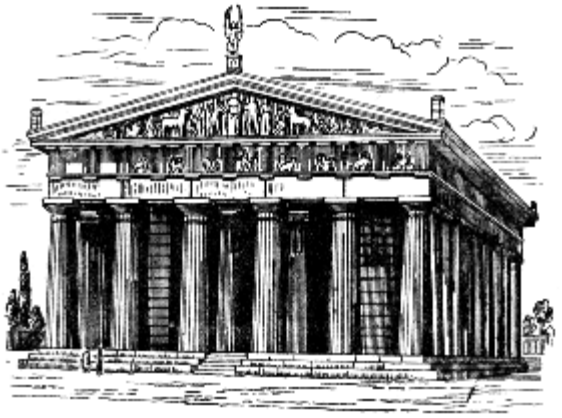
Softwarearchitektur

(Architektur: *αρχή* = Anfang, Ursprung + *tectum* = Haus, Dach)

1. Einleitung und Überblick



Vorlesung Wintersemester 2010 / 2011

Technische Universität München
Institut für Informatik
Lehrstuhl von Prof. Dr. Manfred Broy

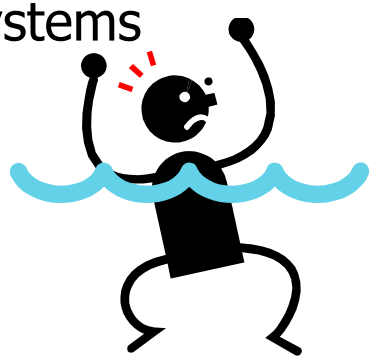
Dr. Klaus Bergner, Prof. Dr. Manfred Broy, Dr. Marc Sihling

Inhalt

- Einleitung
  - Motivation
  - Organisatorisches zur Vorlesung
- Softwarearchitektur als Designergebnis
  - Die Rolle der Softwarearchitektur für das System
  - Beispiel
  - Begriffsdefinition
- Softwarearchitektur als Tätigkeit im Projekt
  - Die Rolle des Softwarearchitekten im Projekt
  - Architekturanalyse und –entwurf
  - Architekturmodelle

Problemstellung

- Beobachtungen:
  - Der Alltag ist immer stärker durchdrungen von Software
  - Software bestimmt immer mehr den Geschäftserfolg
  - Die Größe von Systemen wächst kontinuierlich (siehe beispielsweise Toll-Collect, FISCUS, etc.)
  - Unsere Fähigkeit, Softwaresysteme zu entwickeln wächst langsamer als der Bedarf (sog. „Software-Krise“)
  - Nur etwa 25% aller Entwicklungsprojekte verlaufen erfolgreich (hinsichtlich Zeit, Qualität, Kosten)
  - Bis zu ca. 80% der Gesamtkosten eines Softwaresystems fallen nach Fertigstellung an



Ansatz: Software-Engineering

- Softwareentwicklung wird aufgefasst als Ingenieursdisziplin, mit Methoden, Techniken und Werkzeugen, die teilweise von traditionellen Disziplinen (z.B. Maschinenbau oder Architektur) „abgeschaut" werden.
- Komplexität wird begegnet mit dem Handwerkszeug des Ingenieurs
  - Teile und Herrsche: ein Problem wird in kleinere Häppchen zerlegt, die versteh- und handhabbar sind (-> Modul, Komponente)
  - Abstraktion: die für die aktuelle Zielsetzung irrelevanten Details werden ausgeblendet (-> Schnittstelle)
  - Wiederverwendung: die Situation wird mit verstandenen Problemen korreliert, für die Lösungen bekannt sind
- Softwarearchitektur ist eine Teildisziplin des Software-Engineerings, für die sich überraschend viele Analogien zur klassischen Architektur finden lassen

Architektur-Beispiel: das Münchner Olympiagelände

- Gewinner des damaligen Architekturwettbewerbs: G. Behnisch
- Gebaut in den Jahren 1966-1972
- Charakteristisches Zeltdach, gestützt von 58 Pylonen
- Konzipiert als „offenes Amphitheater“
- Unter Denkmalschutz seit 1977

Wichtigste Anforderungen

- Langlebigkeit
  - Vielfältige Einsatzmöglichkeiten (7.400 Veranstaltungen seit ´72)
  - Dauerhaftigkeit der Strukturen
  - Geringe Wartungskosten (vgl. andere Städte!!!)
- Ästhetik
  - Imagetransport, auch über Olympia hinaus
  - Unterstützung der Funktion
- Berücksichtigung lokaler Gegebenheiten
  - Brachgelände mit angrenzendem Schuttberg
- Balance zwischen Qualität, Zeit und Kosten
  - Begrenztes Budget
  - Hoher Zeitdruck
- Und viele mehr….

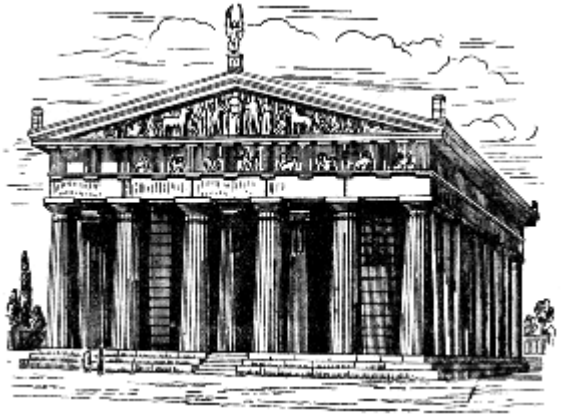
Leistung der Architekten

- Spektakuläre Ingenieursleistung
  - Dachkonstruktion nicht „berechenbar“
  - Einfache Simulationsmodelle
- Eleganz, Transparenz, Leichtigkeit
  - Organische Einbettung der Sportanlagen in die Umgebung
- „Blick über den Tellerrand“
  - Konzeption eines dauerhaften Naherholungsgebiets
  - Vielfältige Einsatzmöglichkeiten (Studentenstadt, Konzerte und Fußballspiele im Stadion, etc.)

Die Disziplin „Architektur“

»Architektur ist die Kombination von *utilitas*, *firmitas* und *venustas*.« frei nach Vitruvius (Römischer Architekt 90-20 v. Chr.)

