Software-Architekturen

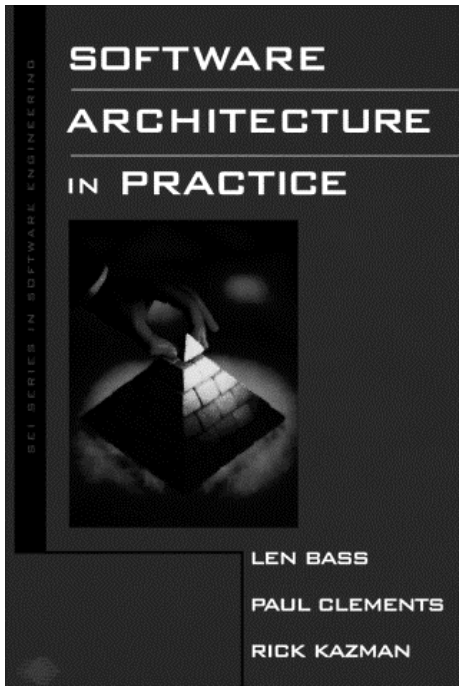
- Begriffe, Elemente von Software-Architekturen
- Notation
- Architektur großer Systeme

- Architektur-Verwendung
  - Beispiel aus DeMarco: Der Termin
- Architektur-Stile
- Schlussbemerkung

Architectural Design = Establishing the overall structure of a software system

Ian Sommerville. *Software Engineering*. 2001

Definition (Produkt)



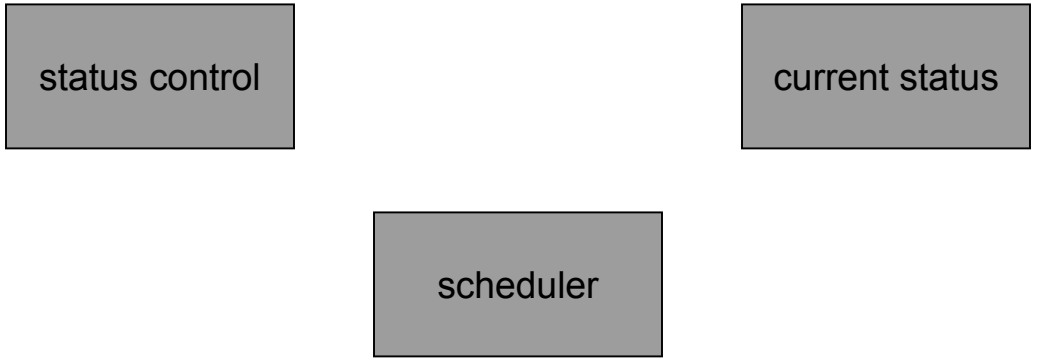
The Software Architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.

By *externally visible* properties, we are referring to those assumptions other components can make of a component, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on.

Bass, Clements, and Kazman.
Software Architecture in Practice. 1998

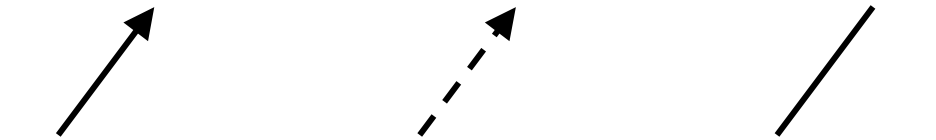
Software-Architektur-Design 1/3

- Architektur-Design ist ein kreativer Prozess, in dem Datenstrukturen und Algorithmen auf Subsysteme verteilt werden, um das gewünschte Verhalten zu erreichen
- Anforderungen werden auf eine Menge von Komponenten (*computational units*) abgebildet, die
  - in einer angemessenen Zeitspanne implementiert werden können und
  - die Verteilung der Implementierung auf Teams ermöglichen



Software-Architektur-Design 2/3

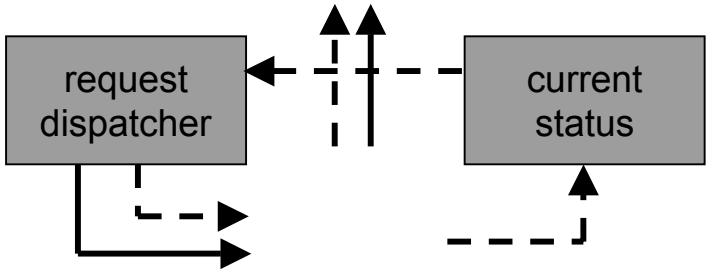
- Die Interaktionen zwischen den Komponenten werden durch Konnektoren beschrieben, in Form von
  - Kontrollfluss,
  - Datenfluss oder
  - Abhängigkeiten
- Konnektoren definieren, welche Komponenten wie zusammenarbeiten



Kontrollfluss Datenfluss Abhängigkeit

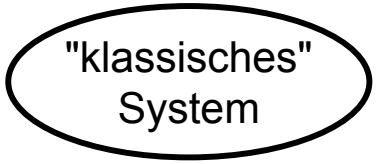
Software-Architektur-Design 3/3

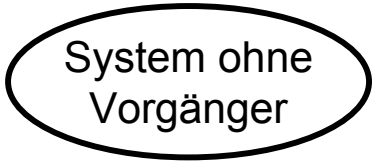
- Eine Konfiguration beschreibt die Topologie der Komponenten-Verbindungen
- Die Anforderungen sind auf die Komponenten verteilt, die miteinander mittels der Konnektoren interagieren, um die Anforderungen zu erfüllen

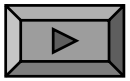


Herkunft von Software-Architekturen

- Einflussfaktoren
  - Erfahrung des Architekten
  - Neuheit des Systems / der Systemart







Fragen?

Partner-Interview: Eigene Erfahrungen

- Interviewen Sie einen Partner, indem Sie ihm die folgenden Fragen stellen
  - Wie groß (etwa) war das größte Programm, das Sie bisher entwickelt haben?
  - Wie groß (etwa) war das größte Programm, an dem Sie bisher mitgearbeitet haben?
  - Wie groß (etwa) war Ihr Anteil darin?
  - Beschreiben Sie / zeichnen Sie die Struktur dieses Programms (Ihres eigenen oder desjenigen, an dem Sie mitgearbeitet haben) in einer Minute!

- Dauer: 4 Minuten, dann Wechsel

Beschreibung von Software-Architekturen

- Architekturen werden aus verschiedenen Sichten (*architectural views*, *structures* in der Definition von BCK) beschrieben
- Beispiele für Sichten
  - Statische Sicht: zeigt die wesentlichen System-Komponenten
  - Dynamische Sicht: zeigt die Prozess- oder Ablauf-Struktur des Systems
  - Entwickler-Sicht: zeigt die Module/Klassen/Packages des Systems
  - Physische Sicht: zeigt die physischen Komponenten (Prozessoren, Sensoren etc.) des Systems
  - etc.
- Verwendung von Sichten: je nach Bedarf!
- Prinzip: nicht alles in eine Darstellung "wurschteln"

