SPRINGER COMPASS

Herausgegeben von P. Schnupp und H. Strunz

Timm Grams

Denkfallen und Programmierfehler

Mit 17 Abbildungen

Springer-Verlag Berlin Heidelberg New York
London Paris Tokyo Hong Kong

Professor Dr. Timm Grams
Fachhochschule Fulda
Fachbereich Angewandte Informatik und Mathematik
Marquardstraße 35
D-6400 Fulda

ISBN-13: 978-3-642-75325-1 e-ISBN-13: 978-3-642-75324-4
DOI: 10.1007/978-3-642-75324-4

CIP-Titelaufnahme der Deutschen Bibliothek
Grams, Timm:
Denkfallen und Programmierfehler / Timm Grams. - Berlin ; Heidelberg ; New York ; London ; Paris ; Tokyo ; Hong Kong : Springer, 1990
(Springer compass)
ISBN-13: 978-3-642-75325-1

2145/3140-543210 Gedruckt auf säurefreiem Papier

Vorwort des Herausgebers

Was ist der Unterschied zwischen einem guten und einem schlechten Programmierer? Daß der gute keine Fehler mehr macht? Ganz falsch! Der bessere Programmierer macht nur bessere Fehler.

Wenn Sie nun fragen, was die besseren von den schlechteren Fehlern unterscheidet: Natürlich ist der bessere Fehler derjenige, der sich besser versteckt, denn ein Fehler, den man sofort sieht, ist ja keiner.

Das ist natürlich ärgerlich. Je besser wir werden, desto länger suchen wir nach unseren Fehlern!

Dieses Phänomen muß seinen Grund haben. Vielleicht liegt es daran, daß uns zwar viele Autoren erklären, wie man programmiert, aber nur wenige, wie man Fehler macht. Wahrscheinlich, weil die meisten meinen, Fehlermachen können wir alle sowieso.

Sicherlich - bloß um etwas zu vermeiden, muß man wissen, wie es funktioniert. Deshalb ist es gut, daß es endlich ein Buch über die Theorie und Praxis des Programmierfehlers gibt. Sind die Schwächen erst einmal erkannt, ist ihre Beseitigung kein großes Problem mehr. Auch dafür finden Sie hier praxiserprobte Hinweise.

Wenn Ihnen also noch immer Ihre Fehler die Freude an Ihren Programmen verderben: Jetzt können Sie etwas dagegen tun!

München, April 1990 Peter Schnupp

Vorwort

Wer einige Erfahrungen im Programmieren hat, wer sich schon einmal gewundert hat, daß er eine bestimmte Art von Programmierfehlern bereits zwei- oder gar dreimal gemacht hat, und wen interessiert, wie solche Programmierfehler zustande kommen und wie man sie vermeiden kann, für den ist dieses Buch gemacht.

Bei vielen Programmierfehlern sind Denkfallen im Spiel, und der Unvorbereitete fällt nahezu zwangsläufig herein, so wie jedermann den optischen Täuschungen erliegt. Paradoxerweise ist es nicht die Unvollkommenheit seines Denkens, die den Irrtum verursacht. Ganz im Gegenteil: Oft stellt sich heraus, daß gerade ein normalerweise sehr nützlicher Denkmechanismus den Fehler hervorgebracht hat. Der Mechanismus ist schon in Ordnung, er war nur fehl am Platz.

Solche Denkfallen werden analysiert, und es wird gezeigt, mit welchen Programmiertechniken Reinfälle zu vermeiden sind. Es geht um die Fragen:

- Inwieweit läßt sich der Programmierstil ändern, so daß die typischen Programmierfehler immer seltener auftreten?
- Welche Techniken zur Verbesserung des Programmierstils gibt es, und wie wirksam sind sie?
- Wie wirksam sind Methoden der Fehlererkennung und Fehlerkorrektur hinsichtlich der typischen Programmierfehler?
- Wie lassen sich die Methoden am besten kombinieren?