- ***utilitas*** : das Gebäude erfüllt seine Funktion
- **firmitas** : das Gebäude ist stabil

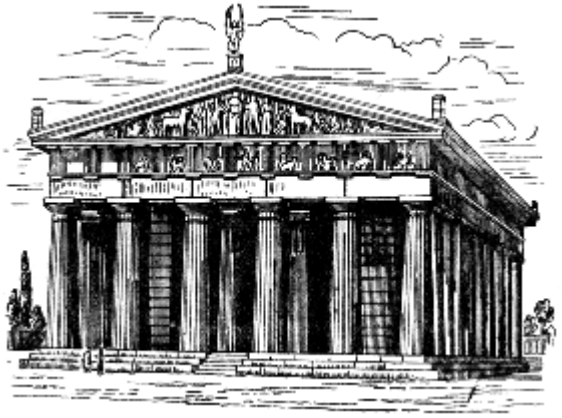


- **venustas** : das Gebäude ist ästhetisch gestaltet

Die Disziplin „Architektur“

»Architektur ist die Kombination von *utilitas*, *firmitas* und *venustas*.« frei nach Vitruvius (Römischer Architekt 90-20 v. Chr.)

- ***utilitas*** : das Gebäude erfüllt seine Funktion
  (das Softwaresystem erfüllt seine Anforderungen)
- **firmitas** : das Gebäude ist stabil
  (das Softwaresystem ist langlebig und „stabil“ gegenüber Änderungen)
- **venustas** : das Gebäude ist ästhetisch gestaltet
  (das Softwaresystem weist klare, logische Strukturen auf, die das Verständnis vereinfachen)



Organisatorisches zur Vorlesung „Softwarearchitektur“

- Ort: PH HS LMU, Raum 5109
- Zeit: Dienstags von 12:00 – 14:00 Uhr
- Website: http://www4.in.tum.de/lehre/vorlesungen/sw_arch/WS1011/
- Mailverteiler: savorles@in.tum.de
- Büro: FMI 01.11.044 (Sekretariat Broy)

Überblick über die Vorlesung

- 19.10.10: Einleitung und Überblick
- 26.10.10: Grundlagen der Softwarearchitektur
- 02.11.10: Beschreibungstechniken
- 09.11.10: Sichten (1)
- 16.11.10: Sichten (2)
- 23.11.10: Ausprägungen (1)
- 30.11.10: Ausprägungen (2)
- 07.12.10: Betriebliche Informationssysteme (1)
- 14.12.10: Betriebliche Informationssysteme (2)
- 11.01.11: Betriebliche Informationssysteme (3)
- 18.01.11: Eingebettete Systeme (1)
- 25.01.11: Eingebettete Systeme (2)
- 01.02.11: Eingeladener Industrievortrag

- 18.02.11: Prüfung

Zielgruppe der Vorlesung

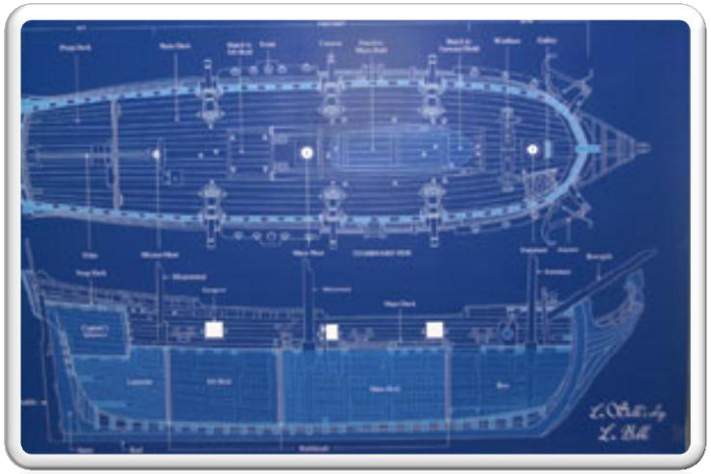
- Hörerkreis:
  - 5./6. Semester Bachelor
  - 1./2. Semester Master
- Nützlich:
  - Erfahrungen bei der Programmierung größerer Systeme (nicht nur mit der Technik – auch mit dem sozialen Gefüge eines Teams)
  - Kenntnisse der UML

Inhalt

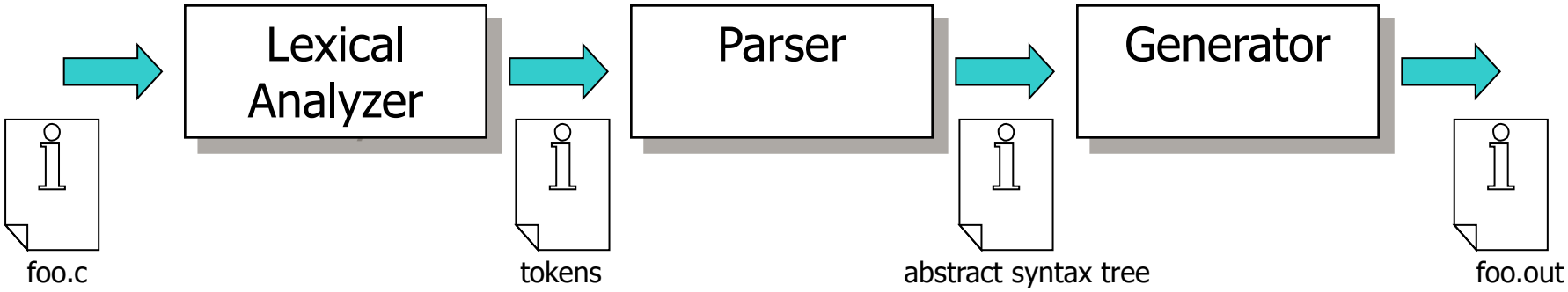
- Einleitung
  - Motivation
  - Organisatorisches zur Vorlesung
- Softwarearchitektur als Designergebnis
  - Die Rolle der Softwarearchitektur für das System
  - Beispiel
  - Begriffsdefinition
- Softwarearchitektur als Tätigkeit im Projekt
  - Die Rolle des Softwarearchitekten im Projekt
  - Architekturanalyse und –entwurf
  - Architekturmodelle

Bedeutung der Softwarearchitektur

- Die Architektur
  - ist Grundlage für das Verständnis über das System im Projektteam
  - definiert die Entwicklungsprozesse im Team und beeinflusst somit die Effizient im Projekt
- Die Qualität der Architektur
  - entscheidet über den Projekterfolg
  - entscheidet über die Qualität des Produkts
  - entscheidet über die Wartbarkeit und Erweiterbarkeit des Systems