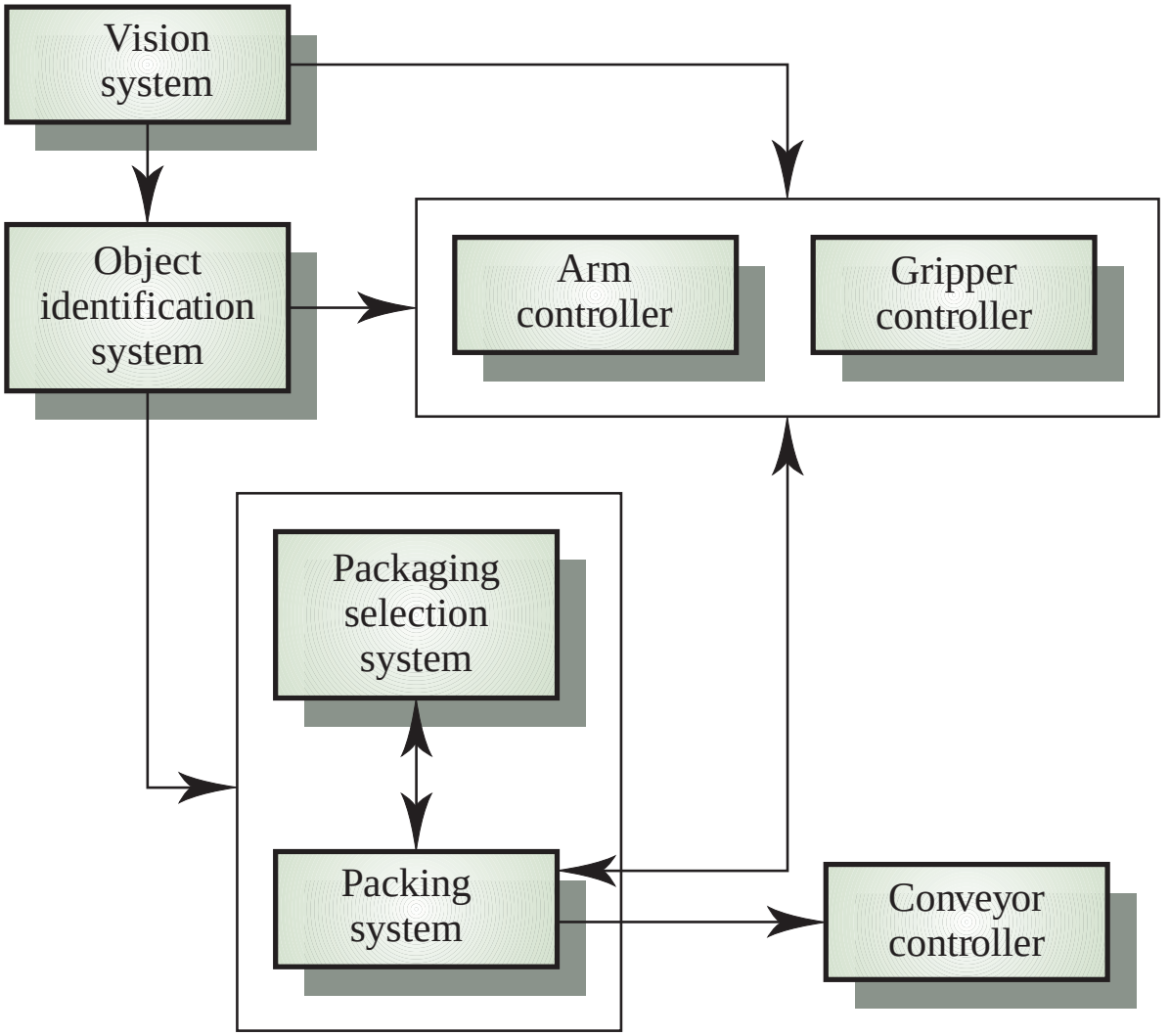
Architektur-Beschreibung: Spezielle Sprachen

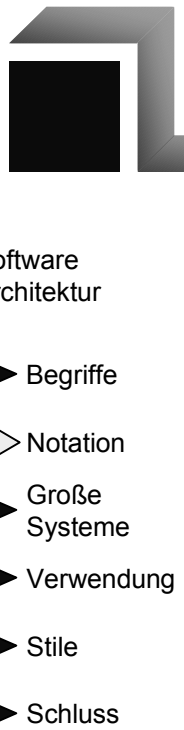
- Architektur-Beschreibungs-Sprachen (*architecture description languages*, ADL)
  - Formale Sprachen für die Beschreibung von Software-Architekturen
  - Teilweise mit Unterstützung für die Integration existierender (Teil-) Architekturen oder Komponenten
  - Teilweise mit Unterstützung für die Architektur-Bewertung
- Kommerzielle Beispiele
  - RoseArchitect (UML), Koala (bei Philips)
- Forschungsbeispiele
  - ACME, Aesop, C2, Darwin, MetaH, Rapide, SADL, Unicon, Wright, Z

Architektur-Beschreibung: "Boxes and Lines"

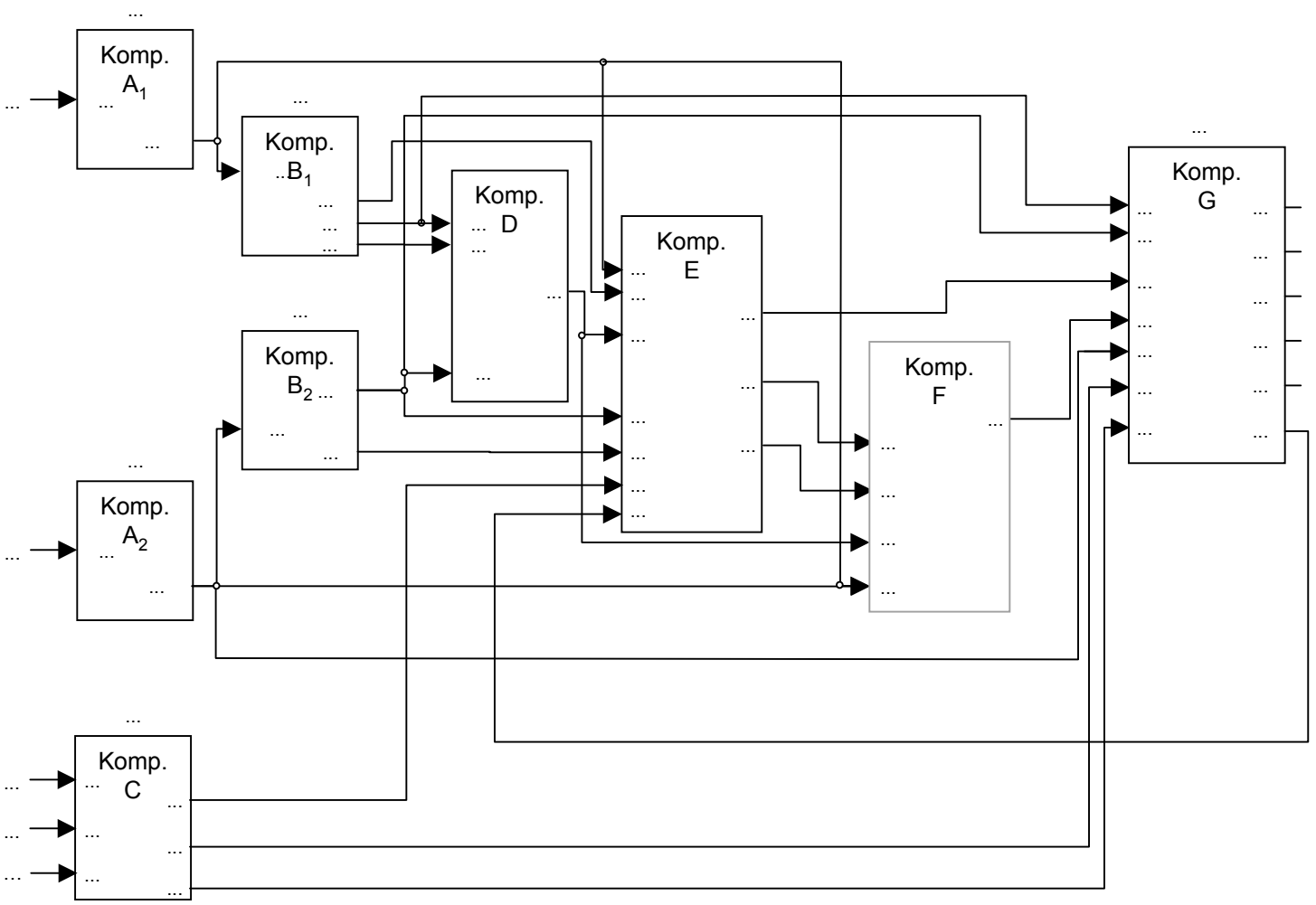
- Oft benutzte Darstellung von Kästen und Linien/Pfeilen
- Vorteil:
  Graphische Darstellung unterstützt das schnelle Verständnis
- Nachteil:
  Oft gibt es keine eindeutige Definition, was Kästen/Pfeile bedeuten
- Die meisten ADLs unterstützen eine graphische Darstellung

Packing robot control system
[Sommerville]