Im Mittelpunkt der Betrachtungen steht die Fehleranalyse. Ihr Zweck ist, alle häufiger auftretenden Programmierfehler auf einige wenige Prinzipien und Mechanismen unserer Wahrnehmung und unseres Denkens zurückzuführen. Eine solche Fehleranalyse läßt sich nicht vollständig im Rahmen der Computerwissenschaft abhandeln. Es fließen Erkenntnisse der evolutionären Erkenntnistheorie und der Denkpsychologie ein.

Die Fehleranalyse ist ein sehr effizienter Weg zum besseren Programmierstil. Sie zeigt, wie man optimal aus den Fehlern der Vergangenheit lernen und Fehler zukünftig wirksam vermeiden kann. Auch wenn es das Patentrezept zur Erstellung absolut fehlerfreier Software nicht gibt: Der hier eingeschlagene Weg ist erfolgversprechend, weil er den Problemen an die Wurzel geht.

Wirklichen Nutzen wird der Leser nur haben, wenn er sich aktiv mit der Methode auseinandersetzt. Deshalb enthält der Text eine Reihe von Denksportaufgaben, Übungen und Programmierstudien. Um zu verhindern, daß der Leser ungewollt bereits die Lösung studiert, noch bevor er so richtig zum Nachdenken gekommen ist, wird der Lösungsteil vom Aufgabenteil durch eine Linie und die Aufforderung HALT abgetrennt.

Die Darstellung logischer Ausdrücke und Prädikate geschieht so, daß die Leistungsfähigkeit heutiger Textverarbeitungssysteme für die Umformungsarbeit (Kopieren, Verschieben, Löschen, Ersetzen, Einfügen) gut genutzt werden kann: Jeder Ausdruck ist ein einfacher Text, eine lineare Anordnung von Zeichen also, der hoch- und tiefgestellte Abschnitte enthalten kann. Das soll den Leser anregen, sein Textverarbeitungssystem in derselben Weise direkt für die Programmentwicklung einzusetzen.

Der Text des Buches wurde mit dem Textverarbeitungsprogramm Word geschrieben und für den Lichtsatz konvertiert. Die Programme der Beispiele und Übungen wurden, bis auf wenige Ausnahmen, in Turbo Pascal erstellt und erprobt.

Bei den Literaturhinweisen, insbesondere zu den biologischen und physiologischen Grundlagen, wurde nicht versucht, immer die Originalquellen aufzuzeigen. Hier sind meist allgemein zugängliche Werke aufgeführt, die Einführungs- oder Übersichtscharakter haben und die Hinweise auf weiteres Material für ein vertieftes Studium geben.

Meinen Gesprächspartnern möchte ich danken: Wolfgang Ehrenberger (Fulda) machte mich darauf aufmerksam, daß das Problem der häufig auftretenden Programmierfehler gerade im Zusammenhang mit sicherheitsrelevanter Software von größtem Interesse ist. Er und Francesca Saglietti (Garching) eröffneten mir einen schnellen Zugang zu Literatur über Programmierfehler und Fehlertoleranztechniken. Robert L. Baber zeigte, welche erfrischenden Aspekte das Konstruieren von Software hat, wenn man Logik von Anfang an betreibt. Eine ausführliche Darstellung eines Softwarefehlers im Zusammenhang mit der Voyager 2 Mission verdanke ich William I. McLaughlin (Pasadena, Kalifornien). Für eine Diskussion psychologischer Aspekte bin ich Oswald Huber (Salzburg) dankbar. Hartmut Siebert (Mannheim) und Udo Voges (Karlsruhe) gaben mir Gelegenheit zur Diskussion der mathematischen Modelle und der Verläßlichkeitsbewertung von Software. Dem Verlag danke ich, daß er die erste Fassung des Buches einer gründlichen Kritik unterzog.

Fulda, April 1990 Timm Grams

Inhaltsverzeichnis