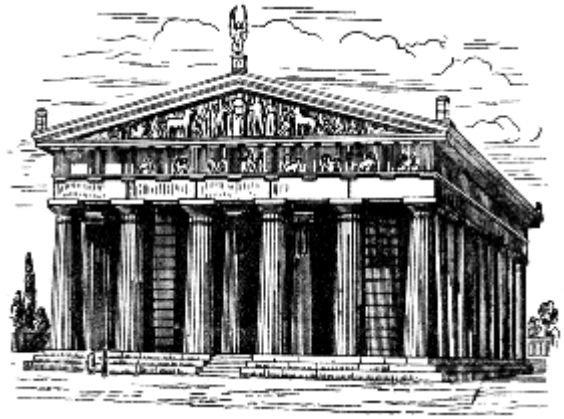
Softwarearchitektur

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

1. Einleitung und Überblick



Vorlesung Wintersemester 2005 / 2006

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

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

Inhalt

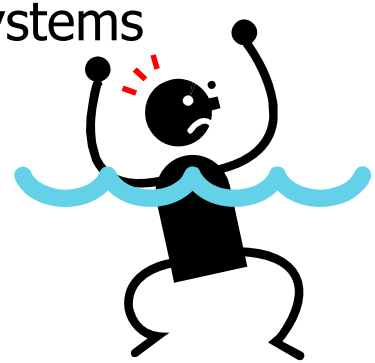
- Motivation und Analogie zur Architektur
- Der Begriff der Softwarearchitektur
- Softwarearchitektur am Beispiel
- Inhalte der Vorlesung
- Zielgruppen
- Organisatorisches
- Zusammenfassung und Ausblick

Inhalt

- Motivation und Analogie zur Architektur
- Der Begriff der Softwarearchitektur
- Softwarearchitektur am Beispiel
- Inhalte der Vorlesung
- Zielgruppen
- Organisatorisches
- Zusammenfassung und Ausblick

Problemstellung

- Beobachtungen:
  - Der Alltag ist immer stärker durchdrungen von Software
  - Software bestimmt immer mehr den Erfolg (von Produkten, oä.)
  - 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. 75% 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

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.)

Konkretes Beispiel: Bungalows im Olydorf

- 800 Bungalows
- Fertigstellung 1.5.1971
- Geplant und gebaut von Werner Wirsing (Preis für vorbildliches Bauen vom Bund deutscher Architekten)
- Grundstück: ehemaliger Flughafen Oberwiesenfeld
- Auftraggeber: Deutsches Studentenwerk

Sich ändernde Anforderungen der Bungalows

- Anforderung: Sportlerheim
- Trennzaun zwischen Männer- und Frauenbereich
- Aktivitäts- (Schwimmbad) und Erholungsbereiche
- „Sportlermensa"
- Leichte Erreichbarkeit der Sportstätten

- Anforderung: Studentendorf
- Gemeinschaftszentrum mit Disco, Tee- und Bierstube, Kino
- Günstige Unterkunft
- Hohe Fluktuation

- Einfaches Navigationskonzept von Otl Aicher über Farben an den Eingangstüren und im Außenbereich („Media-Linien")
- Moderne Infrastruktur (Parkplätze, U-Bahn-Anschluß, etc.)
- Ansprechende Gestaltung

Die zentrale Rolle des Architekten

- Aufsammeln, Bewerten und Priorisieren der teils mannigfaltigen Anforderungen

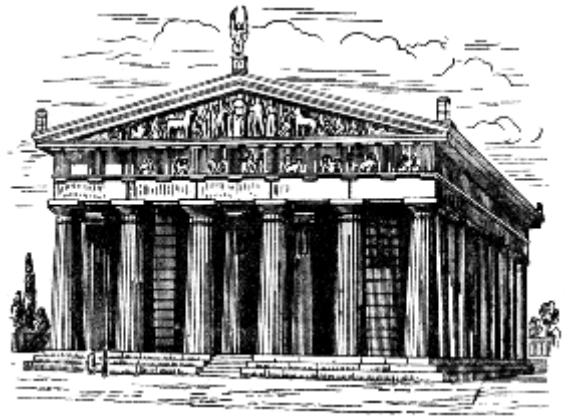
- Konzeption und Umsetzung einer qualitativ hochwertigen Architektur…

- … unter Berücksichtigung von Zeit und Kosten!