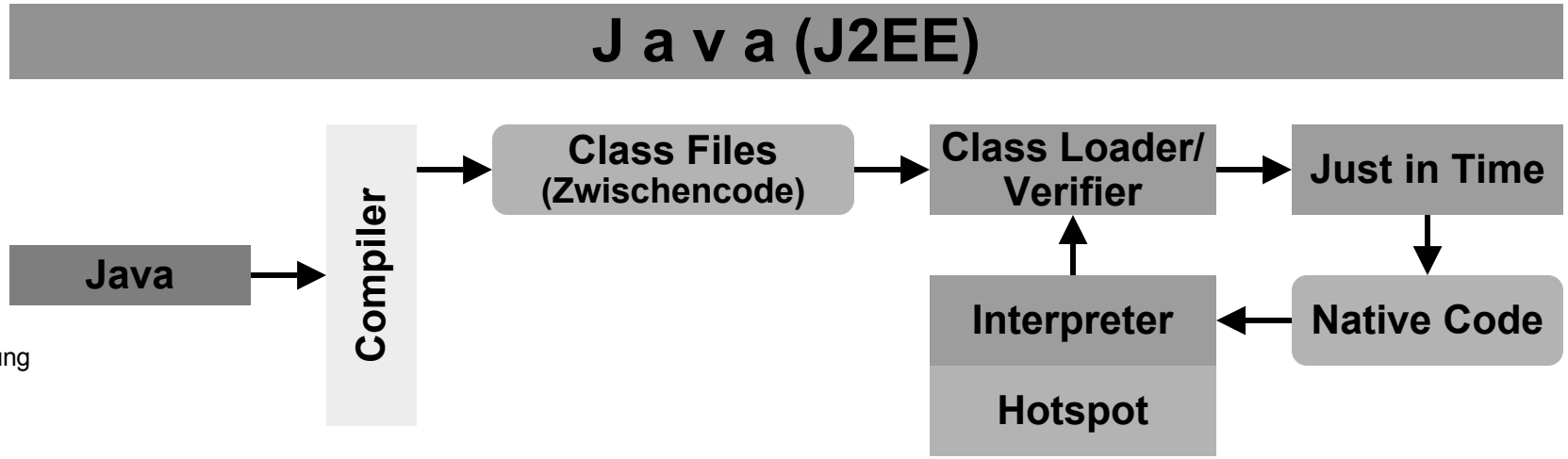


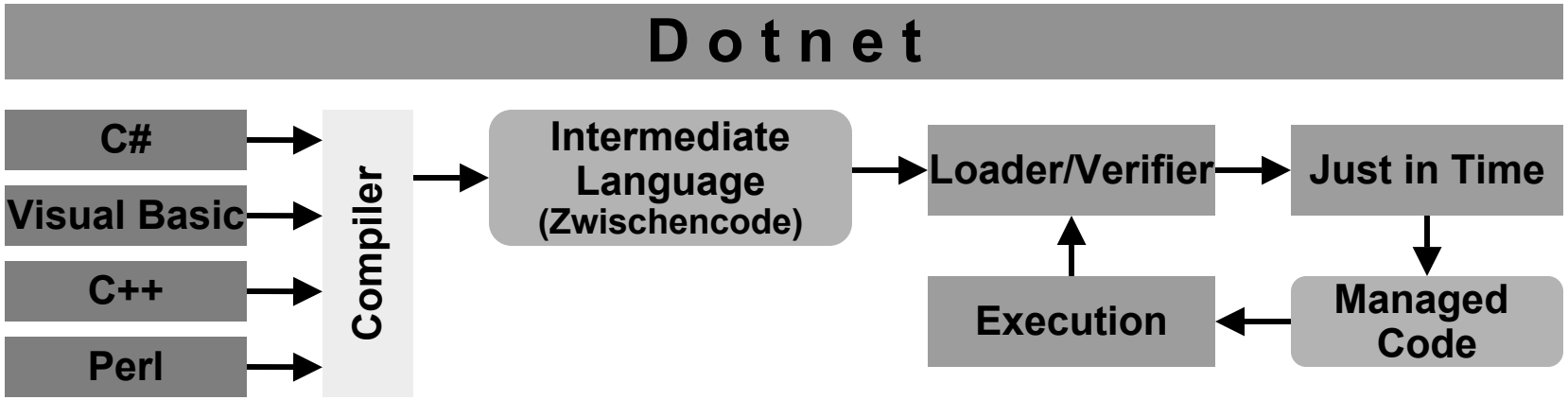
Beispiel für "Box and Lines"-Notation



J2EE als Vorlage für MS-Dotnet (.net)

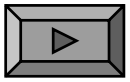
[COMPUTER ZEITUNG 22/2002]

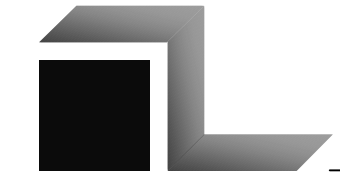




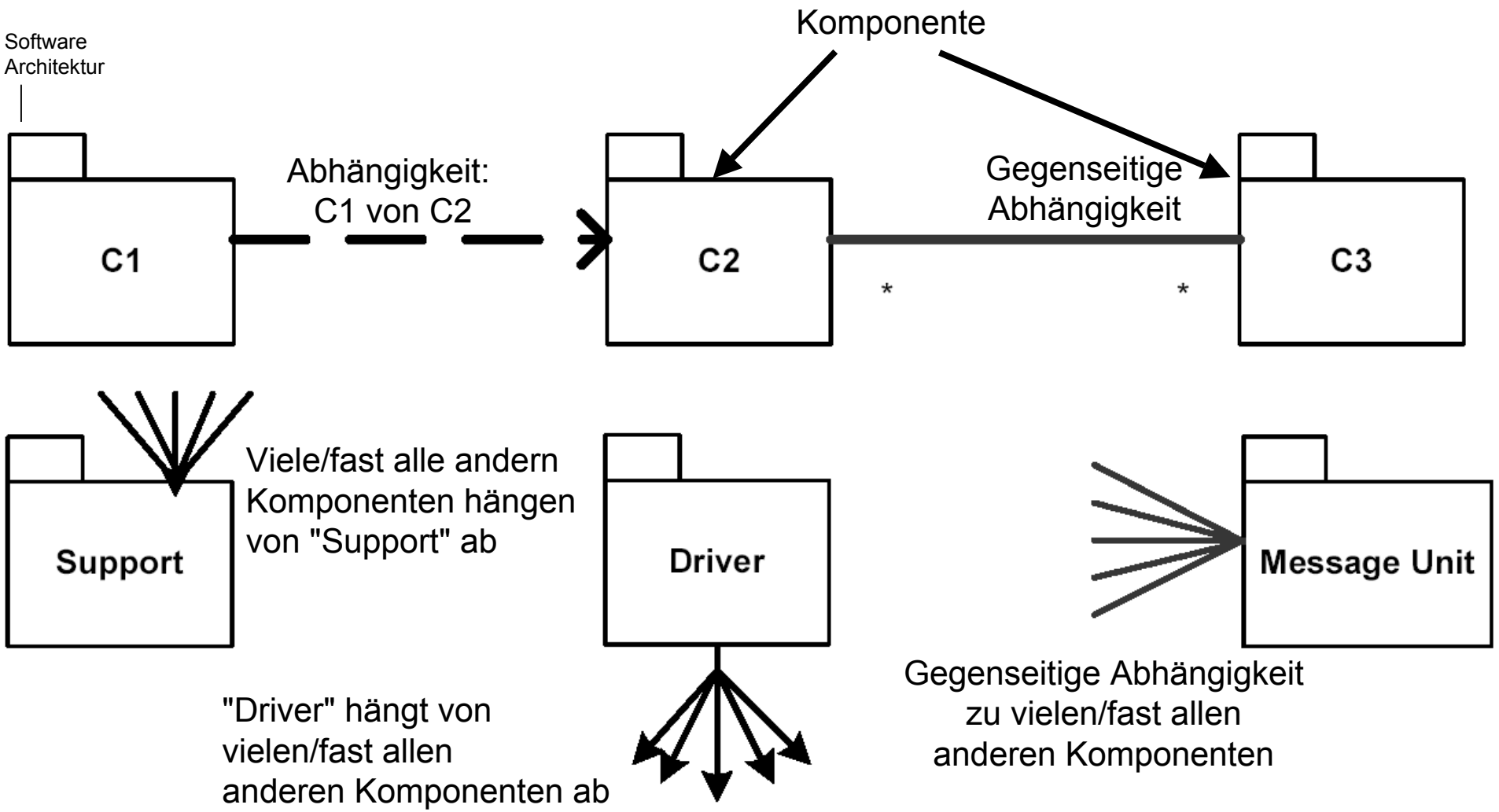
Große Software-Systeme

System	Domäne	Größe [kLoC]	Anzahl der Dateien	Anzahl der Komponenten (Top-Level)	Anzahl der Level
tcsh	Unix Shell				
Apache	Web Server				
AOL Server	Web Server				
Linux Kernel	OS				



