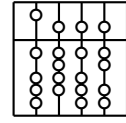


Technische Universität München
Fakultät für Informatik



Systementwicklungsprojekt

Entwicklung eines Datenerfassungs- und analysewerkzeugs zur
Experimentauswertung

Aufgabensteller: Prof. Dr. Dr. h.c. Manfred Broy

Bearbeitern: Guy-Armand Kouam Wotchung
Qi Zhu

Betreuer: Dipl.-Inf. Stefan Wagner

Abgabetermin: 9. Februar 2005

Inhaltsverzeichnis

1	Einleitung	3
1.1	Allgemein	3
1.2	Ziel des Projekts	3
2	Grundlegende Bezeichnungen	5
2.1	QS-Maßnahme	5
2.1.1	Konstruktive QS-Maßnahmen	5
2.1.2	Analytische QS-Maßnahmen	5
2.2	Ausfalldaten und Time between Failures	5
2.3	Metriken und Komplexitätserfassung	6
3	Erfassungstool	7
3.1	Überblick und Rolle	7
3.2	Eingaben	7
3.2.1	Ausfälle (CSV-Dateien)	7
3.2.2	TBF (Txt-Datei)	8
3.2.3	Fehler (CSV-Dateien)	8
3.3	Berechnungen	8
3.4	Ausgaben	9
3.4.1	CSV-Dateien für TBF's	9
3.4.2	XML-Datei	9
3.5	GUI	11
3.5.1	Die Editoren	11
4	Analysetool	14
4.1	Überblick	14
4.2	Eingaben	14
4.3	Berechnung der Effizienzen	15
4.3.1	Mean-value Methode	15
4.3.2	failure-intensity Methode	16
4.4	Ausgaben	17
4.4.1	Tabelle der Effizienzen	17
4.4.2	Chart	17
4.5	GUI	17
5	Zusammenfassung und Ausblick	21

1 Einleitung

1.1 Allgemein

Softwarebasierte Systeme sind mittlerweile ein integraler Bestandteil unseres täglichen Lebens. Sie übernehmen zunehmend Aufgaben, von denen die Existenz von Unternehmen oder sogar Menschenleben abhängen. Software-Entwickler sind deshalb einem immer stärker werdenden Druck ausgesetzt, zuverlässige und sichere Software zu erstellen. Der Begriff *Qualität* wird in *DIN 55350*, Teil 11 [1] folgendermaßen definiert:

„Qualität ist die Gesamtheit von Eigenschaften und Merkmalen eines Produkts oder einer Tätigkeit, die sich auf deren Eignung zur Erfüllung gegebener Erfordernisse bezieht.“

Mit der Durchführung von gezielte, erprobte und gute Qualitätssicherungsmaßnahmen kann die Qualität eines Software-Produkts objektiv sichergestellt werden. Dies ist notwendig, weil ohne explizite Überprüfung der Qualität eines Softwareprodukts, meistens seine Qualität nicht gewährleistet werden kann.

1.2 Ziel des Projekts

Hintergrund dieses Projekts sind verschiedene Experimente im Bereich Software Engineering, die am Lehrstuhl von Prof. Broy durchgeführt werden sollen. Die Experimente beziehen sich auf die Effizienz von Qualitätssicherungsmaßnahmen, wie z.B. Tests und Reviews. Das Ziel dieses Projektes war, deshalb ein Werkzeug zu entwickeln, das für die Erfassung und die Analyse von Experimentdaten eingesetzt werden kann. Da die Software im Rahmen eines Forschungsprojekts eingesetzt werden soll, mußte ein gewisser Qualitätsgrad der Software erreicht werden. Es sollte zuverlässig und effizient arbeiten. Der Einsatz in der Forschung bringt es auch mit sich, dass sehr wahrscheinlich später Änderungen an der Software durchgeführt werden muss, d.h. sie mußte modular, leicht erweiterbar und gut dokumentiert sein. Das hier beschriebene Tool ist in Java implementiert. Das Programm ist deshalb auf jeder PC-Hardware nutzbar und kann auf jedem Betriebssystem laufen, worauf eine Java Virtualmaschine installiert ist. Das System besteht im Wesentlichen aus 2 Komponenten (Siehe Abbildung 1):

1. Erfassungstool: Dieser Teil ist für die Erfassung der Experimentdaten zuständig.
2. Analysetool: Dieser Teil ist für die Analyse der erfassten Experimentdaten zuständig.

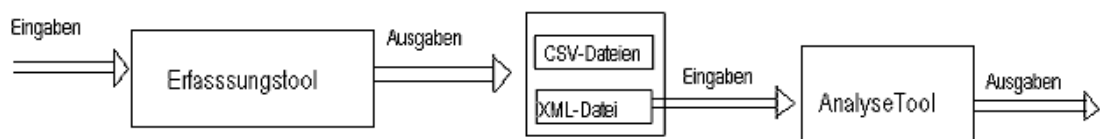


Abbildung 1: Der Aufbau des Tools

2 Grundlegende Bezeichnungen

2.1 QS-Maßnahme

Mit steigendem Softwareanteil in unserem Leben nimmt die Entwicklung von Software eine Schlüsselstellung in der Informatik ein. Die Software-Qualität bestimmt in hohem Maße die Gesamtqualität von mehreren Produkten. Deshalb wird es immer wichtiger, sich zunehmend mit der Qualität unsere Software zu befassen.

Qualität ist die Gesamtheit der Eigenschaften oder Merkmale einer Tätigkeit oder eines Produktes. Sämtliche Eigenschaften oder Merkmale müssen die gesetzten Anforderungen erfüllen und dafür geeignet sein. Qualität ist auch die Erfüllung der Kundenwünsche und die Eignung zur Erfüllung gegebener Erfordernisse, wobei aufzupassen ist dass man nicht übertreibt und vor allem dass kein Overengineering betrieben wird.

Die Qualitätssicherung befasst sich mit der Einhaltung zugesicherter Eigenschaften und Anforderungen. Eine unabhängige entwicklungsbegleitende Qualitätssicherung hat deshalb zum Ziel, Mängel im Vorfeld zu erkennen und dadurch zu vermeiden, sodass eine Software unter Berücksichtigung der anerkannten Regeln der Informatik den vereinbarten und zugesicherten Eigenschaften entspricht. Es gibt mehrere Arten von QS-Maßnahme [4], wie etwa Review/Audit, Statische Analyse, Programmverifikation, Inspektion, Walkthrough, Simulation und Tests.

