

SPRINGER COMPASS

Herausgegeben von
M. Nagl P. Schnupp H. Strunz

Manfred Nagl

Softwaretechnik: Methodisches Programmieren im Großen

Mit 136 Abbildungen und 13 Tabellen



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

Prof. Dr. Manfred Nagl
Lehrstuhl für Informatik III
RWTH Aachen, Ahornstr. 55
D-5100 Aachen

ISBN-13: 978-3-642-95625-6 e-ISBN-13: 978-3-642-95624-9
DOI: 10.1007/978-3-642-95624-9

CIP-Titelaufnahme der Deutschen Bibliothek

Nagl, Manfred: Softwaretechnik : methodisches Programmieren im Grossen /
Manfred Nagl. - Berlin; Heidelberg; New York; London;
Paris; Tokyo; Hong Kong; Barcelona: Springer, 1990
(Springer compass)
ISBN-13: 978-3-642-95625-6

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 vom 9. September 1965 in der jeweils gültigen Fassung zulässig. Sie ist grundsätzlich vergütungspflichtig. Zuwiderhandlungen unterliegen den Strafbestimmungen des Urheberrechtsgesetzes.

© Springer-Verlag Berlin Heidelberg 1990
Softcover reprint of the hardcover 1st edition 1990

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.

2145/3140 (3011) 543210 - Gedruckt auf säurefreiem Papier

Vorwort

Dieses *Buch wendet sich an* alle, die größere Softwaresysteme erstellen oder warten oder an der Erstellung bzw. Wartung beteiligt sind und sich über die Struktur des betreffenden Softwaresystems Gedanken machen wollen. Somit wendet sich dieses Buch in erster Linie an *Software-Praktiker* im Berufsleben oder an Studenten, denen diese Tätigkeit bevorsteht, und die sich das hierzu nötige Rüstzeug aneignen wollen. Es setzt dabei nicht unbedingt ein Informatik-Studium an einer Hochschule voraus. Der Stil der Erläuterungen ist so gehalten, daß das Buch auch Praktikern, die sich autodidaktisch Kenntnisse erwerben wollen und hierfür das nötige Interesse aufbringen, verständlich sein sollte.

Die *Zielsetzung des Buches* besteht darin, geeignete *Konzepte* einzuführen, mithilfe derer über die *Architektur von Softwaresystemen* nachgedacht und diskutiert werden kann. Hierfür wird eine *Architekturbeschreibungssprache* eingeführt, die diese Konzepte widerspiegelt, es wird der methodische Umgang mit ihr anhand vieler Beispiele eingeübt, und es wird diskutiert, wie solche Konzepte in die tagtägliche Arbeitswelt übertragen werden können, die von Programmiersprachen wie FORTRAN, Cobol, C oder sogar Assemblersprachen geprägt ist.

Die durch eine Architekturbeschreibung festgehaltene Struktur eines bestehenden oder sich in der Entwicklung befindlichen Softwaresystems ist *Grundlage* dafür, daß über die *Struktur* dieses *Softwaresystems* nachgedacht werden kann. Auch die *Wartungsüberlegungen* finden zum großen Teil auf der Architektur des Softwaresystems statt und nicht auf dem ausformulierten Programmsystem. Insbesondere kann anhand der Architektur über *Qualitätseigenschaften* dieses Systems diskutiert werden. Die Flexibilität, d.h. die Anpaßbarkeit an veränderte Anforderungen und die Übertragbarkeit auf neue Maschinen, stehen dabei im Vordergrund. Erst eine flexible Architektur ermöglicht die Wartung des entsprechenden Softwaresystems! Ferner können anhand der Architekturfestlegung immer wiederkehrende Teilarchitektursituationen erkannt werden, d.h. es können allgemeine, wiederverwendbare Architekturbausteine in Form von Modulen oder Teilsystemen identifiziert werden. So wird die fortschreitende wissenschaftliche Diskussion über solche Fragen hoffentlich dazu führen, daß in Zukunft bei der Erstellung eines größeren Softwaresystems nicht tagtäglich das "Rad neu erfunden wird". Man wird dann vielleicht in der Lage sein, für bestimmte Klassen von Problemen auch zugehörige Standardarchitekturen anzugeben. Kurzum, auf der Architektur finden auch alle fundierten Überlegungen zur *Wiederverwendbarkeit* statt.

