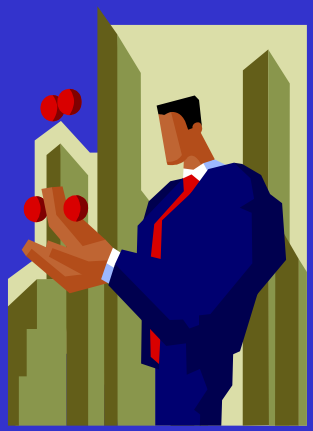


Softwarearchitektur - ein kurzer Überblick

28.05.2004

Maik G. Seewald

Softwarearchitektur – Ein kurzer Überblick
Agenda

1. Begriffsdefinition
2. Stellenwert in Unternehmen und Projekten
3. Rolle der Softwarearchitektur
4. Notation und Darstellung
5. Architekturdefinition und Konzepte
6. Die 3-Schichten-Architektur als Beispiel
7. Zusammenfassung

Softwarearchitektur
1. Ziel der Präsentation und Begriffsdefinition

Ziel: Ein kurzer Überblick zur Thematik Softwarearchitektur (SWA), einem kritischen Erfolgsfaktor für Softwareprojekte

Herausforderung: Umfassende, sich ständig weiter entwickelnde Thematik

- Keine allgemeingültige Definition existent
- Viele Definitionen aus unterschiedlichen Quellen (UML, IEEE, …)

"The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the external properties of those components, and the relationship among them." [Bass, Clements, Kazman 1998]

UML 1.3: Architecture is the organizational structure of a system. An architecture can be recursively decomposed into parts that interact through interfaces, relationships that connect parts, and constraints for assembling parts. Parts that interact through interfaces include classes, components and subsystems.

Die Softwarearchitektur ist das Bindeglied zwischen den Anforderungen der Fachdomäne und der technischen Realisierung eines Systems. Aus dem Anforderungsmanagement heraus kann eine Softwarearchitektur abgeleitet und erarbeitet werden. Hierbei wird die Struktur des Softwaresystems definiert und erste technologische Entscheidungen getroffen.

Softwarearchitektur
2. Stellenwert in Unternehmen und Projekten

1. Motivation - Organisatorische Vorteile:	▪ SWA bündelt unternehmensweites Know-How (Modelle, Dokumentationen, etc.) ▪ SWA ermöglicht Wiederverwertbarkeit, Reduziert Kosten für Weiterentwicklung und Wartung (ROI) ▪ SWA dokumentiert Systeme und erzeugt Transparenz ▪ SWA ist Grundlage für die Definition von Arbeitspaketen und unterstützt damit die Projektplanung
2. Motivation - Technische Vorteile:	▪ Beherrschbarkeit von Komplexität ▪ SWA ist Basis für Wiederverwertbarkeit, Flexibilität und Erweiterbarkeit
3. Externe Einflüsse:	▪ Projektbeteiligte (Alle Stakeholder) ▪ Unternehmensstrategie, Bestehende Projekte, Standards und Frameworks ▪ Maximierung des Kundennutzens ▪ Organisatorische Faktoren (Budget, Zeitrahmen) ▪ Bestehendes Know-How, Lernkurve

Softwarearchitektur
3. Rolle der Softwarearchitektur

Die Softwarearchitektur hat Schnittstellen zu folgenden Prozessen innerhalb von Softwareprojekten: Anforderungsmanagement, Design, Implementierung, Projektplanung, Qualitätssicherung und Hardwarearchitektur.

Schlüsselschnittstelle zur Anforderungsanalyse:

- Fachliche und technische Anforderungen als Resultat der Anforderungsanalyse
- Berücksichtigung funktionaler und nicht-funktionaler Anforderungen (Qualität, FURPS)
- Rückkopplung hinsichtlich Machbarkeit und Kosten

Allgemeines Vorgehen:

- Kommunikation und Abstimmung mit oben genannten Stakeholdern
- SWA ist ein zentrales Bestandteil des Entwicklungsprozesses mit z.T. fest definierten Rollen und Artefakten (RUP, Agile Dev./XP, V-Modell)
- Typische Schritte innerhalb eines iterativen, inkrementellen Vorgehens der Architekturerstellung: Machbarkeitsstudie, Risikoanalyse, Priorisierung der Anforderungen, Architekturkonzept, Architekturbewertung

Grundlagen: Modelle, Heuristiken, Muster (Pattern, Idiome, “Best Practices”)