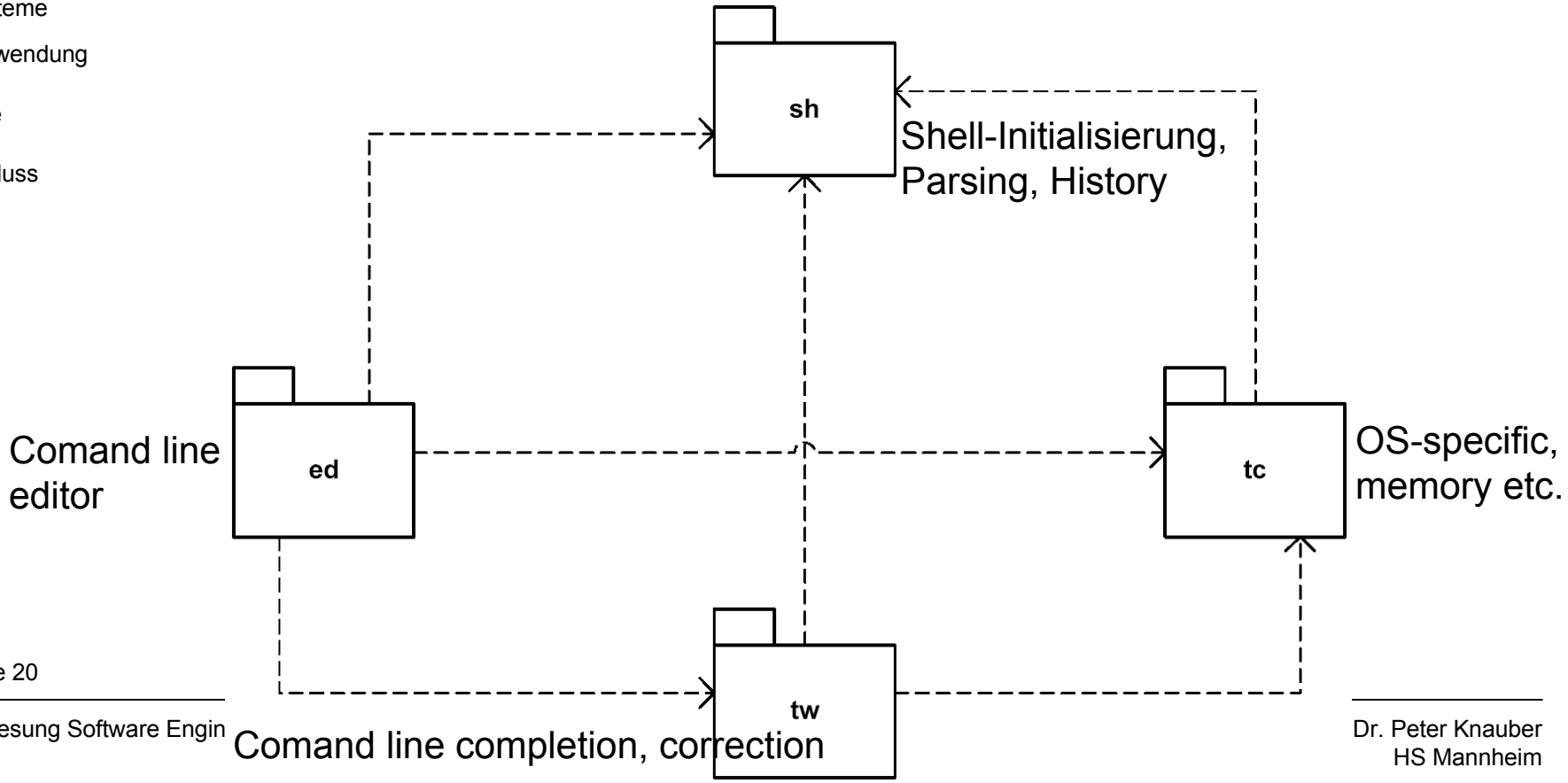


Eine mögliche Notation für die Darstellung der (statischen) Architektur



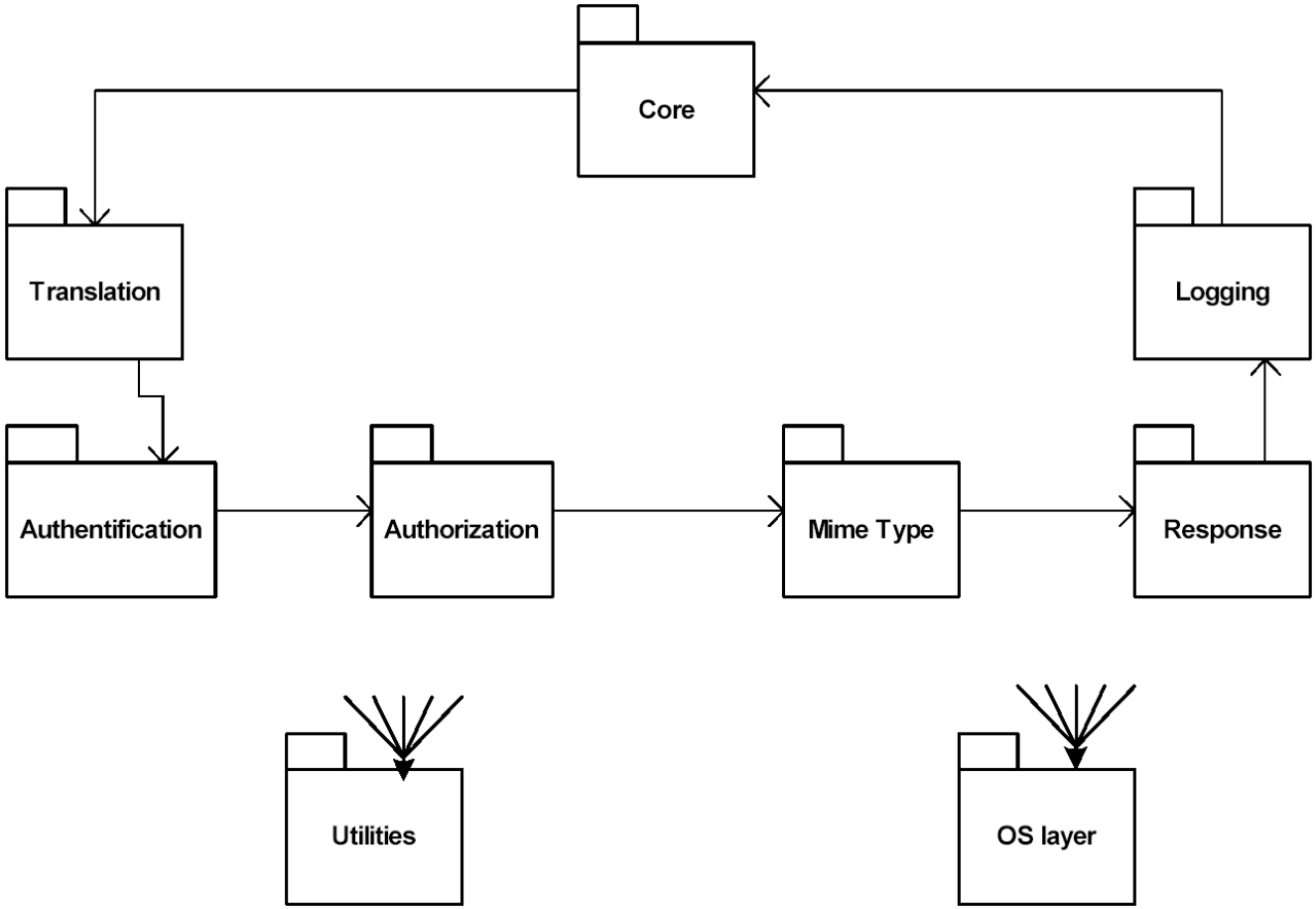
Architektur großer Systeme: tcsh – Unix Shell

