Software-Architektur (1)

Vorlesung: Software-Engineering für große, betriebliche Informationssysteme

für Universität Leipzig, Sommersemester 2004

Institut für Software- und Systementwicklung
Professur Wirtschaftsinformatik, insbes. Informationsmanagement

Hans Hartmann (Generali VIS Informatik Ges.m.b.H., Wien)
Wolfgang Keller (AMB Generali Informatik Services GmbH, Aachen)

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

- Motivation
- Einführung, Begriffe
- Architektursichten
  - Facharchitektur
  - Anwendungsarchitektur
  - Systemarchitektur
- Architekturmuster
  - Schichten-Architektur
  - Fat-Client, weitere Schnitte
- Distribution
  - Phoenix (Fat Client Smalltalk)
  - Generali Bank

Motivation:

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
- Wissen Sie, dass man für das Design von großen Projekten noch viel mehr Entwurfsmuster benötigt, als die heute vorgestellten
- und freuen sich hoffentlich auf den Teil Software Architektur 2

Motivation:

Ein typisches EDV-Großprojekt ..

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

Motivation:

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?

Motivation:

Warum braucht man Software-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

Einführung / Begriffe:

Software-Architektur

- Definition
- Nichtfunktionale Anforderungen
- Architektonische Stile & Entwurfsmuster
- Balancieren von Kräften

Einführung / Begriffe:

Definition 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 :-)

Fach-, Anwendungs-, Systemarchitektur ? Das ist alles ein bisschen komplizierter!

Facharchitektur

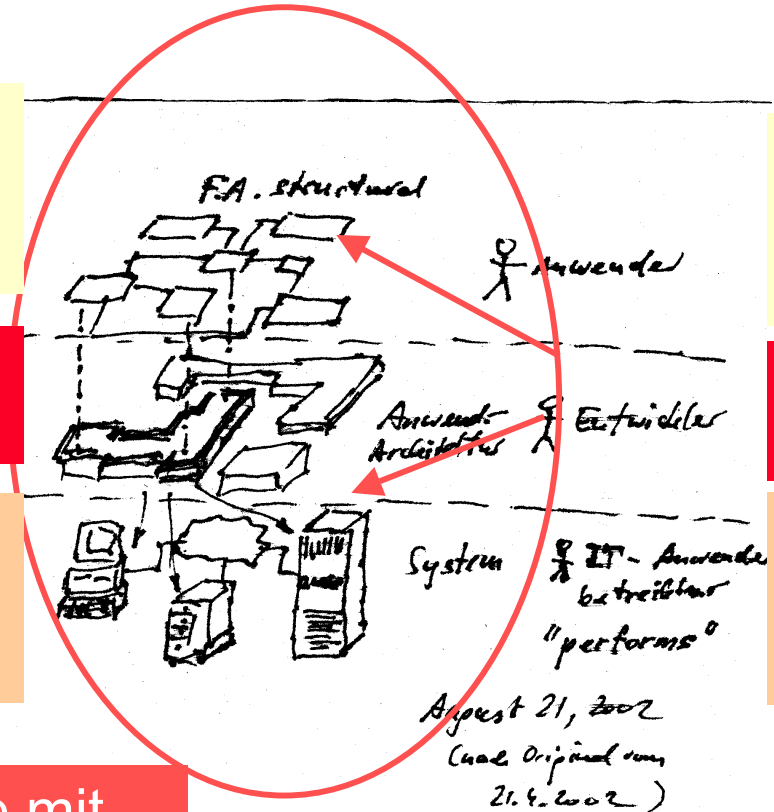
Abstrakte Systeme

Anwendungs-A.

Reale Komponenten

System-A.

Betreibbare K.



