

Software-Architektur I

Vorlesung: Software-Engineering für große Informationssysteme

TU-Wien, Sommersemester 2002

Wolfgang Keller

Software-Architektur

I am more convinced than ever. Conceptual integrity is central to product quality. Having a system architect is the most important single step toward conceptual integrity... After teaching a software engineering laboratory more than 20 times, I came to insist that student teams as small as four people choose a manager, and a separate architect.

Fred Brooks, The Mythical Man-Month (20th Anniversary Edition. 1995)

Überblick

- Warum braucht man Software-Architektur?
- Eine Reise durch die Begriffe der Software-Architektur
- Fallstudie: Software-Architektur in Phoenix
- Einige Muster (Styles), die wir verwenden
 - Drei-Schichten-Architektur
 - MVC
 - Fat-Client, weitere Schnitte

Lernziele

Nach dieser Vorlesung ...



- Kennen Sie die Probleme, die die Beschäftigung mit Software-Architektur nötig machen
- Kennen Sie Begriffe, die in der „Forschungsrichtung“ Software-Architektur verwendet werden und wissen, warum man sich damit beschäftigt
 - und können in der angegebenen Literatur weitere Details nachlesen.
- Wissen Sie, daß man für das Design von großen noch viel mehr Entwurfsmuster benötigt, als die heute vorgestellten
 - und freuen sich hoffentlich auf die VL5, Entwurfsmuster

Warum braucht man Software-Architektur?

Ein typisches EDV-Großprojekt ..



- 60-70 Anwendungskernobjekte,
 - wie Kunde, Auftrag, Auftragsposition, Mitarbeiter etc.
- 100-200 Dialoge
- Projektdauer ca. 2-3 Jahre
- Teamstärke 2 -> 5 -> 10-20 -> 4
- Budget 35-100 Mio. ATS
- potentiell hunderte Benutzer im Dialogbetrieb
- entscheidend für das Kerngeschäft der Organisation, die das Projekt beauftragt hat

Warum braucht man Software-Architektur?

Stellen Sie sich bitte vor



- 15 Entwickler starten gleichzeitig mit der Implementierung
- 12 der 15 Entwickler sind Uni-Absolventen, die keine Erfahrung mit MVS, COBOL, CICS, Smalltalk u.ä. haben
- 3 der 15 Entwickler haben Projekterfahrung in mehreren Projekten
- Trotzdem wird das Projekt ein Erfolg!
- Warum?

Warum braucht man Software-Architektur? Architektur



- Wenige Architekten entwerfen ein Gebäude
 - der Bauplan ist generisch und kann in vielen Situationen analog wiederverwendet werden
- Viele Spezifikateure entwerfen die Detailpläne
 - sie bringen die funktionalen Anforderungen des Kunden ein
- Noch mehr “Bauarbeiter” realisieren das Gebäude



- Ein solches Vorgehen ist sinnlos bei Kleinstgebäuden

Eine Reise durch die Begriffe der Software-Architektur



- Definition: Software-Architektur
- Nichtfunktionale Anforderungen

- Erfahrung: Architektonische Stile & Entwurfsmuster
- Balancieren von Kräften
- Herleitungen aus nichtfunktionalen Anforderungen:
 - Unit Operations
- Darstellung: Architektonische Sichten

Begriff: Software-Architektur

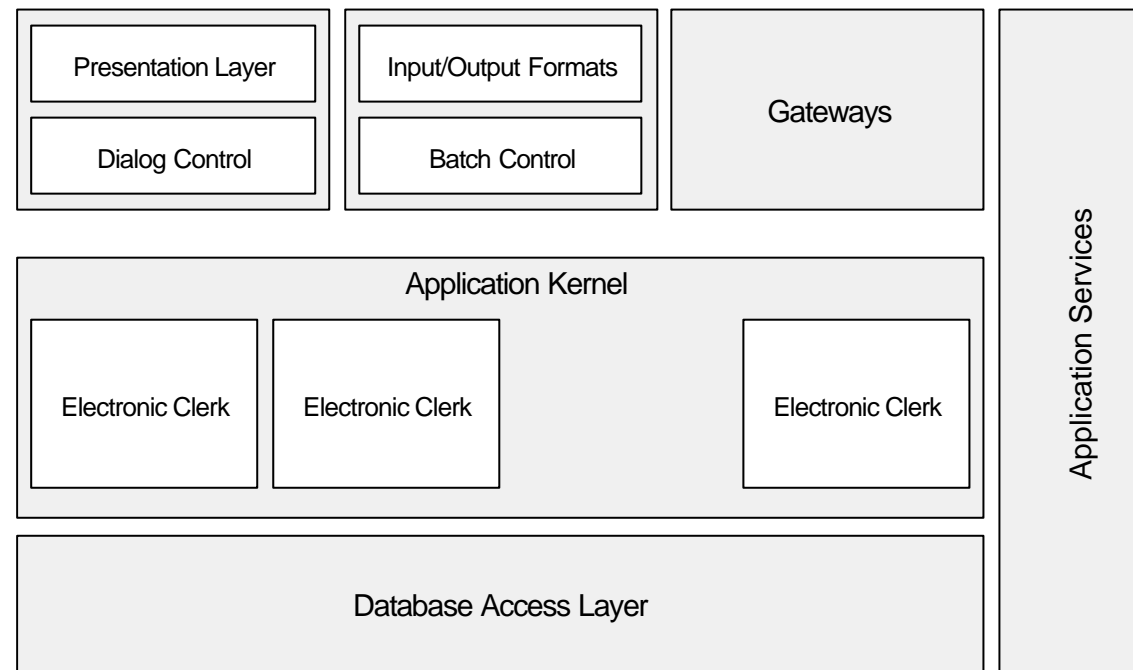
A software architecture provides a model of a whole software system that is composed of internal behavioral units (i.e. components) and their interaction, at a certain level of abstraction. All postulated requirements that are relevant to the later construction of the system have to be incorporated in this model.

Mehr gefällig? -> <http://www.sei.cmu.edu/architecture/definitions.html>

Dort finden sie ca. 50 Kilobytes an Definitionen :-)

Vertrautes Beispiel

Eine Repräsentation einer 3-Schichten-Architektur ...



Was ist Software-Architektur?

- es geht also um die Komponenten eines Software-Systems und deren Interaktion
- aber wie und wann und was soll ich denn nun designen?
- Woher kommen diese Kästen



Nichtfunktionale Anforderungen

Einige Beispiele ...