Die hier eingeführten *Konzepte* auf Architekturebene haben ihren gedanklichen Ursprung in klassischen Sprachen moderner Ausprägung wie Ada, Modula-2 bzw. in eingeschränkterem Maße aber auch in objektorientierten Sprachen wie Smalltalk. Die Diskussion in diesem Buch geht aber hauptsächlich um die Frage, wie man mit solchen Konzepten sinnvoll und methodisch umgehen sollte. Diese Erkenntnisse können somit nicht Büchern über die obengenannten Sprachen entnommen werden, sie sind z.B. für einen Ada-Programmierer keineswegs selbstverständlich. Trotz der gedanklichen Nähe zu obigen Sprachen ist dieses Buch *programmiersprachenunabhängig*. Aus diesem Grunde wird die Übertragung der hier vorgestellten Konzepte in die heute üblichen Programmiersprachen auch vorgeführt.

Dieses Buch spiegelt einen *Zwischenzustand* wider, da die *Diskussion* über und die Suche nach *Konzepten* für *Programmsystemstrukturen* in der Wissenschaft noch *voll im Gange* ist. Insbesondere nach Erscheinen der Sprachen Ada und Smalltalk-80 hat hier eine breite Diskussion eingesetzt (was nicht bedeuten soll, daß die als neu empfundenen Konzepte in diesen Sprachen zuerst auftauchten). So ist die Frage, einen einheitlichen gedanklichen Rahmen für Architekturbeschreibungen zu finden, der die Ideen aus klassischen, objektorientierten, aber auch weiteren Sprachen vereint, m.E. zur Zeit noch nicht befriedigend gelöst. Wenn dieses Buch auch einen einigermaßen breiten Rahmen spannt, so bleiben doch bestimmte Softwareerstellungs-Paradigmen (wie etwa die logische Programmierung) hier außer acht. Der Leser ist aufgefordert, sich mit solchen Ansätzen, entsprechenden Programmiersprachen und ihren Auswirkungen auf der Ebene von Softwarearchitekturen auseinanderzusetzen (vgl. Literaturabschnitt 5).

Die in diesem Buch vorgestellten *Ideen* und *Erkenntnisse* haben eine lange *Vorgeschichte*, die hier nicht unerwähnt bleiben soll, um auch den Beitrag anderer zu dokumentieren. Diese Vorgeschichte ist am Anfang des Kapitels 4 aufgeführt. Ferner sei darauf hingewiesen, daß es gerade in neuerer Zeit eine Reihe ähnlicher Ideen in der Literatur gibt (vgl. Literaturabschnitt 4). Die *Diskussion* um Konzepte und Sprachen für Architekturen und deren methodische Anwendung wird auch mit Sicherheit *weitergehen*. Das hier vorgestellte didaktische Konzept der Vermittlung von Architekturüberlegungen ist ebenfalls über lange Jahre, anhand einiger Vorlesungen und etlicher Industrieminare, gereift. Es hat damit einen Teil seiner Bewährungsprobe bereits hinter sich.

Dieses *Buch* setzt *keine Kenntnisse* der oben erwähnten Programmiersprachen Ada, Modula-2, Smalltalk etc. oder Kenntnisse über Softwaretechnik *vor*aus. Solche Kenntnisse sind aber natürlich sehr nützlich und werden das Verständnis erleichtern und vertiefen. Programmiersprachenkenntnisse auf der Basis von Pascal oder C sollten aber vorhanden sein. Wie die Praxis gezeigt hat, geht es auch mit Erfahrungen in FORTRAN, Cobol oder Assembler. Als nützlich für die Bereitschaft zur Aufnahme der hier vorgestellten Ideen hat sich die Erfahrung im Umgang mit großen Softwaresystemen und ihrer Wartungsproblematik erwiesen. Je nach Vorkenntnissen sollte

dieses Buch auch unterschiedlich gelesen werden. Bei Vorliegen von Erfahrung in neueren Programmiersprachen und Softwaretechnik genügt es, die ersten zwei Kapitel zu überfliegen und dann das intensive Studium zu beginnen. Liegt beides nicht vor, so ist, vom Umfang her, für einen ersten Durchgang Kapitel 1 bis 4, 6 und der Anfang von Kapitel 7 ausreichend. Die restlichen Kapitel sollten erst nach einer Phase der Nacharbeit und der Anwendung des in diesen Kapiteln präsentierten Stoffs gelesen werden.

