


Softwarearchitekturen und modellgetriebene Softwareentwicklung - Architektur

Modellbildung in der Entwicklung
Prof. Dr. Dr. h.c. Manfred Broy
Gemeinsam mit Dr. Bernhard Schätz
Fakultät für Informatik, TU München
SS 2007

Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering

1




Inhalt

- 1) Einführung
- 2) Beschreibung von SW-Architekturen
- 3) Formale Modellierung
- 4) Entwurf von SW-Architekturen
- 5) Wiederverwendung
- 6) Architektur-Evolution

Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering

2



Definitionen

Einführung
 Beschreibung
 Formale Modellierung
 Entwurf
 Wiederverwendung
 Evolution

„Architecture is defined [...] as

- the **fundamental organization** of a system,
- embodied in its **components**,
- their **relationships** to each other and the environment, and
- the **principles governing** its design and evolution.“

(IEEE-Standard 1471-2000)


„Architecture is a **framework** for the disciplined introduction of **change**.“

(Tom de Marco)

→ weitere Definitionen unter: <http://www.sei.cmu.edu/architecture/definitions.html>

Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering

3



Ausprägungen von Architekturen

- Systemarchitektur
- Hardware-Architektur
- Software-Architektur

Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering

4

Bedeutung von SW-Architekturen

Einführung

Wiederverwendung

Evolution

- „erste Entwurfsentscheidung“
- Brücke zwischen Anforderungsmanagement und Entwicklung
- Grundlage für Erfüllung von Qualitätsanforderungen
- Kommunikationsmedium für Kunden, Projektleiter, Entwickler, Tester, Benutzer, ...
- Grundlage für Planung, Management und Ausführung der Systementwicklungsaktivitäten
- Dokumentation
- Grundlage für Wiederverwendung

Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering

5

Perspektiven

Einführung

Wiederverwendung

Evolution

- **Fachlich**
 - Zerlegung (Dekomposition) des Systems in **fachliche Komponenten** (z.B. in Vertragsverwaltung, Kundenverwaltung ...)
 - Festlegung der fachlichen **Außenschnittstellen**
- **Technisch**
 - Festlegung der technischen **Infrastruktur** (GUI-Toolkit, Datenbanksystem, App. Server, Frameworks, Hardware, ...)
 - Festlegung der **Grobstruktur** des Systems (Schichten, Tiers, Referenz-Architektur, ...)
 - Festlegung der **Entwicklungsumgebung** (SEU) (Programmiersprache, Umgebung, Build-Tool, Bugtracker, ...)
- **Organisatorisch**
 - Festlegung der Arbeitspakete, **Arbeitspaketstruktur** und Teamstruktur zur Entwicklung des Systems

Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering

6

Beispiel 1 (Pipes&Filters)

Einführung

Wiederverwendung

Evolution

Architektur eines Übersetzers:

Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering

7

Beispiel 2 (Schichten)

Einführung

Wiederverwendung

Evolution

ISO – OSI-Referenzmodell (ISO/IEC 7498):

Layer	Protocols
Anwendung	FTP, TelNet
Darstellung	ASN.1, XDR
Kommunikation	RPC
Transport	TCP, UDP
Vermittlung	IP, X.25
Sicherung	CSMA/CD
Bitübertragung	X.21, RS232

Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering

8

Beispiel 3 (Schichten)

AUTOSAR (AUTomotive Open System ARchitecture):

Quelle: www.autosar.org, Overview of Software Layers (Component View)

Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering

Architekturbeschreibung

- Wie Architektur beschreiben?
- Welche Modelle?

Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering

Beschreibungssprachen

- Grafische Beschreibungssprachen, wie
 - Box-And-Arrow – Diagramme (ad hoc Notationen)
 - UML 2.0 (neue Diagrammtypen, verbesserte Notation für Komponenten, XML)
 - Klassendiagramme
 - UML-Architektur-Profile (=Architektursprachen auf der Basis der UML)
- Architecture Description Languages (ADL)
 - ACME, Rapide, UNICON, Wright, ... (haben sich nicht durchgesetzt)
- Programmiersprachen
 - Java, C++, (CORBA-IDL), ...
- Formale Spezifikationstechniken
 - Focus, B-Method, (Petri-Netze), ...

Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering

Informelle Beschreibung - Beispiele

Informell:
Was bedeutet ein Rechteck?
Syntax der Beschreibungssprache?

Semi-Formal:
Syntax der Beschreibungssprache.
Semantik? Code-Generierung?

Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering

Systemmodell

System besteht aus

- benannten Komponenten (mit verkapselten Zuständen)
- benannten Kanälen

mit diskretem globalem Zeitmodell

TUM TECHNISCHE UNIVERSITÄT MÜNCHEN Fakultät für Informatik Lehrstuhl IV: Software & Systems Engineering 13

Modellierung der Kanäle: Ströme

Terminologie

- Ein Kanal verbindet zwei Teilsysteme oder ein System mit seiner Umgebung (Ein- oder Ausgabekanal)
- Ein Strom modelliert die Kommunikationsgeschichte auf einem Kanal

Da in zusammengesetzten Systemen Ströme durch rekursive Gleichungen definiert werden, brauchen wir eine Theorie der Ströme, die rekursive Gleichungen unterstützt.

TUM TECHNISCHE UNIVERSITÄT MÜNCHEN Fakultät für Informatik Lehrstuhl IV: Software & Systems Engineering 14

Ströme

Sei M eine Menge von Nachrichten

M^* Menge der endliche Sequenzen über M

$\langle \rangle$ leere Sequenz

M^ω Menge der unendlichen Sequenzen über M

$\text{IN} \setminus \{0\} \rightarrow M$

M^ω Menge der Ströme

$M^\omega = M^* \cup M^\omega$

$(M^*)^\omega$ Menge der gezeiteten Ströme - eine Sequenz von Nachrichten pro Zeitintervall

TUM TECHNISCHE UNIVERSITÄT MÜNCHEN Fakultät für Informatik Lehrstuhl IV: Software & Systems Engineering 15

Gezeitete Ströme

Gezeitete Ströme: Semantisches Modell für Schnittstellenverhalten

Nachrichtensmenge:
 $M = \{a, b, c, \dots\}$

Nachrichten in Zeiteintervall t

Unendliche Kanalgeschichte

TUM TECHNISCHE UNIVERSITÄT MÜNCHEN Fakultät für Informatik Lehrstuhl IV: Software & Systems Engineering 16

Schnittstellen-Modell

Verhaltensmodellierung

Formale Modellierung

Entwurf

Wiederverwendung

Evolution

$I = \{x_1, x_2, \dots\}$ Menge der Eingabekanäle

$O = \{y_1, y_2, \dots\}$ Menge der Ausgabekanäle

Schnittstellenverhalten

$F : \vec{I} \rightarrow \mathcal{P}(\vec{O})$

Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering

17

Komponentenschnittstellen und Kausalität

Verhaltensmodellierung

Formale Modellierung

Entwurf

Wiederverwendung

Evolution

Schnittstellenverhalten $F : \vec{I} \rightarrow \mathcal{P}(\vec{O})$

Kausalität

$x, z \in \vec{I}, y \in \vec{O}, t \in \mathbb{N}$

$x \downarrow_t = z \downarrow_t \Rightarrow \{y \downarrow_{t+1} : y \in F(x)\} = \{y \downarrow_{t+1} : y \in F(z)\}$

Eine kausale Komponente F ist total, d.h. $F.x \neq \emptyset$ für alle x, oder $F.x = \emptyset$ für alle x

Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering

18

Komposition / Dekomposition von Systemen

S1

S2

S

Komposition

Abstraktion

Kompositionalität:

$S = S1 \otimes S2$

$\alpha : \text{System} \rightarrow \text{Verhalten}$

$\alpha(S) = \alpha(S1) \otimes \alpha(S2)$

Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering

19

Beispiel: Arithmetisches Mittel

Verhaltensmodellierung

Formale Modellierung

Entwurf

Wiederverwendung

Evolution

MeanGraph

$i_1 : \mathbb{N}$

$i_2 : \mathbb{N}$

Mean

$o : \mathbb{N}$

Name der Spezifikation

Mean

in $i_1, i_2 : \mathbb{N}$ ← Eingabekanäle

out $o : \mathbb{N}$ ← Ausgabekanal

$\#o = \min\{\#i_1, \#i_2\} \wedge \forall j \in \text{dom.} o : o.j = \frac{i_1.j + i_2.j}{2}$

Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering

20

Beispiel

Formale Modellierung

MeanCompGraph

Entwurf

Add

in $i_1, i_2 : \mathbb{N}$
out $c : \mathbb{N}$

$\#c = \min\{\#i_1, \#i_2\} \wedge \forall j \in \text{dom}.c : o.j = i_1.j + i_2.j$

DivBy2

in $c : \mathbb{N}$
out $o : \mathbb{N}$

$\#o = \#c \wedge \forall l \in \text{dom}.o : o.l = \frac{c.l}{2}$

Wieder-Verwendung

untimed

Evolution

untimed

TUM TECHNISCHE UNIVERSITÄT MÜNCHEN
Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering
21

Beispiel

Formale Modellierung

MeanComp

in $i_1, i_2 : \mathbb{N}$
out $o : \mathbb{N}$
loc $c : \mathbb{N}$

$(c) := \text{Add}(i_1, i_2) \quad (o) := \text{DivBy2}(c)$

Entwurf

MeanComp

in $i_1, i_2 : \mathbb{N}$
out $o : \mathbb{N}$
loc $c : \mathbb{N}$

$\#c = \min\{\#i_1, \#i_2\} \wedge \forall j \in \text{dom}.c : o.j = i_1.j + i_2.j$ **Konjunktion (Komposition)**
 $\#o = \#c \wedge \forall l \in \text{dom}.o : o.l = \frac{c.l}{2}$

Wieder-Verwendung

glass-box

Evolution

glass-box

TUM TECHNISCHE UNIVERSITÄT MÜNCHEN
Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering
22

Komposition von Spezifikationen

Formale Modellierung

Composed components spec

in $x_1 : M_1, x_2 : M_2, \dots$
out $y_1 : N_1, y_2 : N_2, \dots$

$\exists c_1, c_2, \dots : P_1 \wedge \dots \wedge P_n$

Entwurf

Composed Component

Wieder-Verwendung

Systemkomposition = logisches UND
Channel Hiding = Existenzquantifikation

Evolution

TUM TECHNISCHE UNIVERSITÄT MÜNCHEN
Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering
23

Komponenten-Hierarchie

Formale Modellierung

Entwurf

Wieder-Verwendung

Evolution

TUM TECHNISCHE UNIVERSITÄT MÜNCHEN
Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering
24

Entwurfsziele

Anforderung
 Anforderungsänderung
 Problem-Modifizierung
 Entwurf
 Wiederverwendung
 Evolution


Allgemeine Ziele:

- Angemessenes Kosten / Nutzenverhältnis
- Angemessene Modifizierbarkeit, Wartbarkeit
- Entwickelbarkeit, Verständlichkeit, Testbarkeit

Erfüllung nicht-funktionaler Anforderungen, die sich auf den Benutzer auswirken z.B.

- Verfügbarkeit (Zeiten verkürzen, in denen das System nicht verwendbar ist)
- Performance und Effizienz (kurze Antwortzeiten, hoher Durchsatz, wenig Ressourcen Verbrauch)
- Sicherheit (Schutz gegen nicht autorisierten Zugriff, ...)
- (Benutzbarkeit)


Vgl. Kapitel über Qualitätsmodelle bzw. Qualitätssicherung


Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering
25

Prinzipien für Architekturentwurf

Anforderung
 Anforderungsänderung
 Problem-Modifizierung
 Entwurf
 Wiederverwendung
 Evolution


- Abstraktion
- Sichten/Perspektiven
- Modularisierung („Divide et Impera“)
- Kapselung („Information Hiding“)
- Hierarchische Dekomposition
- „Separation of Concerns“
- Einheitlichkeit (keine „Speziallösungen“)


Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering
26

Identifikation von Komponenten

Anforderung
 Anforderungsänderung
 Problem-Modifizierung
 Entwurf
 Wiederverwendung
 Evolution