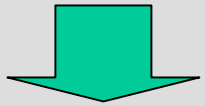
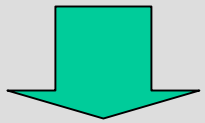
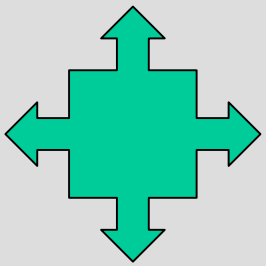
Softwarearchitektur
4. Notation und Darstellung

Motivation: Notationsform, mit höherem Abstraktionsgrad als Sourcecode um effektive Kommunikationsstrukturen im Team zu gewährleisten (Artefakte: Graphische Darstellungen, Dokumente)

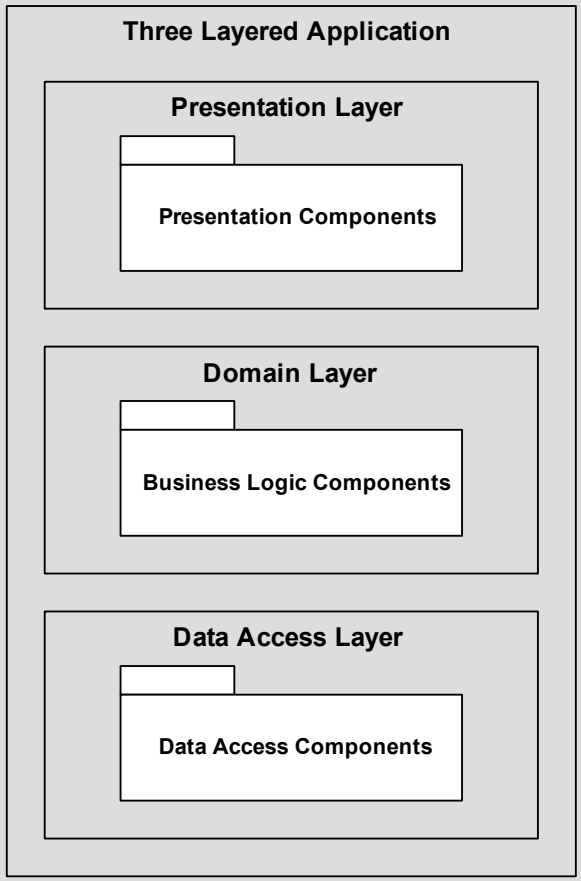
Allgemeine Sichtweisen:

- UML als standardisierte Notationsform für Software mit dem Ziel der Bereitstellung von unterschiedlichen Sichten auf das System (Anforderungen, Systemkontext, Systemdesign, Infrastruktur, Verteilung, Informationsfluss, …)
- UML beschreibt statische und dynamische Elemente mit Hilfe von Klassen, Komponenten, Pakete, Aktionen, Interaktionen, Use Cases und
  - Statischen Diagrammen (Klassen-, Komponenten-, Verteilungs-, Strukturdiagramme, ...) zur Beschreibung der Struktur
  - Dynamischen Diagrammen (Aktivitäts-, Sequenz-, Kommunikations-, Use-Case-Diagrammen, …) zur Beschreibung des Verhaltens

Softwarearchitektur
5. Architekturdefinition und Konzepte

Kriterien und Dimensionen: 	▪ Datenorientierte o. Ereignisorientierte Systeme ▪ Art der Anwendung (Embedded, Hostbasiert, C/S, Verteilt, …) ▪ Technische Rahmenbedingungen (OS, Hardware, Datenhaltung, Infrastruktur, Netzwerk, Security, Transaktionssicherheit, MMI, …)
Grundlagen der Entscheidungen: 	▪ Designmuster u. Architekturstile (Pattern, ”Best Practices”) z.B.: Layered Application, Façade Pattern, Strukturierung durch “Separation of Concerns”), Deployment Patterns ▪ Generierungstechniken (MDA), Wiederverwendbarkeit ▪ Andere Einflüsse (siehe Folie 2: Know How, Standards)
Resultate: 	▪ Entwurf und Unterteilung in Schichten, Komponenten ▪ Deployment; Aufteilung des Systems auf logische/physische Plattformen (C/S- Architektur, Fat-, Thin-Client) ▪ Technologieentscheidungen (OS, Tools, RAD, Programmiersprachen, Komponentenmodelle, Application Server, Plattformen [J2EE, .NET]) ▪ Make or Buy Entscheidung, 3rd-Party Software ▪ Teamstruktur (Distributed Development)

Softwarearchitektur
6. Die 3-Schichten-Architektur als Beispiel



Beispiel:

- 3 Schichten Architektur einer typischen web-basierten Anwendung (Darstellung in UML)
- Schichten werden genutzt um die Applikation zu organisieren und die Komponenten anzuordnen
- Vorteile:
  - Skalierbarkeit
  - Verteilbarkeit
  - Wiederverwertbarkeit der Business Logik
  - Robustheit
  - Wartbarkeit
- Andere Verteilung möglich, siehe Folie 12