Die Disziplin „Architektur“

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

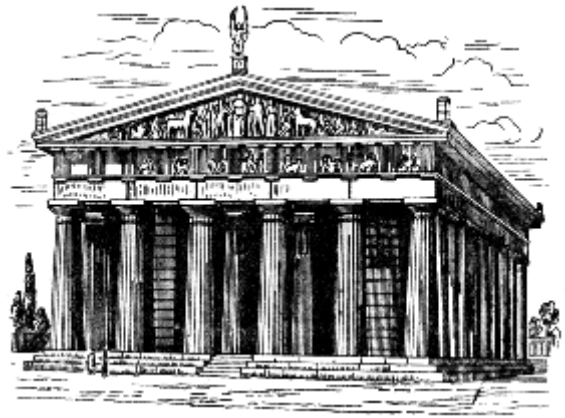
- ***utilitas*** : das Gebäude erfüllt seine Funktion
- **firmitas** : das Gebäude ist stabil
- **venustas** : das Gebäude ist ästhetisch gestaltet



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)



Inhalt

- Motivation und Analogie zur Architektur
- Der Begriff der Softwarearchitektur
- Softwarearchitektur am Beispiel
- Inhalte der Vorlesung
- Zielgruppen
- Organisatorisches
- Zusammenfassung und Ausblick

Softwarearchitektur als Teil der Lösung

- 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

- Begriffsdefinition: *„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.

Weitere Definitionen (1)

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

- „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)

Weitere Definitionen (2)

- „…the **highest level concept** of a system in ist environment. The architecture of a software system is its **organization or structure** of significant **components** interacting through **interfaces**, those components being **composed** of successively smaller components and interfaces.“

  (Rational Unified Process)

- „In most successful software projects, the expert developers working on that projects have a **shared understanding of the system design**. This shared understanding is called „architecture“. This understanding includes how the system is divided into **components** and how the components interact through **interfaces**…“

  (Ralph Jonson, 2003)

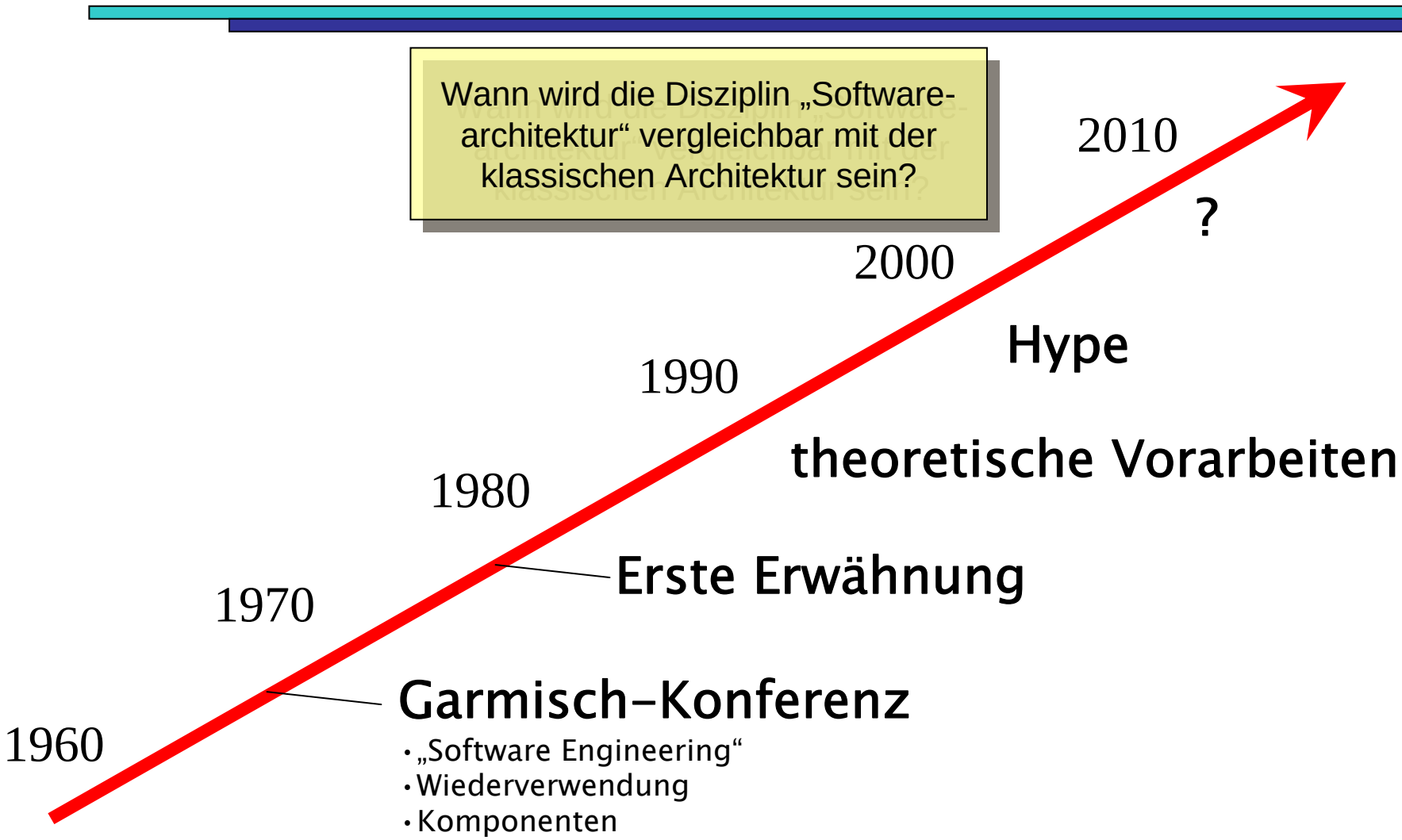
- „ Architecture is a **framework for change**“