2.1.1 Konstruktive QS-Maßnahmen

Konstruktive Maßnahmen sorgen dafür, dass die Produkte bzw. der Erstellprozess von vornherein bestimmte Eigenschaften erfüllen bzw. erfüllt. Konstruktive QS-Maßnahmen umfassen sowohl technische als auch organisatorische Maßnahmen:

- Technische Maßnahmen: Methoden, Werkzeuge und Sprachen
- Organisatorische Maßnahmen: Richtlinien, Standards und Checklisten

2.1.2 Analytische QS-Maßnahmen

Analytische QS-Maßnahmen bringen keine Qualität direkt ins Produkt oder den Entwicklungsprozess. Durch sie wird die existierende Qualität gemessen. (Prüfung & Bewertung der Qualität). Es existieren analysierende und testende Verfahren:

- Analysierende Maßnahmen: Statische Analyse, Programmverifikation, Review/Audit, Inspektion, Walkthrough
- Organisatorische Maßnahmen: Dynamischer Test, Symbolischer Test, Simulation, Schreibtischtest

2.2 Ausfalldaten und Time between Failures

1. **Ausfalldaten:** Ausfalldaten werden beim Testen von Software gemäß systematischer Testtechniken gewonnen. Diese Daten beschreiben Ausfälle die während des Tests einer Vorrichtung oder einer Software aufgetreten sind. Diese Daten können dann erfasst und ausgewertet werden.

2. **Time between Failures (TBF):** Dies ist die Zeit, zwischen dem Auftreten von zwei Ausfällen bei einer arbeitenden Vorrichtung oder Software.

2.3 Metriken und Komplexitätserfassung

Die Komplexität von Software-Quellcode ist ein starker Indikator für die Zuverlässigkeit der Software. Um diese zu bewerten, stehen verschiedene Komplexitätsmetriken zur Verfügung. Metriken helfen bei der Analyse eines unbekanntes Source Codes und liefern wichtige Informationen bevor Sie einen einzigen Blick in den Sourcecode geworfen haben. Diese Informationen sind nicht nur reine Masszahlen wie Anzahl der Codezeilen oder Anzahl der Klassen. Sie beinhalten auch Hinweise auf die Komplexität des Codes und seine innere Struktur. Mit Hilfe von Metriken können Sie die Wiederverwendbarkeit von Code wie auch den Aufwand für Funktionstests sowie Wartungsarbeiten abschätzen. Dieses Werkzeug berechnet mehrere Softwaremetriken wie etwa: Halstead [2], McCabe [3] oder die "Cyclomatic Complexity" [7]

3 Erfassungstool

3.1 Überblick und Rolle

Diese Komponente dient dazu, verschiedene Daten über eine Qualitätssicherungsmaßnahme zu erfassen. Diese Daten können folgendermaßen eingeteilt werden:

- Die Daten, die vor der QS-Maßnahme auftreten. Das sind Ausfall-Daten, die beim Testen der Software vor der QS-Maßnahme erfasst werden.
- Die Daten, die bei der QS-Maßnahme erfaßt werden.
- Die Daten, die nach der QS-Maßnahme auftreten. Das sind Ausfall-Daten, die beim Testen der Software nach der QS-Maßnahme erfasst werden.

Aus den, durch den Benutzer eingegebenen Daten, werden dann verschiedene Werte ausgerechnet. Das ganze wird dann am Ende in eine XML-Datei geschrieben.

3.2 Eingaben

Bei der Benutzung des Erfassungstools, gibt es mehr Eingaben, die über eine entsprechende Datei eingegeben werden können. Man speichert die Daten in einer Datei mit der entsprechenden Struktur und gibt den Pfad der Datei an das Tool weiter, damit dieses die Daten herauslesen kann. Es besteht auch für alle diese Eingaben die Möglichkeit, sie interaktiv in einer Tabelle einzugeben. Dabei passiert dann folgendes: Die eingegebene Daten werden dann in einer Datei gespeichert, die anschließend eingelesen wird.

Es gibt natürlich auch Eingaben die nur interaktiv (über Eingabefelder) eingegeben werden können. Hier werde ich aber nur die behandeln, die über Dateien eingegeben werden können.

3.2.1 Ausfälle (CSV-Dateien)

Die Ausfälle, die vor der QS-Maßnahme oder die nach der QS-Maßnahme aufgetreten sind, werden in entsprechenden CSV-Dateien gespeichert. Die Pfade der Dateien werden dann an das Tool weitergegeben, damit dieses die Daten herauslesen kann. Eine CSV-Datei mit den Ausfalldaten muss folgende Struktur haben:

Ausfall;Ausfallzeit;Schwere

Dabei ist die Spalte "Ausfall" nur für die Auflistung der Daten gedacht. Die Spalte "Ausfallzeit" enthält die Zwischenzeit zwischen dem jetzigen Ausfall und dem folgenden. Die Spalte Schwere beinhaltet die zu den Ausfällen gehörende Schwere.

Ein Beispiel für den Inhalt einer Ausfalldatei wäre:

```
Ausfall;Ausfallzeit;Schwere
Ausfall1;100;3
Ausfall2;300;4
Ausfall3;400;5
Ausfall4;800;6
Ausfall5;1300;7
```

3.2.2 TBF (Txt-Datei)

Die TBF werden in Text-Dateien gespeichert. Die Text-Datei mit den Ausfalldaten muss folgende Struktur haben:

Nummer *Wert*

Dabei steht "Nummer" für die Reihenfolge der Daten. "Wert" steht natürlich für die "Zeit zwischen zwei Ausfälle"

Ein Beispiel für den Inhalt einer TBF-Datei wäre:

```
1 3
2 30
3 113
4 81
5 115
```

3.2.3 Fehler (CSV-Dateien)

Das Tool erfasst auch Angaben über die Fehler die bei der QS-Maßnahme gefunden und behoben worden sind. Die Fehler werden in einer CSV-Datei gespeichert. Die CSV-Datei mit den Fehlern muss folgende Struktur haben:

Art;Anzahl

Dabei enthält die Spalte "Art" der Typ beziehungsweise die Benennung des Fehlers. Die Spalte "Anzahl" enthält, wie oft dieser Typ von Fehler aufgetreten ist.

