Buchbesprechung

Mary Shaw und David Garlan; Software Architecture — Perspectives on an Emerging Discipline.[*]

Ernst-Erich Doberkat[†]

29. Mai 2000

Software-Architektur wird von den Autoren verstanden als die Beschreibungsebene, auf der die strukturellen Aspekte eines Software-Systems beschrieben werden. Hierbei handelt es sich um die übergreifenden Aspekte wie die Komponenten, aus denen ein System zusammengesetzt ist, und die Konnektoren, mit deren Hilfe diese Komponenten interagieren: "Abstractly, software architecture involves the description of elements from which systems are built, interactions among those elements, patterns that guide their compositions, and constraints on these patterns. In general, a particular system is defined in terms of a collection of components and interaction among those components. Such a system may in turn be used as a (composite) element in a larger system design" (p.3). Diese sehr allgemeine Architekturauffassung durchzieht das Buch. Es ist gewiß reizvoll, sie gegen die aus der (Bau-)Architektur stammende Auffassung von Alexander zu halten, der Komponenten wohl eher als Vokabular versteht, aus denen die Artefakte des Architekten, eben Gebäude, hervorgehen. Die Alexandersche Auffassung wird im bekannten Buch von Gamma et. al. für Entwurfsmuster zugrunde gelegt. Sie vernachlässigt die Interaktion von Komponenten insofern, als den Konnektoren keine Bürgerrechte erster Klasse verliehen werden, wie dies im Buch von Shaw und Garlan der Fall ist.

Das erste Kapitel hat einführenden Charakter und setzt den Ton für die kommende Diskussion. Neben der oben kurz angedeuteten Antwort auf die Frage nach dem Wesen der Software Architektur wird der Stand der Entwicklung der Softwaretechnik als Ingenieur-Disziplin geschildert. Der im Vergleich zu anderen, klassischen Ingenieur-Disziplinen defizitäre Stand des Software Engineering wird charakterisiert, es wird gezeigt, daß die auf Klassifikation und Synthese zielende Arbeit des Software-Architekten sicher dazu beitragen kann, der Disziplin zu einem soliden Unterbau zu verhelfen. Die Unterschiede zwischen einem algorithmisch geprägten Ansatz (*Programming in the small*) und einem an der Konstruktion von Systemen orientierten Vorgehen (*Programming in the large*) werden sorgfältig herausgearbeitet. Es wird gezeigt, wie sich jeder dieser Zugänge in die Systemkonstruktion integrieren muß, und daß sich die Fragestellungen auf dem Systemniveau erheblich von denen auf der algorithmischen Ebene unterscheiden. Der Fokus, so wird verdeutlicht, liegt bei den Diskussionen des Buchs auf der Ebene der Systeme, nicht der Algorithmen.

Kapitel 2 beschreibt kurz architekturelle Stile in allgemeiner Form und diskutiert einige von ihnen, um das zugehörige Vokabular einzuführen und dem Leser ein Gefühl für den Untersuchungsgegenstand zu geben. Die Diskussion zielt hier nicht auf spezielle Anwendungssysteme, sie versucht vielmehr, eine eher abstrakte Charakterisierung mit Tiefenschärfe zu geben. Es fällt auf, daß zwar der Stil `Process Control` diskutiert wird,

[*]Prentice Hall, Upper Saddle River, N. J. 1996 (ISBN 0-13-182957-2)

