Softwarearchitekturstile (Stichworte)

Udo Kelter

01.07.2009

Zusammenfassung dieses Lehrmoduls

Die Architektur eines Softwaresystems ist entscheidend für das Management der Systementwicklung und für die Verstehbarkeit und Wartbarkeit des Systems. Ein Softwarearchitekturstil legt fest, aus welchen Komponenten ein System besteht, wie diese interagieren und welche Entwurfskriterien angewandt werden. Vorgestellt werden die wichtigsten Softwarearchitekturstile mit einer kurzen Einschätzung ihre Anwendungsbereichs und ihrer Vor- und Nachteile.

Vorausgesetzte Lehrmodule:

empfohlen: – Software-Architekturen

Stoffumfang in Vorlesungsdoppelstunden: 1.0

Inhaltsverzeichnis

1 Einleitung **3**
1.1 Einordnung und Stellenwert 3
1.2 Definition “Software-Architektur” 3
1.3 Relevanz der Software-Architektur 4
1.4 Architekturbeschreibungssprachen 5
1.5 Architektur: Instanz vs. Muster vs. Typ 5
1.6 Qualität von Architekturen 6

2 Architekturstile **6**
2.1 Funktionsorientierter Architekturstil (Structured Design) . . 7
2.2 ADT-orientierter Architekturstil 7
2.2.1 Verteilte Varianten 8
2.3 Objektorientierter Architekturstil 9
2.4 Datenflußorientierte Architektur 10
2.5 Komponentenorientierte Architektur 11
2.6 Interpreter-Architekturen 12
2.7 Blackboard-Architektur 12
2.8 Domänenspezifische Architekturen 13

Literatur 13
Index 13

1 Einleitung

1.1 Einordnung und Stellenwert

- Berufsbezeichnung "Softwarearchitekt"
- architekturzentrierte SW-Entwicklung (u.a. im Zusammenhang mit dem Unified Process)

  Steigerung: architekturzentrierte modellgetriebene Softwareentwicklung ...

Institutionen:

- GI-Fachgruppe Software-Architektur im Fachbereich Softwaretechnik
  `www.fg-swa.de/`
  von dort:
  `http://www.handbuch-softwarearchitektur.de/`
  s. Kap. 1 Einleitung
- weitere GI-Ausschüsse / AGs:
  u.a. GI-Fachausschuß Architektur von Rechensystemen
- andere Fachgesellschaften

1.2 Definition "Software-Architektur"

Umgangssprachliche Definition von Architektur: Gestaltung von Bauwerken (allg.: technischen Systemen) hinsichtlich

- Ästhetik (Kunst)
- Funktion (Technik)
- Tektonik (Statik)
- Ökonomie

IEEE-Standard 1471-2000:

Definition "Software-Architektur": *Die Software-Architektur ist die grundlegende (grobgranulare) Organisation eines Systems, dargestellt durch*
- dessen Komponenten,
- deren Beziehungen zueinander und zur Umgebung sowie
- die Prinzipien, die den Entwurf und die Evolution (!) des Systems bestimmen.

1.3 Relevanz der Software-Architektur

Aus Sicht des Projektmanagements:

- Aufwandsschätzung in den frühen Planungsphasen
- Risiken minimieren: Analyse und Simulation zur Bewertung
- Teilaufgaben voneinander entkoppeln, Arbeitsteilung ermöglichen
- Wiederverwendung von Komponenten, make-or-buy-Entscheidungen
- Kommunikationsmedium / Dokumentation

Aus Sicht eines Betreibers:

- wie installieren? (und deinstallieren)
- auf welchen Rechnern / Plattformen?
- welche Schnittstellen zu anderen Systemen? Integration in Anwendungslandschaften? Integrationstechnologien?
- Administrationsfunktionen (Datensicherung, Wiederaufsetzen nach Störungen, ...)
- Angriffsmöglichkeiten, Sicherheits"lücken"