- Performance
 - Time to market
 - Gesamtkosten der Lösung
 - Robustheit (Fehlertoleranz)
 - Wartbarkeit
 - Veränderbarkeit und Flexibilität
 - Einfachheit
-
- die gewünschte Ausprägung solcher Eigenschaften beeinflusst wesentlich den Aufbau eines Software-Systems

Wie entwickelt man eine Software-Architektur?

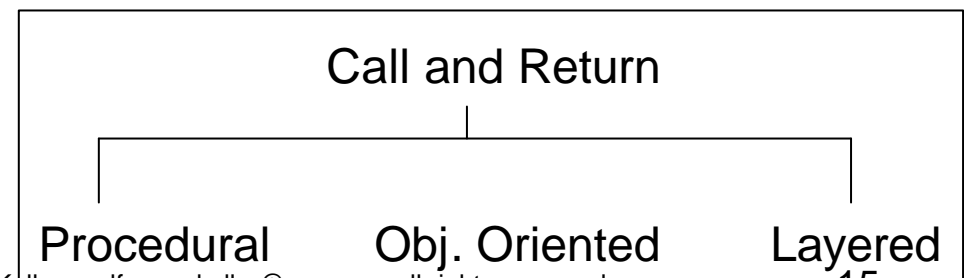
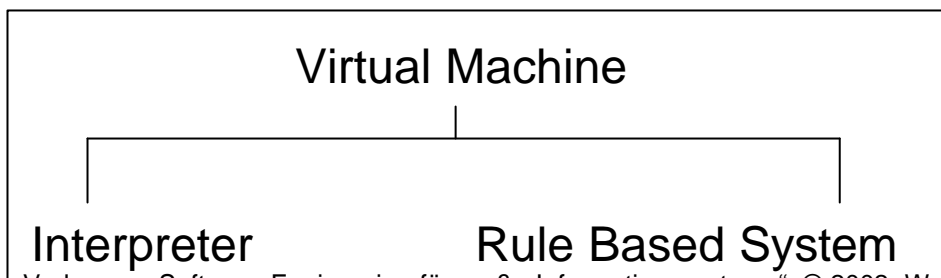
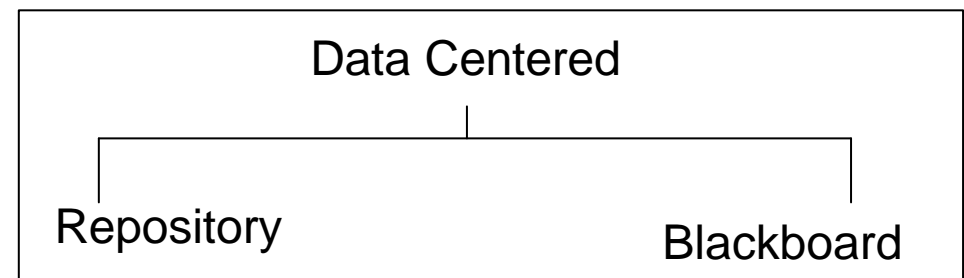
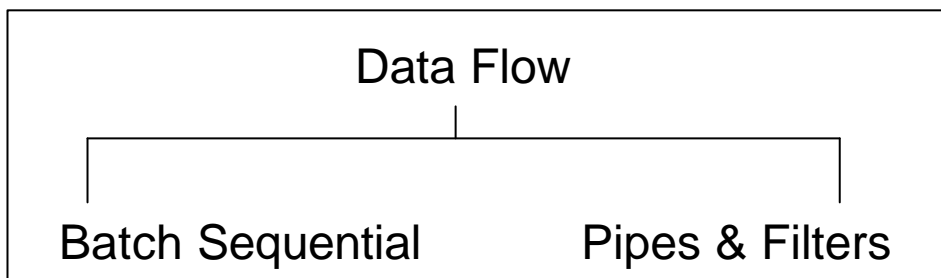
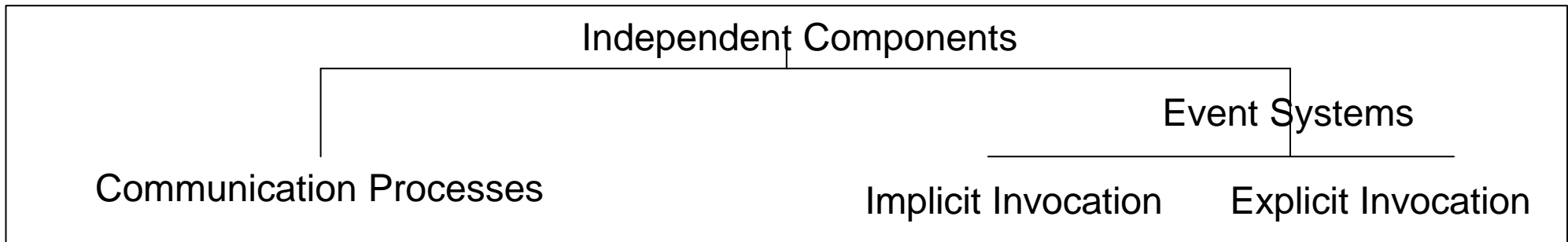
- Und wie bitte macht man aus nichtfunktionalen Anforderungen eine Software-Architektur?



Mehrere Faktoren wichtig ...

- Erfahrung
 - man verwendet bekannte „Architectural Styles“ und „Architectural Patterns“ (Entwurfsmuster)
- Ausbalancieren von Kräften
 - das kann man durch den Umgang mit Mustern (Patterns) lernen.
- Herleitung
 - wenn man ein Design kennt, kann man es meist auch aus einem „big ball of mud“ herleiten - mittels sog. Unit Operations (Architektonische Transformationen“)

Erfahrung Architektonische Stile - ein kleiner Katalog



Erfahrung Architektonische Stile



- Sehr viele davon haben wir in einem großen System
- Man kann sie kombinieren
 - ein „geschichtetes System“ (layered system) ist „objektorientiert“ implementiert und verwendet teilweise auch einen regelbasierten Ansatz (rule based system).
 - Daneben gibt es noch Systemteile, die einen Interpreter darstellen (Beispiel: Produktsystem VP/MS - Laufzeitsystem)