- 51 kLoC, in C programmiert
- 52 Dateien
- 5 Komponenten (4 auf Top-Level, 2 Levels)



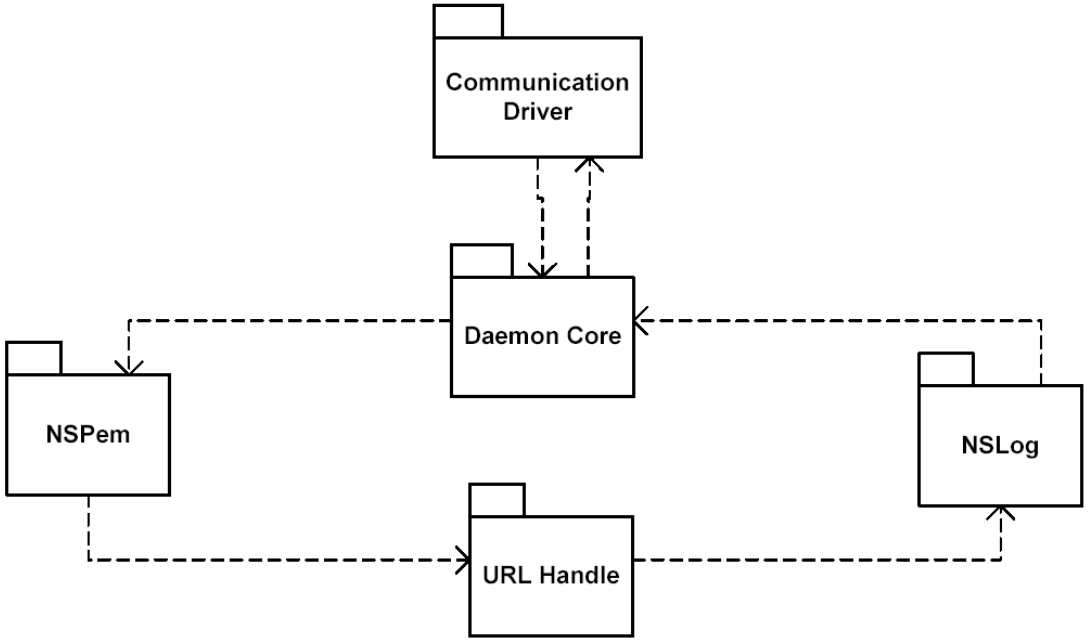
Architektur großer Systeme: Apache – Web Server

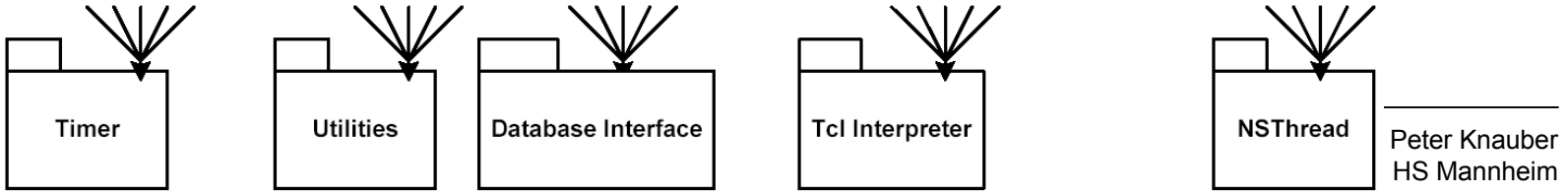
- 80 kLoC, komplett in C programmiert
- 75 Dateien
- 21 Komponenten (9 auf Top-Level, 2 Levels)



Architektur großer Systeme: AOL Server – Web Server

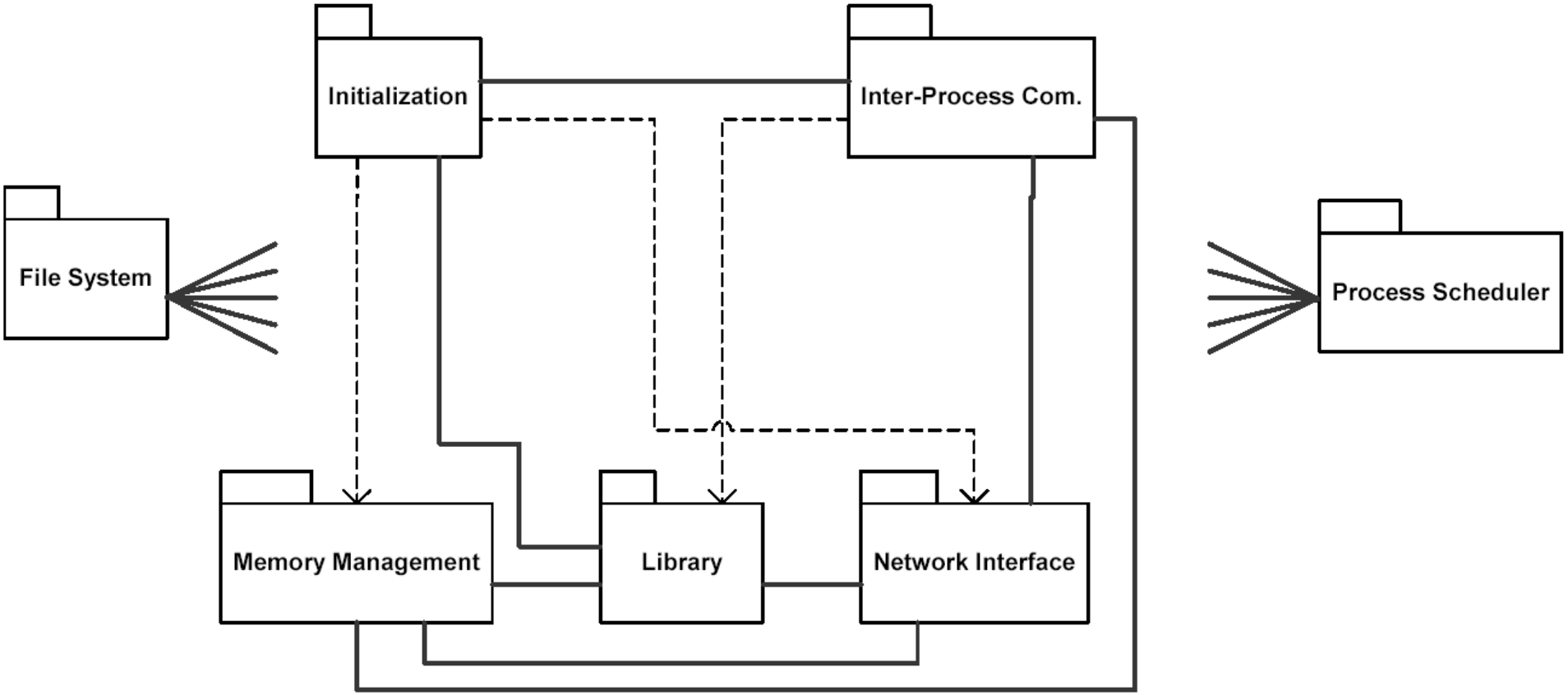
- 164 kLoC, in C programmiert bis auf 4 kLoC Tcl
- 89 Dateien
- 22 Komponenten (10 auf Top-Level, 3 Levels)





Architektur großer Systeme: Linux Kernel

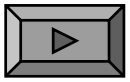
- 800 kLoC in C programmiert
- 557 Dateien
- 128 Komponenten (7 auf Top-Level, 5 Levels)



Fragen?