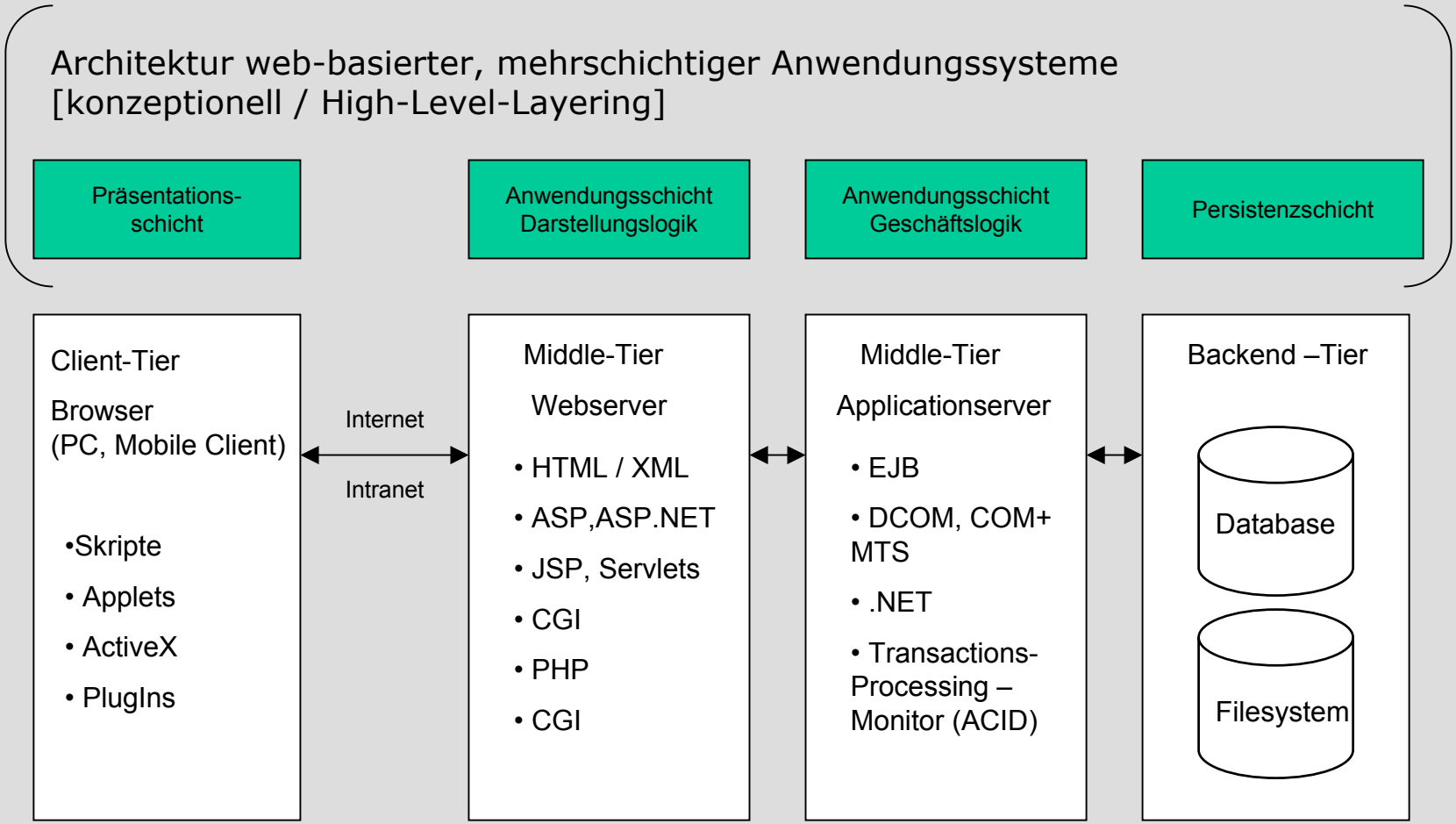
Softwarearchitektur
7. Zusammenfassung

1. Die Softwarearchitektur wird durch fachliche und technische Rahmenbedingungen sowie durch politische und organisatorische Einflüsse auf das Projekt bestimmt.
2. Die Entscheidungsfindung hin zur Softwarearchitektur ist ein inkrementeller, iterativer Prozess und basiert auf einer ständigen Rückkopplung mit allen Stakeholdern mit dem Ziel einer Kompromissfindung.
3. Die Softwarearchitektur ist aufgrund der immer größer werdenden Komplexität ein kritischer Erfolgsfaktor in Softwareprojekten.
4. Die Verwendung von Solution Patterns / „Best Practices" hat sich im Prozess der Definition der Softwarearchitektur als eine effektive Herangehensweise etabliert (siehe Folie 15).