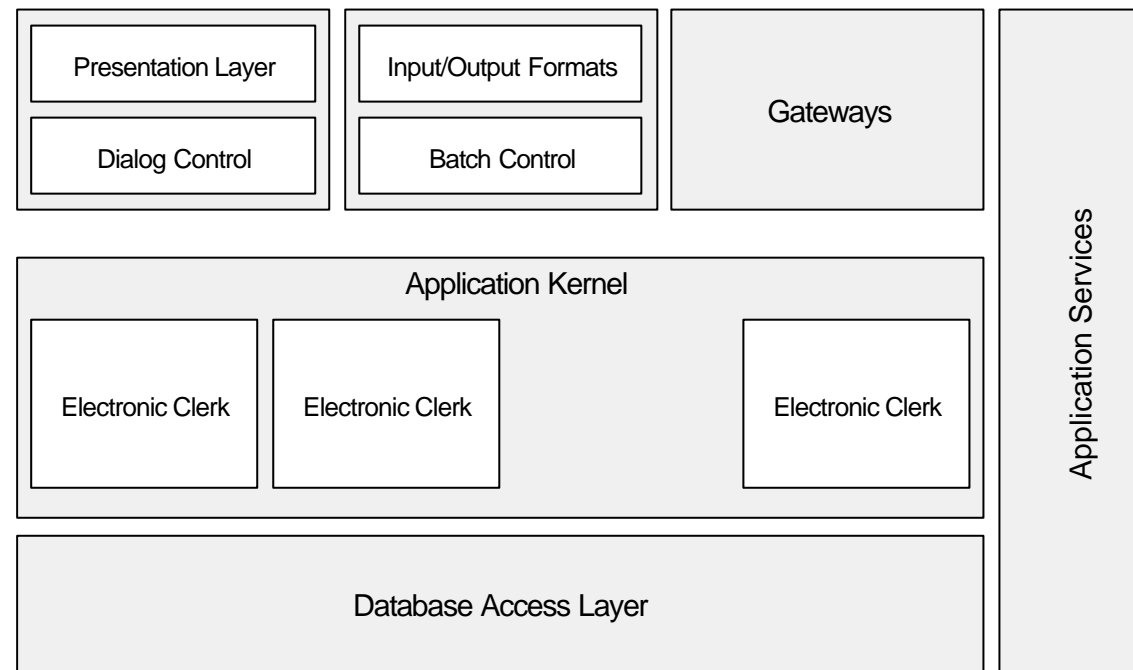
Architektonische Stile

Etwas formaler definiert - man benötigt ...



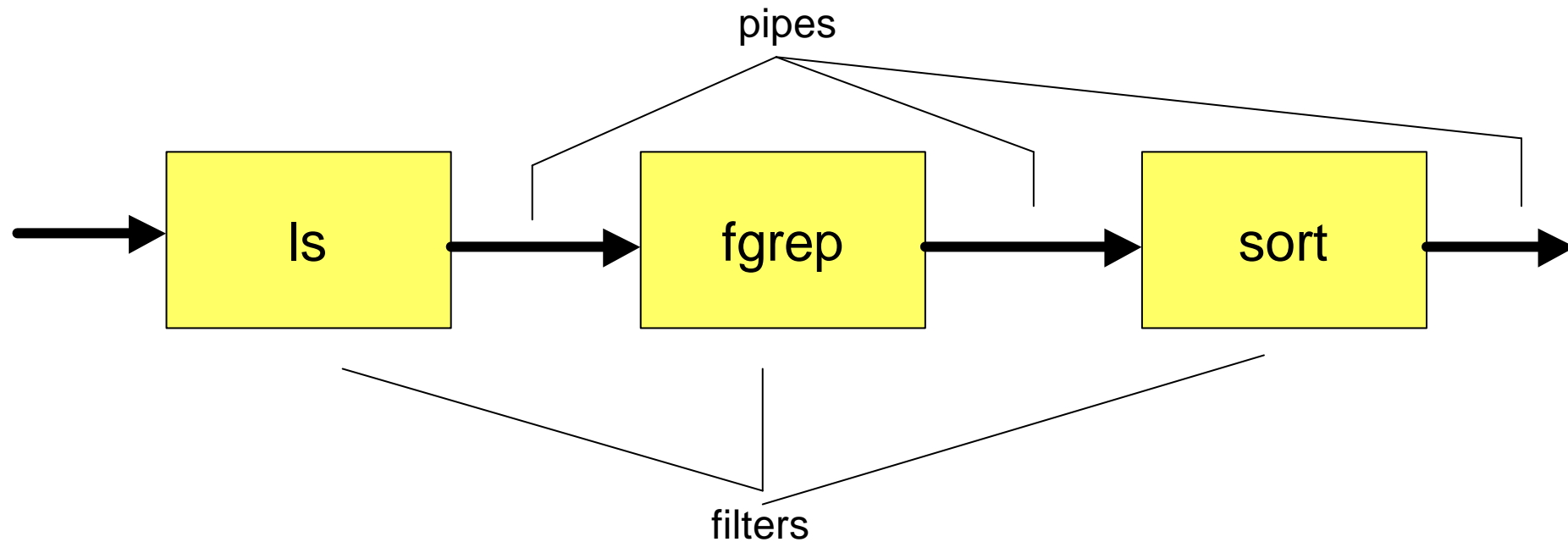
- Eine Menge von „Komponententypen“
 - Beispiel: Objekte organisiert in Schichten
- Eine topologische Anordnung dieser Komponenten
 - Schichten übereinander
- Eine Menge semantischer einschränkender Bedingungen (Constraints).
 - Es dürfen nur Services der nächst tieferen Schicht aufgerufen werden oder solche derselben Schicht - keine Calls nach oben
- Eine Menge möglicher Konnektoren
 - Prozeduraufrufe
- und erhält den Style „layered architecture“

Und wir haben wieder ...