Aus Sicht von Entwicklern:

- Verstehbarkeit des Systems

- Wartbarkeit
- Analyse und Simulation zur Bewertung und Vorhersage
- ggf. beim MDD Eingabe für Generatoren
- "programming-in-th-large", "mega-programming"[1]

1.4 Architekturbeschreibungssprachen

für Dokumentation, Analysen usw.: Architektur muß aufgeschrieben werden

→ Architekturbeschreibungssprache; Beispiele:

- Teile von Programmiersprachen (z.B. Header-Dateien, Modul-/Interface-Spezifikationen)
- UML: Paketdiagramme, Kompositionsstrukturdiagramm, Verteilungsdiagramm, Komponentendiagramm

Bewertung einer Architekturbeschreibungssprache:
Kann die Sprache die relevanten Eigenschaften ausdrücken?
Werden passende Abstraktionen unterstützt? – hängt offenbar von der Art des Systems ab

1.5 Architektur: Instanz vs. Muster vs. Typ

- **1 Architektur**: beschreibt ein ganz konkretes System (Instanz)
  → Architekturbeschreibungen sind i.d.R. Modelle,
  Umkehrung gilt nicht
- **Architekturmuster**: Menge von Gestaltungsregeln oder fest vorgegebenen Komponenten
  ("System X hat eine 3-Schichten-Architektur")
  "**Standardarchitektur**": definiert i.d.R. ein Architekturmuster für 1 Applikationsdomäne

[1] s. Gio Wiederhold, Peter Wegner, Stefano Ceri: Towards Megaprogramming: A Paradigm for Component-Based Programming; CACM; 1996

- **Architekturtyp** (∼ Architekturbeschreibungssprache): definiert
  - verwendete Komponenten- (Modul-) Typen
  - Arten von Kooperation zwischen Komponenten

  und damit die vorgenommenen Abstraktionen;
  ferner z.T. Pragmatik, Muster

Architekturstil ist definiert durch:
- Architekturtyp
- Designregeln / Pragmatik

oft keine klare Trennung zw. Architekturstil - Architekturtyp

1.6 Qualität von Architekturen

eher abstrakte Qualitätsbegriffe

teilweise nur sinnvoll innerhalb eines Architekturstils

i.f. bottom-up-Ansatz: erst Architekturstile kennenlernen

2 Architekturstile

Übersicht:

1. Architekturstile für nichtparallele, nichtverteilte Systeme:
   - funktionsorientierter Architekturstil (Structured Design)
   - ADT-orientierter Architekturstil
   - Objektorientierter Architekturstil

   Systemmerkmale:
   - zentralisiert, Ausführung in einem Prozeß / auf einem Prozessor
   - keine relevante Parallelität

2. Architekturstile für parallele Systeme:
   - Prozeß- bzw. Datenflußnetze

- Komponentenarchitekturen

Systemmerkmale:

- Ausführung in mehreren kommunizierenden Prozessen,
- auf einem oder mehreren Prozessoren (verteilte oder nichtverteilte Systeme)

3. Architekturstile für verteilte Systeme:
   - Blackboard

2.1 Funktionsorientierter Architekturstil (Structured Design)

- Entstehung: 1970er Jahre, inzwischen veraltet
- Komponententypen: einzelne Funktionen in üblichen imperativen Progammiersprachen
- Arten der Kooperation: Funktionsaufruf
- Designregeln:
  - Hierarchie von Funktionen, unten elementare, oben komplexere Funktionen, Hauptprogramm an der Spitze
  - funktionale Abstraktion
  - Datenaustausch über Parameter oder gemeinsame Variablen, ggf. Unterteilung in Daten und Steuerinformation
- Hauptnachteile: fehlende Datenkapselung, zu kleinteilig