Partner-Diskussion: Architektur-Verwendung

- Diskutieren Sie mit einem Partner
  - Welche Gründe für die explizite Dokumentation einer Software-Architektur kennen Sie?
  - Welche davon halten Sie für wichtig, welche für weniger wichtig?
- Dauer: 3 Minuten



Design-Noten

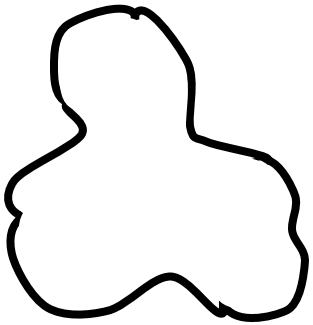
Produkt	A-Team	B-Team	C-Team
Notate	→ Notes (IBM): Team-Organisation/-Kollaboration		
PMill	→ PageMill (Adobe): Web Site/Page Builder		
Paint-It	→ Painter (Corel): Grafik-Software		
P-Shop	→ Photo-Shop (Adobe): Photo Editing		
Quirk	→ Quark-Xpress (Quark): Desktop Publishing		
Quickerstill	→ Quicken (Intuit): Finanzplanung		

Design-Noten

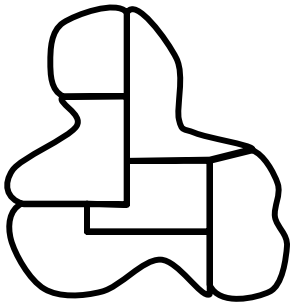
Produkt	A-Team	B-Team	C-Team
Notate	6	1	1
PMill	6	1	1
Paint-It	6	1	2
P-Shop	6	1	1
Quirk	6	2	1
Quickerstill	3	1	1

Das Ganze – Die Teile

Das Ganze

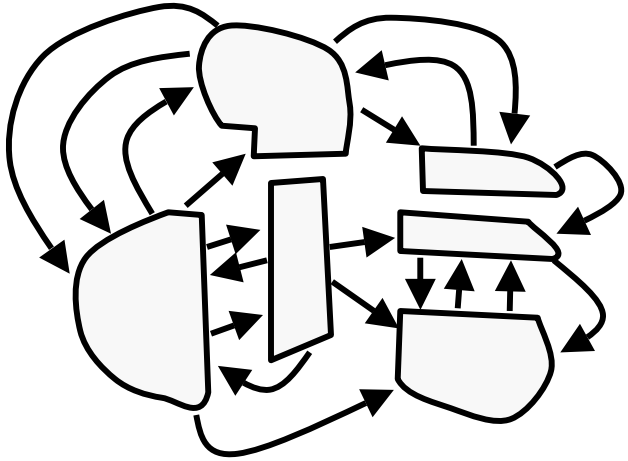


Die Teile

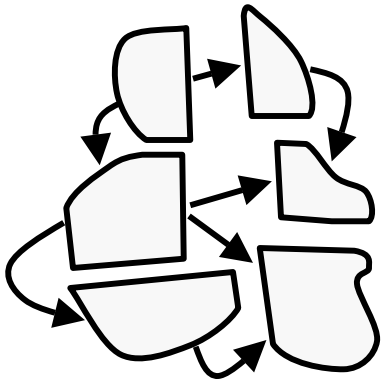


Diese Unterteilung – oder diese?

Diese Unterteilung?

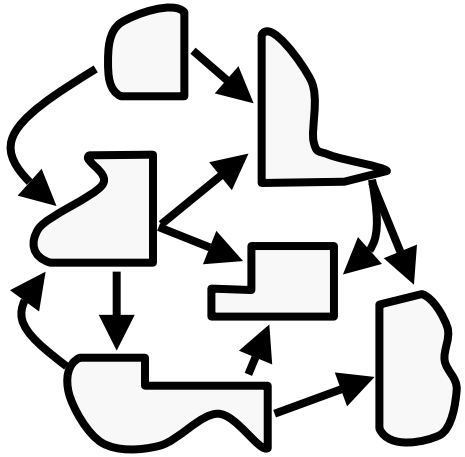


Oder diese?

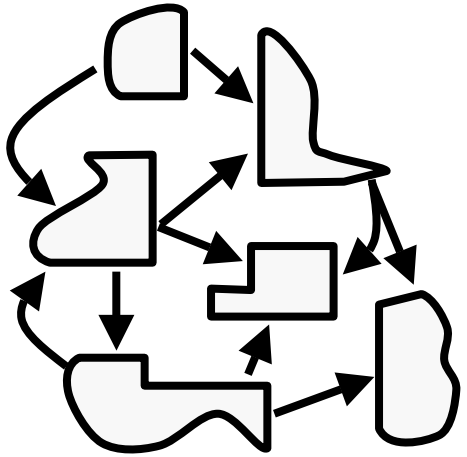


Teile des Produkts – Teile des Projekts

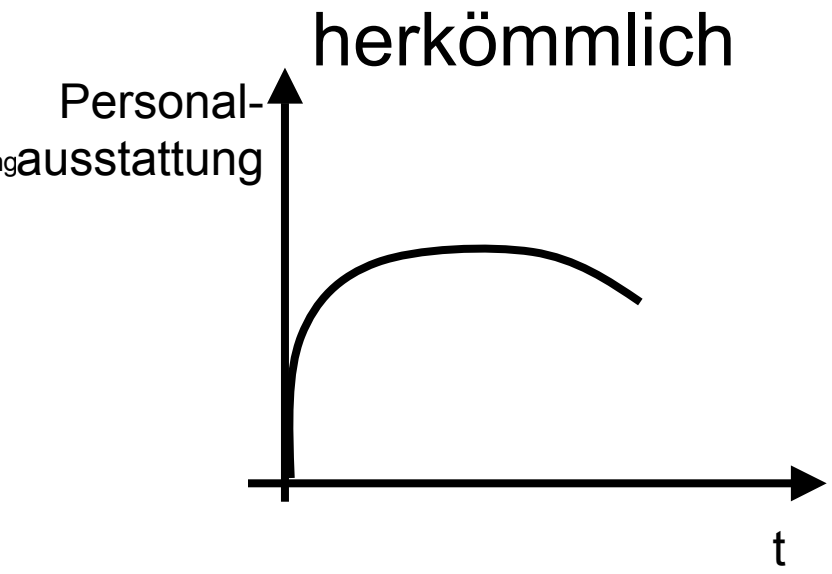
Teile des Produkts

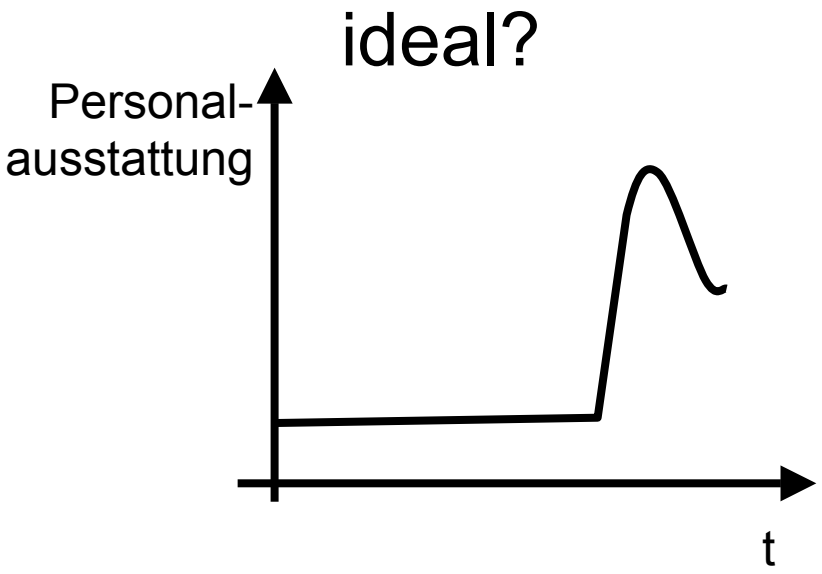


Teile des Projekts



Personalausstattung: herkömmlich – ideal?