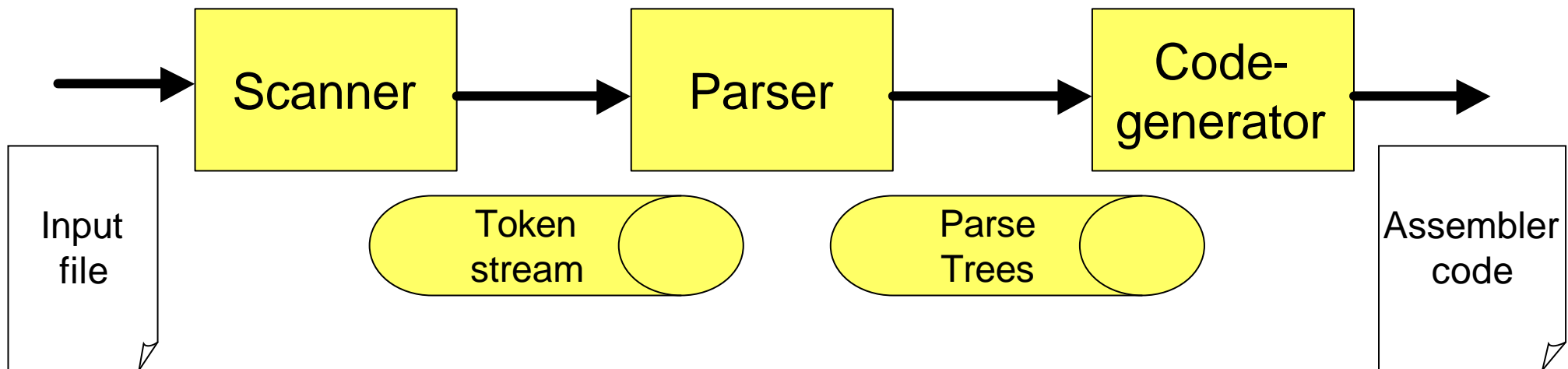


Weiteres Beispiel Pipes and Filters

Aus Unix bekannt: `ls -l | fgrep 'acme' | sort`

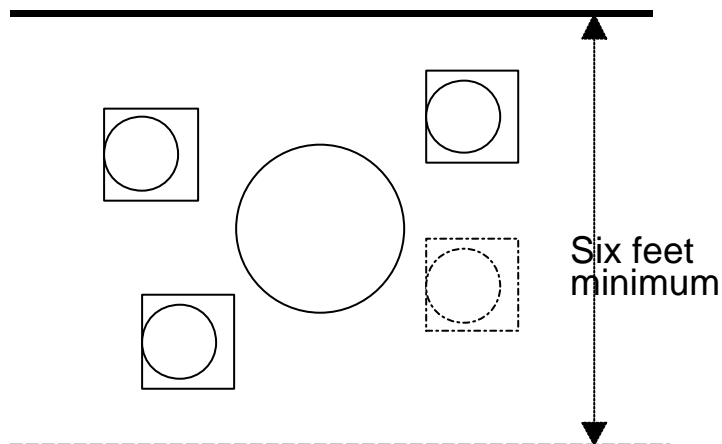


Pipes and Filters findet man auch in Compilern



Erfahrung: Architekturmuster

Beispiel aus der „Bau“architektur



Pattern #167, Alexander, Ishikawa, Silverstein,
A Pattern Language, Oxford University Press 1977

Balconies and porches which are less than six feet deep are hardly ever used

Balconies and porches made very small to save money; but when they are too small, they might as well not be there.

All balcony is only used properly when there is enough room for a small table where they can set down glasses, cups, and the newspaper. ...

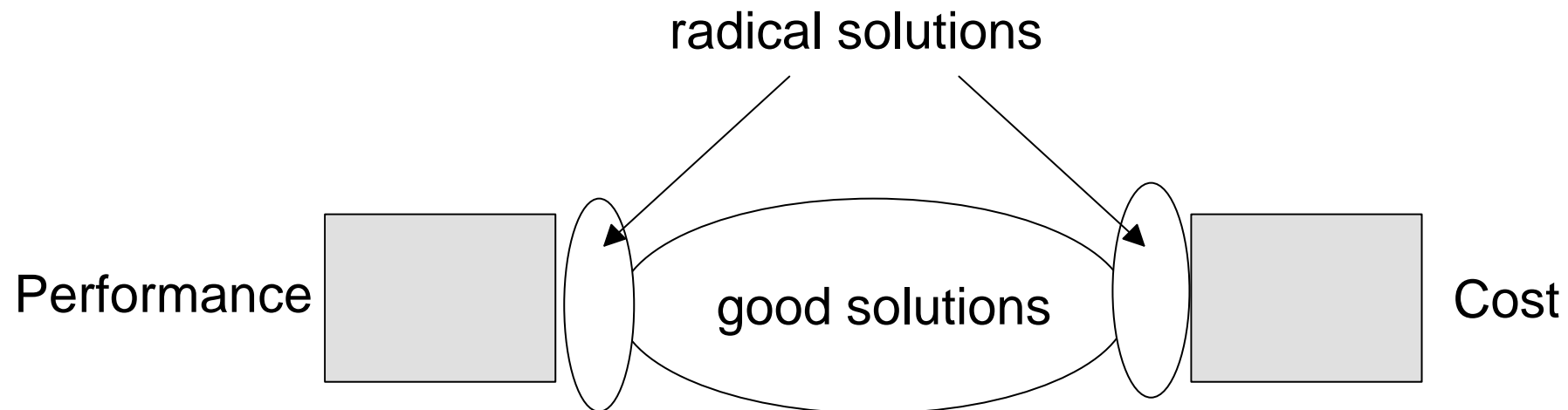
Erfahrung: Architekturmuster

Beispiele aus der Software-Architektur



- Die meisten „Architekturstile“ (Garlan/Shaw) gibt es auch in Form von „Architekturmustern“ beschrieben (Buschmann et al. und auch Garlan/Shaw)
 - Layers
 - Pipes & Filters
 - Blackboard
 - Reflection
- Weitere: MVC, Broker,