2.2 ADT-orientierter Architekturstil

- Entstehung: Ende 1970er Jahre, durch Programmiersprachen wie Modula-2 direkt unterstützt
- Komponententypen: "Module", bestehend i.a. aus:
  Datentypdeklarationen, Konstanten, Operationen;

  abstrakter Datentyp: Modul, das einen relevanten Typ exportiert + einige Designregeln beachtet

  auch reine Funktionsmodule

- Arten der Kooperation: “Benutzung” der exportierten Typen, Konstanten, Operationen
- Designregeln / Entwurfsprinzipien:
  - funktionale Abstraktion
  - Information hiding, Datenkapselung
  - zyklusfreie Benutzungshierarchie (Rekursion nur innerhalb eines Moduls)
  - Schichten, virtuelle Maschinen
- Beispiele (massenhaft):
  - 3- oder 5-Schichtenarchitektur(stil) von Informationssystemen,z s. Lehrmodul SAR!
  - Implementierungshierarchie von Datenbankobjekten,z s. Lehrmodul DBSA
  - IP-Protokollstack
- Bewertung:
  - sehr gute Unterstützung des Information hiding
    → relativ gute Wartbarkeit
  - in der Grundform zentralisiert, liefert Basis für verteilte Systeme (dort aber andere Entwurfskriterien!)
  - Finden der Abstraktionen in den mittleren Schichten kann anspruchvoll sein;
  - Rückverfolgen von Anforderungen ist nicht einfach

2.2.1 Verteilte Varianten

Basis: ADT-orientierter Architekturstil

Grundannahme dort: ganzes System in 1 Prozeß ausgeführt.

Verteilte Varianten: Komponenten auf mehrere Rechner verteilt wegen

- Performance
- Sicherheit
- Zugriff auf entfernte Dienste über lokales GUI

Client-Server-"Architektur":
- System aus 2 Schichten
- jede Schicht auf eigenem Prozessor ausgeführt
- auf n Schichten verallgemeinerbar

entfernte Benutzung:

- logisch äquivalent zu einer lokalen
- technisch deutlich anders
- i.d.R. durch Netzwerktechnologie (RPC, RMI, CORBA usw.) unterstützt

Netzwerktechnologien u.ä. Infrastrukturen:

- sind applikationsunabhängig und daher *nicht Teil der Architektur*
- Ausnahme: aus der Sichtweise des Administrators der Netzwerktechnologie / Infrastruktur

2.3 Objektorientierter Architekturstil

- Weiterentwicklung des ADT-orientierten Architekturstils
- Komponententypen: Module (Klassen) oder Pakete
- Arten der Kooperation: wie ADT-orientierter Architekturstil, zzgl. Typhierarchien
- Designregeln / Entwurfsprinzipien:
  - oo Modellierung
  - funktionale Abstraktion: nur eingeschränkt
  - nur eingeschränkt Information hiding und Datenkapselung: wird bei Subtypbildung verletzt

  - keine zyklusfreie Benutzungshierarchie (wird durch Polymorphie oder inner-Konstrukte durchbrochen)

- Bewertung:
  - bessere Konsistenz mit dem Begriffen in den frühen Phasen, bessere Rückverfolgung von Anforderungen
  - Verhalten nicht mehr sicher spezifizierbar (wg. überschriebener Operationen)
  - normalerweise keine Abstraktionsschichten vorhanden!
    → Ablauf- und Abhängigkeitsstrukturen ggf. wesentlich komplexer und schlechter durchschaubar als in ADT-orientierten Architekturen[2] = massiver Nachteil
  - methodisch anspruchsvoller als ADT-orientierter Entwurf - große Chancen und Risiken
  - mit ADT-orientiertem Architekturstil kombinierbar

2.4 Datenflußorientierte Architektur

- auch als prozeßorientierte Architektur bezeichnet
- eher für nichtinteraktive Systeme; System besteht i.w. aus Netzwerk aktiver Knoten, die über Kanäle verbunden sind, in denen Datenströme transportiert werden
- Komponententypen: einzelner Prozeß (oder Thread oder Filter), der in sich i.d.R. sequentiell ist:

  ausgeführtes Programm kann in einer beliebigen Sprache geschrieben sein
- Arten der Kooperation: Datenflüsse (Ströme) oder Dienstaufrufe

[2] Jahr 1960, Programmieren mit goto: "Spaghettihaufen"; Jahr 2000, Programmieren mit wild interagierenden Klassen: "Tortellinisuppe".