Nun zur *Gliederung des Buches*: Kapitel 1 dient der Einordnung der hier vorgestellten Problematik der Architekturmodellierung in den Gesamtzusammenhang der Softwaretechnik. Kapitel 2 erläutert die Wichtigkeit der Architekturmodellierung, grenzt sie gegenüber anderen, hier nicht behandelten Gebieten ab und zeigt die Zusammenhänge auf. In Kapitel 3 wird ein Beispiel in der heute noch üblichen funktionsorientierten Zerlegung angegeben, wobei sich viele Fehler ergeben, die wir später im einzelnen diskutieren und korrigieren. Kapitel 4 führt die Architekturbeschreibungssprache in einer zunächst einfachen Form ein, in der Bausteine verschiedener Arten, Beziehungen verschiedener Arten zwischen Bausteinen und Konsistenzbedingungen eingeführt werden. Dies ist das erste Hauptkapitel des Buches. Das nächste Kapitel dient zum einen der Erweiterung der Architekturbeschreibungssprache um weitere Konzepte und zum anderen dem Einüben des Umgangs mit dieser Sprache, indem bestimmte Teilarchitektursituationen studiert werden. Kapitel 6 führt vor, wie die hier vorgestellten Konzepte in die heute gängigen Programmiersprachen übertragen werden können, wobei wir durch die Transformation diese Konzepte noch einmal, und zwar aus einem anderen Blickwinkel, kennenlernen und damit vertiefen werden. Wir haben in diesem Buch darauf verzichtet, Werkzeuge für die Architekturmodellierung vorzustellen, die unter anderem diese Transformation übernehmen könnten. Kapitel 7 führt größere Architekturbeispiele vor und dient damit dem Nachweis der Bedeutung der Architekturüberlegungen. Kapitel 8 stellt einige Regeln vor, die bei der Architekturmodellierung zu beachten sind, wenn "intelligentere" Softwaresysteme bezüglich Wartbarkeit und Wiederverwendbarkeit entstehen sollen. Im letzten Kapitel skizzieren wir schließlich die noch offenen Probleme der Architekturmodellierung und ihre Vernetzung mit anderen Arbeitsbereichen, die in diesem Buch nicht erörtert werden. Ein umfangreiches, in Abschnitte eingeteiltes Literaturverzeichnis soll das weiterführende Studium erleichtern, und ein detailliertes Stichwortverzeichnis das Nachschlagen ermöglichen.

An dieser Stelle möchte ich allen herzlich danken, die zu diesem Buch beigetragen haben. Das sind zunächst diejenigen, die über den hier präsentierten Inhalt mitdiskutiert haben, nämlich die Herren Dr. G. Engels, Dr. C. Lewerentz, Dr. W. Schäfer, A. Schürr und B. Westfechtel sowie ungenannte Teilnehmer von Vorlesungen oder Seminaren. Über diese Diskussionen hinaus haben die Herren J. Börstler, Th. Janning und G. Metzen mit sehr viel Engagement die hier vorgestellte Präsentation nachhaltig und positiv beeinflusst. Schließlich haben Frau A. Fleck, Frau M. Hirsch und Frau M. Schiermeyer mit großer Geduld und viel Geschick das Manuskript in eine druckreife Form gebracht.

