

Rainer Typke

Nützlichkeit von Zusicherungen als Hilfsmittel beim Programmieren

Ein kontrolliertes Experiment

Diplomarbeit

Bibliografische Information der Deutschen Nationalbibliothek:

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

Dieses Werk sowie alle darin enthaltenen einzelnen Beiträge und Abbildungen sind urheberrechtlich geschützt. Jede Verwertung, die nicht ausdrücklich vom Urheberrechtsschutz zugelassen ist, bedarf der vorherigen Zustimmung des Verlags. Das gilt insbesondere für Vervielfältigungen, Bearbeitungen, Übersetzungen, Mikroverfilmungen, Auswertungen durch Datenbanken und für die Einspeicherung und Verarbeitung in elektronische Systeme. Alle Rechte, auch die des auszugsweisen Nachdrucks, der fotomechanischen Wiedergabe (einschließlich Mikrokopie) sowie der Auswertung durch Datenbanken oder ähnliche Einrichtungen, vorbehalten.

Copyright © 1999 Diplom.de
ISBN: 9783832423957

Rainer Typke

Nützlichkeit von Zusicherungen als Hilfsmittel beim Programmieren

Ein kontrolliertes Experiment

Rainer Typke

Nützlichkeit von Zusicherungen als Hilfsmittel beim Programmieren

Ein kontrolliertes Experiment

Diplomarbeit

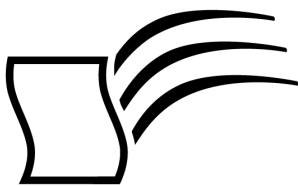
an der Universität Fridericiana Karlsruhe (TH)

Fachbereich Informatik

Prüfer Prof. W. Tichy

Institut für Programmstrukturen und Datenorganisation,

April 1999 Abgabe



Diplomarbeiten Agentur

Dipl. Kfm. Dipl. Hdl. Björn Bedey

Dipl. Wi.-Ing. Martin Haschke

und Guido Meyer GbR

Hermannstal 119 k

22119 Hamburg

agentur@diplom.de

www.diplom.de

ID 2395

Typke, Rainer: Nützlichkeit von Zusicherungen als Hilfsmittel beim Programmieren: Ein kontrolliertes Experiment / Rainer Typke · Hamburg: Diplomarbeiten Agentur, 2000
Zugl.: Karlsruhe, Technische Universität, Diplom, 1999

Dieses Werk ist urheberrechtlich geschützt. Die dadurch begründeten Rechte, insbesondere die der Übersetzung, des Nachdrucks, des Vortrags, der Entnahme von Abbildungen und Tabellen, der Funksendung, der Mikroverfilmung oder der Vervielfältigung auf anderen Wegen und der Speicherung in Datenverarbeitungsanlagen, bleiben, auch bei nur auszugsweiser Verwertung, vorbehalten. Eine Vervielfältigung dieses Werkes oder von Teilen dieses Werkes ist auch im Einzelfall nur in den Grenzen der gesetzlichen Bestimmungen des Urheberrechtsgesetzes der Bundesrepublik Deutschland in der jeweils geltenden Fassung zulässig. Sie ist grundsätzlich vergütungspflichtig. Zuwiderhandlungen unterliegen den Strafbestimmungen des Urheberrechtes.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, daß solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Die Informationen in diesem Werk wurden mit Sorgfalt erarbeitet. Dennoch können Fehler nicht vollständig ausgeschlossen werden, und die Diplomarbeiten Agentur, die Autoren oder Übersetzer übernehmen keine juristische Verantwortung oder irgendeine Haftung für evtl. verbliebene fehlerhafte Angaben und deren Folgen.

Dipl. Kfm. Dipl. Hdl. Björn Bedey, Dipl. Wi.-Ing. Martin Haschke & Guido Meyer GbR
Diplomarbeiten Agentur, <http://www.diplom.de>, Hamburg 2000
Printed in Germany



Diplomarbeiten Agentur

Wissensquellen gewinnbringend nutzen

Qualität, Praxisrelevanz und Aktualität zeichnen unsere Studien aus. Wir bieten Ihnen im Auftrag unserer Autorinnen und Autoren Wirtschaftsstudien und wissenschaftliche Abschlussarbeiten – Dissertationen, Diplomarbeiten, Masterarbeiten, Staatsexamensarbeiten und Studienarbeiten zum Kauf. Sie wurden an deutschen Universitäten, Fachhochschulen, Akademien oder vergleichbaren Institutionen der Europäischen Union geschrieben. Der Notendurchschnitt liegt bei 1,5.