Ein Beispiel für den Inhalt einer Fehler-Datei wäre:

```
Art;Anzahl
Art1;12
Art2;9
Art3;5
Art4;7
Art5;7
```

3.3 Berechnungen

Nachdem die Daten (Entwicklername, Programmname, Ausfälle, etc. . .) von der GUI gesammelt wurden, rechnet das Erfassung je nach Wunsch des Benutzers mehrere Werte aus:

- Falls die Ausfälle eingegeben worden sind, dann berechnet das Erfassungstool die dazu gehörenden "Time between Failures"
- Die Metriken werden mit Hilfe des Metriktools berechnet.
- Die gesamte Anzahl von Fehlern wird berechnet

3.4 Ausgaben

3.4.1 CSV-Dateien für TBF's

Falls man bei der Konfiguration, die Ausgabe der TBF's in einer CSV-Datei ausgewählt hat (siehe Abbildung 2), dann werden sie in einer CSV-Datei ausgegeben die folgende Struktur hat:

Failure-Number;TBF

Dabei enthält die Spalte "Failure-Number" die Nummer des TBF's. Die Spalte "TBF" enthält dann den entsprechenden Wert.

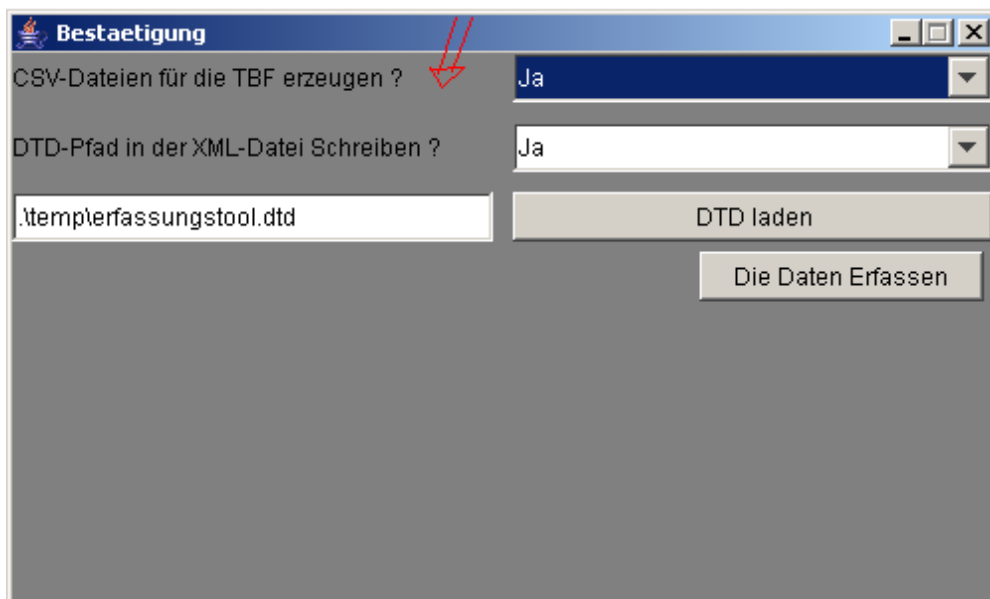


Abbildung 2: Ausgabe der TBF's in einer CSV-Datei

Ein Beispiel für die Ausgabe von TBF's wäre:

```
Failure-Number;TBF
1;100
2;200
3;100
4;400
5;500
```

3.4.2 XML-Datei

Am Ende werden alle die eingegebene und alle die berechnete Daten in einer XML-Datei geschrieben. Diese XML-Datei ist nach einer DTD namens "Erfassung.dtd" gebaut. Der Inhalt dieser DTD-Datei sieht folgendermaßen aus:

```

<!ELEMENT SEIZED_DATA (FIRST_VERSION,QAMEASURE,SECOND_VERSION)>
<!ELEMENT FIRST_VERSION (DATE,DEVELOPER_NAME,PROGRAM_NAME,MEMORY_PLACE,
FAILURES,TBFs,METRICS)>
<!ELEMENT QAMEASURE (KIND,DURATION,FAULTS)>
<!ELEMENT SECOND_VERSION (DATE,DEVELOPER_NAME,PROGRAM_NAME,MEMORY_PLACE,
FAILURES,TBFs,METRICS)>
<!ELEMENT DATE (#PCDATA)*>
<!ELEMENT DEVELOPER_NAME (#PCDATA)*>
<!ELEMENT PROGRAM_NAME (#PCDATA)*>
<!ELEMENT MEMORY_PLACE (#PCDATA)*>
<!ELEMENT FAILURES (FAILURE)*>
<!ATTLIST FAILURES QUANTITY ID #REQUIRED TIME_UNIT ID #IMPLIED>
<!ELEMENT TBFs (TBF)*>
<!ATTLIST TBFs QUANTITY ID #REQUIRED>
<!ELEMENT METRICS (METRIC1,METRIC2,METRIC3,METRIC4,METRIC5,METRIC6,METRIC7,METRIC8)>
<!ELEMENT FAILURE (NUMBER,TIME,GRAVITY)>
<!ELEMENT NUMBER (#PCDATA)*>
<!ELEMENT TIME (#PCDATA)*>
<!ELEMENT GRAVITY (#PCDATA)*>
<!ELEMENT TBF (NUMBER,VALUE)>
<!ELEMENT NUMBER (#PCDATA)*>
<!ELEMENT VALUE (#PCDATA)*>
<!ELEMENT METRIC1 (LINE)*>
<!ATTLIST METRIC1 METRICSNAME ID #REQUIRED>
<!ELEMENT METRIC2 (LINE)*>
<!ATTLIST METRIC2 METRICSNAME ID #REQUIRED>
<!ELEMENT METRIC3 (LINE)*>
<!ATTLIST METRIC3 METRICSNAME ID #REQUIRED>
<!ELEMENT METRIC4 (LINE)*>
<!ATTLIST METRIC4 METRICSNAME ID #REQUIRED>
<!ELEMENT METRIC5 (LINE)*>
<!ATTLIST METRIC5 METRICSNAME ID #REQUIRED>
<!ELEMENT METRIC6 (LINE)*>
<!ATTLIST METRIC6 METRICSNAME ID #REQUIRED>
<!ELEMENT METRIC7 (LINE)*>
<!ATTLIST METRIC7 METRICSNAME ID #REQUIRED>
<!ELEMENT METRIC8 (LINE)*>
<!ATTLIST METRIC8 METRICSNAME ID #REQUIRED>
<!ELEMENT LINE (#PCDATA)*>
<!ELEMENT KIND (#PCDATA)*>
<!ELEMENT DURATION (#PCDATA)*>
<!ATTLIST DURATION TIME_UNIT ID #IMPLIED>
<!ELEMENT FAULTS (QTY_OF_FAULTS)*>
<!ATTLIST FAULTS QUANTITY ID #IMPLIED>
<!ELEMENT QTY_OF_FAULTS (#PCDATA)*>
<!ATTLIST QTY_OF_FAULTS KIND ID #IMPLIED>

```

3.5 GUI

Für die Entwicklung der GUI Komponenten haben wir Swing benutzt. Swing erlaubt eine einfache Erstellung von Prototypen und eine sehr schnelle Generierung von verschiedene GUI Fensters mit den Standarten Komponenten wie z.B Table, List usw. Wenn man das Tool (Erfassungstool) startet, dann bekommt man ein Fenster zu sehen wo man zwischen zwei Modi auswählen kannn (siehe Abbildung 3):

- Der erste Modus, wo die Ausfälle eingegeben werden müssen, und aus diesen Ausfälle berechnet dann das Tool die TBFs
- Der Zweite Modus, wo die TBFs selber vom Benutzer eingegeben werden.