- fachliche Gesichtspunkte
 - Wähle fachliche Konzepte, die in der Realität unabhängig voneinander sind (Konto vs. Vertrag), erzeuge daraus Komponenten
 - Wähle fachliche Konzepte, die möglichst stabil sind, erzeuge daraus Komponenten
 - Schneide Komponenten nach unabhängiger Nutzbarkeit (nach Mehrwert für den Kunden)
 - Schneide Komponenten nach Grenzen von Geschäftsprozessen oder nach den Abteilungsgrenzen (z.B. für jede Abteilung eine Komponente, z.B. Rechnungswesen vs. Inkasso)
- Kommunikationsorientiert
 - Pro externer Schnittstelle eine Komponente
 - Pro Benutzersorte eine Komponente
 - Pro externem Ereignis eine Komponente
 - ..
- Organisatorische Gesichtspunkte
- ...


Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering
27

Wiederverwendung 1/2


Anforderung
 Anforderungsänderung
 Problem-Modifizierung
 Entwurf
 Wiederverwendung
 Evolution

- Guidelines
 - Empfehlungen, die auf Erfahrung beruhen
 - oft firmenspezifisch, nicht öffentlich
- Architekturstile/Architekturmuster (Pipes and Filters, Client-Server, ...)
 - Prinzipielle Lösungsstrukturen, die für ein Element einer Architektur durchgängig und mit weitgehenden Verzicht auf Ausnahmen angewandt werden.
 - Allgemeine, bewährte Lösung für häufig wiederkehrende Probleme
- Framework (z.B. Java Swing)
 - Definieren die SW-Architektur in ihrem Anwendungsbereich
 - „Halbfertiges“ Softwaresystem, das noch vervollständigt werden muss


Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering
28


Wiederverwendung 2/2

- Referenzarchitekturen (z.B. AUTOSAR, Quasar (sd&m AG))
 - Abstrakte SW-Architektur, die Strukturen und Typen von SW-Elementen sowie deren erlaubte Interaktionen und ihre Verantwortlichkeiten speziell für einen Anwendungsbereich definiert.
 - Strukturen sind jeweils für alle Systeme innerhalb einer Domäne anwendbar
- Produktlinien
 - Produktfamilien
 - Geplante Wiederverwendung unterschiedlicher Software-Artefakte zur Erstellung eines verwandten Software-Produkts


Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering
29


Überprüfung und Evolution

- Überprüfung
 - Software-Systeme neigen dazu Architekturregeln zu verletzen
 - Unwissen der Entwickler
 - Termindruck
 - mangelnde Unterstützung der Programmiersprachen
 - kontinuierliche Überprüfung ist notwendig
 - Reviews
 - statische Analyse-Tools
- Evolution
 - auch die beste Architektur ist nicht für die Ewigkeit
 - strukturierter Architektur-Evolutions-Prozess ist notwendig


Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering
30

Literatur

- [BCK03] Bass, Clements, Kazman: Software Architecture in Practice, 2nd Edition, Addison-Wesley, 2003
- [BRO01] Broy, Stolen: Specification and Development of Interactive Systems, Springer, 2001
- [CBB+03] Clements, Bachmann, Bass, et al. : Documenting Software Architectures: Views and Beyond, Addison-Wesley, 2003
- [CKM02] Clements, Kazman, Klein: Evaluating Software Architectures, Methods and Case Studies, Addison-Wesley, 2002
- [HNS99] Hofmeister, Nord, Soni: Applied Software-Architecture, Addison-Wesley, 1999
- [HS00] Herzum, Sims: Business Component Factory: A Comprehensive Overview of Component-Based Development for the Enterprise, Wiley, 2000
- [PBG04] Posch, Birken, Gerdorn: Basiswissen Software-Architektur, Dpunkt Verlag, 2004
- [RH06] Reussner, Hasselbring: Handbuch der Software-Architektur, Dpunkt Verlag, 2006
- [SGM02] Szyperski, Gruntz, Murer: Component Software, Beyond Object-Oriented Programming, 2nd Edition Addison-Wesley, 2002
- [Sid04] Siedersleben, Moderne Software-Architektur - Umsichtig planen, robust bauen mit Quasar, Dpunkt Verlag, 2004
- [Sta02] Starke, Effektive Software-Architekturen, Hanser, 2002


Fakultät für Informatik
Lehrstuhl IV: Software & Systems Engineering
31