Wettbewerbsvorteile verschaffen – Vergleichen Sie den Preis unserer Studien mit den Honoraren externer Berater. Um dieses Wissen selbst zusammenzutragen, müssten Sie viel Zeit und Geld aufbringen.

<http://www.diplom.de> bietet Ihnen unser vollständiges Lieferprogramm mit mehreren tausend Studien im Internet. Neben dem Online-Katalog und der Online-Suchmaschine für Ihre Recherche steht Ihnen auch eine Online-Bestellfunktion zur Verfügung. Inhaltliche Zusammenfassungen und Inhaltsverzeichnisse zu jeder Studie sind im Internet einsehbar.

Individueller Service – Gerne senden wir Ihnen auch unseren Papierkatalog zu. Bitte fordern Sie Ihr individuelles Exemplar bei uns an. Für Fragen, Anregungen und individuelle Anfragen stehen wir Ihnen gerne zur Verfügung. Wir freuen uns auf eine gute Zusammenarbeit

Ihr Team der Diplomarbeiten Agentur

Dipl. Kfm. Dipl. Hdl. Björn Bedey —
Dipl. Wi.-Ing. Martin Haschke —
und Guido Meyer GbR —

Hermannstal 119 k —
22119 Hamburg —

Fon: 040 / 655 99 20 —
Fax: 040 / 655 99 222 —

agentur@diplom.de —
www.diplom.de —

Inhaltsverzeichnis

1	Einleitung	6
1.1	Zusicherungen	6
1.2	Grundidee des Experiments	8
1.3	Verwandte Arbeiten	8
1.3.1	Störk: jContract	8
1.3.2	Leveson, Cha et al.: empirische Studie	9
1.3.3	Schneider: Concurrent Programming	9
1.3.4	Luckham et al.: Two-dimensional Pinpointing	10
1.3.5	McKim: Designing for correctness	10
1.4	Nützlichkeit eines Experiments	10
1.5	Gliederung der Ausarbeitung, Rohdaten	10
2	Die verwendeten Zusicherungswerkzeuge	12
2.1	APP	12
2.2	jContract	16
3	Beschreibung des Experiments	19
3.1	Fragestellung und Hypothesen	19
3.2	Aufbau des Experiments	20
3.2.1	Versuchspersonen	21
3.2.2	Klassifizierung und Vorsortierung der Versuchspersonen	21
3.2.3	Auswahl der Aufgaben	23
3.2.4	Teilaufgabe mit Neuentwicklungscharakter	26
3.2.5	Teilaufgabe mit Wartungscharakter	26
3.2.6	Reihenfolge der Aufgaben, Ablauf des Experiments	27
3.2.7	Unterschiede in den C- und Java-Aufgaben	28
3.2.8	Training der Teilnehmer vor dem Experiment	28
3.3	Bedrohungen der internen Gültigkeit	29
3.4	Bedrohungen der externen Gültigkeit	32
3.5	Technischer Versuchsaufbau	32
3.5.1	Kooperationsmöglichkeiten	32
3.5.2	Syntaxkurs und Skripte zum Übersetzen und Testen	33
3.5.3	Erfasste Daten	35
4	Ergebnisse	37
4.1	Signifikanztest und Diagramme	37
4.1.1	Signifikanztest	37
4.1.2	Der Boxplot	37
4.1.3	Trendgeraden	38
4.2	Schwächen von jContract	38
4.3	Bereitschaft zur Verwendung von Zusicherungen	39
4.4	Zeitbedarf	40
4.4.1	Zur Ermittlung der Zeitdaten	40
4.4.2	Zeitbedarf mit und ohne Zusicherungen	42
4.4.3	Rückschlüsse aus dem Zeitbedarf	48

4.4.4	Zusammenhang zwischen Anzahl der Zusicherungen und Zeitbedarf	48
4.5	Wiederverwendung von Funktionen	51
5	Zusammenfassung und Ausblick	55
A	Anhang	57
A.1	Daten	57
A.1.1	Gruppeneinteilung	57
A.1.2	Daten aus den Protokollen	58
A.1.3	Daten des interaktiven Kurses	59
A.1.4	Zusicherungen, Übersetzungsläufe	60
A.1.5	Wiederverwendung	60
A.2	Das Syntax-Lernprogramm	64
A.2.1	Lernprogramm für APP	64
A.2.2	Lernprogramm für jContract	78
A.3	Die Aufgaben	93
A.3.1	Beispiel: C, String mit APP / Kettenregel ohne APP	93
A.3.2	Beispiel: Java, Kettenregel mit / Menge ohne jContract	99
	Literaturverzeichnis	104