Beispiel: Architektur eines Übersetzers



Wer heutzutage einen Übersetzer programmiert, profitiert von einem etablierten Verständnis der einzelnen Bestandteile und ihrer jeweiligen Aufgaben, Verantwortlichkeiten sowie Abhängigkeiten untereinander. Das realisierte System wird daher wohl ganz ähnlich zur obigen Abbildung aussehen…

Kernpunkte der Übersetzerarchitektur

- Einfach und verständlich
  - ein „Datenstrom“ durchläuft verschiedene „Stationen“ mit klar definierten Eingabe- und Ausgabeformaten
  - Funktionsprinzip kann schnell verstanden und kommuniziert werden
- Effizient realisierbar
  - Zerlegung erlaubt die parallele Entwicklung in mehreren Teams
  - Betriebssysteme wie Unix vereinfachen die Implementierung
- Langlebig
  - Das System kann sehr einfach um neue Funktionalität erweitert werden, beispielsweise um eine Station, die den abstrakten Syntaxbaum optimiert
- Skalierbar
  - Im Rahmen einer verteilten Umgebung kann die Anwendung leicht parallelisiert und mehrere Stationen redundant ausgelegt werden

Wege der Wiederverwendung der Architektur

- Als Referenz-Architektur für den Bau von Übersetzern: Festlegung der einzelnen Systemteile, ihrer Verantwortung und des gemeinsamen Zusammenspiels. Erweiterungsmöglichkeiten werden aufgezeigt.

- Als Architektur-Stil: allgemeines Strukturprinzip, bekannt als **pipes-and-filter**-Architektur. Datenströme werden über **pipes** von einem **filter** zum nächsten transportiert und dort weiterverarbeitet. Das funktioniert nur bei Anwendungen mit
  - sequentialisierbaren Bearbeitungsschritten
  - Abhängigkeiten nur zwischen benachbarten Bearbeitungsschritten

- Als Framework: für die Realisierung eines Übersetzers muss diese „halbfertige Implementierung" nur noch an wenigen Stellen angepasst werden.

(Noch) offene Fragen…

- Erfüllt die Architektur die Anforderungen an das System?
- Existieren bessere Architekturen für den Übersetzerbau?
- Wie ist der Zusammenhang zwischen bestimmten Merkmalen einer Architektur und bestimmten Anforderungen?
- Auf welche Weise wird die Architektur am besten dokumentiert und kommuniziert (mit Kästchen und Pfeilen?)
- Wie können besonders gute Eigenschaften der Übersetzerarchitektur auf andere Systeme übertragen werden?
- Welche Bestandteile des Übersetzers kann man wiederverwenden für die Realisierung weiterer Systeme? (Tatsächlich findet sich diese Architektur in vielen Übersetzern wieder.)

Softwarearchitektur als Prozess und Ergebnis

- Eine Softwarearchitektur
  - unterteilt das Anwendungssystem in Bestandteile mit definierten Verantwortlichkeiten und einem festgelegten Zusammenspiel
  - ermöglicht die Beurteilung der Qualität des Gesamtsystems anhand der Qualitäten der Einzelteile
  - definiert Schnittstellen für zukünftige Erweiterungen
- Eine **Softwarearchitektur** besteht aus einer Menge von Komponenten und ihren Beziehungen untereinander.
- Die Softwarearchitektur beeinflusst maßgeblich Entwicklungs- und Weiterentwicklungsprozesse.
- Die **Disziplin Softwarearchitektur** wird als Teil des Software-Engineerings verstanden und definiert Methoden, Techniken und Werkzeuge zur effizienten Entwicklung qualitativ hochwertiger Softwarearchitekturen.

Was ist eine Komponente?

- Eine **Komponente**
  - ist ein Baustein des Systems
  - kapselt ihr Inneres und kommuniziert mit anderen Komponenten über präzise definierte Schnittstellen
  - kann, muss aber nicht tatsächlichen Modulen der Implementierung entsprechen. Insbesondere kann man auch Komponenten definieren, ohne in der Implementierung ein entsprechendes Konzept hierfür zu haben.
- Beispiele für Komponenten
  - Enterprise Java-Bean
  - Terminal mit Touchscreen und Treibersoftware
- Alternative Definition
  - A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only.
    A software component can be deployed independently and is subject to composition by third parties

Weitere Definitionen

Es existieren über 90 Definitionen des Begriffs „Softwarearchitektur“ (siehe http://www.sei.cmu.edu). Zwei prominente Beispiele:

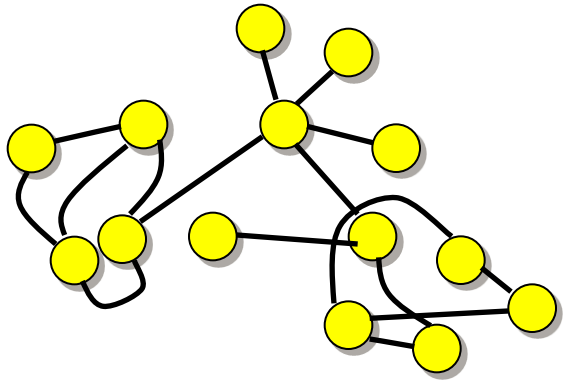
- „Architecture is defined [...] as the **fundamental organization** of a system, embodied in its **components**, their **relationships** to each other and to the environment, and the **principles governing its design and evolution**.“

  (IEEE Std. No. 1471-2000)

- „The **structure** of the **components** of a program/system their **interrelationships**, and principles and **guidelines governing their design and evolution** over time.“

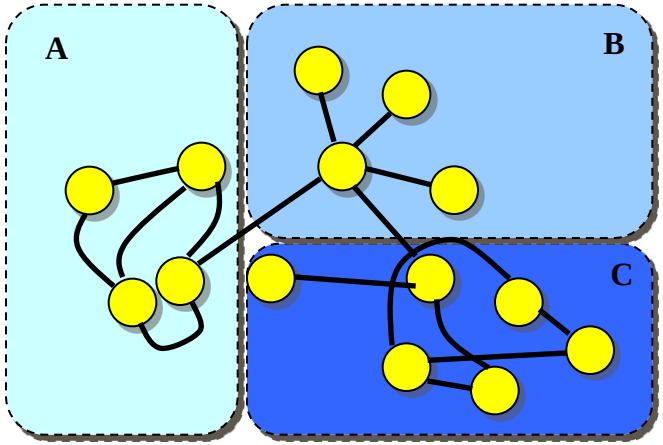
  (Garlan und Perry, 1995)

Ein Beispiel:
wie kommt man zur Softwarearchitektur?