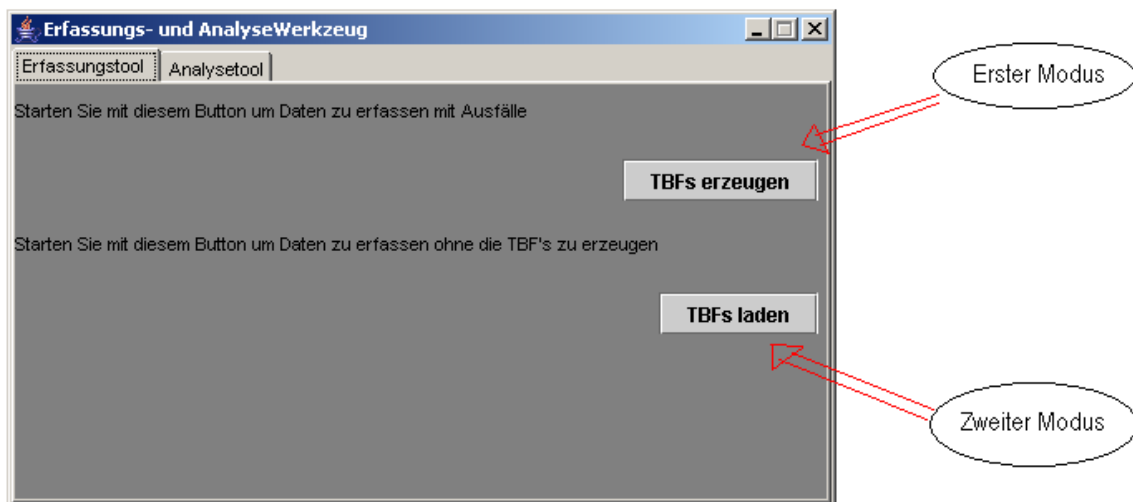


Abbildung 3: Auswahl zwischen zwei Modi

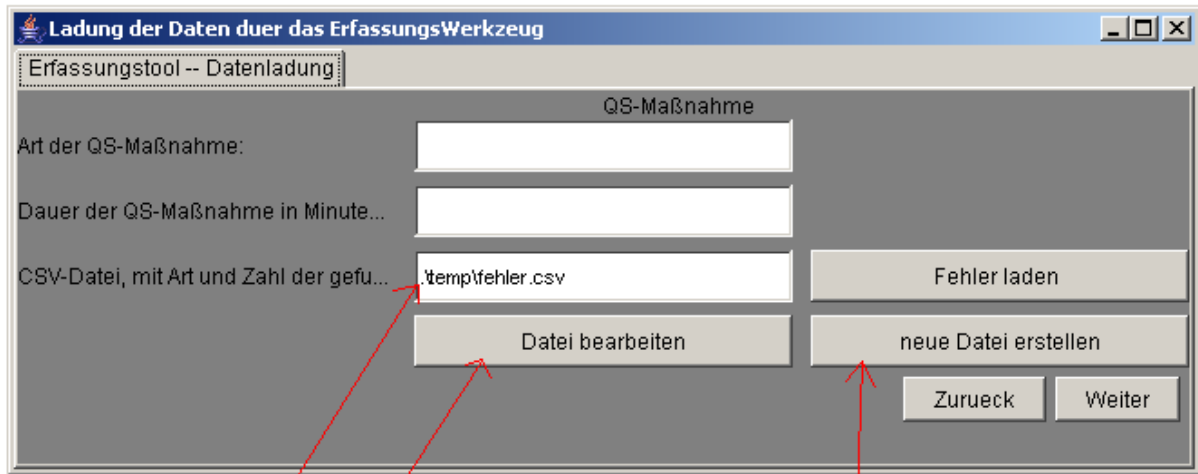
Nachdem man sich für einen Modus entschieden hat, muß man nur noch die verschiedenen Eingabefelder gut ausfüllen. Was man noch anmerken sollte ist die Benutzung der Eingebauten Editoren.

3.5.1 Die Editoren

Zu den eingebauten Editoren gelangt man auf zwei Arten (siehe Abbildung 4)

- Indem man auf den Button "neue Datei erstellen" drückt: Somit erfaßt man eine ganz neue Datei
- Indem man auf den Button "Datei bearbeiten" drückt: Somit wird eine existierende Datei weiter bearbeitet.

Ein neu eingegebener Datensatz in einem Datenfeld wird erst registriert, wenn man auf ein anderes Datenfeld drückt. Wenn man auf den Button "Abspeichern und weiter" drückt, wird



Editiert die Datei im Eingabefeld

Editiert eine Neue Datei

Abbildung 4: Datei editieren

dann die Datei im entsprechenden Speicherort (durch ein Eingabefeld bestimmt) gespeichert (siehe Abbildung 5). Und die Sammlung der Eingaben wird weitergeführt.

