

VDI-GIS (Hrsg.): Software-Zuverlässigkeit

Software-Zuverlässigkeit

Grundlagen

Konstruktive Maßnahmen

Nachweisverfahren

Herausgegeben vom

VDI-Gemeinschaftsausschuß

Industrielle Systemtechnik (VDI-GIS)

VDI VERLAG

Die Deutsche Bibliothek — CIP-Einheitsaufnahme

Software-Zuverlässigkeit : Grundlagen, konstruktive
Massnahmen, Nachweisverfahren / hrsg. vom VDI-
Gemeinschaftsausschuss Industrielle Systemtechnik (VDI-GIS).

— Düsseldorf : VDI-Verl., 1993

ISBN-13: 978-3-540-62305-2 e-ISBN-13: 978-3-642-95800-7

DOI: 10.1007/978-3-642-95800-7

NE: Verein Deutscher Ingenieure / Gemeinschaftsausschuss Industrielle
Systemtechnik

© VDI-Verlag GmbH, Düsseldorf 1993

Alle Rechte, auch das des auszugsweisen Nachdruckes, der auszugsweisen oder vollständigen photomechanischen Wiedergabe (Photokopie, Mikrokopie) und das der Übersetzung, vorbehalten.

ISBN-13: 978-3-540-62305-2

Vorwort

Die ersten Computer, die vor etwa 50 Jahren ihre Arbeit begannen, waren zunächst nur anfällig für Hardware-Fehler. Mit der Steigerung ihrer Leistung und auch ihrer technischen Zuverlässigkeit wurde die Software bald wichtiger als die Hardware.

Zunächst machte man sich nur wenig Mühe, auch die Programme auf Richtigkeit gründlich zu prüfen. Das führte eine Zeitlang zu einer Computergläubigkeit, welche auf einer zu hohen Einschätzung der Zuverlässigkeit von Hardware und Software basierte. Die Entwicklung zeigte aber, daß gerade die Software immer schwerer zu durchschauen war, und ihre Entwicklung wurde zu einer eigenen Wissenschaft. Zuverlässige Programme zu schreiben, wurde zu einer hohen Kunst. Bei der heutigen immer stärker werdenden Verflechtung der Computernetze mit ihren weitverzweigten Auswirkungen ist eine zuverlässige Software aber eine Existenzfrage für unsere ganze Gesellschaft.

Das vorliegende VDI-Buch soll dazu dienen, die Voraussetzungen für eine gesunde Entwicklung zu schaffen. Es ist zu wünschen, daß es eine weite Verbreitung bei allen Beteiligten findet.

Konrad Zuse, im Juni 1992

Inhaltsverzeichnis

1	Einleitung	1
2	Übersicht und Leitfaden	5
2.1	Motivation	5
2.2	Geschichtliche Entwicklung programmtechnischer Stilrichtungen	7
2.2.1	Anfänge der Programmierung	7
2.2.2	Strukturierte Programmierung	7
2.2.3	Modulare Programmierung	8
2.2.4	Objektorientierte Programmierung	9
2.3	Software-Zuverlässigkeit und andere Software-Qualitätsmerkmale	11
2.4	Software-Lebenszyklus	12
2.5	Schritte auf dem Weg zu zuverlässiger Software	13
2.6	Zielsetzung und Zielgruppe des Buchs	15
2.7	Hinweise zum Gebrauch des Buchs	16
2.8	Abgrenzung des Buchs	19
3	Konzeptionelle Grundlagen der Software-Zuverlässigkeit	21
3.1	Einführung	21
3.2	Grundsätze der Zuverlässigkeitsbeschreibung bei physikalischen Fehlern	23
3.3	Statistisch-phänomenologische Behandlung von inhärenten Fehlern auf der Basis des Anforderungsprofils	25
3.4	Die Versagensrate und abgeleitete Kenngrößen bei Software	28
3.4.1	Nicht reparierbare Systeme	29
3.4.2	Reparierbare Systeme	30
3.4.3	Ausgewählte Zuverlässigkeitswachstumsmodelle	33
3.4.3.1	Das geometrische Modell von Moranda	36

