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 Einleitung 1](#_Toc165994982)

[2 Auftrag 1](#_Toc165994983)

[2.1 Problembeschreibung (Ist- Zustand) 1](#_Toc165994984)

[2.2 Projektziel (Soll- Zustand) 1](#_Toc165994985)

[3 Anforderungen 2](#_Toc165994986)

[4 Projektziele 3](#_Toc165994987)

[4.1 Blackbox- und Whitebox-Testing 3](#_Toc165994988)

[4.2 Statische Tests 3](#_Toc165994989)

[4.3 Dynamisch Tests 4](#_Toc165994990)

[4.4 Zusammenfassung der Testarten 5](#_Toc165994991)

[4.5 Testobjekte 5](#_Toc165994992)

[5 Zeitplanung 5](#_Toc165994993)

[6 Wirtschaftlichkeit 6](#_Toc165994994)

[7 Projektverlauf 7](#_Toc165994995)

[8 Fazit 8](#_Toc165994996)

[9 Anhang A](#_Toc165994997)

[9.1 Glossar A](#_Toc165994998)

[9.2 Erfolgreicher Lauf von TPTP B](#_Toc165994999)

# Einleitung

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.

# Auftrag

## Problembeschreibung (Ist- Zustand)

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.

## Projektziel (Soll- Zustand)

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.

# Anforderungen

Da die IDE Eclipse im Unternehmen als Standardwerkzeug zur Erstellung von Java-Applikationen eingeführt wurde, sollten nach Möglichkeit auch die Tests selber in Eclipse erstellt, ausgeführt und ausgewertet werden. Dies würde den Vorteil bieten, das ein ständiges Wechseln der Werkzeuge entfällt und auch die Lernkurve zum Erstellen von Tests wesentlich steiler verläuft, da die IDE an sich bekannt ist und nur die neuen Möglichkeiten zum Erstellen der Tests erlernt werden müssen.

Alle diese Anforderungen erfüllt TPTP. Dabei handelt es sich einerseits um ein Framework um Testsoftware zu schreiben. Andererseits bietet es mit seiner API zusätzlich die Möglichkeit eigene Erweiterungen für TPTP zu erstellen, um bestimmte Testsituationen abzudecken. Dies war bei diesem Projekt aber nicht von Nöten.

Ein weiterer Vorteil ist, dass das weit verbreitete und bekannte JUnit, welches auch zur Auswahl als Testentwicklungsumgebug stand, als Teil von TPTP zum Einsatz kommt. JUnit ist den meisten Entwicklern bekannt und so erlernen sie die neuen Möglichkeiten sehr schnell, da es auf bekanntem fußt.

Außerdem bietet TPTP eine Reihe von vorgefertigten statischen Tests um die Codequalität zu bewerten und mögliche Fehlerquellen zu erkennen.

Diese Möglichkeit bietet zwar auch Hamurapi, welches ebendfalls zur Auswahl stand. Dieses Werkzeug bietet aber ausschließlich statische Tests und lässt sich nicht ohne weiteres in die Eclipse-IDE integrieren.

So lassen sich mit TPTP sowohl dynamische als auch statische Tests erstellen. Dies ist sehr gut, da beide Testbereiche nötig sind und unterschiedliche Bereiche der Softwarequalität garantieren. Auch bietet TPTP ausführliche und aussagekräftige Berichte über die durchgeführten Tests um Mögliche Probleme schnell und sicher zu identifizieren.

# Projektziele

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-Methoden nötig, welche hier besprochen und durchgeführt wurden:

## Blackbox- und Whitebox-Testing

Das Gesamtprojekt wurde von den Entwicklern mittels Whitebox-Testing schon überprüft. D.h. die Entwickler kennen ihre Softwarte und kennen die verschiedenen Ausführungspfade, die je nach übergebenem Parameter in jeder Methode zum Tragen kommen. Dies ist einerseits positiv zu bewerten, da so sehr gut gewährleistet ist, dass alle denkbaren Ausführungswege einer Methode durchlaufen werden.

Andererseits besteht das Problem, dass der Entwickler seiner Software „zu nah“ steht und so ungewollte Lücken in seiner Softwareübersieht und diese auch nicht testet.

Die Abschlusstests die im Rahmen dieses Teilprojektes durchgeführt wurden, wurden mittels Blackbox-Testing durchgeführt. Dies bedeutet einerseits, dass möglicherweise nicht alle Ausführungspfade der getesteten Methoden durchlaufen werden. Der weitaus positivere Effekt, der den Nachteil bei weitem überwiegt, ist jedoch, dass der Entwickler der Blackbox-Tests keine Kenntnis über den innen Aufbau einer Methode hat. Er kann sich nur anhand der Dokumentation ein Bild davon machen, welche Rückgabewerte eine Methode liefern sollte. Dies nimmt er auch an und überprüft jede Methode mit allen denkbaren Parametern und kontrolliert, ob die Rückgabewerte korrekt sind.

Blackboxtests werden im Rahmen dieses Teilprojektes durchgeführt, um die korrekte Funktion der einzelnen Methoden zu garantieren und damit zu gewährleisten, dass die Software als ganzes auch den Erwartungen entspricht.