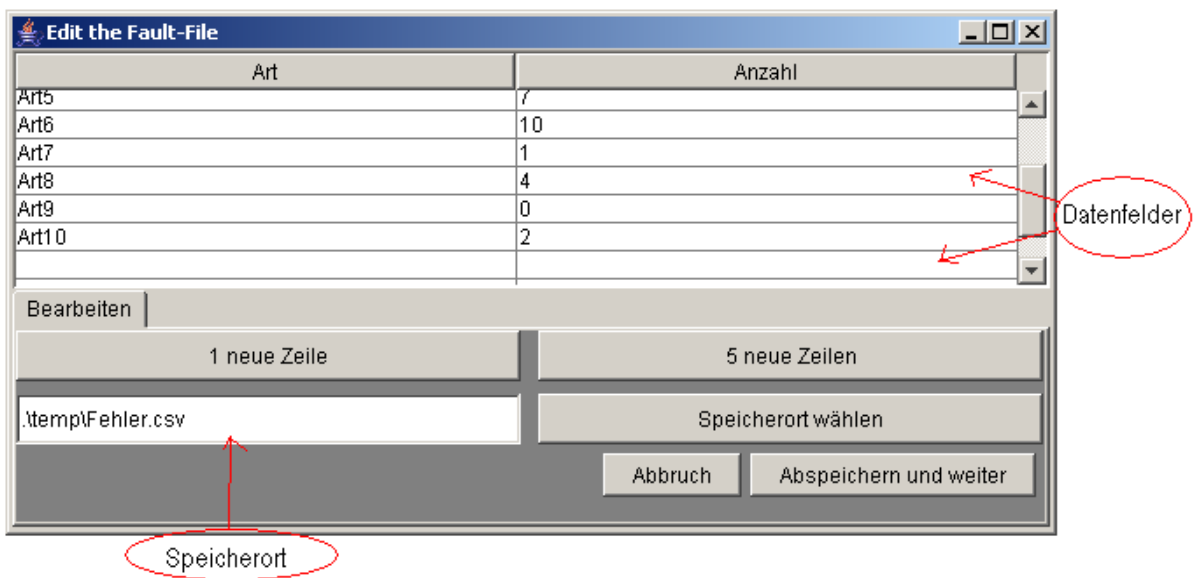


Abbildung 5: Der eingebaute Editor

4 Analysetool

4.1 Überblick

Nach der Erfassung u.a. der TBF-Daten wurde für jede QS-Maßnahme ein XML-Datei erstellt. All diese XML-Dateien sind für das Analysetool als Eingaben möglich. Zuerst werden die XML-Dateien analysiert, und die für die Berechnung der absoluten und relativen Effizienzen benötigten Daten werden aus den XML-Dateien herausgeholt. Diese Daten werden dann den beiden Berechnungsmethoden (mean-value Methode und failure-intensity Methode) übergeben. Nach der Berechnung werden die Ergebnisse in Tabellenform in einem Ausgabenfenster präsentiert. Schließlich hat man noch die Möglichkeit, die relative Effizienz graphisch darzustellen.

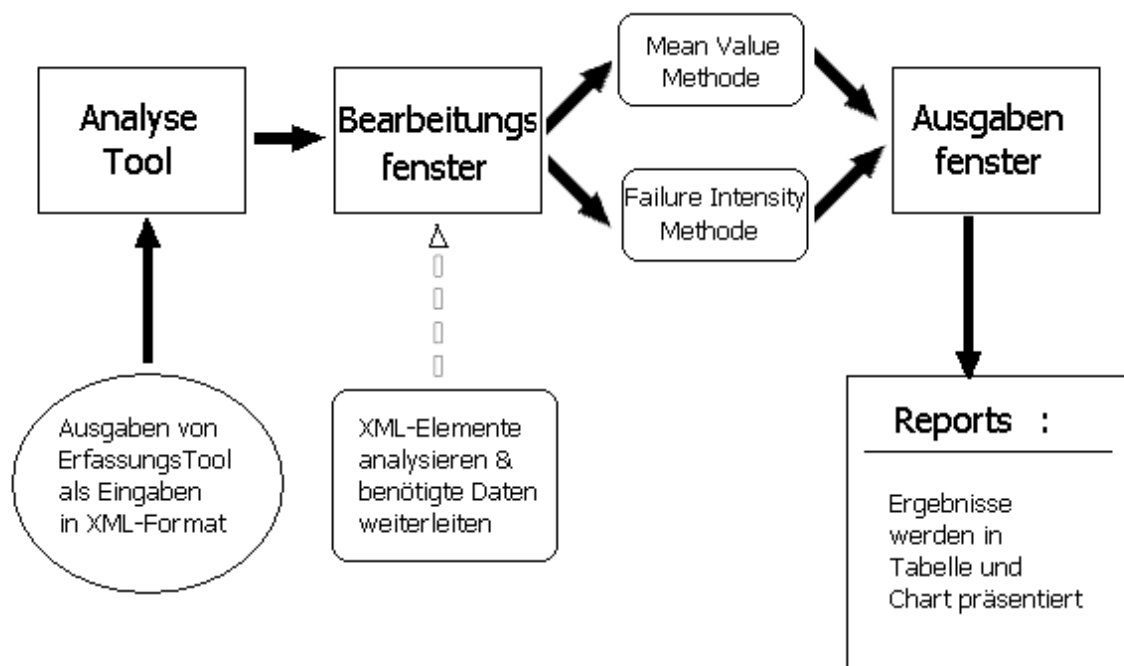


Abbildung 6: Überblick des Analysetools

4.2 Eingaben

Die Eingaben für das Analysetool sind, wie wir früher schon gesagt haben, ausschließlich XML-Dateien, die das Erfassungstool uns liefert.

Dabei muss man beachten, nur ein Teil der Informationen innerhalb dieser XML-Dateien werden bei der folgenden Effizienzberechnung benötigt, z.B. die TBF-Daten, der Name der QS-Maßnahme und die Dauer der QS-Maßnahme.

Das Parsen von XML-Dateien und die Extrahierung der o.g. Informationen ist mit einem externen Tool realisiert, nämlich JDOM (<http://www.jdom.org/>). Durch JDOM lassen sich die XML-Dateien einlesen, manipulieren und dann wieder schreiben.

4.3 Berechnung der Effizienzen

Um die QS-Maßnahmen zu analysieren und die Effizienzen zu berechnen, sind zwei Methoden entwickelt worden, nämlich die mean-value Methode und die failure-intensity Methode:

4.3.1 Mean-value Methode

Die mean-value Methode benutzt das arithmetische Mittel der TBF-Daten vor und nach der Verwendung einer QS-Maßnahme.