Aachen, im März 1990

Manfred Nagl

Inhalt

1. Der Kontext: Softwaretechnik-Grundlagen	1
1.1 Softwarekrise und Softwaretechnik	1
1.2 Aktivitäten und Ergebnisse einzelner Phasen	6
1.3 Diskussion von Lebenszyklusmodellen	10
1.4 Zum Problem der Wartung	13
1.5 Zusammenfassung der Aktivitäten in Arbeitsbereiche	17
1.6 Eigenschaften von Programmsystemen	23
1.7 Die Modellierungsproblematik: Allgemeines	26
1.8 Allgemeine Begriffe der Softwaretechnik	28
1.9 Werkzeuge zur Softwareerstellung	30
1.10 Zum Stand der Softwaretechnik	34
1.11 Zusammenfassung	38
Aufgaben zu Kapitel 1	39
2. Das Problem: Modellieren auf Entwurfsebene	41
2.1 Zur Korrektheit von Programmsystemen	41
2.2 Die Festlegung der Entwurfsspezifikation	43
2.3 Das Architekturparadigma	47
2.4 Zur Bedeutung des Programmierens im Großen	50
2.5 Überblick über das folgende	55
2.6 Zum Einfluß des Softwarestellungs-Paradigmas und der Programmiersprache	59
2.7 Zusammenfassung	61
Aufgaben zu Kapitel 2	62
3. Ein erstes Beispiel: Ohne Vorüberlegung	63
3.1 Die Aufgabenstellung	63
3.2 Eine erste Lösung	68
3.3 Charakterisierung der Lösung	73
3.4 Zusammenfassung	74
Aufgaben zu Kapitel 3	75

4. Die Notation: Ein einfaches Modulkonzept	77
4.1 Module als Bausteine der Architektur	77
4.2 Modularten und Funktionsmodule	84
4.3 Datenabstraktionsprinzip und Datenobjektmodule	89
4.4 Datentypmodule und sonstige Module	105
4.5 Beziehungen zwischen Modulen	117
4.6 Die lokale Benutzbarkeit	120
4.7 Die allgemeine Benutzbarkeit	127
4.8 Zur Darstellung von Architekturen	135
4.9 Konsistenzbedingungen für Architekturen	142
4.10 Zusammenfassung	148
Aufgaben zu Kapitel 4	149
5. Zur Vertiefung: Teilarchitekturüberlegungen und Modulkonzept- Erweiterungen	153
5.1 Datenabstraktion versus funktionale Abstraktion	153
5.2 Zusammenspiel zwischen funktionalen und Datenabstraktionsmodulen	156
5.3 Teilsysteme einer Gesamtarchitektur	167
5.4 Schichtenbildung durch mehrstufige Datenabstraktion	174
5.5 Mehrfache Datenabstraktion: Einträge und Kollektionen	181
5.6 Einige Teilarchitekturen für Einträge und Kollektionen	191
5.7 Generische Module und Teilsysteme	195
5.8 Objektorientierte Programmiersprachenkonstrukte	203
5.9 Objektorientierte Architekturmodellierung	217
5.10 Zusammenfassung	231
Aufgaben zu Kapitel 5	233
6. Vorbereitung für den Einsatz: Übertragung in Programmiersprachen ...	239
6.1 Probleme bei der Übertragung	241
6.2 Sprachen mit unabhängigen Programmeinheiten: Beispiele FORTRAN, C	244
6.3 Blockstrukturierte Sprachen ohne Module: Beispiel Pascal	254
6.4 Sprachen mit Modulen: Beispiel Ada	258
6.5 Zusammenfassung	270
Aufgaben zu Kapitel 6	272

7. Einübung durch Beispiele: Einige Softwarearchitekturen	275
7.1 Wiederaufgreifen des ersten Beispiels: Angabe der Architektur ..	276
7.2 Erweiterungen des Beispiels und zugehörige Architekturmodifikationen	293
7.3 Ein Transformationsproblem: rekursiver Abstiegscompiler	302
7.4 Die Grobarchitektur einer Softwareentwicklungs-Umgebung	313
7.5 Zusammenfassung	322
Aufgaben zu Kapitel 7	324
8. Allgemeine Hinweise: Strategien zur Adaptabilität und Wiederverwendbarkeit	327
8.1 Was haben die bisherigen Überlegungen gebracht?	328
8.2 Einige Strategien zur Erstellung wartungsfreundlicher Architekturen	331
8.3 Zusammenfassung	347
Aufgaben zu Kapitel 8	347
9. Noch nicht gelöst: Offene Probleme und Weiterführung	349
9.1 Modulkonzept-Ergänzungen und andere Architekturerstellungs- Paradigmen	349
9.2 Unterstützung des Erstellungs- und Wartungsprozesses auf Architekturebene	355
9.3 Programmieren im Großen und seine Verzahnung mit anderen Arbeitsbereichen	359
9.4 Zusammenfassung	365
Aufgaben zu Kapitel 9	366
Literaturverzeichnis	367
Stichwortverzeichnis	381