Abbildungsverzeichnis

1	Beispiel für eine Ausgabe des Testprogramms.	25
2	APP-Syntaxkurs: Beispiel für eine Aufgabe	29
3	APP-Syntaxkurs: Beispiel für eine Benutzereingabe	30
4	APP-Syntaxkurs: Kommentare des Lernprogramms zu den Eingaben von Bild 3	31
5	Vorbedingungen	39
6	Nachbedingungen	39
7	Zeitbedarf für die Kettenregel mit / ohne Zusicherungen (C).	42
8	Zeitbedarf für die Kettenregel mit / ohne Zusicherungen (Java).	42
9	Zeitbedarf für die String-Aufgabe mit / ohne Zusicherungen (C).	43
10	Zeitbedarf für die Mengen-Aufgabe mit / ohne Zusicherungen (Java).	43
11	Korrelation der mit dem Syntaxkurs verbrachten Zeit mit der Programmierzeit.	45
12	Relativer Zeitbedarf für die Kettenregel mit / ohne Zusicherungen (C).	45
13	Relativer Zeitbedarf für die Kettenregel mit / ohne Zusicherungen (Java).	46
14	Relativer Zeitbedarf für die String-Aufgabe mit / ohne Zusicherungen (C).	47
15	Relativer Zeitbedarf für die Mengen-Aufgabe mit / ohne Zusicherungen (Java).	47
16	Korrelation der Anzahl hinzugefügter Vor- und Nachbedingungen mit der Programmierzeit.	48
17	Anzahl hinzugefügter Vor- und Nachbedingungen und relative Programmierzeit	50

18	Anzahl hinzugefügter Vor- und Nachbedingungen und Dauer des Syntaxkurses	50
19	Wiederverwendete Funktionen für die Kettenregel, C / APP	53
20	Wiederverwendete Funktionen für die Kettenregel, Java / jContract . . .	53
21	APP-Lernprogramm: Erste Aufgabe	65
22	APP-Lernprogramm: Benutzereingaben zur ersten Aufgabe	66
23	APP-Lernprogramm: Auswertung der Eingaben zur ersten Aufgabe . .	67
24	APP-Lernprogramm: Zweite Aufgabe	68
25	APP-Lernprogramm: Auswertung der zweiten Aufgabe	69
26	APP-Lernprogramm: Dritte Aufgabe	70
27	APP-Lernprogramm: Eingaben zur dritten Aufgabe	71
28	APP-Lernprogramm: Auswertung der Eingaben zur dritten Aufgabe, erster Teil	72
29	APP-Lernprogramm: Auswertung der Eingaben zur dritten Aufgabe, zweiter Teil	73
30	APP-Lernprogramm: Vierte Aufgabe	74
31	APP-Lernprogramm: Auswertung der Eingaben zur vierten Aufgabe . .	75
32	APP-Lernprogramm: Fünfte Aufgabe	76
33	APP-Lernprogramm: Auswertung der Eingaben zur fünften Aufgabe . .	77
34	jContract-Lernprogramm: Erste Aufgabe	79
35	jContract-Lernprogramm: Eingaben für die erste Aufgabe	80
36	jContract-Lernprogramm: Auswertung der Eingaben aus Bild 35	81
37	jContract-Lernprogramm: Zweite Aufgabe mit Eingaben	82
38	jContract-Lernprogramm: Auswertung der zweiten Aufgabe	83
39	jContract-Lernprogramm: Dritte Aufgabe	84
40	jContract-Lernprogramm: Eingaben zur 3. Aufgabe	85
41	jContract-Lernprogramm: Auswertung der Eingaben zur 3. Aufgabe, erster Teil	86
42	jContract-Lernprogramm: Auswertung der Eingaben zur 3. Aufgabe, zweiter Teil	87
43	jContract-Lernprogramm: Vierte Aufgabe (@invariant)	88
44	jContract-Lernprogramm: Auswertung der vierten Aufgabe	89
45	jContract-Lernprogramm: Fünfte Aufgabe (Quantoren)	90
46	jContract-Lernprogramm: Auswertung der fünften Aufgabe	91
47	jContract-Lernprogramm: Mitteilung über Mißerfolg	92

Tabellenverzeichnis

1	Vorsortierung der Teilnehmer für den APP-Teil	22
2	Vorsortierung der Teilnehmer für den jContract-Teil	23
3	Bearbeitungszeiten	41
4	Quartil-Verhältnisse beim Zeitbedarf	43
5	Wiederverwendung von Funktionen	52
6	Gruppenzugehörigkeiten der Teilnehmer	57
7	Manuell eingetragene Daten	58
8	Kursdaten	59

9	Zusicherungen, Zusicherungsverletzungen und Übersetzungsläufe . . .	61
10	Aufrufe wiederverwendeter Funktionen in C-Programmen	62
11	Aufrufe wiederverwendeter Funktionen	62

Hiermit versichere ich, die vorliegende Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet zu haben.