Anwendersicht

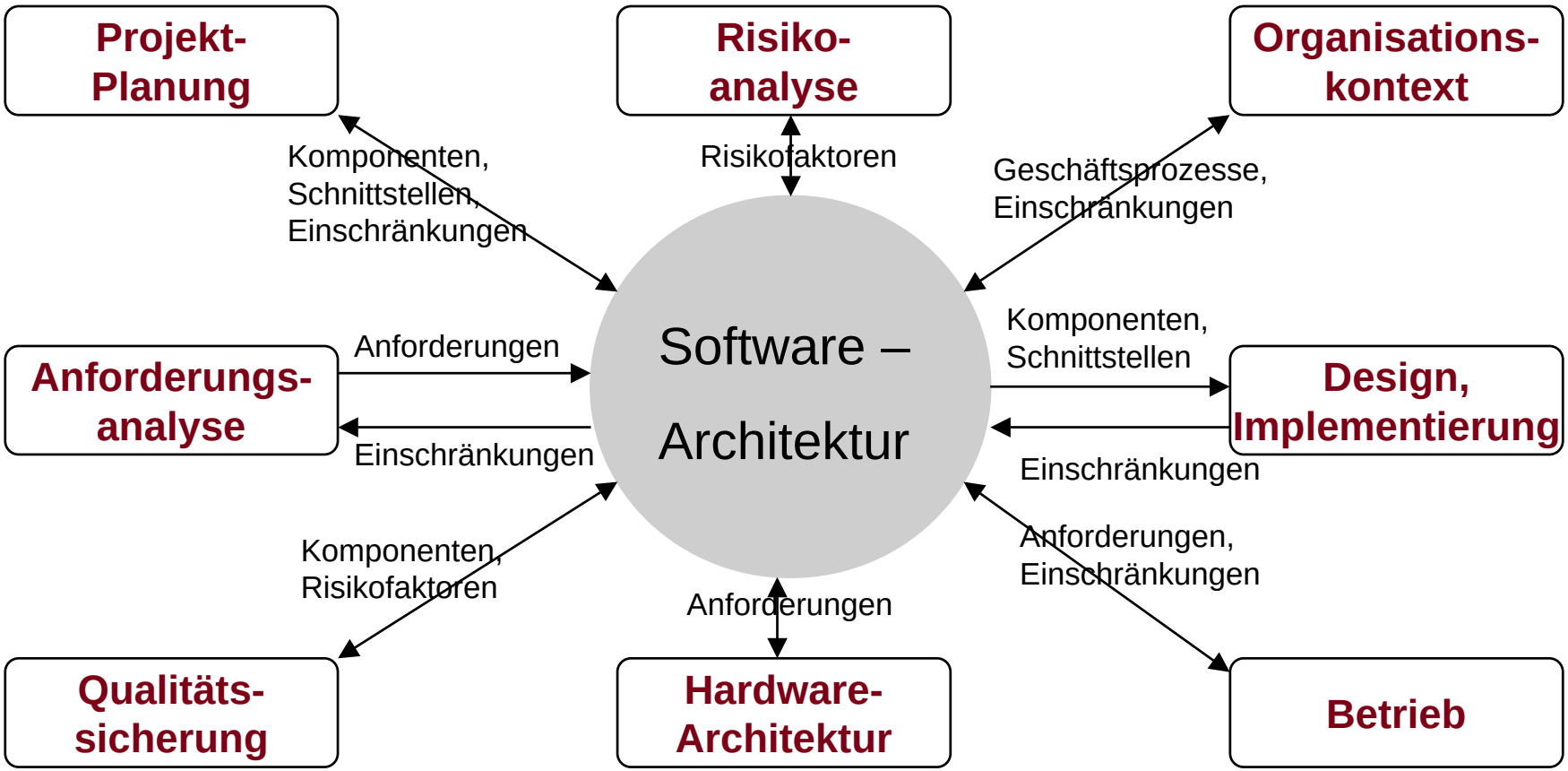
Entwicklersicht

Betreibersicht

Produkte mit niedrigen TCO

Eine Landschaft mit niedrigen TCO

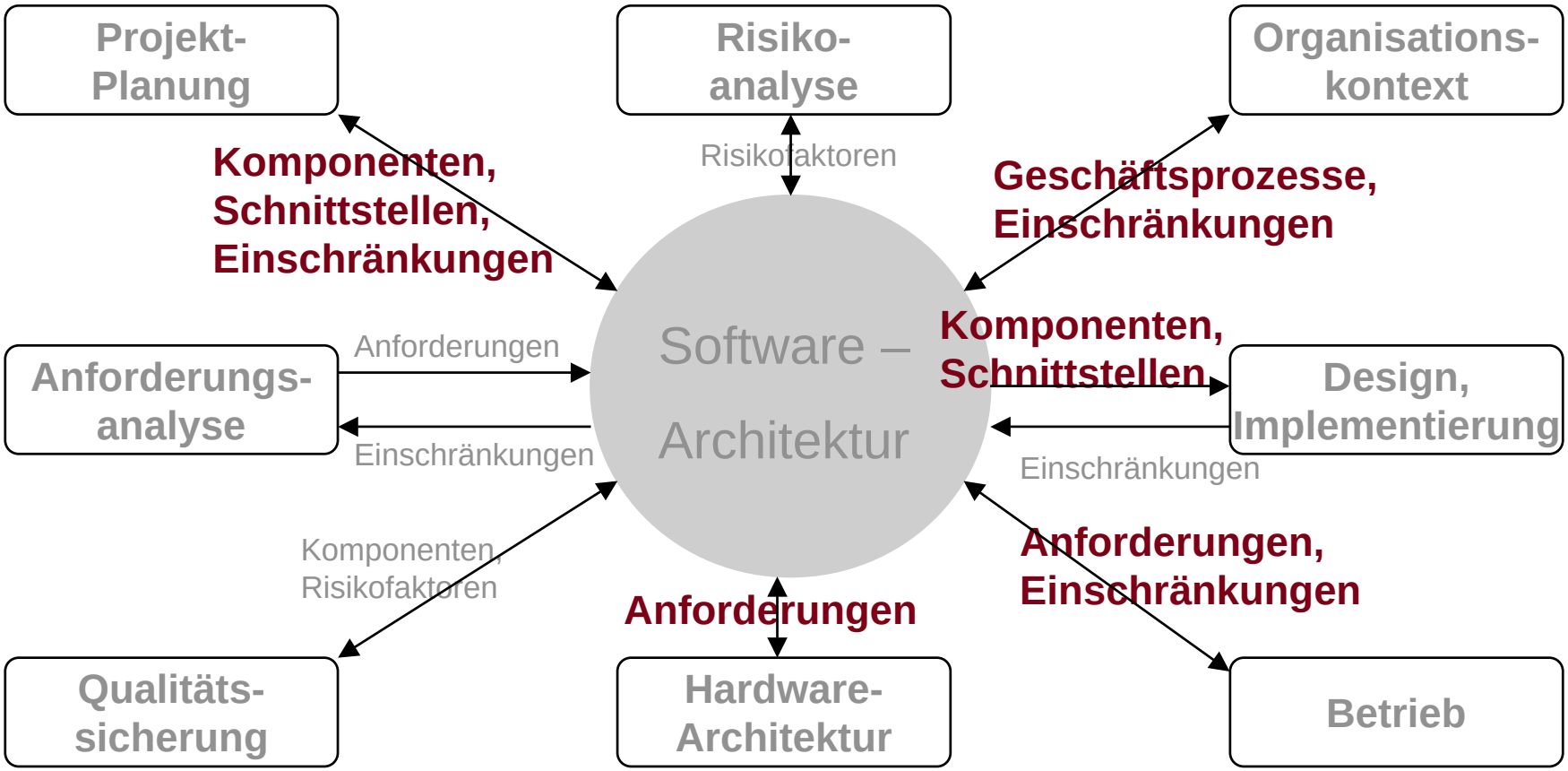
Softwarearchitektur spielt eine zentrale Rolle innerhalb der verschiedenen Entwicklungsaufgaben



Nach *Effektive Software-Architekturen*, Gernot Starke. Hanser,2002

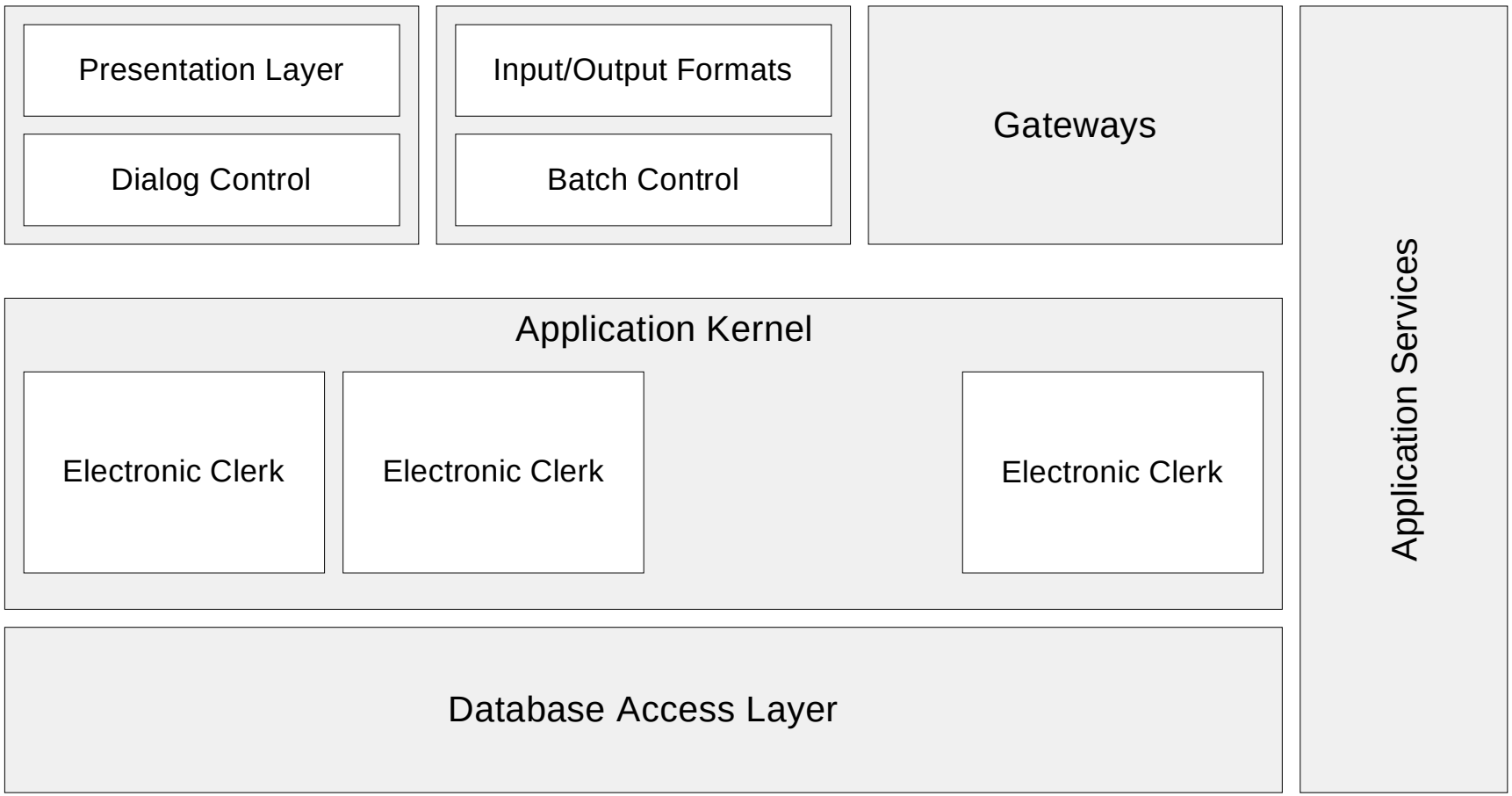
In der Praxis kann eine Architekturabteilung nicht immer alles leisten.

Aber es rächt sich ...



Einführung / Begriffe:

Beispiel: 3-Schichten-Architektur ...



Einführung / Begriffe:

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?

- Und was passiert, wenn ich im Unternehmen mehrere Software-Systeme verheiraten muss?

Software-Architektur spielt sich auf mehreren Ebenen ab.

- Mindestens die Anwendungsarchitektur und die Enterprise-Architektur müssen unterschieden werden.
- In der Folge kümmern wir uns um die Anwendungsarchitektur.

- Als Aufgabe bleibt: wo unterscheidet sich die Enterprise-Architektur von der Anwendungsarchitektur? (Übung)

Einführung / Begriffe:

Nichtfunktionale Anforderungen

Beispiele

- Performance
- Time to Market
- Gesamtkosten der Lösung
- Verfügbarkeit
- 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 ...

- Architekturstile
  - man verwendet bekannte „Architectural Styles“ und „Architectural Patterns“
- 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

Architekturstile:
ein kleiner Katalog

- Distributed
- Event driven
- Reflection
- Pipes & Filters
- Batch
- Blackboard

- Interpreter
- Rule Based
- Layered
- MVC (JSP Model 2)
- Broker
- Service Oriented Architecture

Architekturstile:

- 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)

Architekturstile:

Etwas formaler definiert - man benötigt ...

