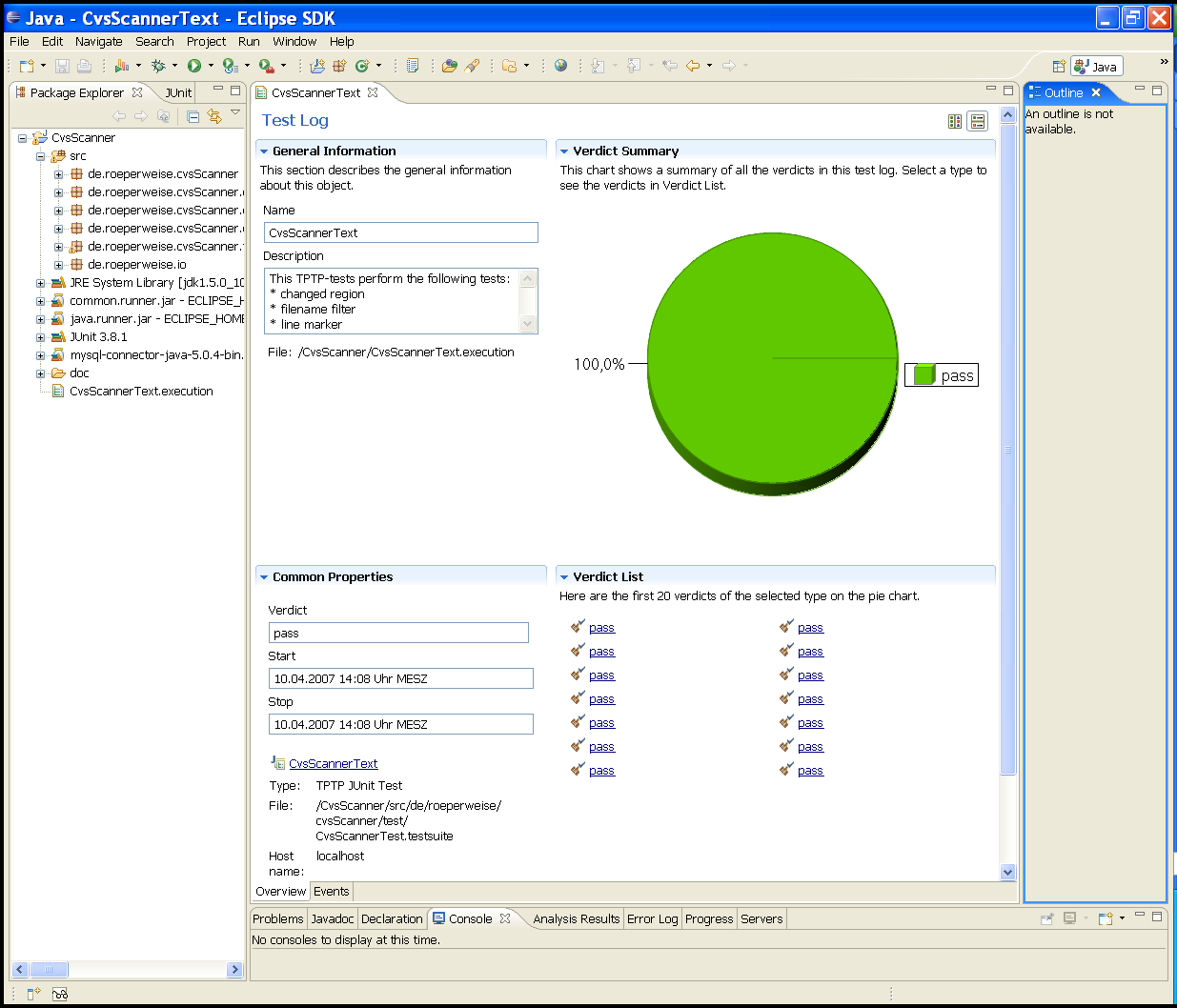
# Abbildungen

## Abb. 01



## Abb. 02

## Abb. 03



# 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>