3.4.3.2	Der logarithmische Poissonprozeß nach Musa und Okumoto	37
3.4.3.3	Ein einfaches Regressionsmodell zur Parameterschätzung	38
3.5	Teilsystem- und Gesamtsystem-Zuverlässigkeit	41
4	Konstruktive Maßnahmen zur Erreichung zuverlässiger Software	43
4.1	Prinzipien der Phasen Anforderungsspezifikation, Entwurf (Design) und Codierung	45
4.1.1	Prinzip der Verständlichkeit	46
4.1.2	Prinzip der Modularisierung	47
4.1.3	Prinzip der losen Kopplung	49
4.1.3.1	Datenkopplung	50
4.1.3.2	Trampdaten	51
4.1.3.3	Strukturkopplung	52
4.1.3.4	Kontrollkopplung	54
4.1.3.5	Kopplung über globale Daten	55
4.1.3.6	Inhaltliche Kopplung	56
4.1.4	Prinzip der hohen Kohäsion	56
4.1.4.1	Funktionale Kohäsion	57
4.1.4.1.1	Verletzung der funktionalen Kohäsion durch Verbindung artfremder Methoden in einer Klasse	58
4.1.4.1.2	Verletzung der funktionalen Kohäsion durch Steigerung der Komplexität	58
4.1.4.2	Sequentielle Kohäsion	59
4.1.4.3	Kommunikative Kohäsion	60
4.1.5	Prozedurale Kohäsion	61
4.1.5.1	Zeitliche Kohäsion	62
4.1.5.2	Zufällige Kohäsion/Klassenbildung	64
4.1.6	Prinzip der 'heilen' Welt	66
4.1.7	Prinzip der 'Magischen Sieben'	67
4.2	Regeln für den Software-Entwurf	68
4.2.1	Änderungsfreundlichkeit	69
4.2.2	Grundsätze des Entwurfs	70
4.2.3	Einzelheiten des Programmentwurfs	73
4.3	Regeln für die Codierung (Programmerstellung)	77
4.3.1	Grundsätze	78

4.3.2	Allgemeine Gesichtspunkte	80
4.3.3	Datenspezifische Gesichtspunkte	84
4.3.4	Arithmetische Berechnungen	87
4.3.5	Echtzeitaspekte	89
4.3.6	Modularisierung, Gliederung in Bausteine	90
4.3.7	Online-Zwischenprüfungen	95
4.3.8	Code-Aufschreibung	98
4.3.9	Kontrollflußspezifische Regeln	101
4.3.10	Sprachspezifische Regeln	102
4.3.11	Prüfen auf Verletzung der Codierungsregeln	105
4.3.12	Implementierungsspezifische Rahmenbedingungen	106
4.4	Programmpflege	107
5	Nachweis der Software-Zuverlässigkeit	109
5.1	Einführung	109
5.2	Rechtsfragen bei Verträgen über Software	113
5.3	Informelle Nachweisverfahren	122
5.3.1	Inspektion	126
5.3.2	Review (Durchsicht)	131
5.3.3	Walkthrough (Durchgang)	132
5.3.4	Schreibtischprüfung	135
5.3.5	Zusammenfassung	135
5.4	Statische Analysen	137
5.4.1	Einleitung	137
5.4.2	Eigenschaften der Statischen Analyse	138
5.4.3	Werkzeuge zur Unterstützung der Analyse	139
5.4.4	Auswahlgesichtspunkte für Analyseverfahren und Analysatoren	140
5.4.5	Analyseergebnisse	142
5.5	Programmkorrektheitsbeweis	144
5.5.1	Einleitung	144
5.5.2	Begriffsbildung	149
5.5.3	Zusammenfassung	151
5.6	Tests	154
5.6.1	Methoden	155
5.6.1.1	Funktionale (Black-Box-) Tests	156
5.6.1.2	Strukturelle (White-Box-) Tests	158

5.6.1.3	Nichtinkrementelles Testen	159
5.6.1.4	Inkrementelles Testen	159
5.6.1.5	Akzeptanz- und Abbruchkriterien	160
5.6.2	Planung und Organisation	160
5.6.2.1	Unterteilung der Software in unabhängige Teilpakete	160
5.6.2.2	Erstellen einer Testspezifikation	161
5.6.2.3	Festlegen des Ablaufs	161
5.6.3	Durchführung	164
5.6.3.1	Begleitende Dokumentation	164
5.6.3.2	Auswerten und Bewerten der Testergebnisse	164
5.6.4	Testwerkzeuge	165
5.6.5	Fehlerbearbeitung nach den Integrationstests	166
5.7	Einsatz der Software und Betriebsbewährtheit	166
5.7.1	Fehlererfassung	166
5.7.2	Fehlerbehebung	167
5.7.3	Auslieferung der geänderten Software	168
5.7.4	Auswertung der Fehlerdaten	168
5.7.5	Definition der Betriebsbewährtheit	168
5.7.6	Nachweis der Betriebsbewährtheit	169
5.8	Quantitativer Nachweis der Zuverlässigkeit	170
5.8.1	Sammeln der Daten	170
5.8.2	Kontrolle der Verwendbarkeit der gesammelten Daten	171
5.8.3	Zeit- oder ereignisorientierte Ansätze	172
5.8.4	Schätzung der Modellparameter	172
5.8.5	Berechnung der Zuverlässigkeit	174
5.8.6	Nutzen der berechneten Zuverlässigkeit	174
6	Schlußbetrachtung	177