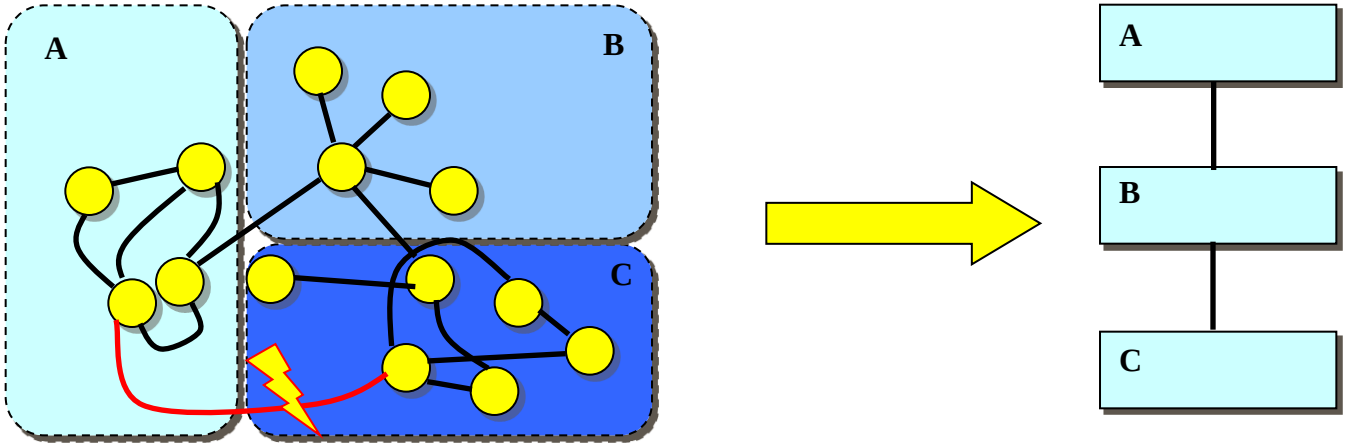
- Ein gängiger Weg, um die Komponenten einer Architektur zu ermitteln, besteht in der Betrachtung von Abhängigkeiten

Ein Beispiel:
wie kommt man zur Softwarearchitektur?



- Ein gängiger Weg, um die Komponenten einer Architektur zu ermitteln, besteht in der Betrachtung von Abhängigkeiten
- Komponenten „gruppieren" Elemente mit starker Abhängigkeit

Ein Beispiel:
wie kommt man zur Softwarearchitektur?



- Ein gängiger Weg, um die Komponenten einer Architektur zu ermitteln, besteht in der Betrachtung von Abhängigkeiten
- Komponenten „gruppieren" Elemente mit starker Abhängigkeit
- Abhängigkeiten zwischen Elementen werden zu Schnittstellen
- Ist die Architektur einmal festgelegt, gibt sie Richtlinien vor für die Einführung neuer Abhängigkeiten (hier: keine Verbindung zwischen den Komponenten A und C)

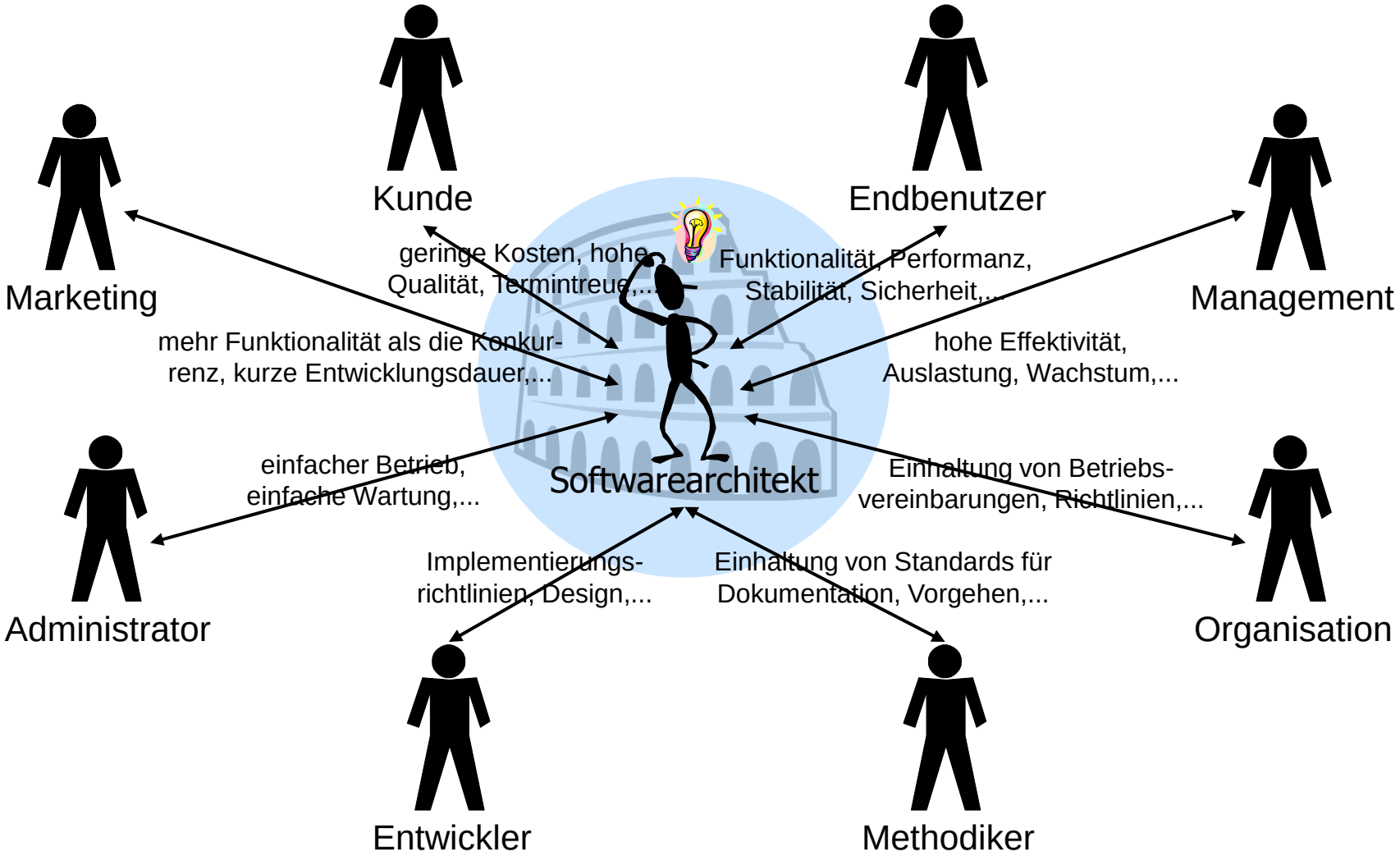
Inhalt

- Einleitung
  - Motivation
  - Organisatorisches zur Vorlesung
- Softwarearchitektur als Designergebnis
  - Die Rolle der Softwarearchitektur für das System
  - Beispiel
  - Begriffsdefinition
- Softwarearchitektur als Tätigkeit im Projekt
  - Die Rolle des Softwarearchitekten im Projekt
  - Architekturanalyse und –entwurf
  - Architekturmodelle