  (Tom De Marco)

Ebenso wichtig: die Rolle des Softwarearchitekten

- 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

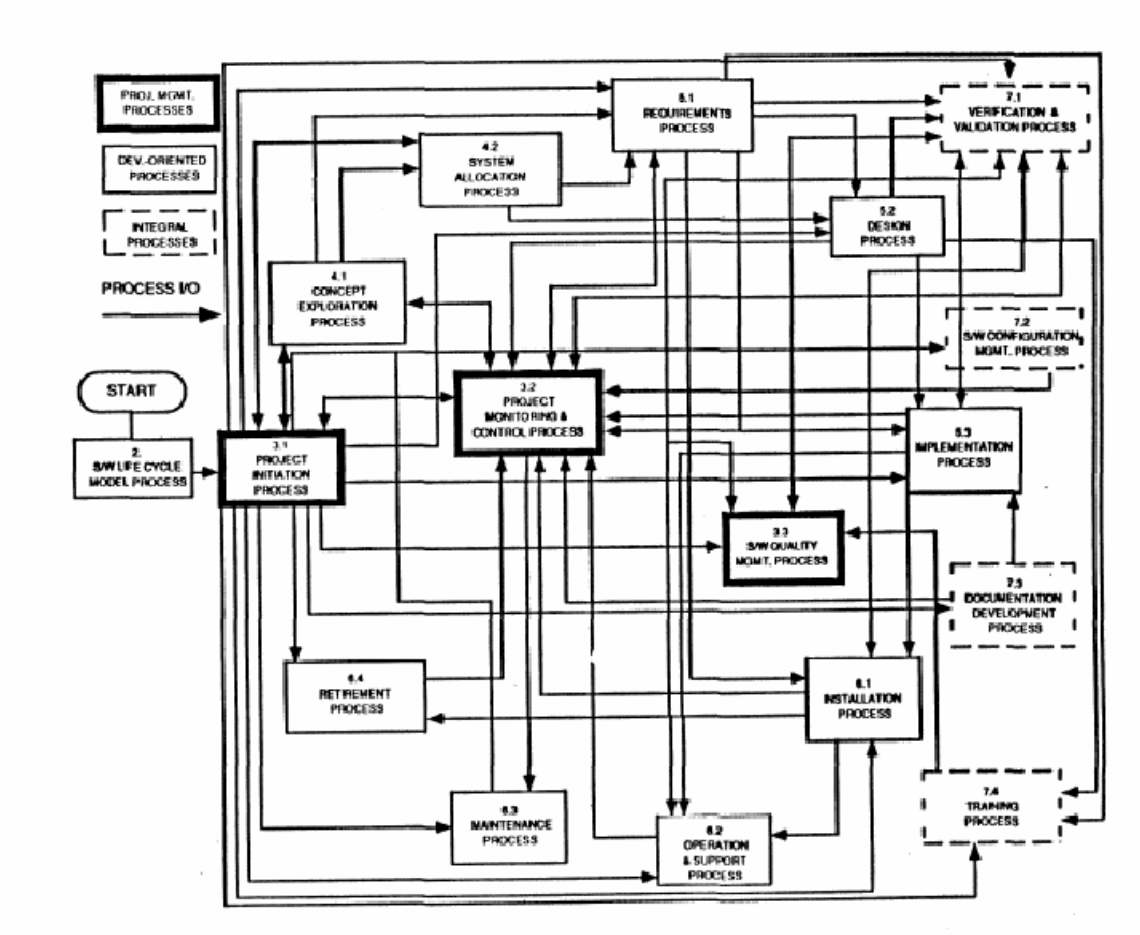
Geschichte des Begriffs „Softwarearchitektur“