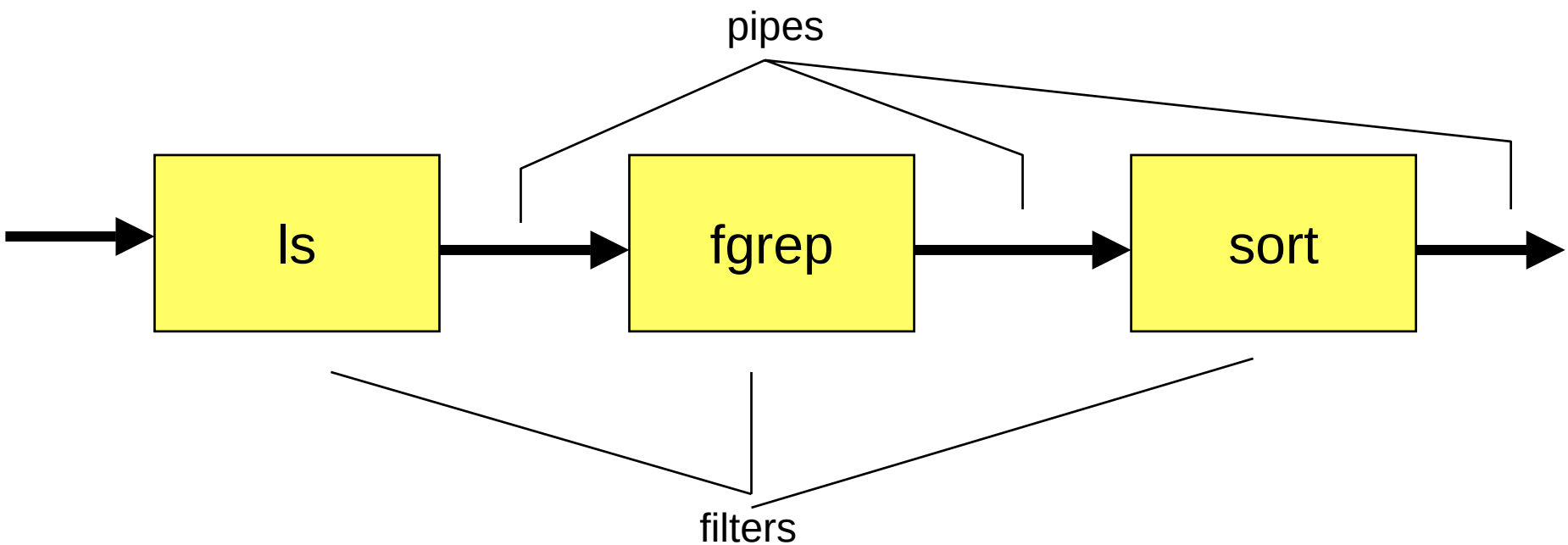
- Eine Menge von „Komponententypen“
  - Beispiel: Objekte organisiert in Schichten
- Eine topologische Anordnung dieser Komponenten
  - Schichten übereinander
- Eine Menge semantischer einschränkender Bedingungen (Constraints).
  - Es dürfen nur Services der nächst tieferen Schicht aufgerufen werden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

...und erhält z.B. den Style „layered architecture“

Architekturstile:

Pipes and Filters

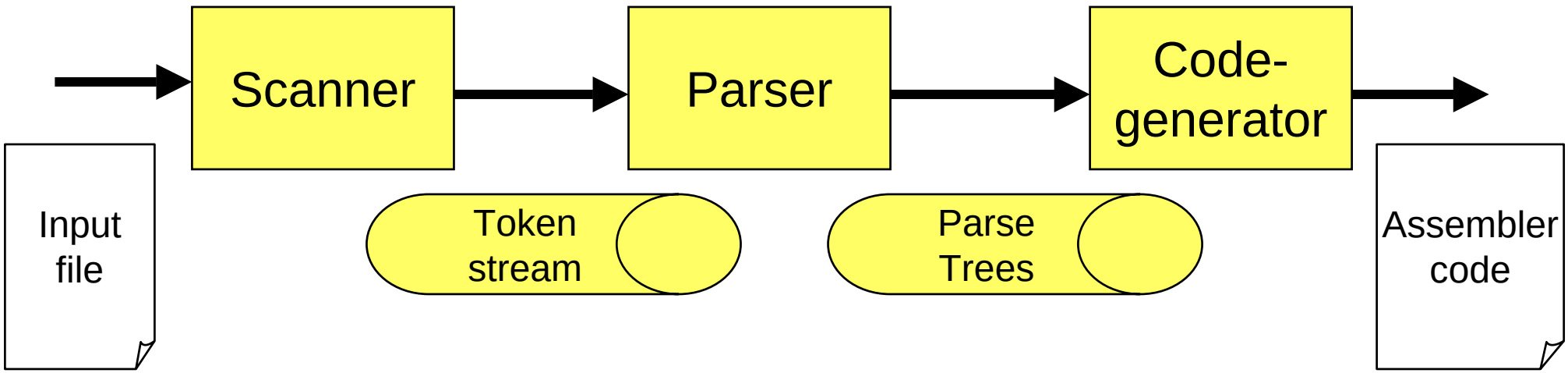
Aus Unix bekannt: ls -l | fgrep ‘acme‘ | sort



Architekturstile:

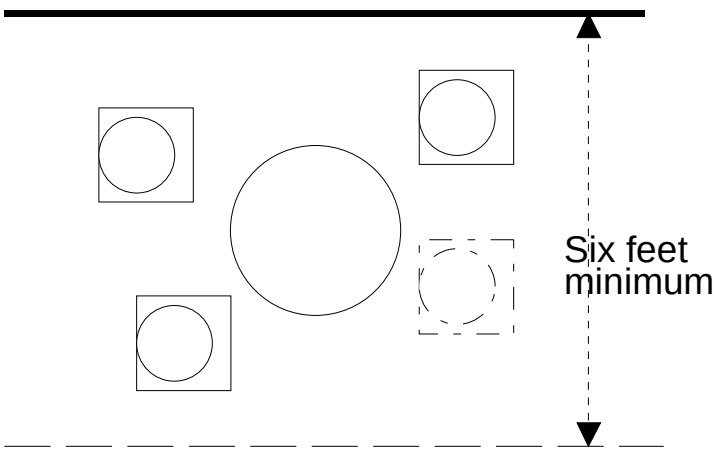
Pipes and Filters

...findet man auch in Compilern



Architekturstile:

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

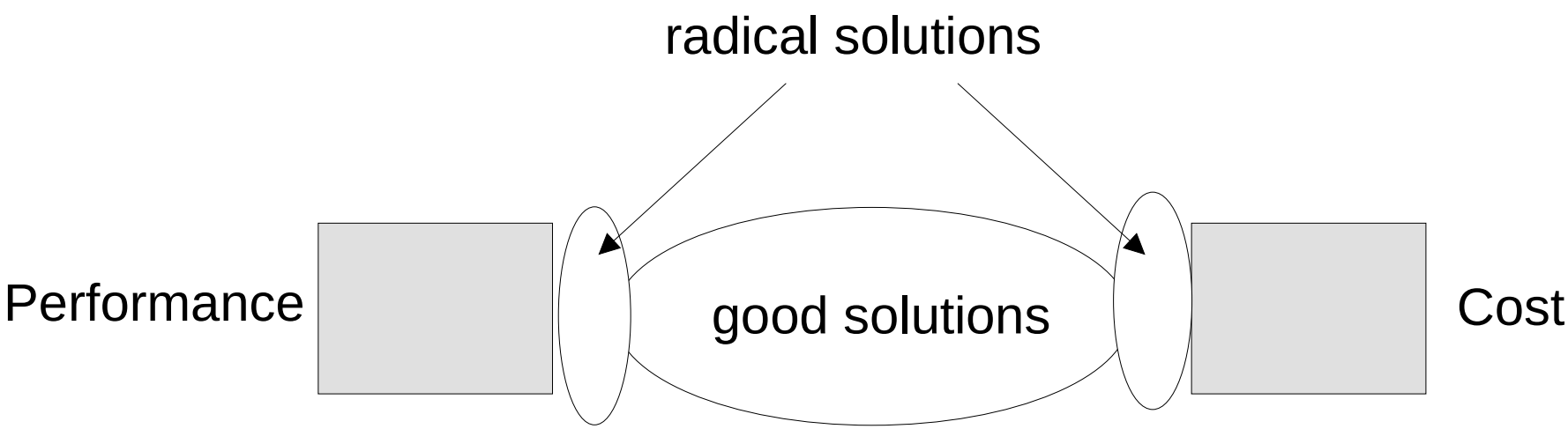
Balconies and porches made very small to save money; but when they are too small, they might as well not be there.
A balcony is only used properly when there is enough room for a small table where they can set down glasses, cups, and the newspaper. ...