Die Rolle des Softwarearchitekten (nach V-Modell XT)

- Aufgaben und Befugnisse
  - Entwurf der System- bzw. Softwarearchitektur
  - Umsetzung der Anforderungen an die Komponenten
  - Verantwortlichkeit für Implementierungs-, Integrations- und Prüfkonzept
- Fähigkeitsprofil
  - Kennt Anwendung, Rahmenbedingungen und Einsatzgebiete des Systems
  - Kennt die Schnittstellen des Systems
  - Kennt Architekturprinzipien und verschiedene SW-Architekturen
  - Kennt Standard-Software, COTS, MOTS, etc.
  - Kennt Methoden und Werkzeuge
  - Hat Fähigkeit, Schwachstellen und Risiken zu erkennen
  - Hat Fähigkeit, Probleme unter adäquater Berücksichtigung der SW/HW zu analysieren und entsprechende Lösungsvorschläge auszuarbeiten
  - Hat Fähigkeit, zu abstrahieren und zu vereinfachen
  - Hat Fähigkeit, Abhängigkeiten zu erkennen
  - Hat Kommunikationsfähigkeit mit Anwendern und Team

Spannungsfeld Architekturentwurf

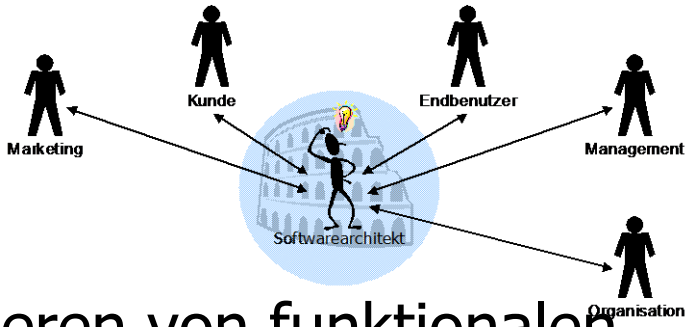


Ziel des Architekturentwurfs

- Eine Softwarearchitektur besteht aus einer Menge von Komponenten und deren Beziehungen untereinander.
- Ziel des Architekturentwurfs ist:
  - Den Bauplan des Systems zu erstellen, nach dem sich dann Erstellung, Wartung, Pflege und Weiterentwicklung ausrichten
  - Eine vollständige und prägnante Architekturbeschreibung erstellen
  - Dabei die geforderten funktionalen und nicht funktionalen Anforderungen gewährleisten
- Die Architekturbeschreibung dient als
  - Kommunikations- und Diskussionsplattform
  - Design- und Implementierungsplan

Architekturbeschreibung: Kommunikations- und Diskussionsplattform

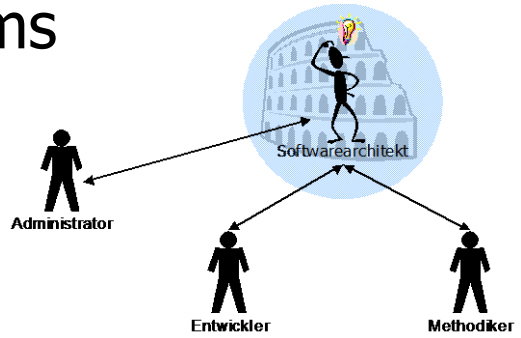
- Abstrakte Beschreibung des Systems
- Beherrschbarkeit der Komplexität



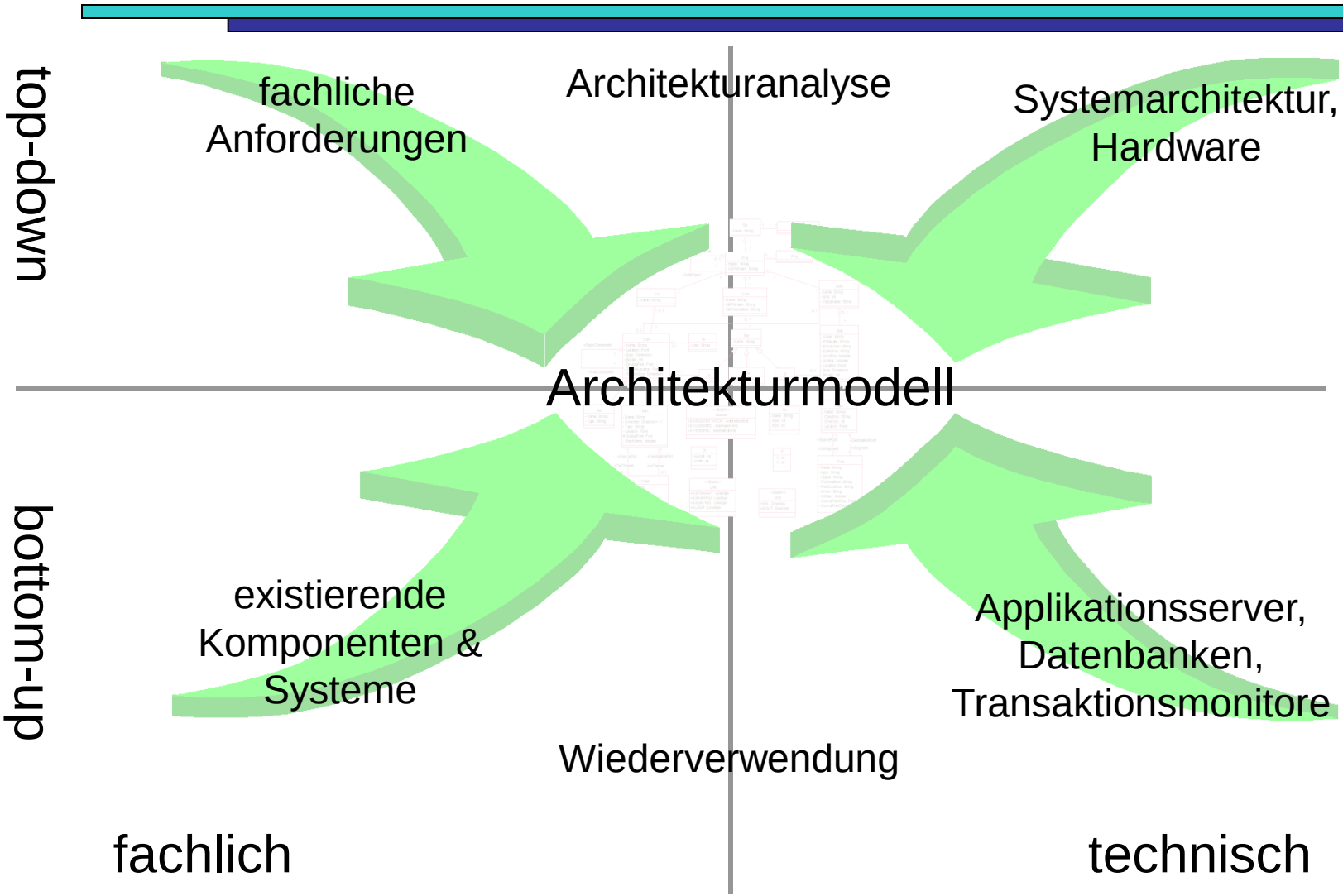
- Zuordnen, Priorisieren und Reflektieren von funktionalen und nicht funktionalen Anforderungen
- Integration von existierenden Lösungen und Systemen
- Abgleich mit den Anforderungen an Systemarchitektur und Hardware
- Erarbeiten, Evaluieren und Bewerten von alternativen Lösungsansätzen