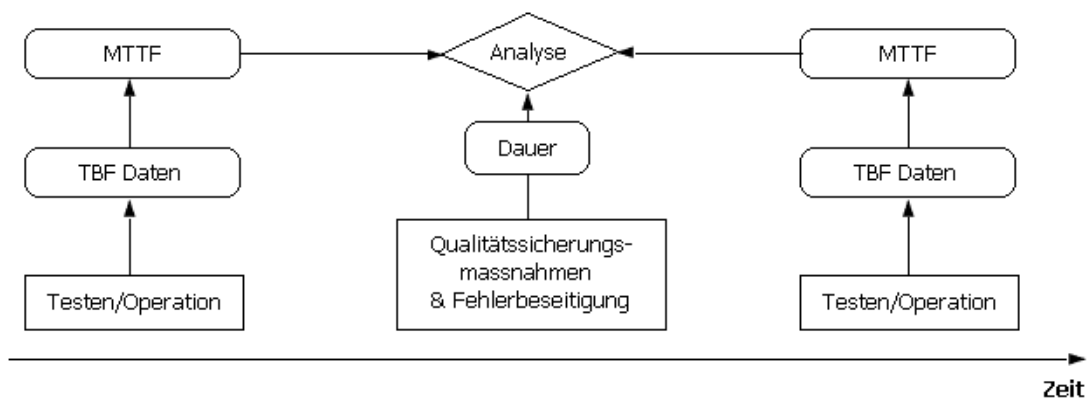


Abbildung 7: Die Analyse der QS-Maßnahme mit der mean-value Methode

Was diese Methode betrifft, fokussieren wir auf die TBF-Daten der Ausfälle, die zwischen den Anwendungen zweier QS-Maßnahmen vorkommen.

Eine solche Reihe von Daten bezeichnen wir als S_{T_n, T_k} für die TBF-Daten zwischen der n -ten und k -ten Anwendung der QS-Maßnahme. Und der Mittelwert dieser Reihe, also auf Englisch "mean time to failure" kurz ($MTTF$), sieht so aus:

$$\mathbf{MTTF}_{T_n} = \overline{S_{T_{n-1}, T_n}}$$

Und der absolute und relative Effekt \mathbf{Fa} und \mathbf{Fr} lassen sich mit folgender Formel berechnen:

$$\mathbf{Fa} = \mathbf{MTTF}_{T_{n+1}} - \mathbf{MTTF}_{T_n}$$

$$\mathbf{Fr} = \frac{\mathbf{Fa}}{\mathbf{MTTF}_{T_n}}$$

4.3.2 failure-intensity Methode

Die failure-intensity Methode benutzt den mittleren Wert der Anzahl von Ausfällen in einer bestimmten Zeiteinheit vor und nach der Verwendung einer QS-Maßnahme. Sie wird in [5] und [6] beschrieben

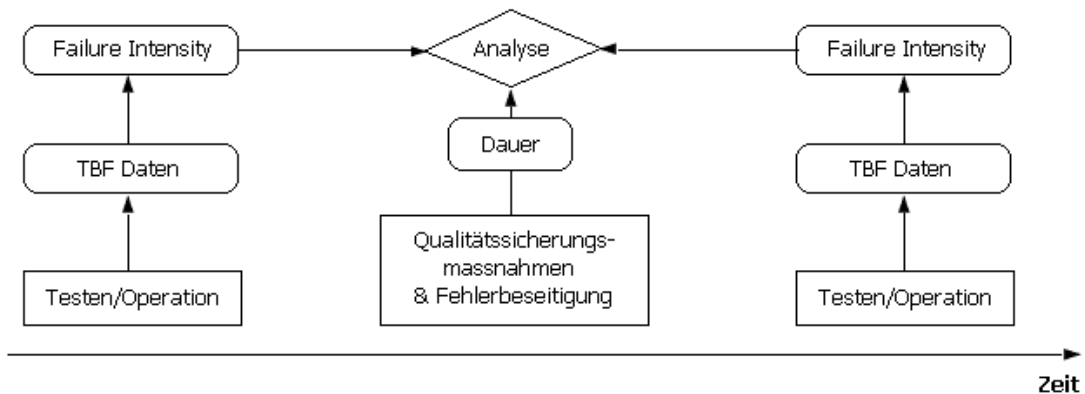


Abbildung 8: Die Analyse der QS-Maßnahme mit der failure-intensity Methode

Bei der zweiten Methode sind nur die Anzahl und die gesamte Länge des Zeitraums der Ausfälle von Bedeutung. Der Unterschied zu der ersten Methode besteht nur darin, dass diese Methode die exakte Timinginformation nicht braucht.

Die failure-intensity bezeichnen wir mit λ . Der Mittelwert der Anzahl von Ausfällen in einer Zeiteinheit vor einer bestimmten QS-Maßnahme T_n ist:

$$\lambda_{T_n} = \frac{f_{T_{n-1}, T_n}}{T_n - T_{n-1}}$$

wobei f_{T_{n-1}, T_n} die Anzahl von Ausfällen zwischen zwei QS-Maßnahmen T_n und T_{n-1} bezeichnet.

Der absolute und relative Effekt \mathbf{Fa} und \mathbf{Fr} lassen sich erneut mit folgendem Formular berechnen:

$$\mathbf{Fa} = \lambda_{T_{n+1}} - \lambda_{T_n}$$

$$\mathbf{Fr} = \frac{\mathbf{Fa}}{\lambda_{T_n}}$$

Schließlich bekommt man die absolute und relative Effizienz E_a und E_r mit der Dauer (D) der Anwendung von QS-Maßnahme wie folgendes:

$$E_a = \frac{F_a}{D}$$

$$E_r = \frac{F_r}{D}$$

4.4 Ausgaben

4.4.1 Tabelle der Effizienzen

Die berechneten absoluten und relativen Effizienzen werden am Ende der Rechnung dem Reportfenster übergeben und in einer Tabelle präsentiert.

Die Tabelle lässt sich so beschreiben: die erste Spalte zeigt die Reihenfolge von QS-Maßnahmen; in der zweiten Spalte werden die Namen der QS-Maßnahmen aufgelistet; die nächsten zwei Spalten stellen die absoluten und relativen Effizienzen dar; und die letzte Spalte ist für die mögliche Erweiterung der Software in der Zukunft gedacht. Damit man die Möglichkeit hat, nur die ausgewählten Zeilen graphisch darzustellen.

4.4.2 Chart

Die graphische Darstellung von relativen Effizienzen in einem Koordinatensystem wird durch ein externes Tool JFreeChart (<http://www.jfree.org/jfreechart>) realisiert.

Die X-Achse zeigt die Reihenfolge der Anwendung von QS-Maßnahmen, während die Y-Achse die relativen Effizienzen darstellt.

4.5 GUI

Nach dem Starten des gesamten Programms wählt man das Analysetool (siehe Abbildung 9). Mit dem Button "XML-Dateien Laden" lädt man all die XML-Dateien, die man zu berechnen hat. Dabei werden die Pfade und Namen der Dateien auch im Fenster aufgelistet.

Anschließend mit dem Button "Analysieren" werden die Pfade und Namen der XML-Dateien dem nächsten Fenster (siehe Abbildung 10) übergeben.

