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

Softcover reprint of the hardcover 1st edition 1990

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 Industrieseminare, 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 *voraus*. 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 Wiederverwenbarkeit 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