Architekturbeschreibung: Design- und Implementierungsplan

- Festlegung der Komponenten des Systems sowie deren Schnittstellen und Interaktionsmuster
- Integration von neuen Technologien und innovativen Lösungsansätzen
- Erarbeitung, Einführung, Schulung und Überprüfung von Programmierrichtlinien
- Erstellung von Prototypen und exemplarischen Teillösungen
- Wiederverwendung von existierenden Implementierungen und Teilimplementierungen
- Rahmen für die Projektorganisation und -planung



Architekturmodellierung



Eingangsprodukte des Entwurfs: Anforderungen

- Anforderungskategorien
  - Funktionale Anforderungen: Verwaltung von Kunden, Verwaltung von Verträgen, etc.
  - Nicht funktionale Anforderungen
    - Qualitätsanforderungen (IEEE 1061): Benutzbarkeit, Stabilität, Wartbarkeit, Erweiterbarkeit, Performanz, etc.
    - Randbedingungen
      - System / Produkt: Vorgaben zur Verteilung, physikalische Anforderungen, Umweltanforderungen, Materialanforderungen, etc.
      - Technologie: Vorgaben zur Hardware, Vorgaben zur Software, Vorgaben zu Standardtechnologie, etc.
      - Prozess: Vorgaben zum Vorgehensmodell, Anforderungen bzgl. Einführung und Migration, etc.
      - Organisation / Management: Vorgaben bzgl. Zeit und Budget, Vorgaben bzgl. Personal, Vorgaben bzgl. Ressourcen

Ergebnisprodukte der Architekturanalyse

- Beschreibung der Architekturanforderungen
  - Identifizieren, Einordnen und Beschreiben der Anforderungen
  - Charakterisierung der Anforderungen bzgl. Flexibilität und Änderbarkeit
  - Auswirkungsanalyse von Anforderungsänderungen
- Beschreibung der Architekturtreiber
  - Identifizieren der zentralen Architekturtreiber und Zuordnen der Anforderungen
  - Entwicklung von (alternativen) (Teil-) Lösungen und Strategien bzw. Strategiebausteinen
  - Identifizieren von Strategien und Lösungen, die in Beziehung stehen

Beschreibung der Anforderungen

<Name der Anforderung>	
Kategorie:	<Kategorie der Anforderung: funktionale Anforderung, nicht funktionale Anforderung (Qualitätsanforderung), Rahmen-bedingung System/Produkt, usw.; Gegenebenenfalls Verweis auf das Anforderungsanalyse- bzw. Domänenanalyse-dokument>
Beschreibung:	<Kurze Beschreibung der Anforderung>
Priorität:	<muss | soll | kann>
Flexibilität und Änderbarkeit:	<Beschreibung der Flexibilität der Anforderung, d.h. in wie weit akzeptiert der Anforderungsträger, dass diese Anforderung verändert werden. Beschreibung der Änderbarkeit der Anforderung, d.h. in wie weit sind Änderungen dieser Anforderung durch den Anforderungsträger abzusehen.>
Auswirkungen:	<Auswirkungen auf andere Anforderungen und auf die Architektur bzw. auf Teile der Architektur bei Änderungen dieser Anforderung>

Beschreibung der Architekturtreiber

<Name des Architekturtreibers>	
Beschreibung:	<Kurze Beschreibung des Architekturtreibers>
Anforderungen:	<Liste der Anforderungen, die der Architekturtreiber adressiert.>
1. Testszenario:	<Beschreibung eines Anwendungsszenarios um eine korrekte Realisierung des Architekturtreibers zu überprüfen.>
N. Testszenario:	<dito>
Lösung:	<Kurze Lösungsbeschreibung>
1. Strategiebaustein:	<Beschreibung eines Strategiebausteins. Mehrere Strategiebausteine können sowohl alternativ als auch kombiniert verwendet werden.>
N. Strategiebaustein:	<dito>
Verwandte Strategien:	<Liste anderer Architekturtreiber bzw. Strategiebausteinen, die in Bezug zu diesem Architekturtreiber stehen>

Anwendungsbeispiel: eVersicherung 2010

- Neues Informationssystem „eVersicherung 2010“ der „Versicherungs AG“
- Produkte der Versicherungs AG
  - Lebensversicherungen
  - Rentenversicherungen
  - Unfallversicherungen
- Funktionalität der eVersicherung 2010
  - Versicherungsangebote verwalten
  - Versicherungsverträge verwalten
  - Versicherungspolicen verwalten
- Anwender des Systems sind Endkunden, Makler und Mitarbeiter der Versicherungs AG

Beispiel einer Architekturanforderungsbeschreibung

A1: Verwaltung von Kundendaten	
Kategorie:	funktionale Anforderung
Beschreibung:	Das System ermöglicht Benutzern Kunden zu erzeugen, deren Daten (Name, Adresse, etc.) zu bearbeiten, zu suchen und zu löschen.
Priorität:	muss
Flexibilität und Änderbarkeit:	Externe Kundendatenbanken müssen in zukünftigen Versionen des Systems eingebunden werden können.
Auswirkungen:	Die nachträgliche Integration externer Kundendatenbanken kann große Auswirkungen auf die Architektur und somit auch auf das System sowie die damit verbundenen Entwicklungssaufwände haben.
A2: Verwaltung von Vertragsdaten	
Kategorie:	funktionale Anforderung
.....	