## Statische Tests

Statische Tests betrachten nur den Source-Code und versuchen daran potentielle Schwachstellen in der Software ausfindig zu machen. Zusätzlich überprüfen sie den Code auch auf stilistische Konventionen um die Wiederverwendung, Lesbarkeit und Erweiterbarkeit des Codes zu gewährleisten.

Dabei bauen die statischen Tests auf Datenbanken auf, die die potentiellen Schwachstellen und Konventionen enthalten.

TPTP enthält bereits eine Datenbank der „Java Best Practises“ (Link???) und bietet somit „von Haus aus“ eine weitreichende Basis an statischen Tests.

Dieses Datenbank lässt sich um weitere Einträge erweitern um zusätzliche (z.B. firmeninterne) Konventionen und Regeln zu befolgen und Schwachstellen auf Grund von Source-Code zu erkennen.

Statische Tests werden im Rahmen dieses Projektes durchgeführt, um eine gute Codequalität zu sichern und potentielle Schwachstellen im Code zu erkennen und zu beseitigen.

## Dynamisch Tests

Dynamische Tests werden bei der Laufzeit der Applikation in einer Testumgebung erstellt. Dabei werden verschiedene (parametresierbare) Parameter der Applikation überwacht und gespeichert.

Als ein wichtiger Punkt wird die Laufzeit jeder einzelnen Methode gespeichert und ermöglicht so später besonders rechenintensive Methoden zu isolieren und ggf. zu verbessern.

Ein anderer Punkt die die Überwachung des Arbeitsspeichers um potentielle Speicherlecks (Speicher wird angefordert aber nie wieder freigegeben) zu isolieren und zu beheben.

Als letzter Punkt ist die Codeabdeckung zu nennen. Dabei wird überwacht ob und wie häufig die einzelnen Methoden aufgerufen werden .So lassen sich z.B. alte Methoden finden die unnötiger Weise noch im Source-Code vorhanden sind aber keine Funktion mehr erfüllen.

Dynamische Tests werden im Rahmen dieses Projektes durchgeführt, um später in produktiver Umgebung stabil zu laufen. Auch dienen sie dazu die Laufzeit der einzelnen Teilprogramme möglichst gering zu halten. Dies ist nötig, da der Benutzer teilweise direkt mit Programmen von CvsScanner in Berührung kommt und nicht zu lange auf ein Ergebnis warten soll.???

## Zusammenfassung der Testarten

Bei den Tests des Projektes CvsScanner werden statische und dynamische (Blackbox) Tests durchgeführt.

Dabei wird ausschließlich TPTP mit all seinen Möglichkeiten eingesetzt, was jedoch mehr als ausreichend ist um die geschilderten Punkte zu testen und zu bewerten.

## Testobjekte

Einerseits sollen Anhand der statischen Tests eine ausreichende Code-Qualität gewährleistet werden. Den wichtigeren Aspekt stellen jedoch die dynamischen Tests dar, da sie den fehlerfreien Ablauf des Programmes sicherstellen und so dem Kunden ein fehlerfreies Produkt garantieren.

Um den Sinn der dynamischen Tests zu verstehen ist eine nähere Betrachtung von CvsScanner nötig:

Der Kunde wünscht sich eine Software, die die LOC-Änderungen pro Mitarbeiter errechnet um die Produktivität des Mitarbeiters zu bestimmen. Dieses Errechnet das Projekt CvsScanner.

Dabei gab der Kunde für das Projekt vor, dass alles als Codeänderung gezählt wird bis auf das Einfügen und Löschen von Leerzeilen. So z.B. auch das Einfügen eines Whitespaces in einer Zeile. Diese Anforderung wurde bei den Tests berücksichtigt.

Wie genau der Algorithmus zum bestimmen der LOCs arbeitet ist bei der Erstellung der Tests nicht bekannt (vgl. Blackbox-Testing). Allerdings wurden Testdateien erstellt und deren LOC per Hand mittels der Anforderungen ermittelt. Mit Hilfe dieser Testdateien lassen sich auch (für das Programm CvsScanner) extreme Situationen nachstellen, die in der Realität eher nicht vorkommen.

# Zeitplanung

# Wirtschaftlichkeit

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 Stundensatz von €70,00 pro Stunde. Zzgl. Kosten für Internetzugang, Abschreibung der benutzen Geräte etc. belaufen sich die Kosten auf rund €5.000,00.

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

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.

# Projektverlauf

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:

# Fazit

Insgesamt kann das Teilprojekt als erfolgreich bezeichnet werden. Es wurden zwar keine neuen Fehler aufgedeckt, jedoch hat die Betrachtung der Wirtschaftlichkeit in Kapitel 5 gezeigt, dass das Projekt aus wirtschaftlichen Gesichtspunkten durchaus positiv zu bewerten ist.

Auch die dokumentierten Erkenntnisse über die Verwendung von TPTP zu effizienten Erstellung von Softwaretest werden in den betrieblichen Alltag Einfluss nehmen und die Qualität der Software weiter erhöhen.

Das Ziel des Projektes wurde erreicht und der Kunde kann ein gutes Produkt in empfang nehmen.

# Anhang

## Glossar

|  |  |
| --- | --- |
| 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. |
| 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. Ein Repository beheimatet eine beliebige Menge dieser Dateien in einer Ordnerstruktur. Die Nutzung von CVS ist besonders im Rahmen von Softwareentwicklung sehr verbreitet. |
|  |  |

## Erfolgreicher Lauf von TPTP