Balancieren von Kräften

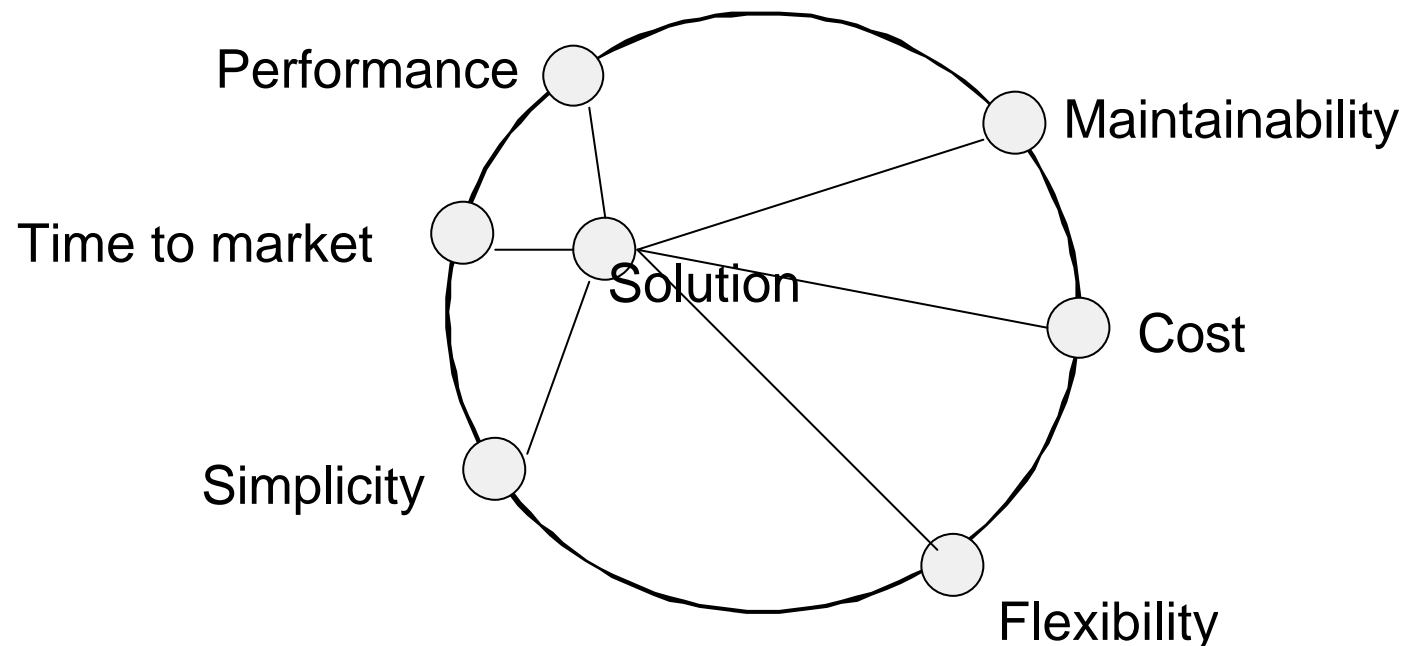


Balancing Forces is similar to politics: The most radical solution is seldom ever the best solution

Bad solutions can be found “near the walls”

Balancieren von Kräften

Conflicting Goals



Herleitung Unit Operations



- Separation
 - ::= Menge von Funktionalität nehmen und in einer Komponente mit definierter Schnittstelle zusammenfassen
- Uniform Decomposition
 - ::= Funktionalität einer Komponente wird in mehrere Subkomponenten zerlegt
 - Ausprägungen: Part-whole und Is-a
- Replication
 - ::= eine Komponente wird dupliziert (vervielfacht) - Benutzt um die Fehlertolerenz zu erhöhen

Herleitung Unit Operations



- Abstraction
 - ::= Bildung einer „virtuellen Maschine“ - zur Emulation von Funktionalität, zum Verstecken von Komplexität
- Compression
 - ::= Entfernung von Schichten oder Schnittstellen: Ziele - bessere Performance, Umgehen der Schichtung, wenn die untere Schicht benötigte Dienste nicht hat, schnellere Entwicklung (quick & dirty)
- Ressource Sharing
 - ::= Mehrere Clients greifen auf eine gemeinsame Ressource zu

Herleitung Kritik an den Unit Operations



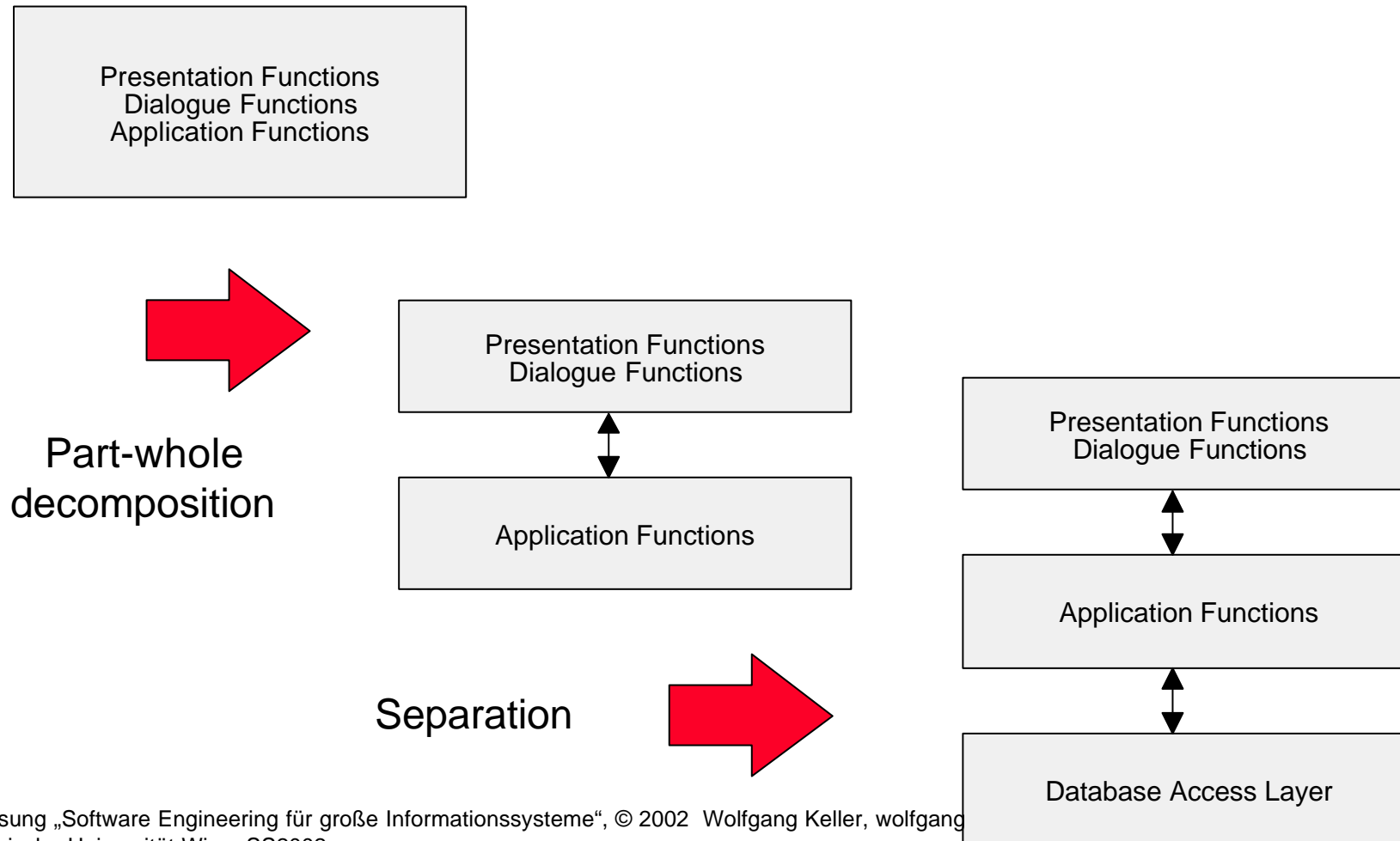
- Unit Operations sind sehr schwer gegeneinander abzugrenzen
- Man sieht selten jemand, der sie bewußt einsetzt, um ein Design herzuleiten
- Aber das Wissen, daß es sie gibt, ist nützlich, weil man ein Design damit begründen und herleiten kann.