Beispiel einer Architekturanforderungsbeschreibung

A3: Gleichzeitiges Arbeiten mehrer Benutzer	
Kategorie:	Qualitätsanforderung
A4: Internet-basiertes System mit grafischer Oberfläche	
Kategorie:	Qualitätsanforderung
A5: Benutzer haben Zugriff auf aktuelle Daten	
Kategorie:	Qualitätsanforderung
A6: Time-to-market ist zwei Jahre	
Kategorie:	Rahmenbedingung System/Produkt
A7: Anforderungen sind priorisiert	
Kategorie:	Rahmenbedingung Prozess
A8: Release-Zyklus wichtiger als Funktionalität	
Kategorie:	Rahmenbedingung Management
.....	

Beispiel einer Architekturtreiberbeschreibung

Aktuelle, konsistente, mehrbenutzerfähige Datenbearbeitung	
Beschreibung:	Mehrere Benutzer müssen gleichzeitig mit dem System die Daten bearbeiten und auf aktuelle und konsistente Daten zugreifen können. Auf Datenänderungen anderer Benutzer muss ein möglichst zeitnaher Zugriff möglich sein.
Anforderungen:	A1, A2, A3, A4, A5, A6, A8,
1. Testszenario:	Das System stellt dem Benutzer eine Kundenliste zur Verfügung. Der Benutzer wählt einen Kunden zur Bearbeitung aus und ändert den Namen des Kunden. Mit der Bestätigung der Namensänderung, ändert sich auch der Name in der dargestellten Kundenliste.
2. Testszenario:	Benutzer A bearbeitet die Daten eines Kunden. Ein anderer Benutzer B ermittelt gleichzeitig ein Liste mit Kunden deren Versicherungsvolumen ein bestimmtes Limit übertrifft. In dieser Kundenliste taucht auch der veränderte Name des Kunden auf, der gerade von Benutzer A bearbeitet wurde.
Lösung:	????

Beispiel einer Architekturtreiberbeschreibung

Lösung:	Kombination von Model/View/Controller Muster und Transaktionsmechanismus. Zuerst eine Kombination aus dem 1. und dem 6. Strategiebaustein. Dann von dem 1. zum 2. und später zum 3. Strategiebaustein erweitern, entsprechend dem Architekturtreiber „Kurze Entwicklungszyklen“.
1. Strategiebaustein:	Benutzer kann die Aktualisierung der Daten auf Knopfdruck aktivieren.
2. Strategiebaustein:	In bestimmten Zeitintervallen werden die angezeigten Daten aktualisiert.
3. Strategiebaustein:	Nach einer Datenänderung werden die Daten anderer Benutzer automatisch aktualisiert.
4. Strategiebaustein:	Eine Transaktion pro Benutzer
5. Strategiebaustein:	Eine Transaktion pro Dialog / Fenster
6. Strategiebaustein:	Eine Transaktion pro Anwendungsfall
Verwandte Strategien:	Architekturtreiber: Kurze Entwicklungszyklen; 1. Strategie: Einfaches, inkrementelles Hinzufügen von Funktionalität

Ergebnistyp der Architekturmodellierung: Architekturbeschreibung bzw. Architekturspezifikation

- Eine Architekturbeschreibung bzw. Architektur-spezifikation beschreibt ein Architekturmodell (meist in Form eines Softwarearchitekturdokumentes)
- Ein Softwarearchitekturdokument sollte enthalten:
  - Entwurfsziele, Richtlinien und Muster der Architektur
  - Abstrakte, grafische Beschreibungen mit verschiedenen Sichten
  - Zusätzlicher, erklärender Text
  - Entwurfsentscheidungen und Alternativlösungen
  - Funktionale und nicht funktionale Testfälle
  - Erweiterungsmöglichkeiten
- Ein Softwarearchitekturdokument sollte so einfach wie möglich sein, aber so umfangreich wie notwendig.
- “You know you've achieved perfection in design, not when you have nothing more to add, but when you have nothing more to take away.”

[nach Antoine de Saint-Exupéry]

Techniken für den Architekturentwurf

- Ziel des Architekturentwurfs
  - Komponenten identifizieren
  - Schnittstellen der Komponenten beschreiben
  - Komposition der Komponenten beschreiben
- Techniken
  - Anwendung von Heuristiken zur Komponentenidentifikation
    - Hauptwörter in Anforderungen werden zu Entitäten und Verben werden zu Aktivitäten
    - Bündelung von Aktivitäten und Entitäten in Komponenten
  - Von Szenarien zu vollständigen Modellen
    - Von UML-Sequenzdiagrammen zu UML Klassendiagrammen
    - **C**lass **R**esponsibilities **C**ollaborations Workshop, CRC-Karten
  - Wiederverwendung von bewährten Architekturen (Architekturstile, Frameworks, etc.)