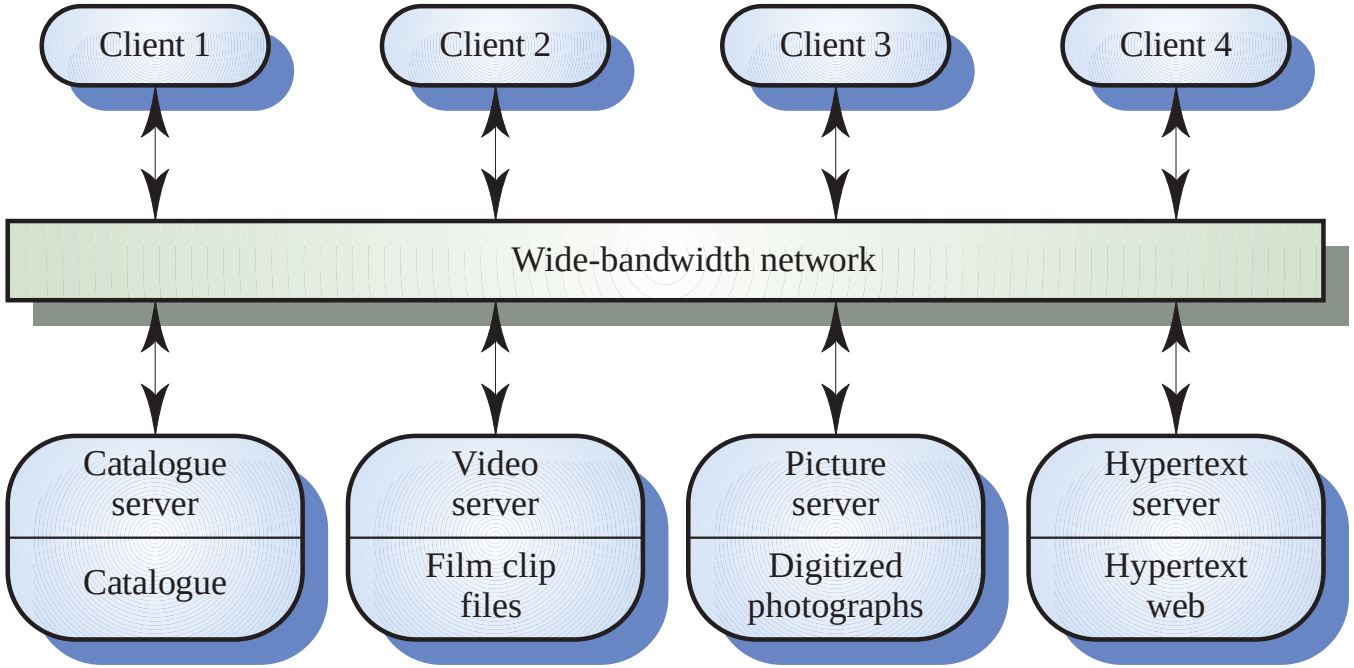
Fragen?

Architektur-Stile

- Ein Architektur-Stil definiert die Grundlage für das Design
- Jeder Stil bietet ein bestimmtes *Design-Vokabular* für eine Familie ähnlicher Systeme zusammen mit den geeigneten Regeln und Anwendungen
- Beispiele
  - Client-Server
  - Blackboard / Repository
  - Pipes and Filter
  - Layer / Abstrakte Maschinen
  - Problem-spezifische Stile für spezifische Domänen

Architektur-Stile: Client-Server

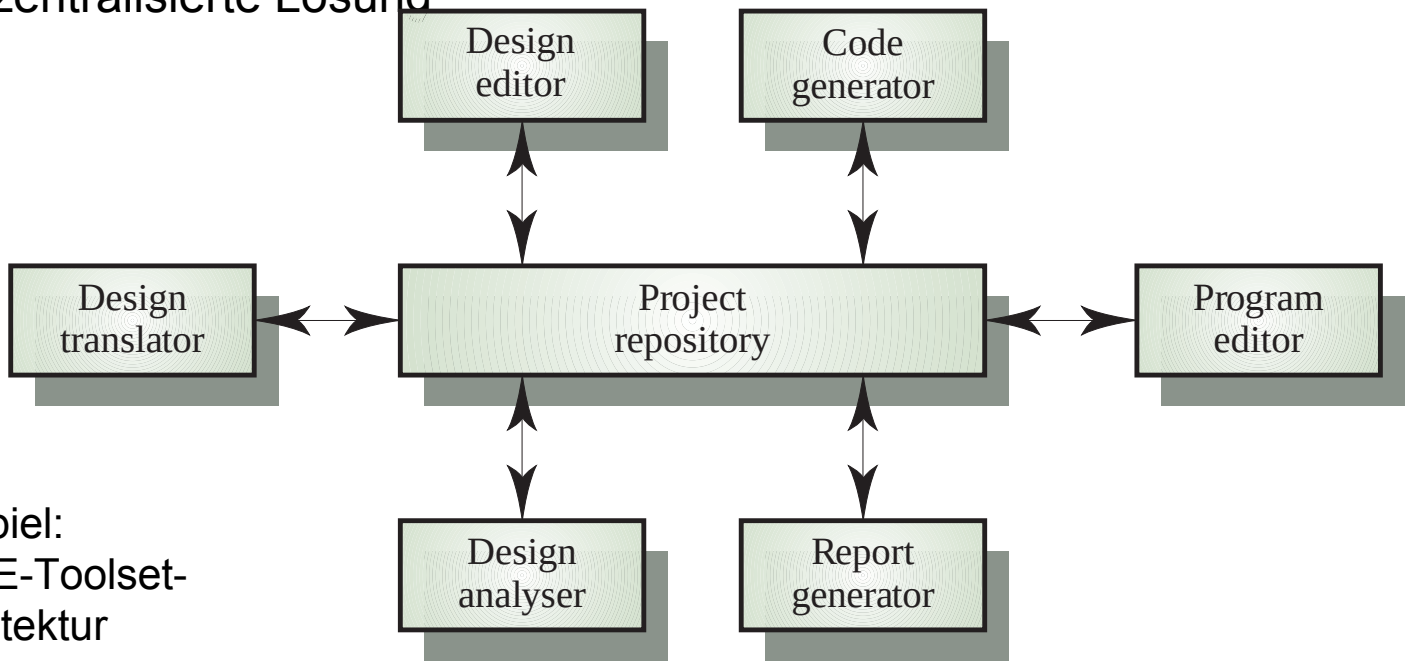
- Einsatzgebiet:
  Datenhaltung und Rechenaufgaben sollen auf verschiedene Rechner verteilt werden; Lösung:
  - Verschiedene (eigenständige) Server bieten verschiedene Dienste an (Drucken, Datenmanagement, Backup etc.)
  - Eine Reihe von (wechselnden) Clients (Klienten) nutzen diese Dienste
  - Ein Netzwerk erlaubt den Clients den Zugang zu den Servern



Beispiel: Film- und Bild-Archiv-Architektur

Architektur-Stile: Blackboard / Repository

- Einsatzgebiet:
  Subsysteme müssen Daten austauschen; 2 Lösungsalternativen
  - Die Subsysteme teilen sich ihre Daten in einem zentralen Pool (Repository, Blackboard)
  - Jedes Subsystem hält seine eigenen Daten und reicht notwendige Daten explizit an andere Subsysteme weiter
- Aus Performanzgründen wählt man bei großen Datenmengen die zentralisierte Lösung

