

Ein Kosten-Nutzen-Modell für die Softwareprüfung

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik
der Universität Stuttgart zur Erlangung der Würde eines Doktors der
Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Tilman Hampp

aus Ludwigsburg

Hauptberichter: Prof. Dr. rer.nat. J. Ludewig

Mitberichter: O. Univ.-Prof. Dr. R. Mittermeir

Tag der mündlichen Prüfung: 19. Juli 2010

Institut für Softwaretechnologie der Universität Stuttgart

2010

Berichte aus der Softwaretechnik

Tilman Hampp

**Ein Kosten-Nutzen-Modell
für die Softwareprüfung**

D 93 (Diss. Universität Stuttgart)

Shaker Verlag
Aachen 2010

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Zagl.: Stuttgart, Univ., Diss., 2010

Copyright Shaker Verlag 2010

Alle Rechte, auch das des auszugsweisen Nachdruckes, der auszugsweisen oder vollständigen Wiedergabe, der Speicherung in Datenverarbeitungsanlagen und der Übersetzung, vorbehalten.

Printed in Germany.

ISBN 978-3-8322-9567-7

ISSN 1433-9986

Shaker Verlag GmbH • Postfach 101818 • 52018 Aachen

Telefon: 02407 / 95 96 - 0 • Telefax: 02407 / 95 96 - 9

Internet: www.shaker.de • E-Mail: info@shaker.de

Danksagung

Zu dieser Arbeit haben viele Personen beigetragen, denen ich an dieser Stelle danken möchte. Mein ganz besonderer Dank gilt meinem Doktorvater Prof. Jochen Ludewig, der diese Arbeit ermöglicht hat. Seine Sicht auf die Software-Qualitätssicherung, seine Anregungen und seine Kritik haben diese Arbeit geprägt. Mein besonderer Dank geht auch an Prof. Roland Mittermeir für sein Interesse an der Arbeit, die freundliche Übernahme des Zweitgutachtens und die wertvollen Hinweise. Prof. Erhard Plödereder danke ich für seinen Einsatz als Zweitprüfer.

Herzlich danken möchte ich auch allen Kollegen und Kolleginnen der Abteilung Software Engineering für die angenehme Zusammenarbeit, die ermutigenden Gespräche und die anregenden Diskussionen.

Die Partner in der Industrie haben mir in den Gesprächen und mit den Projektdaten einen tiefen Einblick in ihre Projekte gewährt. Vielen Dank! Ebenso danke ich den Teilnehmern und Hilfskräften des Software-Praktikums 2007 für die Datenerhebung und die Unterstützung bei der Analyse. Mein Dank geht auch an die Studenten, die durch ihre Diplom- und Studienarbeiten und Fachstudien zu dieser Arbeit beigetragen haben.

Nicht zuletzt danke ich allen, die mich während der Arbeit begleitet haben, für ihr Vertrauen, ihre Geduld und ihre Unterstützung.

Zusammenfassung

Prüfungen können große Teile des Budgets eines Software-Projekts aufzehren, erlauben aber, die Produktqualität zu beurteilen und zu verbessern. Sie dürfen nicht vernachlässigt werden, da Defizite der Produktqualität nach Projektende teuer werden können. Projektleiter und Verantwortliche für die Qualität müssen bereits in der Planung über Prüfungen entscheiden. Sie sind in einer schwierigen Situation, weil sie dabei viele komplexe und langfristig wirkende Entscheidungen über Prüfungen und über einzelne Parameter der Prüfungen treffen müssen. Die Kosten der Prüfungen sind früh sichtbar und messbar. Im Gegensatz dazu wird der Nutzen durch schwierig zu messende Qualitätsverbesserungen erreicht, die zu langfristigen Einsparungen führen. Zusätzlich hängen Kosten und Nutzen von der Projektsituation ab. Für jedes Projekt ist darum ein individueller Kompromiss zwischen den Kosten für Prüfungen und ihrem Nutzen nötig, so dass minimale Gesamtkosten erreicht werden.

Um diese Entscheidungen zu unterstützen, wird in dieser Arbeit ein Kosten-Nutzen-Modell für Softwareprüfungen, CoBe, entwickelt und validiert. Mit diesem Modell kann untersucht und prognostiziert werden, wie sich Entscheidungen über Prüfungen und über einzelne Parameter der Prüfungen auswirken. Dazu werden die Entscheidungen und die Projektsituation durch Modelleingaben dargestellt. Die Modellresultate sind die Wirkungen dieser Entscheidungen: die Kosten, die durch die Prüfung entstehen, und der daraufhin erreichte Nutzen durch eingesparte Kosten. Kosten und Nutzen zeigen sich während des Projekts, während der Wartung des Produkts und beim Einsatz des Produkts. Damit Kosten und Nutzen abgewogen und Gesamtkosten minimiert werden können, werden die Modellresultate als Geldwerte berechnet. Zur Projektplanung werden Kosten und Nutzen durch Aufwand, Dauer und Personalbedarf einzelner Aktivitäten dargestellt. Dazu enthält CoBe feingranulare Prüfungsmodelle aus einzelnen, quantitativen Wirkungszusammenhängen.