Anhänge

A	Objektorientierte Programmierung	179
A.1	Elemente der objektorientierten Programmierung	179
A.2	Objektorientierte Programmierung als Verallgemeinerung der strukturierten Programmierung	185
A.3	Objektorientierte Software-Konstruktion	187
A.4	Übersicht über vorhandene Implementierungen	189
B	Programmkorrektheitsbeweise	191
B.1	Liste der verwendeten Symbole	191
B.2	Beispielprogramm	192
B.3	Die symbolische Ausführung	193
B.3.1	Einführung in die symbolische Ausführung	193
B.3.2	Anwendung der symbolischen Ausführung auf das Beispielprogramm	197
B.4	Beweisverfahren mit axiomatischen Ansätzen	201
B.4.1	Allgemeine Einführung in Beweisverfahren mit axiomatischen Ansätzen	201
B.4.2	Einführung in die Methode der schwächsten Vorbedingung	202
B.4.3	Einführung in die Methode der stärksten Nachbedingung	204
B.4.4	Einführung in die Methode der induktiven Zusicherungen	205
B.4.5	Einführung in die axiomatische Methode von Hoare	206
B.4.6	Beispiel für den Beweis mit der Methode der induktiven Zusicherungen	208
B.5	Eine vereinfachte, praxisorientierte Kombination der theoretischen Ansätze	215
B.5.1	Theoretische Grundlage	215
B.5.1.1	Allgemeine Darstellung	215
B.5.1.2	Definitionen	216
B.5.1.3	Beweisregeln	216

B.5.1.4	Terminierung	223
B.5.2	Beweisbeispiel	225
B.5.2.1	Aufgabenstellung	225
B.5.2.2	Partielle Korrektheit	226
B.5.2.2.1	Die Schleifeninvariante I	226
B.5.2.2.2	Initialisierung der Schleife	227
B.5.2.2.3	Die Invarianz von I	228
B.5.2.2.4	Die Wahrheit der Nachbedingung nach Terminierung	230
B.5.2.3	Vollständige Korrektheit	231
B.5.2.4	Schlußfolgerung: Externe Anforderungsspezifikation des Programms	233
B.5.2.5	Weitere Bemerkungen	234
B.6	Programmkonstruktion (Beispiel)	235
B.6.1	Einführung	235
B.6.2	Grundgedanken zur Programmstruktur und zur Nachbedingung	235
B.6.3	Die Schleifeninvariante	237
B.6.4	Die Initialisierung der Schleife	238
B.6.5	Die while-Bedingung	238
B.6.6	Der Schleifenkern	239
B.6.7	Das gesamte Programm	240
B.6.8	Der Korrektheitsbeweis	240
B.6.9	Schlußbemerkungen	240
C	Theoretische Grundlagen der Zuverlässigkeitswachstumsmodelle	243
C.1	Zuverlässigkeitskennwerte	243
C.1.1	Generelle Einführung der Versagenswahrscheinlichkeit als Kenngröße	243
C.1.2	Die Gegebenheiten im Problemfeld der Software- Zuverlässigkeit	245
C.1.3	Korrektheitswahrscheinlichkeit als Kenngröße	247
C.2	Zur Annahme exponentialverteilter Lebensdauern bei Software	248
C.2.1	Herleitung der Exponentialverteilung	249
C.2.2	Konsequenzen für die zu wählende Teststrategie	250
C.3	Ausgewählte Zuverlässigkeitswachstumsmodelle	250

C.3.1	Das binomiale Modell	251
C.3.2	Das Jelinski-Moranda-Modell	256
C.3.3	Das geometrische De-Eutrophikationsmodell von Moranda	256
C.3.4	Das Modell von Littlewood und Verrall	258
C.3.5	Der nicht-homogene Poissonprozeß	260
C.3.6	Ein Modell zur Berücksichtigung von Abhängigkeiten	261
C.4	Ein einfaches Regressionsmodell	267
C.5	Schlußbemerkungen	269
D	Glossar	271
D.1	Einführung	271
D.2	Schlagworte	272
E	Literaturverzeichnis	287
	Index	295
	Arbeitsgruppen und ihre Mitglieder	301