Softwarearchitektur
Backup 01: Die Rolle des Softwarearchitekten

- RUP: Ein SW Architekt ist die treibende Kraft bei der Systemerstellung. Die wesentliche Aufgabe ist es, eine Folge von suboptimalen Entscheidungen bei unvollständigem Wissen zu treffen. Architekturverantwortung ist Vollzeitbeschäftigung.
- Der SW Architekt sollte eine starke Persönlichkeit mit breiter technischer Erfahrung und ausgeprägten Soft-Skills wie Abstraktionsvermögen, Entscheidungs-, Durchsetzungs-, Team- und Überzeugungsfähigkeit sein.

Softwarearchitektur
Backup 02: Web Based Computing Layering



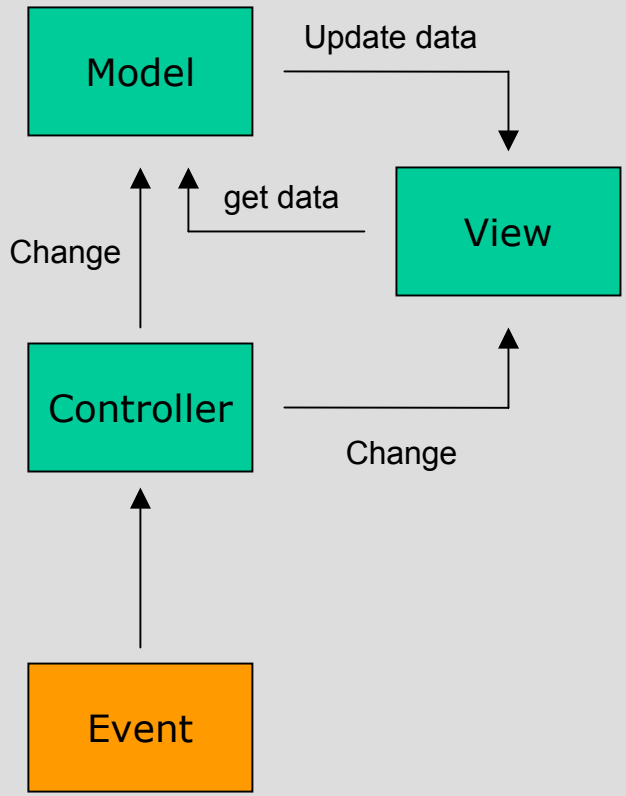
Softwarearchitektur
Backup 03: Java Design Patterns

Pattern Name	Description
Business Delegate	Reduce coupling between Web and Enterprise JavaBeans™ tiers
Composite Entity	Model a network of related business entities
Composite View	Separately manage layout and content of multiple composed views
Data Access Object	Abstract and encapsulate data access mechanisms
Fast Lane Reader	Improve read performance of tabular data
Front Controller	Centralize application request processing
Intercepting Filter	Pre- and post-process application requests
Model-View-Controller	Decouple data representation, application behaviour, and presentation
Service Locator	Simplify client access to enterprise business services
Session Facade	Coordinate operations between multiple business objects in a workflow
Transfer Object	Transfer business data between tiers
Value List Handler	Efficiently iterate a virtual list
View Helper	Simplify access to model state and data access logic

Anmerkung: Hierbei handelt es sich um Design Pattern, nicht um ein Architektur Pattern (also somit um einen anderen Abstraktionsgrad).

Quelle: Sun Microsystems

Softwarearchitektur
Backup 04: MVC Pattern



	Java / J2EE	MS .NET
M	Database Entity Bean	Database ADO.NET
V	Servlet JSP	ASP.NET
C	Stateless Session Bean Stateful Session Bean	COM, COM+

Anmerkung: Hierbei handelt es sich um ein Design Pattern, nicht um ein Architektur Pattern (also somit um einen anderen Abstraktionsgrad).

Softwarearchitektur
Backup 05: Literatur und Links

- Microsoft: Pattern and Practices / .NET oriented:

http://www.microsoft.com/resources/practices/

- Java / J2EE Patterns:

http://java.sun.com/blueprints/patterns

http://www.corej2eepatterns.com/index.htm

http://www.theserverside.com/patterns

- Excellente Bücher von J. Bloch, E.Gamma, Buschmann und anderen

  → "*Each Pattern then depends both on the smaller patterns it contains, and on the larger patterns within which it is contained"* , Christopher Alexander in *The Timeless Way of Building*