Die Validierung des Modells erfolgte mit Daten aus Software-Projekten. Dabei wurden einzelne Zusammenhänge und das gesamte Modell mit Daten aus über 20 studentischen Projekten geprüft. CoBe ist mit Daten aus zwei iterativen Industrieprojekten mit umfangreicher, paralleler Entwicklung validiert. Das Modellverhalten wird durch Sensitivitätsanalyse untersucht, zusätzlich wird das Kosten-Optimum analysiert. Die Validierung zeigt, dass CoBe ausreichend genau beschreibt, wie sich Entscheidungen über Prüfungen auswirken. Da die Resultate der studentischen Projekte deutlich streuen, ergibt sich eine gewisse Abweichung zwischen den Projektergebnissen und den Modellresultaten. Die Resultate sind für die beiden Industrieprojekte genauer. Deutlich wird, dass CoBe für eine bestimmte Umgebung kalibriert werden muss, damit die Resultate ausreichend genau sind. Dazu sind wenige Daten aus abgeschlossenen Software-Projekten notwendig. Die Daten sind oft verfügbar, da sie häufiger als andere Daten erhoben werden. Die Validierung zeigt, dass CoBe gut verallgemeinerbar ist. Die Daten, die für den Einsatz von CoBe notwendig sind, sind in Projekten verfügbar, können gemessen oder erfragt werden.

Abstract

Software quality assurance can consume large parts of a software project's budget. On the other hand, quality assurance permits product quality to be assessed and improved. Cutting quality assurance investments may lead to increased costs after delivery.

Project managers and quality managers have to decide on quality assurance while planning and running a project. They have to make many complex and far-reaching decisions on reviews, tests, and their parameters without having the necessary information available. In particular, quality-related information is hard to get because quality improvements appear as long-term savings, whereas the costs of reviews and tests can be measured early on. As every project has its own special characteristics, a tailored trade-off between costs and benefits of quality-assurance activities is needed to minimise total costs.

In this work CoBe, a quantitative cost-benefit model to support these decisions, is developed and validated. CoBe is able to analyse and predict the effects that decisions on quality assurance activities and on their parameters will have. To do so, decisions on quality assurance and project characteristics are modelled as inputs. The output of the model consists of the costs and benefits resulting from these decisions. Costs and benefits occur during the project, in maintenance, and during product usage. For the purpose of comparing costs and benefits and minimising overall costs, the model results are expressed in monetary values. For project planning, costs and benefits are expressed as effort, duration, and staff of single activities. For calculating the results, CoBe uses fine-grained models built on single quantitative relationships.

CoBe was validated against real-world software project data. Single relationships and the entire model were examined using data from more than 20 student projects. CoBe was validated using data from two iterative industry projects that used extensive parallel development. Model behaviour is subjected to sensitivity analysis, and the optimum cost is analysed. The validation shows that CoBe describes the effects of decisions on quality assurance sufficiently accurate. As student projects scatter, model results differ from project results up to a certain extent. Results are more accurate for industry projects. It is evident that CoBe needs to be calibrated for a certain environment. However, only few data from past projects is required for this. The necessary data is readily available in most projects. Results indicate that CoBe is generalisable, and that the necessary data is either available in the projects or can be measured or obtained by enquiry.

1	Einleitung und Überblick	13
1.1	Motivation	13
1.2	Lösungsansatz	15
1.3	Überblick	15
2	Grundlagen und Begriffe	17
2.1	Modelle	17
2.2	Deduktive und induktive Modelle	18
2.3	Metriken	19
2.3.1	Skalen und Skalentypen	19
2.3.2	Merkmale von Metriken	20
2.4	Entscheidungen und Entscheidungstheorie	21
2.4.1	Modelle in der Entscheidungstheorie	21
2.4.2	Nutzen und Grenzen	22
2.5	Kosten und Nutzen	23
2.5.1	Kosten- und Nutzenbegriffe	23
2.5.2	Kosten-Nutzen-Analyse und andere Verfahren	23
2.6	Software-Projekte	25
2.6.1	Prozess	25
2.6.2	Rollen in Software-Projekten	26
2.6.3	Projektleitung	27
2.6.4	Kosten in Software-Projekten	27
2.6.5	Kostenschätzung in Software-Projekten	28
2.7	Qualität und Software-Qualität	29
2.7.1	Der Qualitätsbegriff	29
2.7.2	Software-Qualität	29
2.8	Software-Qualitätssicherung	31
2.9	Qualitätskosten und Software-Qualitätskosten	32
3	Die Idee eines Kosten-Nutzen-Modells für Prüfungen	33
3.1	Schwierigkeiten mit Entscheidungen über Qualitätssicherung	33
3.2	Ziele des Kosten-Nutzen-Modells für Prüfungen	36
3.3	Unterstützte Entscheidungen	37
3.3.1	Auswahl der Qualitätssicherungsmaßnahmen	37
3.3.2	Entscheidungen über Prüfungen und Prüfparameter	38
3.4	Kosten und Nutzen von Prüfungen	39
3.4.1	Modellierte Kosten und modellierter Nutzen	40