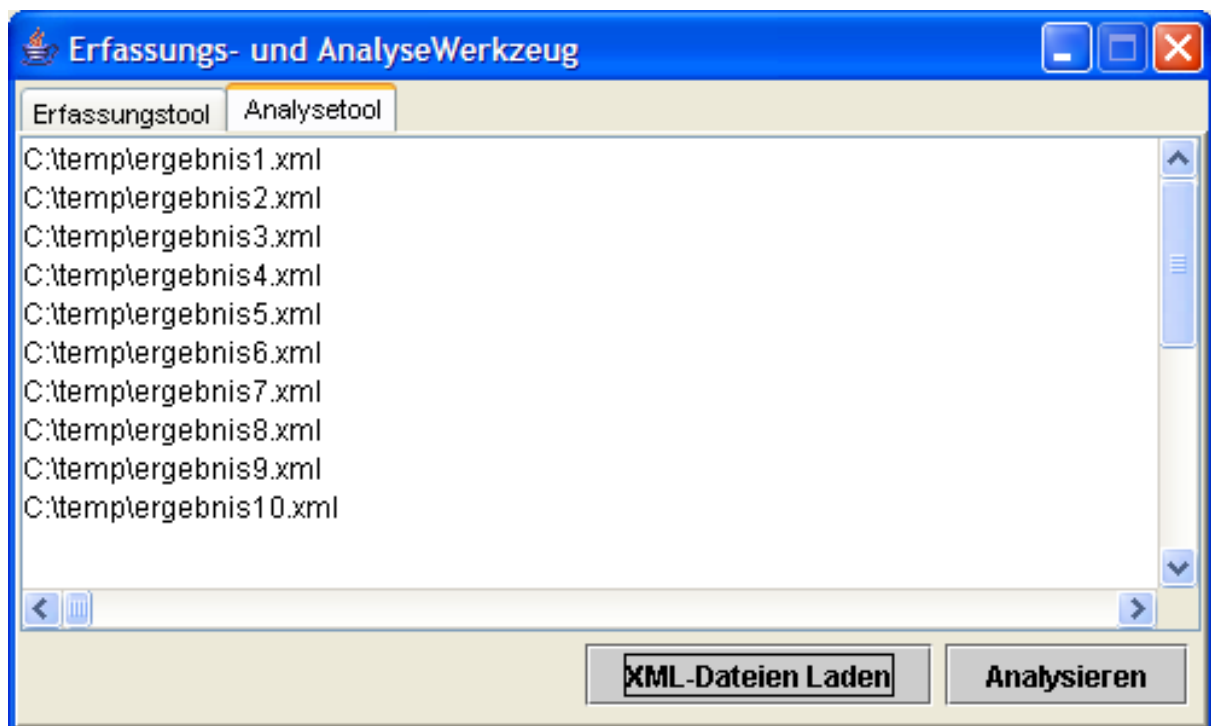


Abbildung 9: Erstes Fenster des Analysetools

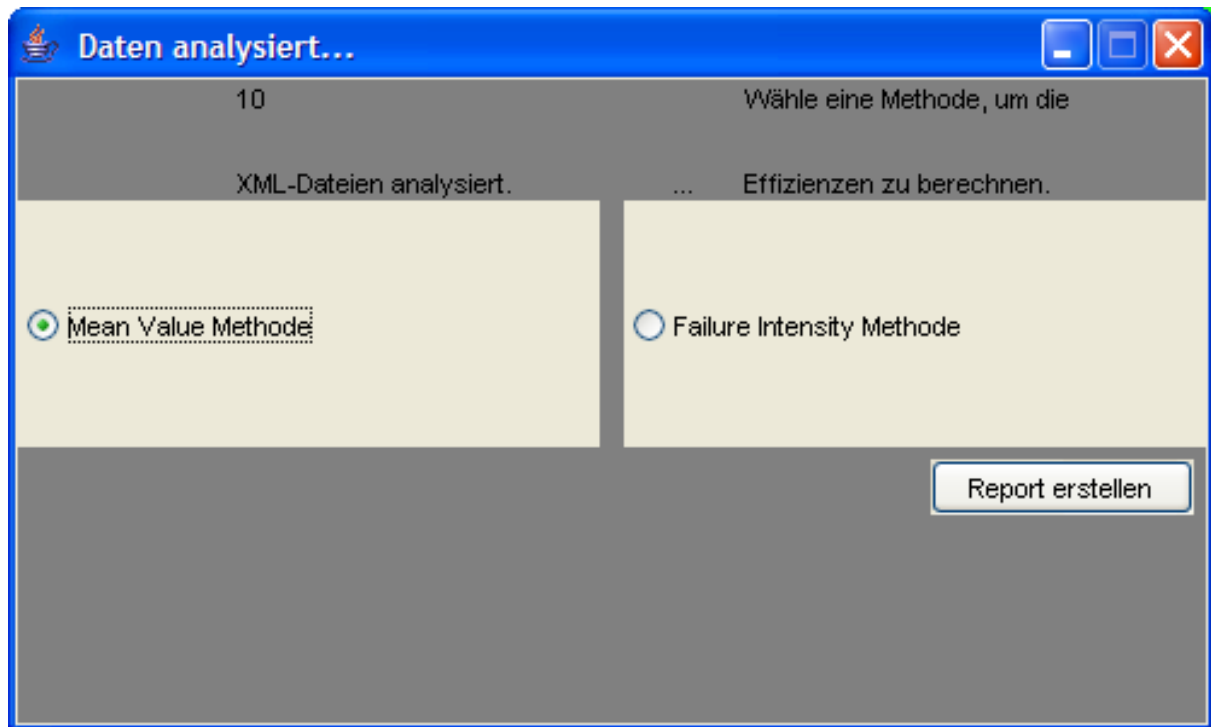


Abbildung 10: Zweites Fenster des Analysetools

Hier kann man sich zwischen den zwei Methoden (mean-value Methode, failure-intensity Methode) entscheiden, wie man die Effizienzen berechnet. Als nächstes wird durch das Anklicken des Buttons "Report erstellen" das Reportfenster (siehe Abbildung 11) aufgerufen.

	QS-Namen	Absolut Effizienz	Relativ Effizienz	Plotten?
Technik1	BB	151.21513	0.6480649	<input checked="" type="checkbox"/>
Technik2	BB	226.82268	1.1341134	<input checked="" type="checkbox"/>
Technik3	Inspection	3332.2314	12.816275	<input checked="" type="checkbox"/>
Technik4	BB	534.6104	2.3208401	<input checked="" type="checkbox"/>
Technik5	BB	805.56067	3.497084	<input checked="" type="checkbox"/>
Technik6	Inspection	403.19998	1.5507692	<input checked="" type="checkbox"/>
Technik7	Inspection	-1007.99994	-3.3600001	<input checked="" type="checkbox"/>
Technik8	WhiteBox	-369.60004	-1.1088002	<input checked="" type="checkbox"/>

Abbildung 11: Drittes Fenster des Analysetools

Als letztes klickt man auf das Button "Plotten", damit das Chartfenster in den Vordergrund tritt.