Erfolgsfaktoren des Architekturentwurfs

- In minimalen Schnittstellen und Komponenten denken
- Syntax und Verhalten von Schnittstellen und Komponenten beschreiben
- Komposition der Komponenten zu einer Architektur getrennt beschreiben

- Modularität
- Lose Kopplung
- Abhängigkeiten sichtbar machen
- Zuerst fachliche Sichten, dann technische
- Geeignete Kombination von bottom-up und top-down

Architekturdokumente nach V-Modell XT

- Die zentralen Produkte der Systemerstellung, der HW- und SW-Entwicklung haben eine einheitliche Themenstruktur.
- … legen richtungsweisende Architekturprinzipien fest
- … untersuchen mögliche Entwurfsalternativen
- … beschreiben die Zerlegung in Komponenten
- … identifizieren Schnittstellen und Beziehungen zwischen Komponenten und zur Umgebung des Systems

📖 **(Unterstützungs-) System- / HW- / SW-Architektur**

- **Architekturprinzipien und Entwurfsalternativen**
- **Dekomposition des (Unterstützungs-) Systems / der HW- / SW-Einheit**
- **Querschnittliche Systemeigenschaften** (nur bei (Unterstützungs-) System)
- **Schnittstellenübersicht**
- **(Übergreifender) Datenkatalog**
- **Designabsicherung**
- **Zu spezifizierende System- / HW- / SW-Elemente**

Designabsicherung

- Ziel: Zeigen, dass die Architektur die Anforderungen erfüllt. Eventuell darüber hinaus Alternativen bewerten
- Modellgetriebene Validierung und Verifikation
  - exemplarische „Ausführung“ der Modelle
  - Modelchecking
  - Beweisen bestimmter Eigenschaften (Theorembeweiser)
- Prototyping
  - experimentelle Prototypen
  - explorative Prototypen
  - evolutionäre Prototypen
- Tests
  - funktionale Tests
  - Lasttests
  - Performance-Tests
- Messungen mit Metriken (siehe [Gil02])

Der Architekt als Programmierer

- Evaluierung von Entwurfsalternativen mit Hilfe von experimentellen Prototypen
- Einführung und Verbreitung der Softwarearchitektur im Entwicklerteam durch evolutionären Prototypen
- Evaluierung und Einführung von neuen innovativen Technologien und Konzepten
- Mitarbeit im Entwicklerteam als Programmierer u. Tester
- Trainer und Berater des Entwicklungsteams
- Koordination und Moderation der Schnittstellen-festlegung während Feindesign und Implementierung
- Initiierung und Etablierung des Dialogs zwischen den Entwicklern im Team

Der Architekt als Analyst

- Maßgebliche inhaltliche Gestaltung der Systemvision: Festlegung der übergreifenden Ziele, Anforderungen und Rahmenbedingungen des Systems und Architektur
- Abstimmung der Systemanforderungen durch explorative Prototypen
- Wirkungsanalyse, Abstimmung und Review von Anforderungen an das System
- Integration von technischen und konzeptionellen Innovationen in die gemeinsame Systemvision
- Verbreitung und Akzeptanz der Systemvision unter allen Beteiligten: Endbenutzer, Kunde, Entwicklungsteam, etc.

Der Architekt als technischer Projektleiter (1)

- Entscheidungskompetenz für alle wesentlichen Entwurfs-entscheidungen und Veränderungen: Entscheidungen so spät wie möglich und so früh wie nötig treffen!
- Auswahl und Beurteilung der technischen Fähigkeiten von Bewerbern und Mitarbeitern
- Auswahl und Festlegung von Schulungen, Werkzeugen und Technologie
- Schulung und Einführung der Softwarearchitektur im Entwicklerteam
- Motivation, Begeisterung und Integration des Entwicklerteams bzgl. des Architekturentwurfs
- Organisation des Entwicklerteams entsprechend dem Architekturentwurf und dem Projektplan

Der Architekt als technischer Projektleiter (2)

- Zuordnen der konkreten Implementierungsaufgaben zu den Entwicklern
- Delegieren von „lokalen“ Entwurfsentscheidungen an Spezialisten und an das Entwicklungsteam
- Verfolgung und Überprüfung der implementierungs-spezifischen Projektmeilensteine
- Verfolgung und Überprüfung der Qualität des Feindesigns der Implementierung
- Überprüfung der Integrität und korrekten Realisierung der Softwarearchitektur
- Organisation von Wartung und Weiterentwicklung der zentralen Schnittstellen des Systems

Der Architekt als „Hüter“ der Softwarearchitektur

- Überprüfung und Sicherung der Einhaltung der Architekturvorgaben im konkreten Entwicklungsprojekt
- Softwarearchitektur und die Rolle des Software-architekten als Bestandteil der Unternehmenskultur etablieren
- Identifikation, Koordination und Erarbeitung von projektübergreifenden Architekturthemen aus der eigenen Projektlandschaft, Industrie und Forschung
- Definition und Integration des Architekturentwurfs in die firmenübergreifende Projektmanagement- und Qualitätssicherungsstandards

Zusammenfassung „Disziplin Softwarearchitektur“

- Ziel des Architekturentwurfs ist es eine vollständige und prägnante Architekturbeschreibung zu erstellen, um die geforderten funktionalen und nicht funktionalen Anforderungen zu gewährleisten
- Der Softwarearchitekt ist DIE Brücke zwischen allen Anforderungsträgern und der Softwaretechnik
- Der Entwurfsprozess steht im Zentrum des gesamten Entwicklungsprozesses; Er hat Schnittstellen zu Anforderungsanalyse, Systemarchitektur und Implementierung
- Der Entwurfsprozess ist ein Zyklus aus Analyse, Modellierung und Validierung der Softwarearchitektur
- Die querschnittlichen Aufgaben des Architekten gehen von der Programmierung bis zur Firmenstrategiegestaltung

Literaturhinweise

- [BRS97] Klaus Bergner, Andreas Rausch, Marc Sihling: Using UML for Modeling a Distributed Java Application, Technischer Bericht der Universität München, TUM-I9735, http://wwwbib.informatik.tu-muenchen.de/infberichte/1997/TUM-I9735.ps.gz,1997
- [AF98] Paul Allen, Stuart Forst: *Component-Based Development for Enterprise Systems: Applying The SELECT Perspective*, Cambridge University Press, 1998
- [BCK+98] Len Bass, Paul Clements, Rick Kazman, Ken Bass: *Software Architecture in Practice (Sei Series in Software Engineering)*, Addison-Wesley Publishing, 1998
- [DW98] Desmond Francis D'Souza, Alan Cameron Wills. *Objects, Components, and Frameworks With UML: The Catalysis Approach*, Addison Wesley Publishing Company, 1998
- [HNS99] Christine Hofmeister, Robert Nord, Dilip Soni: *Applied Software Architecture*, Addison Wesley – Object Technology Series, 1999
- [HS99] Peter Herzum, Oliver Sims. Business Component Factory: *A Comprehensive Overview of Component-Based Development for the Enterprise*, John Wiley & Sons, 1999
- [JRL+00] Mehdi Jazayeri, Alexander Ran, Frank Van Der Linden, Philip Van Der Linden: *Software Architecture for Product Families: Principles and Practice*, Addison-Wesley Publishing, 2000
- [Gil02] Tom Gilb: Competitive Engineering, www.gilb.com, 2002