Operationen. Wirkung der UML Operations auf nichtfunktionale Eigenschaften



| | Operation | | | | | |
|----------------------------|-------------|-------------|--------------------------|--------------------|-------------|-------------------|
| | Abstraction | Compression | Part-whole decomposition | is-a decomposition | Replication | Ressource sharing |
| Scalability | | | | | + | |
| System modifiability | + | - | + | + | | |
| Integratability | + | - | + | + | | + |
| Portability | + | - | | | | |
| Sequential performance | - | + | | - | - | - |
| Concurrent performance | | - | + | + | + | |
| Fault tolerance | + | | | | + | |
| Ease of system creation | + | - | | + | | + |
| Component modifiability | + | - | | | | - |
| Ease of component creation | + | | | + | | + |
| Reusability | + | - | + | | | |

Herleitung: Beispiel einer Herleitung am Beispiel Seeheim Modell (vulgo 3S Arch.)



Wie stellt man eine Software-Architektur dar?

- Kästchen und Pfeile -
ist das alles?



Darstellung Architektonische Sichten



- Ein Architekt kennt nicht nur eine Planungsansicht von einem Haus, sondern es gibt z.B.
 - Gesamtansichten
 - Entwässerungspläne
 - Stromversorgungspläne
 - Statische Pläne
- Analog gibt es auch verschiedene Sichten auf ein Software-System - man nennt diese „architektonische Sichten“ (architectural views)

Beispiele für Architektonische Sichten



| Architektonische Sicht | Gegenstände in der Sicht | Konnektoren |
|--------------------------|----------------------------------|---|
| Konzeptuelle Architektur | Komponenten | allg. Beziehungen |
| Modul-Architektur | Subsysteme, Module | Export- / Import- Beziehungen |
| Code-Architektur | Files, Directories, Libraries | Include- / Contains- Beziehungen |
| Ausführungs-Architektur | Tasks, Threads | Inter-Prozeß- Kommunikation, (Remote) Procedure Calls |

Fallstudie

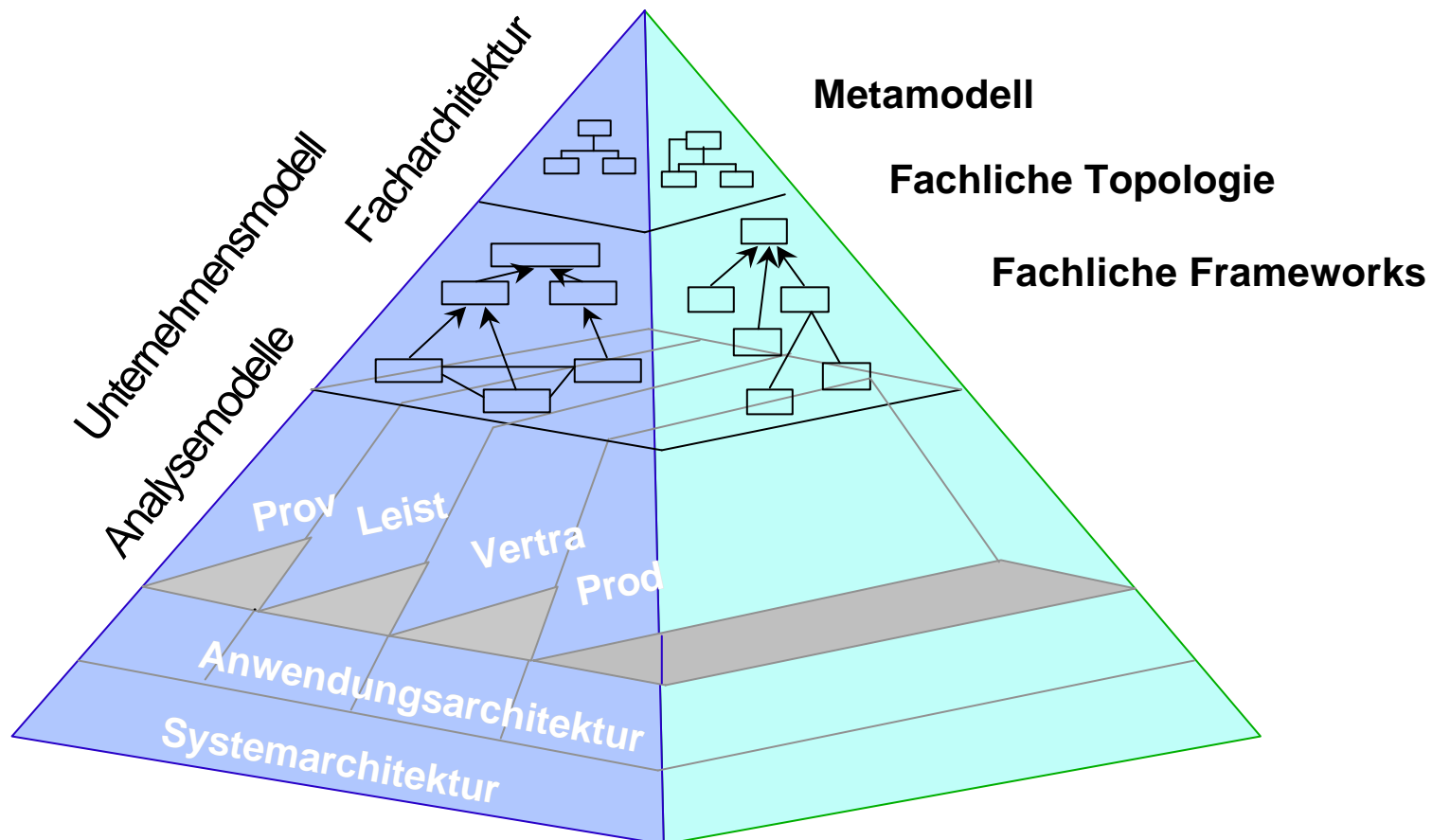
Software-Architektur in Phoenix



- Architektursichten in Phoenix
 - Facharchitektur
 - Anwendungsarchitektur
 - Systemarchitektur
- Muster und ihr Aufscheinen in den Sichten
- Wichtiges und Unwichtiges