3.4.2	Darstellung der Kosten und des Nutzens	42
3.4.3	Abstraktionsebene der Kosten und des Nutzens	42
3.5	Prozess- und Produktmerkmale	42
3.6	Modelleinsatz	43
3.6.1	Modellkalibrierung und -anpassung an Projekte und Prozesse	43
3.6.2	Planung und Kontrolle	45
3.7	Modellierungsansatz	47
3.8	Zusammenfassung	49
3.8.1	Prüfungen und Prüfparameter in CoBe	49
3.8.2	Kosten und Nutzen in CoBe	50
4	Verwandte Arbeiten	51
4.1	Projektsimulation mit SESAM und dem QS-Modell	51
4.1.1	Das Qualitätssicherungs-Modell	53
4.1.2	Einsatz und Anwendung	55
4.1.3	Andere Modelle in SESAM	55
4.1.4	Bewertung und Folgerungen	56
4.2	Kostenschätzung mit COCOMO II	57
4.2.1	Zusammenhänge in COCOMO II	57
4.2.2	Kalibrierung und Validierung	58
4.2.3	Bewertung und Folgerungen	58
4.3	Das Datenarchiv und die Analysen von Jones	59
4.4	Weitere Kosten-Nutzen-Modelle	60
4.4.1	COCOMO-Erweiterungen	60
4.4.2	El Emams Return-On-Investment-Modell	62
4.4.3	Wagners Modelle zur Kosten-Nutzen-Optimierung	62
4.4.4	Müllers Produktlinienmodell	63
4.4.5	Prozesssimulation von Raffo et al.	64
4.5	Bewertungen und Folgerungen	65
5	Analyse der Kosten und des Nutzens von Prüfungen	67
5.1	Begriffe	67
5.2	Analyse der Fehlerentstehung, -entdeckung und -korrektur	70
5.3	Analyse von Fehlerkosten	72
5.3.1	Fehlerbehebungskosten	72
5.3.2	Fehlerfolgekosten und Zuverlässigkeit	74
5.3.3	Organisationsaufwand	75
5.3.4	Aufwand, Dauer und Personalbedarf	75
5.3.5	Geldwerte	76

5.4	Analyse von Reviews	77
5.5	Analyse von Tests	79
5.6	Analyse automatischer statischer Codeanalyse	85
6	Ein quantitatives Modell für Prüfungen: CoBe	87
6.1	Überblick über das Modell CoBe	87
6.2	Die Architektur von CoBe	88
6.2.1	Die Modellkomponenten von CoBe	88
6.2.2	Ein Beispiel zur Illustration von CoBe	91
6.2.3	Die Kalibrierungsparameter von CoBe	93
6.3	Das Basismodell mit den grundlegenden Zusammenhängen	94
6.3.1	Das Umfangsmodell von CoBe	94
6.3.2	Der Fehlerbegriff und das Fehlermodell in CoBe	96
6.3.3	Modellierung der Fehlerentstehung in CoBe	97
6.3.4	Modellierung der Fehlerentdeckung und Fehlerkorrektur in CoBe	99
6.3.5	Das Modell für den Korrekturaufwand in CoBe	104
6.3.6	Das Modell für die Kosten der Prüfwiederholung in CoBe	105
6.3.7	Das Modell für den Aufwandseinfluss in CoBe	109
6.3.8	Das Modell für Dauer und Personal in CoBe	110
6.3.9	Das Geldwerte-Modell von CoBe	112
6.3.10	Das Fehlerfolgekostenmodell von CoBe	112
6.4	Reviews im Modell	116
6.4.1	Eingaben für Reviews	116
6.4.2	Zusammenhänge im Reviewmodell	117
6.5	Automatische statische Codeanalyse	121
6.6	Tests im Modell	121
6.6.1	Eingaben von Tests	121
6.6.2	Zusammenhänge im Testmodell	122
6.7	Zusammenfassung	129
6.7.1	Zusammenhänge im Überblick	129
6.7.2	Vorgehen zur Kalibrierung	133
6.8	Quantifizierung	134
6.8.1	Basismodell	135
6.8.2	Reviews	138
6.8.3	Codeanalyse	141
6.8.4	Tests	141
7	Modellrealisierung, Modellprüfung und Modellverbesserung	145
7.1	Die Realisierung von CoBe	145