Inhalt

- Motivation und Analogie zur Architektur
- Der Begriff der Softwarearchitektur
- Softwarearchitektur am Beispiel
- Inhalte der Vorlesung
- Zielgruppen
- Organisatorisches
- Zusammenfassung und Ausblick

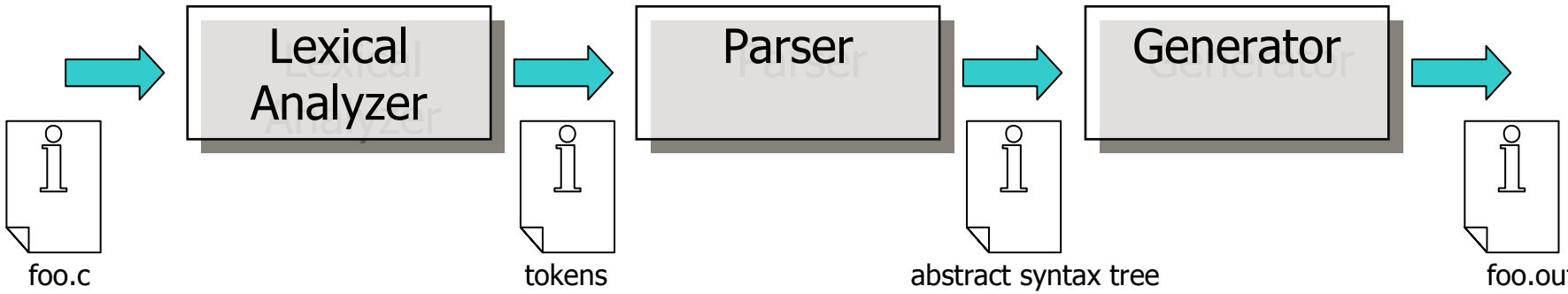
Was macht eine gute Architektur aus?



Negativbeispiel

- „Ich habe vielleicht schon mal eine schlechte Softwarearchitektur gesehen, da sie …“
  - kaum zu verstehen bzw. zu kommunizieren war
  - die Wiederverwendung von Bestandteilen nicht zuließ
  - zu einer schlechten Performanz des Systems führte
  - bei Änderungen der Anforderungen Anpassungen an ganz unterschiedlichen Stellen erforderlich machte
  - nicht erweiterbar war
  - nicht wartbar war
  - sehr umfangreiche Schnittstellen definierte
  - etc....

Positivbeispiel: 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

(Noch) offene Fragen…

- Auf welche Weise 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 eine 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.)

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.

Welche Qualitätseigenschaften werden durch die Softwarearchitektur beeinflusst?

- scalability
- availability / reliability
- security
- safety
- usability
- portability
- performance
- interoperability
- evolvability
- extensability

- Verfahren, um die Qualität einer Architektur beurteilen zu können oder auch um Architekturen vergleichbar zu machen, sind Inhalt aktueller Forschungen…

Inhalt

- Motivation und Analogie zur Architektur
- Der Begriff der Softwarearchitektur
- Softwarearchitektur am Beispiel
- Inhalte der Vorlesung
- Zielgruppen
- Organisatorisches
- Zusammenfassung und Ausblick

Überblick über die Vorlesung

- 17.10.: Einleitung und Überblick
- 24.10.: Ziele und Ergebnisse des Architekturentwurfs
- 07.11.: Grundlagen der Softwarearchitektur
- 14.11.: Beschreibungstechniken
- 21.11.: Sichten
- 28.11.: Model Driven Architecture (MDA)
- 05.12.: Ausprägungen (1)
- 12.12.: Ausprägungen (2)
- 19.12.: Betriebliche Informationssysteme (1)
- 09.01.: Betriebliche Informationssysteme (2)
- 16.01.: Betriebliche Informationssysteme (3)
- 23.01.: Eingebettete Systeme (1)
- 30.01.: Eingebettete Systeme (2)
- 06.02.: Telekommunikation