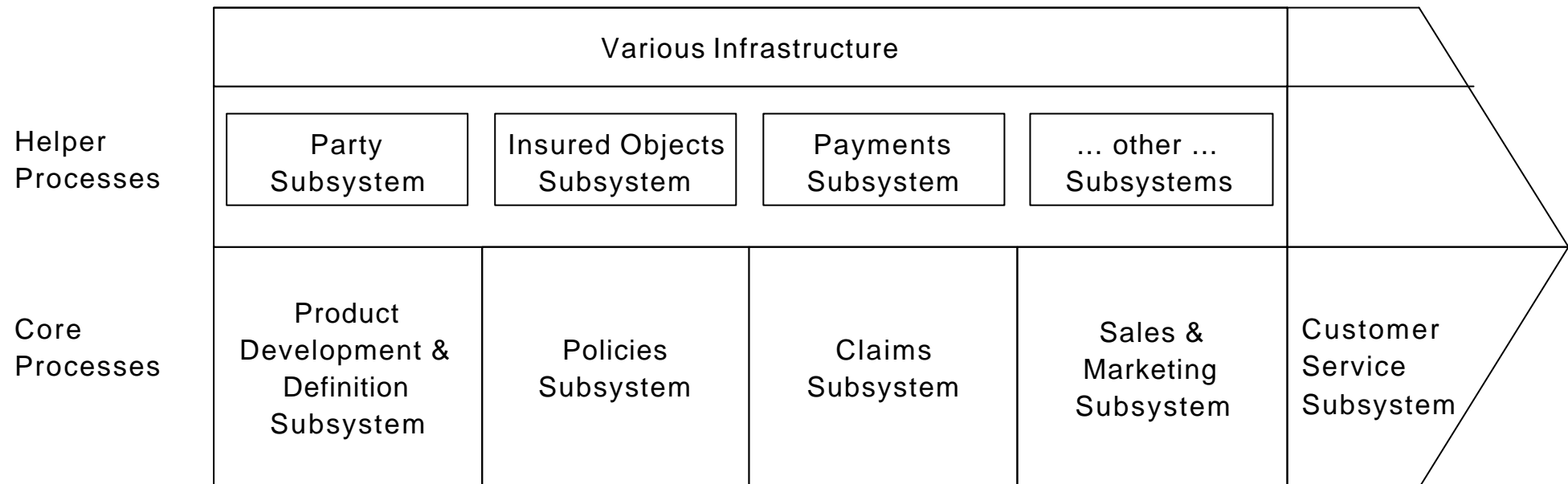
Software-Architektur in Phoenix

Architektonische Sichten



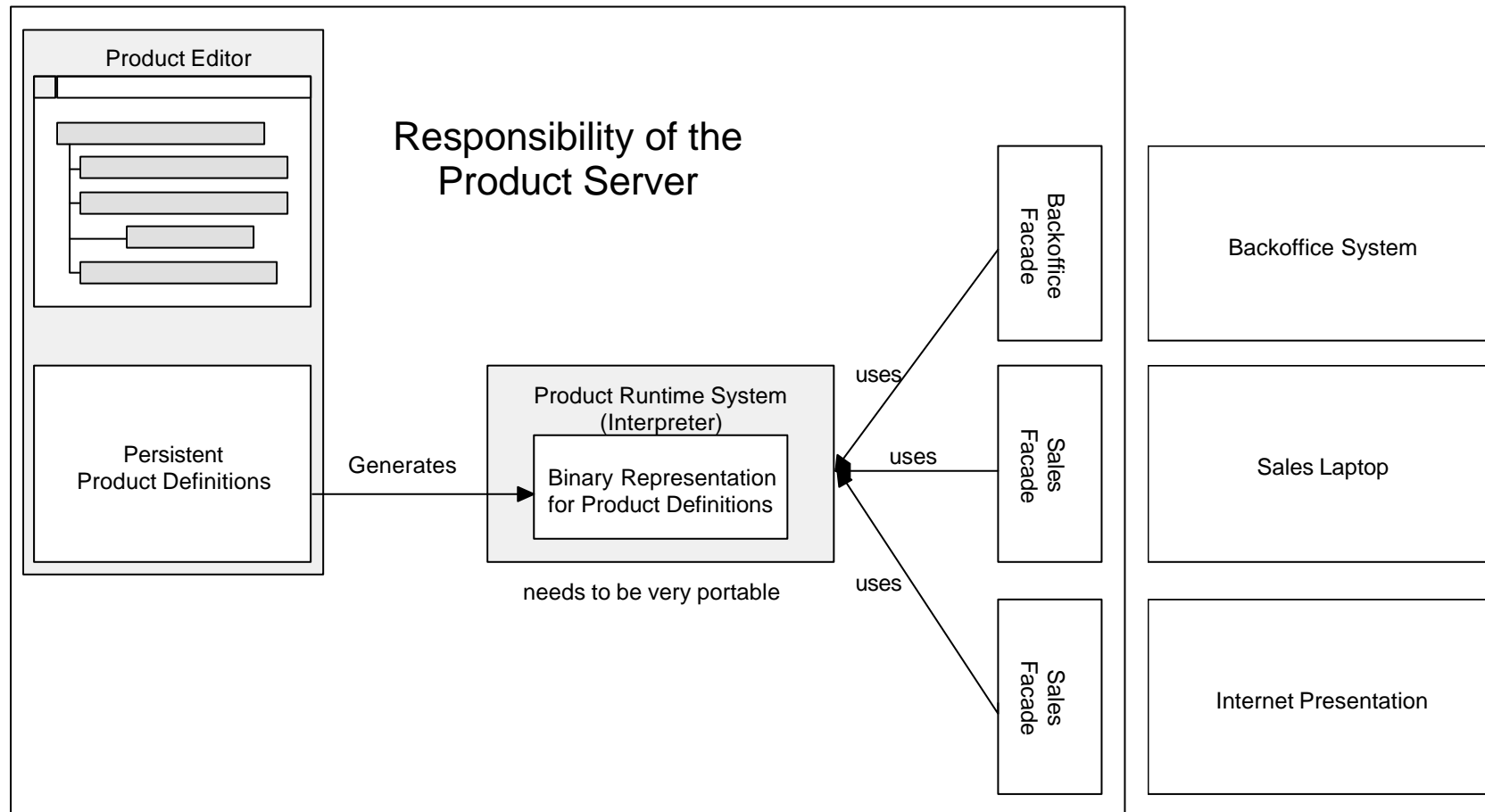
Muster aus der Facharchitektur

Beispiel: Wertkette für Versicherungen



Muster aus der Facharchitektur

Beispiel: Produktserver



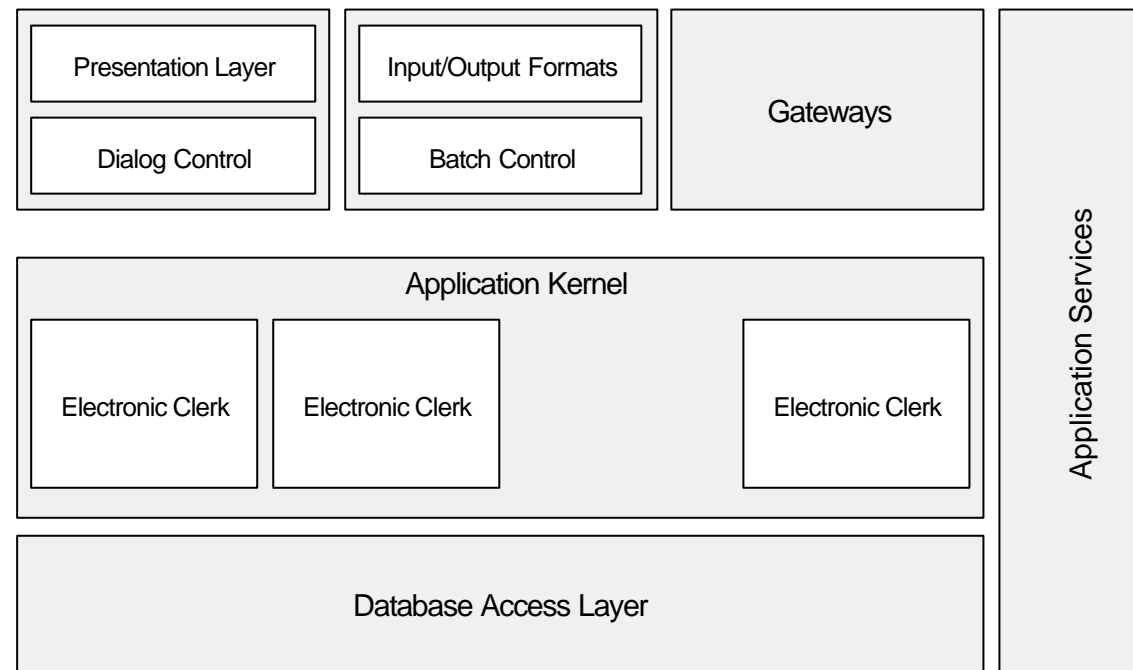
Facharchitektur: In der Facharchitektur finden Sie nicht ..



- Drei-Schichten-Architektur
- n-tier-Architektur
- J2EE Muster
- MVC
- Fat-Client

Dafür aber in der Anwendungsarchitektur

Und wir haben wieder ...



In der Anwendungsarchitektur finden Sie nicht ..



- Fat-Client

Dafür aber in der Systemarchitektur

Baustein Zielarchitektur

Was ist wichtig?



- Technische Architekturen ändern sich schnell
- Technologiegenerationen von 3-5 Jahren.
 - Beispiele: Internet, CORBA, Componentware, DCOM, Java haben vor 5 Jahren bei Architekturüberlegungen noch keine Rolle gespielt
 - Damit müssen sich die Anwendungs- und Systemarchitektur ändern.
- Wichtig ist daher die Facharchitektur
 - sie ändert sich langsamer
 - fachliche Konstruktionsprinzipien sind lange gültig

Fallstudie: Phoenix-Architekturprinzipien

Konstruktionsprinzipien der Generali

Beispiele für verwendete Architekturstile

- versicherungsfachlich
 - Spartenübergreifender Ansatz
 - Produktorientierung
 - Geschäftsprozeßorientierung
- technisch
 - Objektorientierung
 - Komponenten