[†]Lehrstuhl für Software-Technologie, Universität Dortmund

Client/Server-Architekturen aber nur am Rand erwähnt werden. Kapitel 3 konkretisiert diese Diskussion durch die Diskussion einiger Fallstudien. Es werden Beispiele angegeben (KWIC von Parnas; Instrumentenbau bei Tektronix; Robotik; das klassische Problem des Geschwindigkeitsseglers von G. Booch), für jedes dieser Probleme werden architekturelle Lösungen vorgeschlagen und diskutiert. Dieses Kapitel ist sehr instruktiv und hilft dem Leser, die verschiedenen Entwurfsstile zu verstehen und einschätzen zu lernen. Kapitel 4 befaßt sich mit integrativen Aspekten von Informationssystemen, wobei die Integration von Software-Entwicklungssystemen einen breiten Raum einnimmt und diesen Entwurfsstil für Systeme deutlich macht. Als Anwendung integrativer Techniken werden Informationssysteme in der (Bau-) Architektur, also beim Entwurf von Gebäuden beschrieben, ein Thema, das dem zu erwartenden Publikum möglicherweise einigermaßen fremd ist (und leider auch bleibt). Die Varianten zu Datenflußsystemen und zu Repositories sind dem Softwaretechniker schon eher vertraut. Kapitel 5 zum Thema Entwurfsempfehlungen ist von Gastautoren verfaßt, der erste Teil von T. G. Lane faßt wohl die Dissertation des Autors zusammen und gibt einen Einblick in Entwurfsüberlegungen zu Benutzungsschnittstellen — interessant ist hier das Zusammenspiel von Entwurf und Validierung. Ein quantitativ orientierter Entwurfsraum wird von einer studentischen Arbeitsgruppe beschrieben.

Die Frage nach gleichen Bürgerrechten für Komponenten und Konnektoren wird in Kapitel 7 in größerem Detail diskutiert, und es wird gezeigt, wie Spezifikationen in Z für architekturelle Probleme aussehen. Das Ineinandergreifen von Konnektoren und Komponenten wird hier recht durchsichtig dargestellt. Kapitel 8 widmet sich ähnlichen Problemen aus programmiersprachlicher Sicht. Erwünschte Eigenschaften für Architekturbeschreibungssprachen werden genannt und mit Hilfe der in früheren Kapiteln angeführten Beispielen nachvollziehbar gemacht. Ein ausgearbeitetes Beispiel (impliziter Aufruf von Prozeduren) zeigt, wie sich ein architekturell interessantes Konstruktionsprinzip in eine konventionelle objektbasierte Sprache wie Ada integrieren läßt. Das Beispiel wird aus verschiedenen Blickwinkeln diskutiert, einige Lösungsansätze werden gefunden und wieder verworfen, so daß eine recht lebendige Diskussion entsteht.

Das letzte inhaltlich geprägte Kapitel befaßt sich mit Werkzeugen zum architekturellen Entwurf, es diskutiert u.a. zwei Entwurfssprachen und zeigt auf, wie sich die gefundenen Abstraktionen in den Sprachen auf handhabbare Weise realisieren lassen.

Das letzte Kapitel gibt eine Anleitung für eine Vorlesung zum Thema „Software Architektur", die sehr stark von curicularen Gegebenheiten an US-Universitäten geprägt ist und in der beschriebenen Art an einer deutschen Universität wohl nicht realisiert werden kann.

Das Buch bietet eine schöne Grundlage für die Diskussion von Software-Architekturen. Es ist begrifflich recht genau und verliert sich nicht im Nebel- und Anekdotenhaften. Gleichwohl hat es einige Mängel, die seinen Nutzen einschränken. Die starke Betonung von *pipes and filters* und die Vernachlässigung von Client/Server-Architekturen wurde oben schon angedeutet und erscheint als wichtiges Defizit. Didaktisch sind einige Teile ziemlich mißlungen — Abbildungen werden meist nicht erläutert, auch wenn die Essenz einer Architektur darin verkapselt ist (das fand ich in 3.1 besonders störend). Einige triviale quantitative Aspekte insbesondere in 5.2 werden doch arg breitgetreten, und manche formalen Konstruktionen im Zusammenhang mit Z hätten sicher breiter und damit verständlicher dargestellt werden können. Gern gesehen hätte ich auch eine Diskussion des Verhältnisses von Entwurfsmustern und architekturellen Fragen (das Buch von Gamma et al. wird im Literaturverzeichnis angegeben, aber im Text nur kurz angesprochen).