Karlsruhe, den 29. April 1999

Zusammenfassung

Im Rahmen dieser Diplomarbeit wurde ein kontrolliertes Experiment durchgeführt, mit dem die Nützlichkeit von Zusicherungen als Hilfsmittel beim Programmieren untersucht wurde. Zusicherungen sind Bedingungen, die mitten im Programmtext, als Vor- oder Nachbedingung einer Funktion zugeordnet, oder als Invariante auf eine Klasse bezogen auftreten können. Ihre Einhaltung kann während der Laufzeit überwacht werden. In diesem Experiment wurden APP für C und jContract für Java als Zusicherungswerkzeuge eingesetzt. Das Ergebnis legt nahe, daß der Zeitbedarf beim Programmieren durch die Verwendung von Zusicherungen nicht steigt, aber die Qualität der entstehenden Software.

1 Einleitung

1.1 Zusicherungen

Ein bekanntes Problem bei der Softwareentwicklung ist der Konflikt zwischen den Zielen, einerseits möglichst korrekte Programme zu schreiben und andererseits den Aufwand in einem vernünftigen finanziellen und zeitlichen Rahmen zu halten. Es gibt viele Arbeiten (Beispiel: der „Karlsruhe Interactive Verifier“, KIV [Reif 92]) auf dem Gebiet der Softwareverifikation, die darauf abzielen, zu beweisen, daß ein gegebenes Programm korrekt ist, also einer Implementierung einer gegebenen Spezifikation entspricht. Derzeit sind vollständige formale Korrektheitsbeweise nur für kleine Programmteile und unter erheblichem zeitlichem Aufwand zu erreichen. Nicht genügend auf die Softwarekorrektheit zu achten, kann aber unangenehme Folgen haben. Hierzu gibt es viele bekannte Beispiele (in [Prechelt 97] ist eine lange Liste aufgeführt). Es wäre also wünschenswert, einen Mittelweg zu finden, der mit vertretbarem Aufwand einen Gewinn an Softwarequalität verspricht, der dem möglichst nahekommt, was sich mit teuren formalen Verifikationsmethoden erreichen läßt.

Ein solcher Mittelweg, der schon vor mehr als zwei Jahrzehnten von verschiedenen Autoren beschrieben wurde (z. B. [Stucki, Foshee 75] und [Yau, Cheung 75]), beruht auf dem Einfügen von Zusicherungen in den Programmtext, die während der Laufzeit geprüft werden können. Der Programmierer kann also im Programmtext festhalten, daß zu einer bestimmten Zeit eine bestimmte Bedingung gelten muß. In die Programmiersprache C hat diese Idee in Form des in `assert.h` definierten Makros `assert` Eingang gefunden.¹

David S. Rosenblum stellte vor sechs Jahren in [Rosenblum 92] fest, daß diese Grundidee und darauf aufbauende hochentwickelte Systeme wie *Anna* (ANNotated Ada) [Luckham, Henke 85] zwar theoretisch überzeugen, sich aber nicht allgemein durchgesetzt haben. Er sieht vor allem zwei Gründe dafür: mangelhafte Integration der Zusicherungs-Werkzeuge in bestehende Programmierumgebungen und fehlendes Wissen darüber, wie Zusicherungen aussehen müssen, um Fehler möglichst effektiv aufzudecken.

Auch für Eiffel gibt es eine in die Sprache integrierte Möglichkeit, Zusicherungen zu formulieren. Bertrand Meyer beschreibt das Konzept des „Entwurfs per Vertrag“ in seinem Buch „Object-oriented Software Construction“ [Meyer 97]. Dabei wird die Beziehung zwischen einer Klasse und deren Aufrufern als formeller Vertrag betrachtet, in dem für jede Seite Verpflichtungen und Rechte festgelegt werden. Verpflichtungen einer Methode schlagen sich in einer Nachbedingung nieder, die nach der Ausführung gelten muß. Im Gegenzug hat die Methode das Recht, davon auszugehen, daß die Vorbedingung gilt, die ihr zugeordnet ist. Sollte diese Voraussetzung nicht erfüllt sein, ist sie auch nicht an die Nachbedingung gebunden. Ein drittes Instrument ist die Möglich-

¹Das „Linux Programmer’s Manual“ enthält eine nett formulierte Beschreibung dieses Makros. Man findet dort die Warnung: *assert() is implemented as a macro; if the expression tested has side-effects, program behaviour will be different depending on whether NDEBUG is defined. This may create Heisenbugs which go away when debugging is turned on.*