Architekturstile:

Beispiele aus der Software-Architektur

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

Balancieren von Kräften

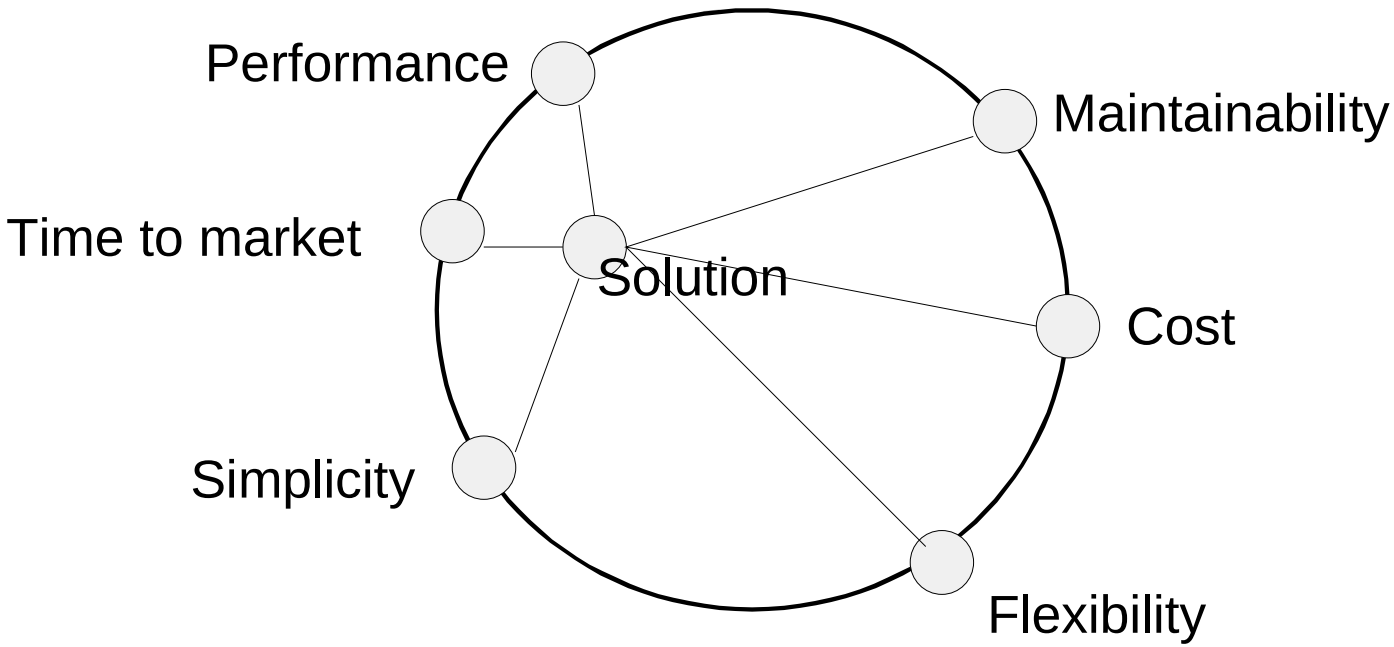


Balancing Forces is similar to politics: The most radical solution is seldom ever the best solution

Bad solutions can be found “near the walls”

Balancieren von Kräften

Conflicting Goals



Wie stellt man eine Software-Architektur dar?

- Kästchen und Pfeile - ist das alles?

Architektonische Sichten:

- Ein Architekt kennt nicht nur *eine* Planungsansicht von einem Haus, sondern es gibt z.B.
  - Gesamtansichten
  - Entwässerungspläne
  - Stromversorgungspläne
  - Statik-Pläne

- Analog gibt es auch verschiedene Sichten auf ein Software-System - man nennt diese „architektonische Sichten“ (architectural views)

Architektonische Sichten:

Beispiele Literatur

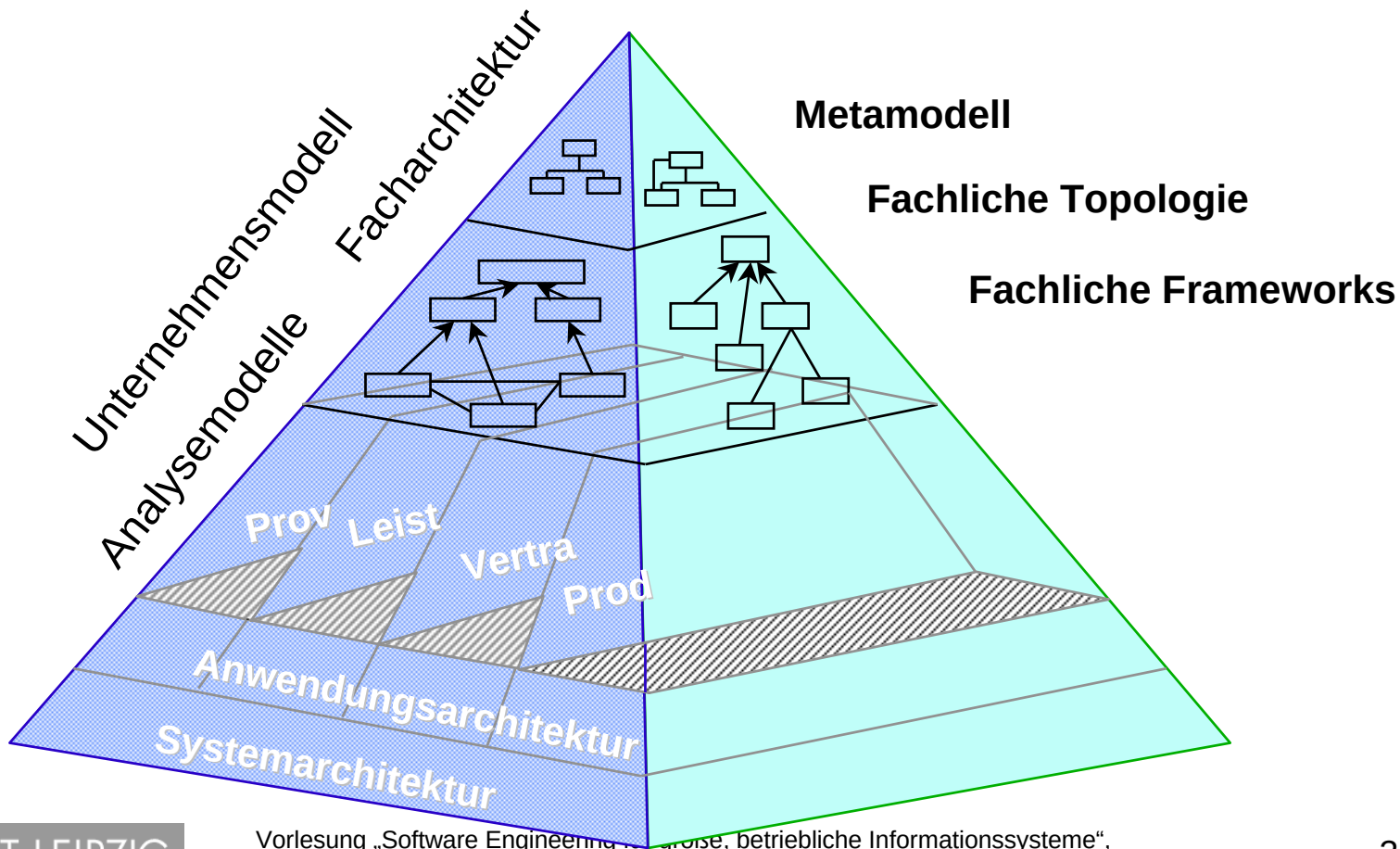
Architektonische Sicht	Gegenstände in der Sicht	Konnektoren
Konzeptuelle Architektur	Komponenten	allg. Beziehungen
Modul-Architektur	Subsysteme, Module	Export- / Import-Beziehungen
Code-Architektur	Files, Directories, Libraries	Include- / Contains-Beziehungen
Ausführungs-Architektur	Tasks, Threads	Inter-Prozess-Kommuniation,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