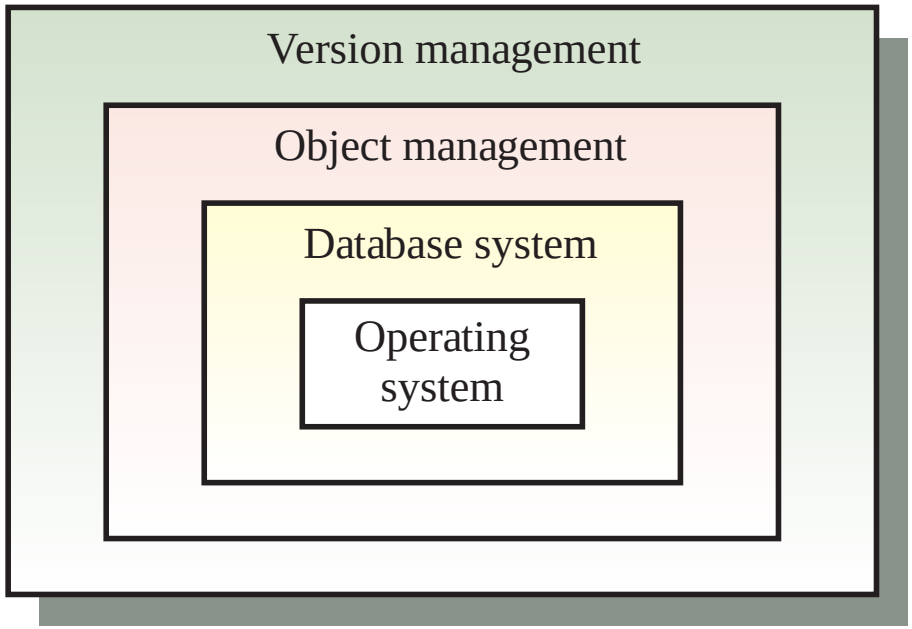


Beispiel: CASE-Toolset-Architektur

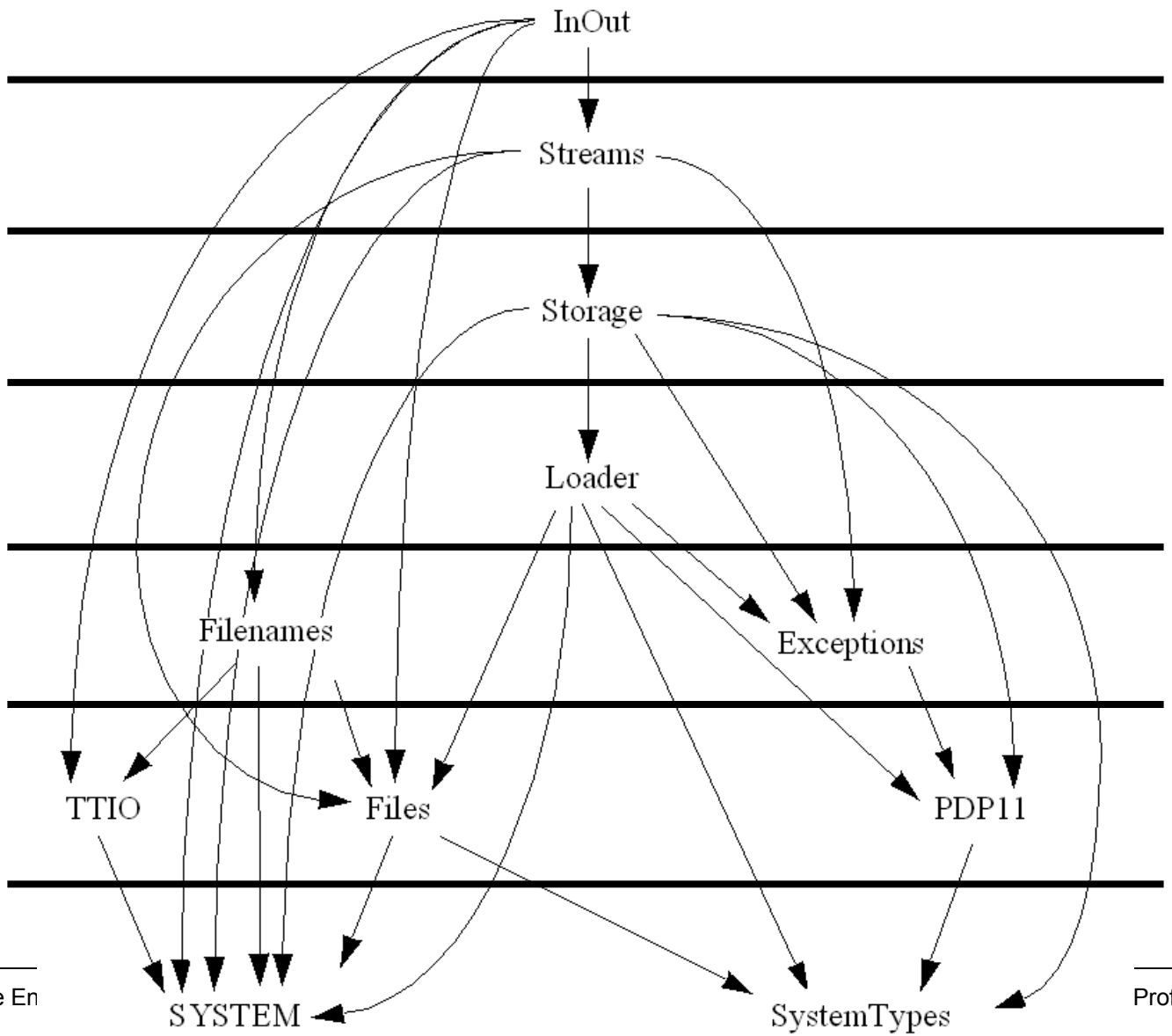
Architektur-Stile: Layer / Abstrakte Maschinen

- Einsatzgebiet:
  Schrittweise Abstraktion durch Schichten (Layer, abstrakte Maschinen)
- Jede Schicht bietet bestimmte Dienste
- Ändern sich die Schnittstellen einer Schicht, ist nur die benachbarte betroffen
- Sehr vorteilhaft für inkrementelle Entwicklung

Beispiel: Versions-Management-System



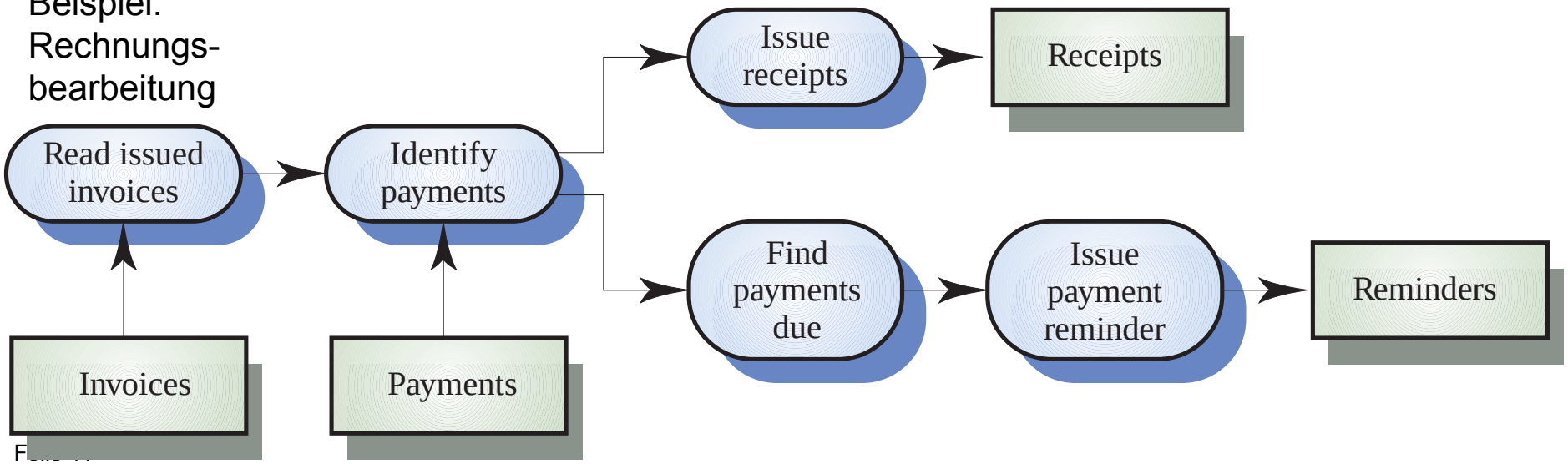
Beispiel für den Layer-Architektur-Stil: Ein- / Ausgabe - Module in MODULA-2



Architektur-Stile: Pipe and Filter

- Einsatz:
  Transformatoren überführen Eingaben in Ausgaben
- Sequenzielle Transformationen entsprechen einem Batch-Betrieb
- Unbrauchbar für interaktive Systeme
- Beispiel: Text-Transformationen in UNIX-Shells (awk, sed, perl)

Beispiel: Rechnungs-bearbeitung



Fragen?

Schlussbemerkung

- Eine gute Architektur kann nicht garantieren, dass ein System, das auf Basis dieser Architektur implementiert wird, die gestellten Anforderungen erfüllt
- Eine schlechte Architektur kann es jedoch unmöglich machen, die gestellten Anforderungen zu erfüllen
- Architekturanalyse kann dazu beitragen, potenzielle Schwachpunkte in einer Architektur zu entdecken und zu vermeiden