7.1.1	Vorgehen zur Realisierung des Modells	145
7.1.2	Realisierung als Tabellenkalkulation	147
7.1.3	Realisierung als Java-Anwendung	149
7.2	Überblick über die Modellprüfung	151
7.2.1	Verifikation und Validierung für quantitative Modelle	152
7.2.2	Schritte der Modellprüfung	154
7.2.3	Kriterien für die Modellprüfung	155
7.3	Studentische Projekte für die Modellvalidierung	158
7.3.1	Ablauf, Datenerhebung und Datenvalidierung	159
7.3.2	Interne und externe Validität	160
7.4	Prüfung ausgewählter Modellzusammenhänge	162
7.4.1	Fehlerentstehung, Fehlerentdeckung, Korrekturaufwand	164
7.4.2	Testfälle, Code-Überdeckung, Fehlerentdeckungsquote	168
7.4.3	Fehlerfolgekosten	181
7.4.4	Folgerungen	184
7.5	Erprobung im Software-Praktikum	185
7.5.1	Vergleich mit Mittelwerten und Kalibrierung	185
7.5.2	Modellverbesserungen	186
7.5.3	Modellresultate und Mittelwerte des Praktikums	187
7.6	Vergleich mit einzelnen Projekten des Software-Praktikums	191
7.6.1	Diagnose einzelner Projekte	192
7.6.2	Kreuzvalidierung als Ersatz für die Prognose	197
7.6.3	Bewertung und Folgerungen	198
8	Evaluation des Modells	203
8.1	Sensitivitätsanalyse	203
8.1.1	Ziele und Hypothesen der Sensitivitätsanalyse	203
8.1.2	Vorgehen zur Sensitivitätsanalyse	204
8.1.3	Szenarien für die Sensitivitätsanalyse	205
8.1.4	Eingaben für die Sensitivitätsanalyse	207
8.1.5	Statistische Sensitivitätsanalyse	207
8.1.6	Graphische Sensitivitätsanalyse	213
8.1.7	Sensitivitätsanalyse für unsichere Eingaben	217
8.1.8	Analyse der Auswirkungen unsicherer Eingaben	219
8.1.9	Zusammenfassung der Sensitivitätsanalyse	220
8.2	Analyse des Optimums mit CoBe	222
8.2.1	Vorgehen	222
8.2.2	Resultate	223
8.3	Vorgehen für die Validierung in der Industrie	227

8.4	Industrieprojekt 1	229
8.4.1	Das Projekt und sein Prozess	230
8.4.2	Die Abbildung in das Modell	232
8.4.3	Modelleingaben im Detail	235
8.4.4	Kalibrierung des Modells	237
8.4.5	Vergleich der Modellresultate mit Istwerten	238
8.4.6	Resultate der Referenzmodelle	242
8.5	Industrieprojekt 2	244
8.5.1	Das Projekt und sein Prozess	244
8.5.2	Die Abbildung in das Modell	247
8.5.3	Modelleingaben im Detail	249
8.5.4	Vergleich der Modellresultate mit Istwerten	251
8.5.5	Resultate des Referenzmodells	255
8.6	Bewertung der Validierung	256
8.6.1	Resultate der Validierung	256
8.6.2	Gültigkeit der Validierungsergebnisse	257
8.6.3	Folgerungen	260
8.7	Modellverhalten und Modelleinsatz	261
8.7.1	Ein fiktives Beispielprojekt	263
8.7.2	Kosten und Nutzen des Entwurfsreviews	264
8.7.3	Langfristiger Nutzen des Entwurfsreviews	265
8.7.4	Prozessverbesserung durch Reviews	268
8.7.5	Prozessverbesserung durch Testautomatisierung	270
8.7.6	Codereview durchführen oder Modultest verbessern	270
9	Zusammenfassung und Bewertung	273
9.1	Zusammenfassung	273
9.1.1	Validierung	274
9.1.2	Aussagen des Modells	275
9.2	Bewertung	275
9.2.1	Modellziele	275
9.2.2	Verallgemeinerbarkeit	276
9.2.3	Kosten und Nutzen des Modelleinsatzes	276
9.2.4	Abgrenzung zu SESAM und zum QS-Modell	277
9.2.5	Grenzen	277
9.3	Ausblick	278
9.4	Schlussbemerkungen	278
	Verzeichnis der Bezeichner	281
	Literatur	285