Architektonische Sichten

Generali

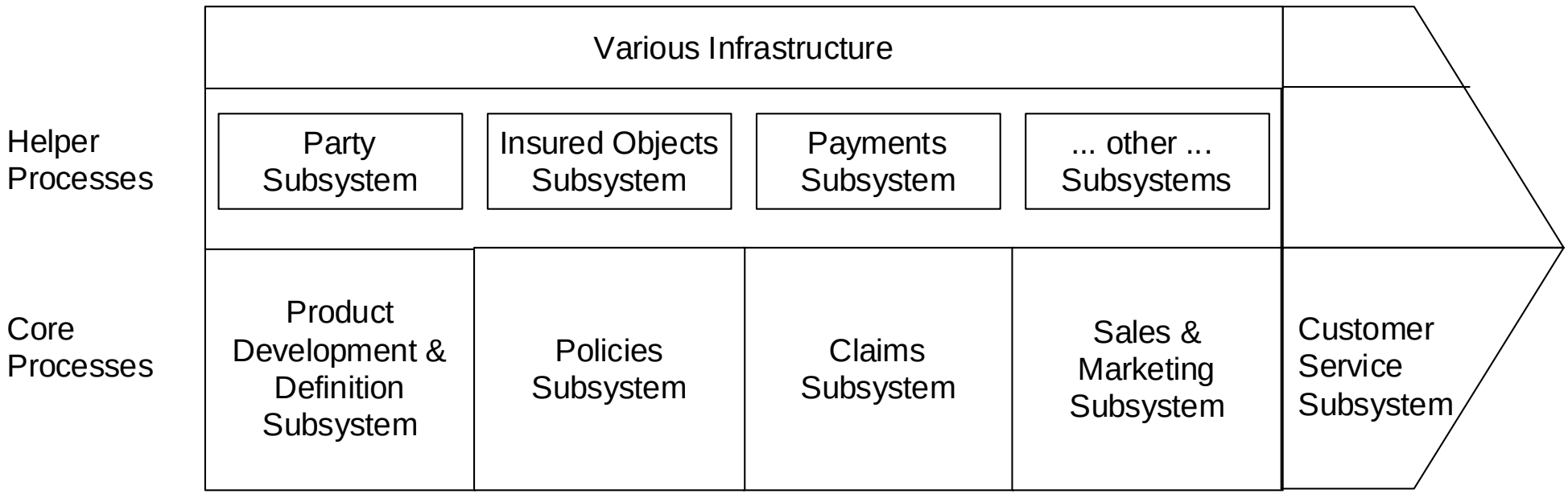
- Facharchitektur (Domain)
- Anwendungsarchitektur (SW Komponenten)
- Systemarchitektur (HW Geräte, Verteilung)

Architektonische Sichten
Beispiel Phoenix



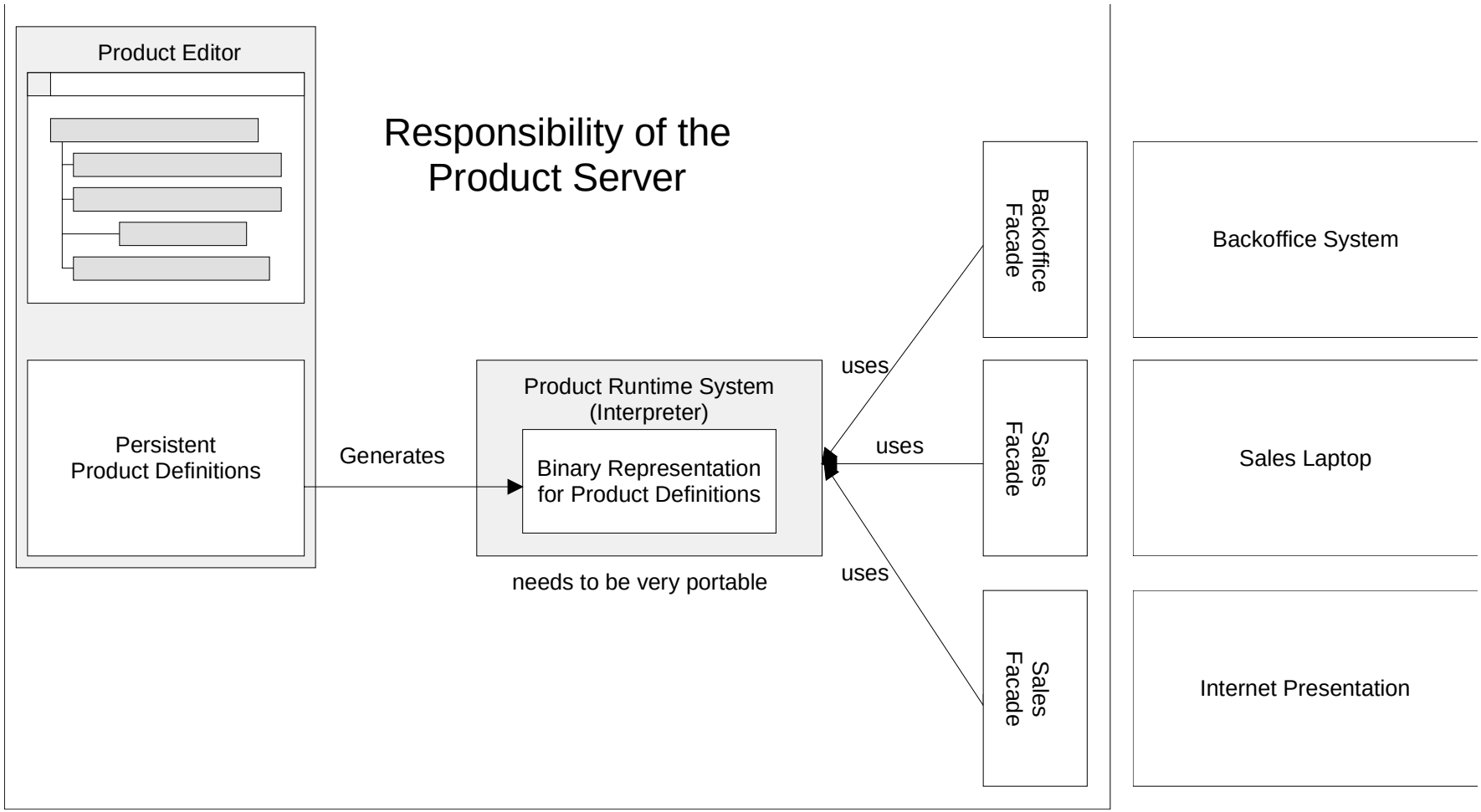
Facharchitektur

Beispiel: Wertkette für Versicherungen



Facharchitektur

Beispiel: Produktserver



Facharchitektur:

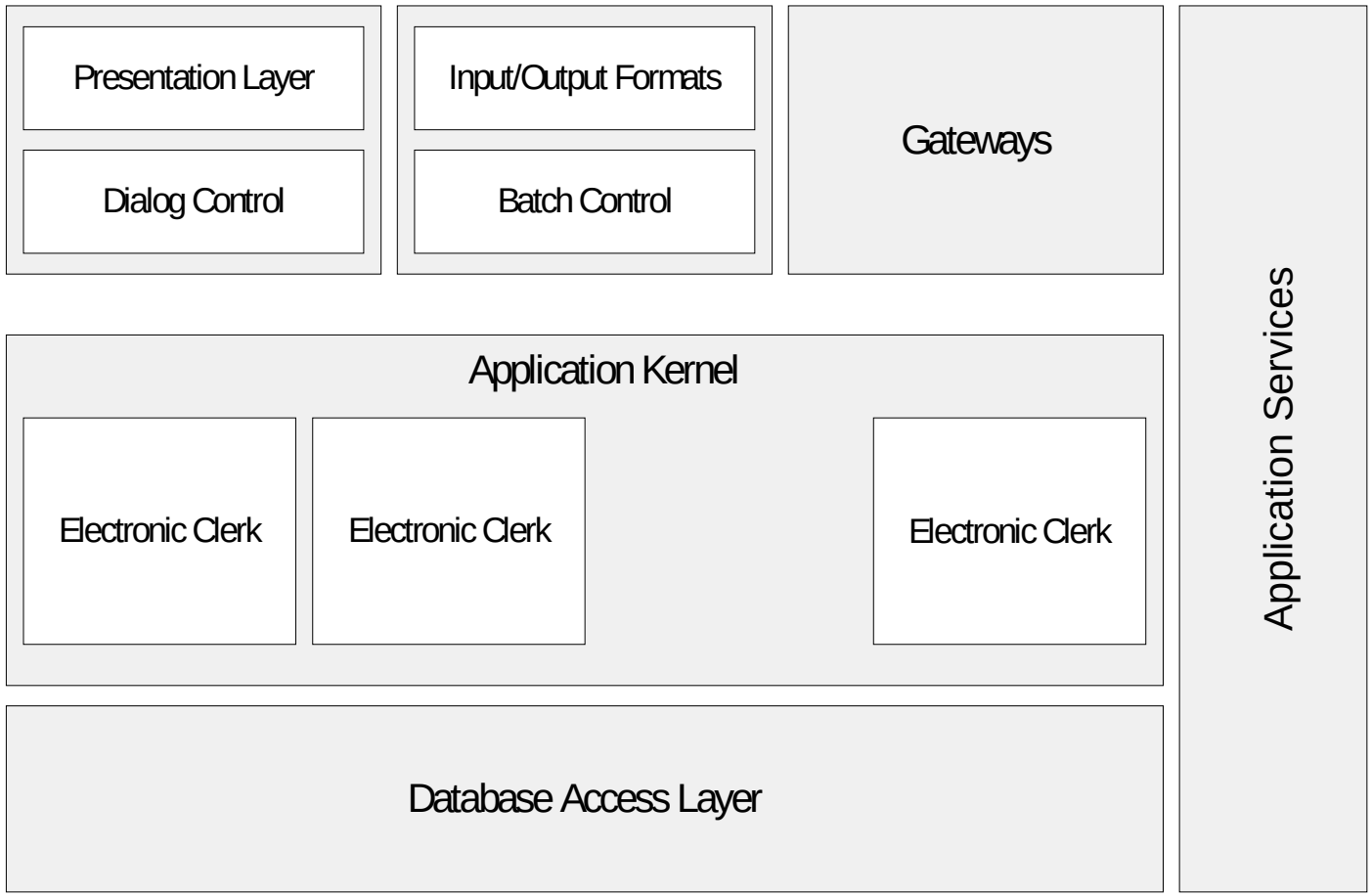
In der Facharchitektur finden Sie nicht ..