Phoenix-Architekturprinzipien

Beispiel für nichtfunktionale Eigenschaft: Produktorientierung



- Neue Produkte müssen innerhalb von Tagen oder Wochen einföhrbar sein
- Sicherheit gewinnt man durch die Zusammensetzung von neuen Produkten aus bekannten, kalkulierten Bausteinen
- Die Einföhrung neuer Produkte geschieht ohne oder mit nur geringer Programmierung

Weiterführende Literatur

Bücher



- Bass, Clements, Kazman: Software Architecture in Practice, Addison Wesley, 1998.
- Shaw, Garlan: Software Architecture, Prentice Hall 1996
- Buschmann et al.: Pattern Oriented Software Architecture, A System of Patterns, Wiley, 1996.

Weiterführende Literatur

Frei verfügbare Surveys



- Hofmann, Horn, Keller, Renzel, Schmidt: The Field of Software Architecture, Technische Universität München Technical Report TUM-I9641, 1996
 - <http://www4.informatik.tu-muenchen.de/reports/TUM-I9641.html>
- Broy, Denert, Renzel, Schmidt (Eds.): Software Architectures and Design Patterns in Business Applications, Technische Universität München Technical Report TUM-I9746 , 1997
 - http://wwwbroy.informatik.tu-muenchen.de/paper_db/reports/TUM-I9746.html

Wichtige Web-Sites

- Software Architecture Resource Sites
 - <http://www2.umassd.edu/SECenter/SAResources.html>
- Carnegie Mellon University Software Architecture Pages
 - http://www.sei.cmu.edu/architecture/sw_architecture.html