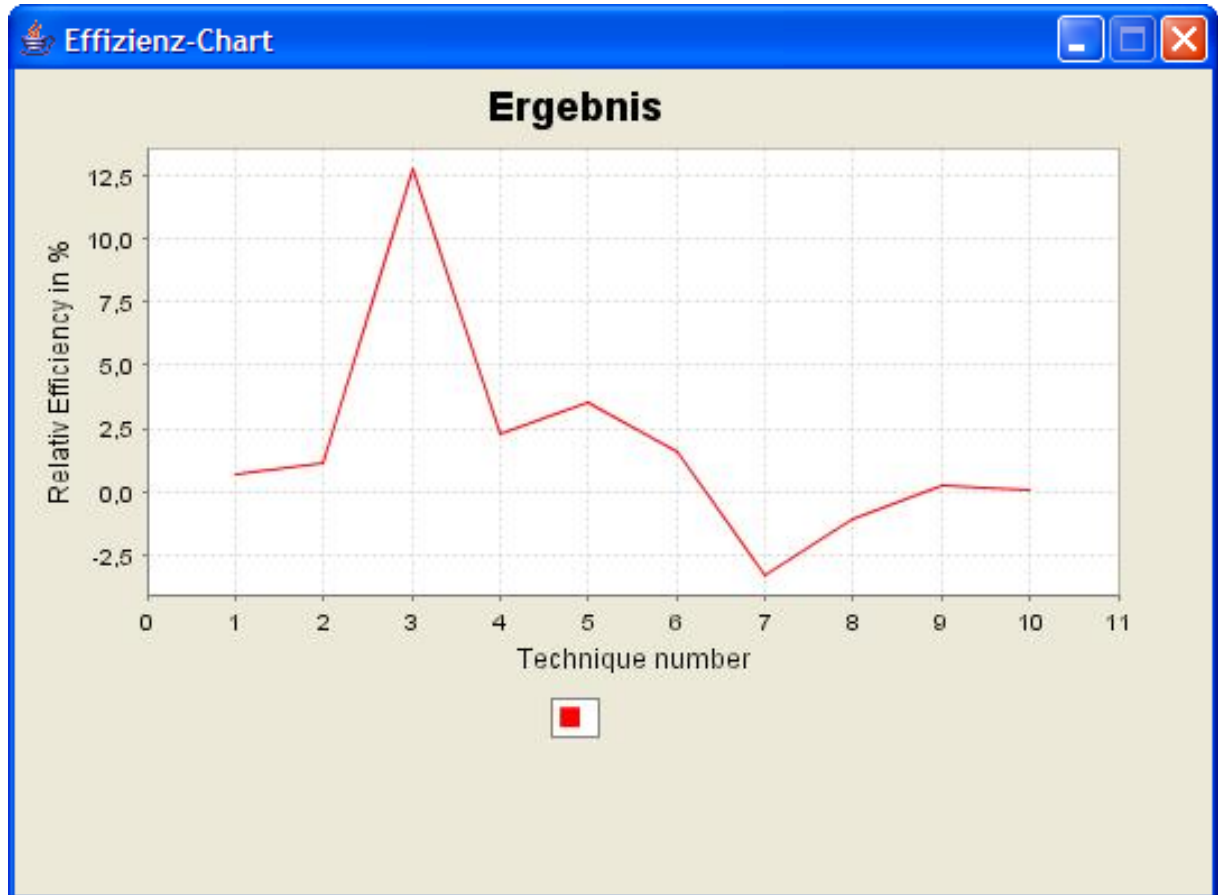


Abbildung 12: Viertes Fenster des Analysetools.

5 Zusammenfassung und Ausblick

In der Softwareentwicklung wird eine riesige Menge von Testdaten und Testprotokollen mit-erzeugt, doch diese Daten sind nicht gleichzusetzen mit neuen nutzbaren Informationen.

In dieser Ausarbeitung wird den Lesern ein Überblick darüber gegeben, was die QS-Maßnahmen der heutigen Softwareentwicklung sind, wie man aus der Menge von Testdaten nutzbare In-formationen ziehen kann, und wie man mit verschiedenen Methoden diese Informationen analysieren und auch die Ergebnisse präsentieren kann.

Das Hauptziel dieser Arbeit liegt darin, die Qualität der QS-Maßnahmen mathematisch zu messen. Sie hat uns auch gezeigt, dass es möglich ist, die Effizienz der QS-Maßnahmen mit echten Projektdaten zu berechnen. Leider gibt es momentan nicht genug echte Daten, beson-ders die Dauer der QS-Maßnahmen fehlt uns.

In der Zukunft hat man noch die Möglichkeit, die Analyse auszubauen, die aus dieser Arbeit resultierten.

Man könnte z.B. noch andere Methode wie die reliability-model Methode (wird in [5] und [6] beschrieben) zur Berechnung der Effizienzen heranziehen. Diese Methode ist völlig anders als die anderen zwei Methoden, weil diese alle Ausfälle vor der jeweiligen QS-Maßnahme, und nicht nur die Ausfälle nach der vorletzten QS-Maßnahme, braucht.

Außerdem bei der Ergebnispräsentation darf man in der Zukunft die berechneten Effizienzen nach bestimmter Forderung grafisch darstellen. Z.B. nur die QS-Maßnahmen gleicher Art sind zu plotten.

Literatur

- [1] Din 55350. Teil 11, August 1995.
- [2] M. Halstead. *Elements of Software Science*. Elsevier computer science library edition, 1977.
- [3] T. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, December 1976.
- [4] B. Rumpe. Vorlesungsscript softwaretechnik. Oktober 2002.
- [5] S. Wagner. Efficiency analysis of defect-detection techniques. Technical report, TU München, 2004.
- [6] S. Wagner. Reliability efficiency of defect-detection techniques: A field study. *Inteational Symposium on Software Reliability Engineering, (ISSRE'04)*, 2004.
- [7] McCabe T. Watson, A. Structured testing: A testing methodology using the cyclomatic complexity metric. *Computer Systems Laboratory National Institute of Standards and Technology*, September 1996.