Sonderfall: Pipeline-Architektur

- Gesamtwirkung des Systems = *lineare* Sequenz von Transformationsschritten
- Varianten der "Datenströme" / Kanäle:
  1. Kanal = *Puffer*, mit beschränkter Größe (geringer Platzverbrauch)
     Stationen der Pipeline laufen parallel
     Beispiel: UNIX-Pipe (= Verbindung zweier Stationen in einer Pipeline)
  2. Kanal = *temporäre Datei*
     Stationen der Pipeline laufen eine nach der anderen,
     wenn eine temp. Datei gelesen worden ist, kann sie sofort gelöscht werden
     Beispiel 1: Compiler
     Beispiel 2: Visualisierungspipeline
  3. Kanal = *Datenobjekt, das "ausgebaut" wird*
     Stationen der Pipeline laufen eine nach der anderen,
     Beispiel: Compiler (teilweise)

  Varianten können gemischt auftreten
- Bewertung: gut geeignet für Systeme, man als Sequenz von Transformationsschritten verstehen kann...

2.5 Komponentenorientierte Architektur

- Weiterentwicklung des objektorientierten Architekturstils
- Komponententypen: Pakete, "Bundles", ganze Applikationen (jedenfalls mehr als 1 Klasse), mit Schnittstelle, die angegeben Dienste beschreibt
  interface definition language, IDL
  Dienste sind gekapselt (Geheimnisprinzip)
  Komponenten typischerweise geographisch verteilt

- Art der Kooperation: Benutzung der Dienste, über einen **Makler** (*broker*):

  kein direkter Aufruf, sondern Beschreibung des gewünschten Dienstes, und der Makler findet einen passenden Anbieter (kann weit über einen reinen Methodendispatcher hinausgehen)

  Diensteanbieter müssen ihre Angebote beim Makler registrieren
- Bewertung:
  - stark entkoppelte Komponenten
  - System leicht umkonfigurierbar
  - Funktionalität im Detail nicht kontrollierbar

2.6 Interpreter-Architekturen

- vor allem für Systeme, die in erheblichem Umfang umkonfigurierbar sein müssen, Systemfamilien u.ä.
- Grundlage: ADT- oder objektorientierte Architektur
- zusätzliche Komponententypen:
  - Interpreter
  - Konfigurationsdaten ("Ressourcen"), die zwar nur Daten sind, aber die Funktion des Systems entscheidend beenflussen und die gleiche Rolle einnehmen wie klassicher Quellcode

2.7 Blackboard-Architektur

- vor allem für Systeme geeignet, in denen Personen verteilt und ohne zentrale Kontrolle kooperieren / kommunizieren müssen
- Komponententypen:

  (a) das Blackboard: zentrale Datenstruktur, in der die relevanten Informationen strukturiert verwaltet

  (b) "Editoren", die entfernt Teile des Blackboards lesen und ggf. Inhalte ändern

- Art der Kooperation: Editoren greifen lesend und schreibend auf das Blackboard zu;

  Blackboard notifiziert Editoren über Änderungen
- strukturell ähnlich wie MVC-Entwurfsmuster, aber im größeren Maßstab

2.8 Domänenspezifische Architekturen

Beispiele:

- Betriebssysteme: klassische Schichtenarchitektur vs. Mikrokernel vs. Hybridarchitekturen
- Frameworks für bestimmte Domänen, z.B. GUI-Konstruktion

Literatur

[SAR] Kelter, U.: Lehrmodul “Software-Architekturen”; 2003