Übersicht Kapitel 2: Ziele und Ergebnisse des Architekturentwurfs

- Das Ziel, ein Softwaresystem mit einer geeigneten Architektur zu unterlegen, beeinflusst nachhaltig Organisation und Entwicklungsprozess
  - Definition und systematische Adressierung bestimmter Qualitätskriterien (z.B. funktionale und technische Anforderungen)
  - „Softwarearchitektur" als zentraler Dreh- und Angelpunkt des Entwicklungsprozesses (Analyse, Design, Bewertung, etc.) mit vielen Querbezügen in begleitende Tätigkeiten (Projektplanung, Projektmanagement, Auslieferung, Wartung, etc.)
  - Softwarearchitekt als zentraler Vermittler und Koordinator (oft auch „technischer Projektleiter")

Übersicht Kapitel 3: Grundlagen der Softwarearchitektur

- Teilweise sehr unterschiedliche Vorstellungen prägen die heutige Begriffswelt im Bereich Softwarearchitektur

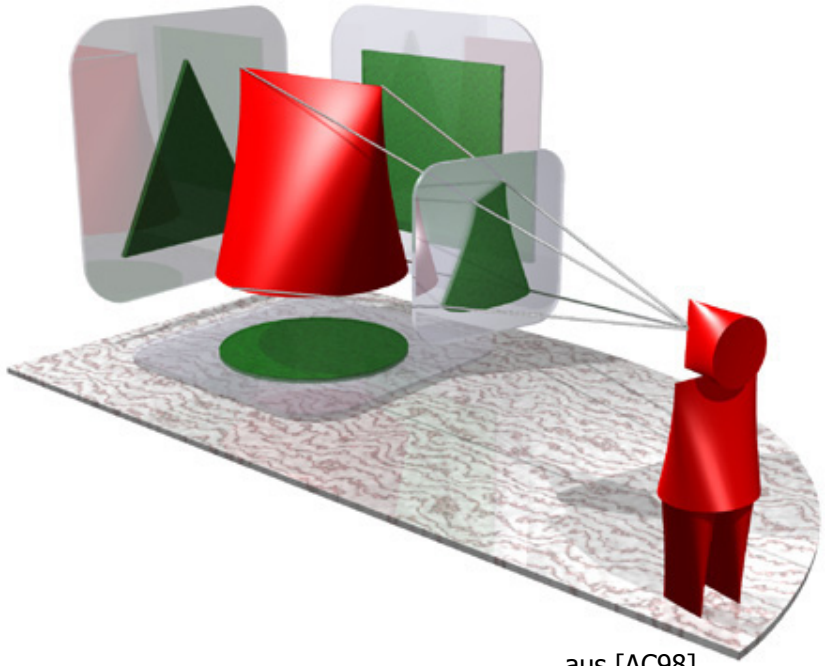
- Wir definieren ein einfaches Architekturmodell
  - als kleinsten Nenner existierender, bewährter Technologien
  - in formaler Form für die präzise, eindeutige Festlegung von Konzepten
  - als gemeinsames Verständnis der zugrunde liegenden Ideen und damit für eine verbesserte Kommunikation
  - für die Beschreibung von Bedeutungen (z.B. von graphischen Beschreibungstechniken, Programmsyntax, etc.)

Übersicht Kapitel 4: Beschreibungstechniken

- Um in der Lage zu sein, über Softwarearchitekturen zu kommunizieren, zu diskutieren, diese zu bewerten und zu vergleichen benötigt man Möglichkeiten, Softwarearchitekturen in geeigneter Form zu repräsentieren
- Ältere „Architekturbeschreibungssprachen" sind oftmals sehr formal und daher umständlich
- Neue Ansätze sind geeigneter:
  - IEEE 1471 standard on architecture representation
  - Unified Modeling Language
- Wir verwenden in der Vorlesung die UML 2.0