- Security
- Authorization
- Database Access Layer
- Dialog Control

Dafür aber in der Anwendungsarchitektur

Anwendungsarchitektur:

Beispiel: Anwendungskomponenten



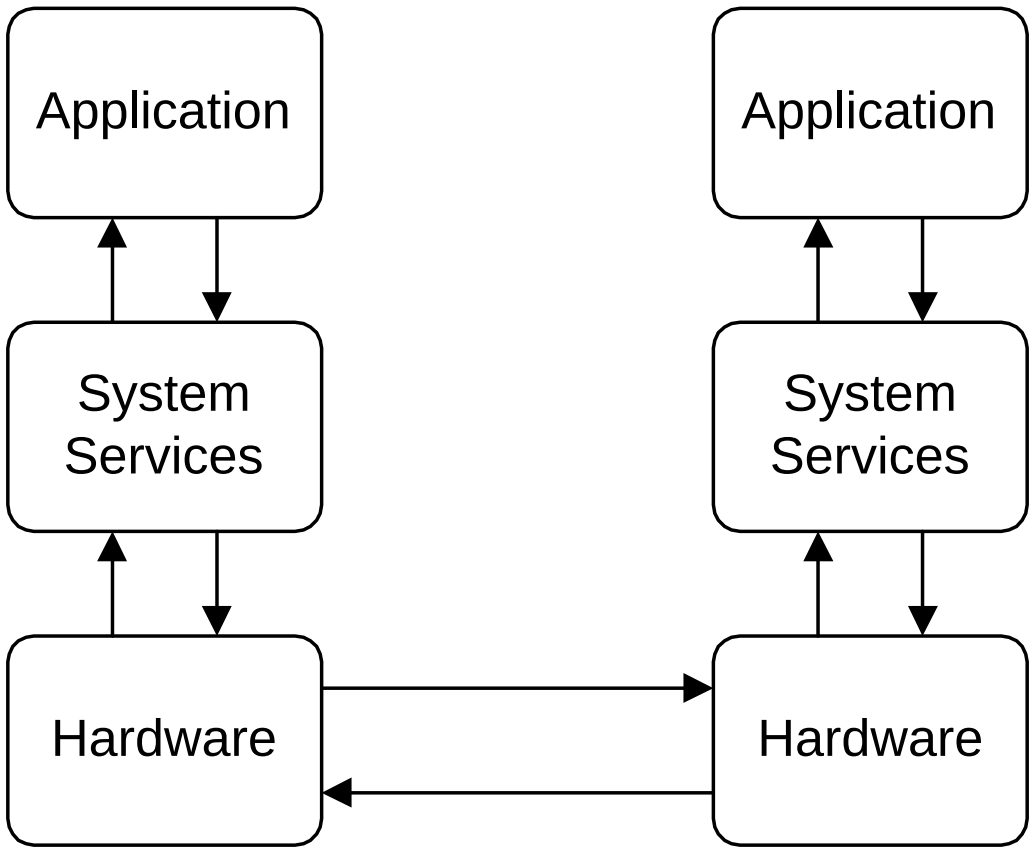
In der Anwendungsarchitektur finden Sie nicht ..

- Fat-Client
- Load Balancing
- Distribution

Dafür aber in der Systemarchitektur

Systemarchitektur:

Beispiel: Distribution

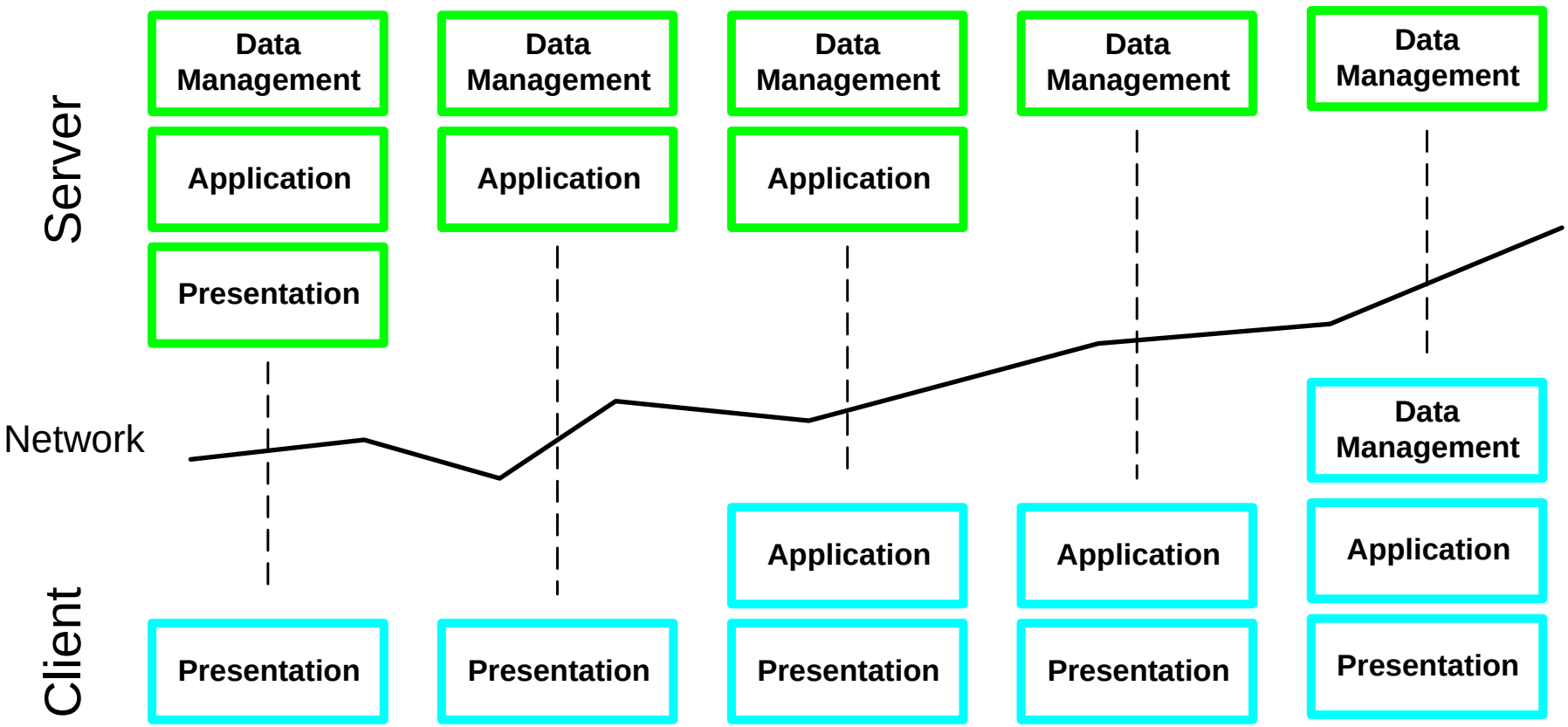


Systemarchitektur: Beispiel: Distribution

OSI Schicht	*Internet Protokoll Suite*						*DOD Schicht*
Anwendung	File Transfer	Electronic Mail	Terminal Emulation	Usenet News	Domain Name Service	Trivial File Transfer	Prozess / Applikation
Darstellung	File Transfer Protocol	Simple Mail Transfer Protocol	Telnet Protocol	Network News Transfer Protocol	Domain Name System	Trivial File Transfer Protocol	
Sitzung	(FTP) RFC 959	(SMTP) RFC 821	(Telnet) RFC 854	(NNTP) RFC 977	(DNS) RFC 1034	(TFTP) RFC 1350	
Transport	Transmission Control Protocol (TCP) RFC 793					User Datagram Protocol (UDP) RFC 768	Host-to-Host
Netzwerk	Address Resolution Protocol (ARP) RFC 826	Internet Protocol (IP) RFC 791				Internet Control Message Protocol RFC 792	Internet
Sicherung	Ethernet, Token Ring, DQDB (802.X), FDDI						lokales Netzwerk oder Netzzugriff
Bit-übertragung	Übertragungsmedium Doppelader, Koaxkabel, Lichtwellenleiter, drahtlose Übertragung						