Übersicht Kapitel 5: Sichten

- Eine Softwarearchitektur kann sich je nach Rolle des Betrachters, Zeitpunkt im Entwicklungsprozess und gewünschtem Fokus unterschiedlich darstellen
- Diese „Sichten" können beispielsweise wie folgt klassifiziert werden:
  - Code View
  - Module View
  - Execution View
  - Conceptual View
- Vgl. [HNS99]



aus [AC98]

Übersicht Kapitel 6-Ende: Ausprägungen und Wiederverwendung

- Auf welche Weise lassen sich erfolgreiche Softwarearchitekturen in neuen Projekten wiederverwenden?
- Sammlung, Katalogisierung und geeignete Präsentation von Fragmenten oder vollständigen Softwarearchitekturen, die sich in der Praxis bewährt haben
  - Frameworks
  - Komponenten
  - Architekturmuster
  - Standards und Mechanismen

Inhalt

- Motivation und Analogie zur Architektur
- Der Begriff der Softwarearchitektur
- Softwarearchitektur am Beispiel
- Inhalte der Vorlesung
- Zielgruppen
- Organisatorisches
- Zusammenfassung und Ausblick

Ziele der Vorlesung

- Wer die Vorlesung (kontinuierlich) besucht
  - weiß, was Softwarearchitektur bedeutet
  - kennt die Kernprobleme sowie geeignete theoretische und praktische Lösungsansätze
  - kann Softwarearchitekturen graphisch aber auch formal darstellen
  - kennt nicht nur wichtige Technologien sondern auch Prozesse insbesondere innerhalb des Entwicklungsteams
  - kann anhand vorgestellter Beispiele und diskutierter Formen der Wiederverwendung selbst Softwarearchitekturen entwickeln und umsetzen
  - kennt das Handwerkzeug des Softwarearchitekten
  - steigert seinen/ihren „Marktwert“

Zielgruppe der Vorlesung

- Voraussetzung:
  - Grundstudium Informatik
  - Vorlesung Softwaretechnik oder Softwareengineering
- Nützlich:
  - Erfahrungen bei der Programmierung größerer Systeme (nicht nur mit der Technik – auch mit dem sozialen Gefüge eines Teams)
  - Kenntnisse der UML

Inhalt

- Motivation und Analogie zur Architektur
- Der Begriff der Softwarearchitektur
- Softwarearchitektur am Beispiel
- Inhalte der Vorlesung
- Zielgruppen
- Organisatorisches
- Zusammenfassung und Ausblick

Organisatorisches

- Die Vorlesung ist prüfbar! (ECTS: 3)
- Ort: MI 00.13.009A
- Zeit: Montags von 14:00 – 16:00 Uhr
- Website:
  http://www4.in.tum.de/lehre/vorlesungen/sw_arch/WS0506/index.shtml
- Mailverteiler: savorles@in.tum.de
- Büro: FMI 01.11.044 (Sekretariat Broy)

Inhalt

- Motivation und Analogie zur Architektur
- Der Begriff der Softwarearchitektur
- Softwarearchitektur am Beispiel
- Inhalte der Vorlesung
- Zielgruppen
- Organisatorisches
- Zusammenfassung und Ausblick

Zusammenfassung

- Architektur und Softwarearchitektur zeigen viele Parallelen: die richtige Kombination aus Ästhetik und Funktionalität führt zum Erfolg
- Die „Disziplin Softwarearchitektur" trägt entscheidend zum Gelingen eines Softwareprojekts bei
  - die Softwarearchitektur beschreibt, auf welche Weise die diversen Anforderungen erfüllt werden und kann anhand wünschenswerter Qualitätskriterien beurteilt werden
  - der Softwarearchitekt nimmt im Entwicklungsprozess eine zentrale Rolle ein. Er behält den Überblick, vermittelt zwischen den Beteiligten und gibt den Rahmen für die Implementierung vor.
- Die Vorlesung behandelt sowohl Grundlagen als auch weiterführende Konzepte

Literaturhinweise

- [AC98] Martín Abadi, Luca Cardelli: *A Theory of Objects – Monographs in Computer Science,* Springer Verlag 1998
- [HNS99] Christine Hofmeister, Robert Nord, Dilip Soni: *Applied Software Architecture*, Addison Wesley – Object Technology Series, 1999
- [Hus94] Norbert Huse: Le Corbusier, Rowohlt Taschenbuch Verlag, Sechste Auflage, 1994
- [SG96] Mary Shaw, David Garlan: *Software Architecture. Perspectives on an Emerging Discipline,* Prentice Hall, 1996