Systemarchitektur:

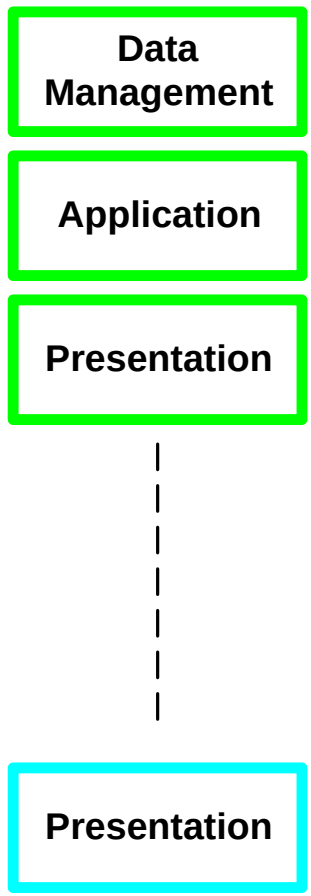
Gartner Styles für Distribution



Systemarchitektur:

Distributed Presentation

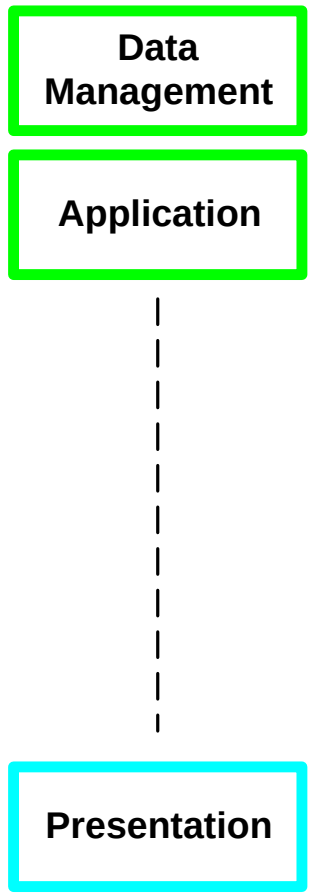
- X-Windows
- 3270 Terminal
- Web browser with HTML
- No drag / drop
- No responsive UI



Systemarchitektur:

Remote Presentation

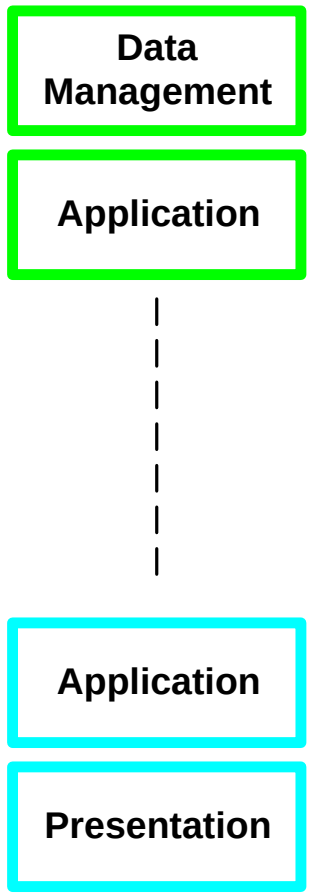
- Client is responsible for complete UI
- screen scraping



Systemarchitektur:

Distributed function

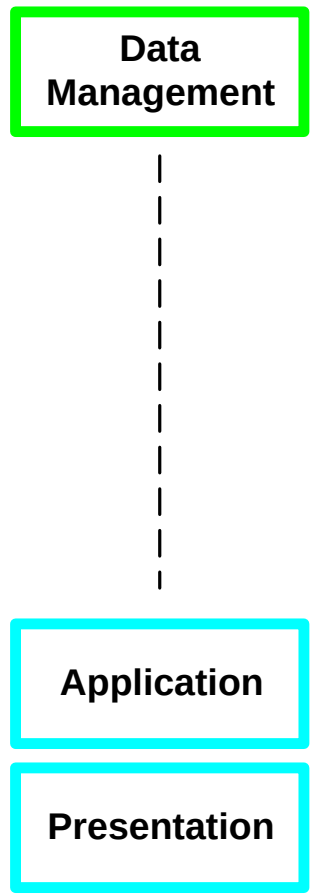
- Application logic split between client and server
- DCE (remote procedure call)
- Microsoft DCOM
- SOAP
- Java EJB / Applets
- Common Object Request Broker Architecture



Systemarchitektur:

Remote Data Access

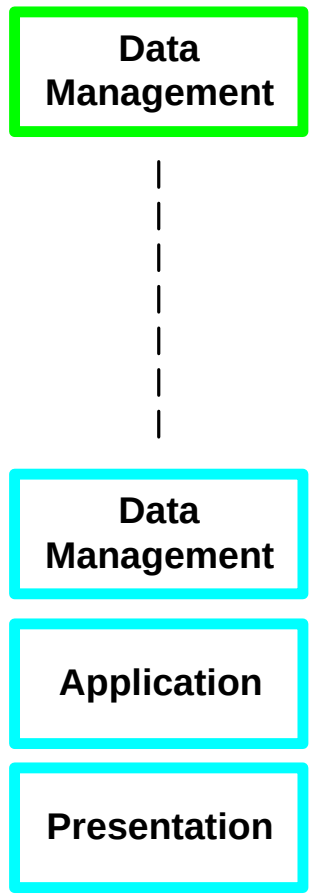
- Presentation and Logic on the Client (Fat Client)
- E.g. SQL APIs to access a remote DB



Systemarchitektur:

Distributed Database

- IBM Distributed Relational Database Architecture (DRDA)
- DB2 Satellite
- Webcasting

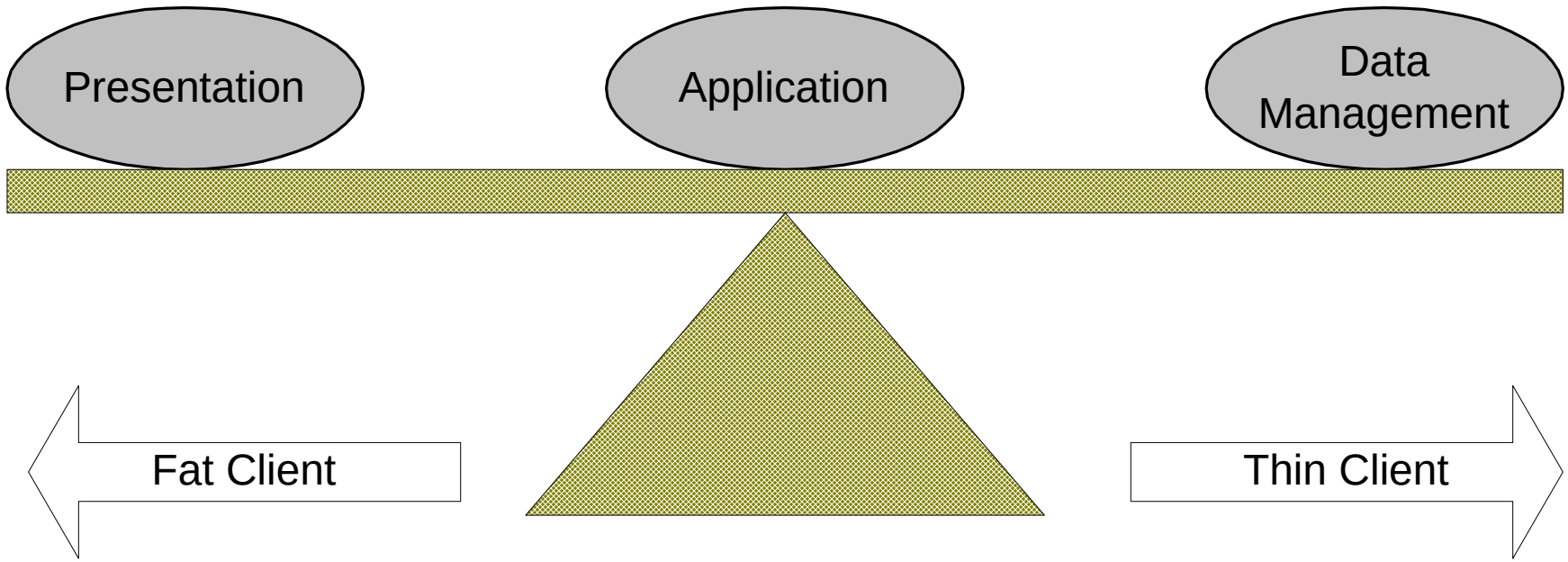


Distribution:

Fat Client - Thin Client

Client

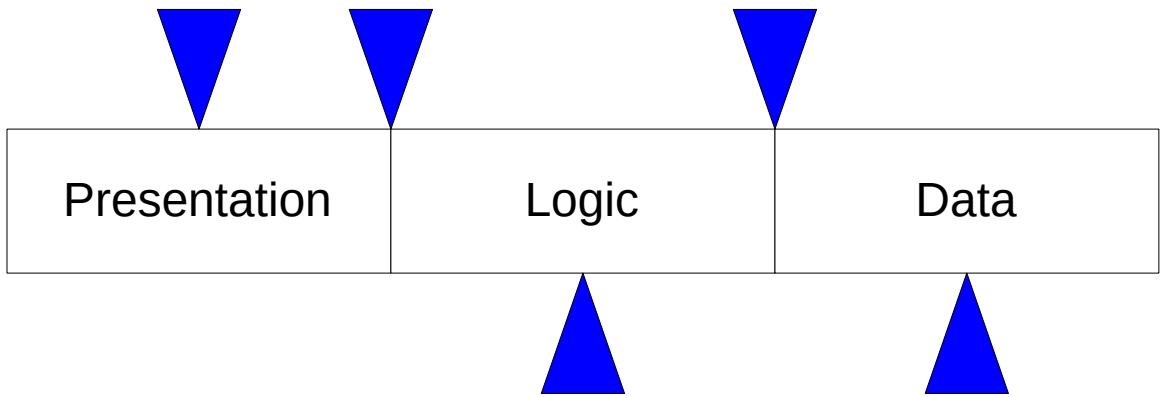
Server



Distribution:

Two-Tier, Three-Tier

- One or two cuts?



Distribution:

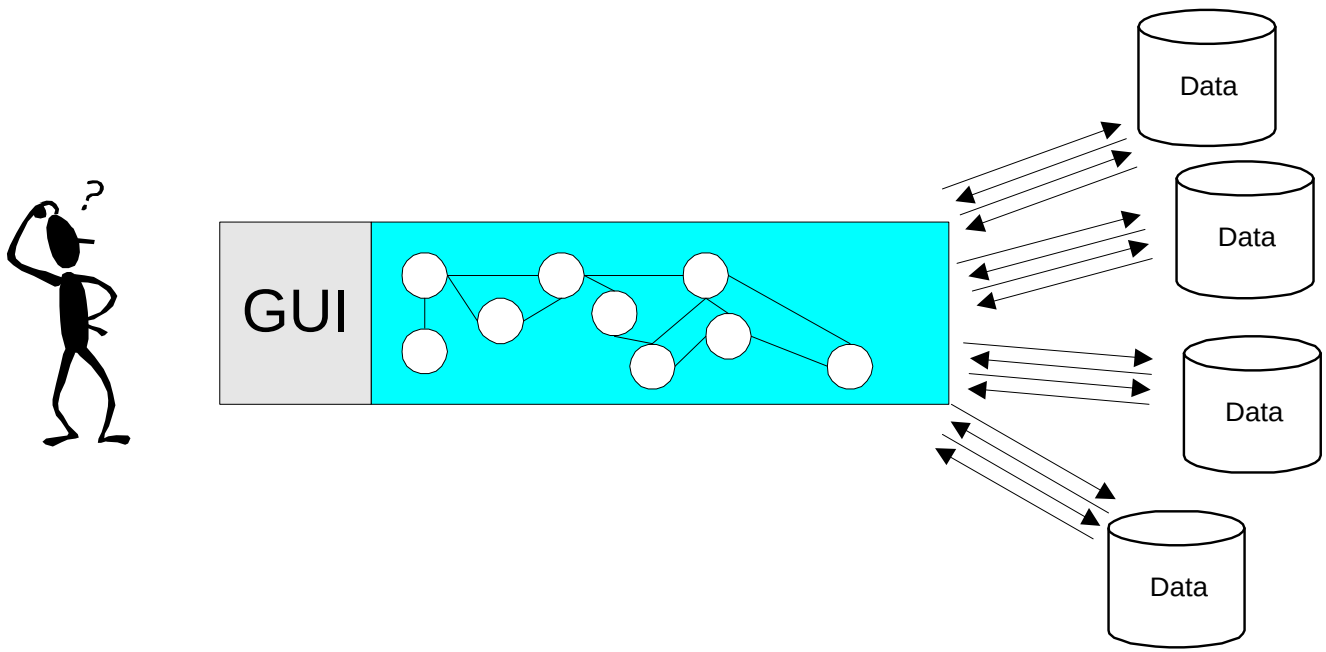
Fat Clients

- Responsiveness
  - Entry verification (e.g. Date)
  - fast checking possible
- Fine grained feedback
  - The user has not to wait until form is complete
  - Feedback during typing, mouse movement
- Error Correction expensive/slow
- Minimized network traffic?
  - client side verification
  - saved bandwidth -> more clients

Distribution:

Can Clients Be Too Fat?

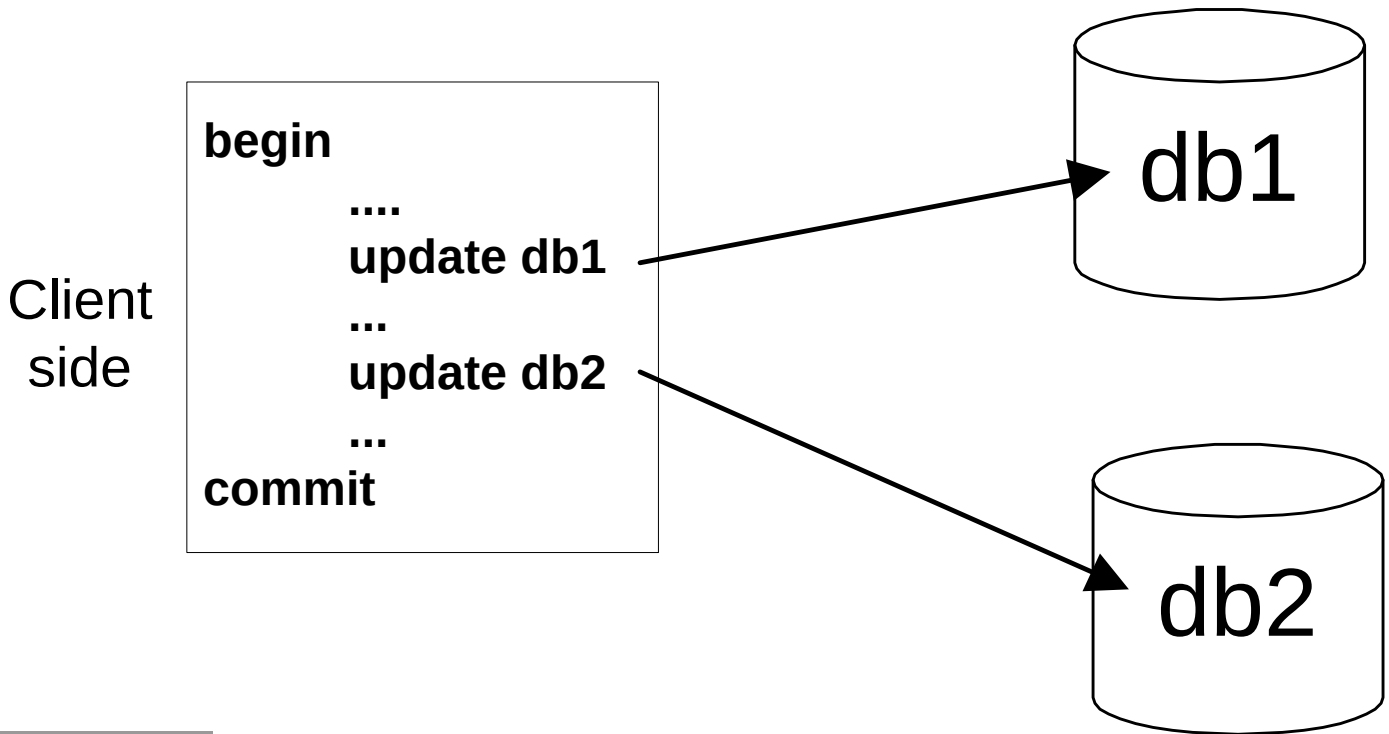
- Middleware becomes bottleneck
- Server might optimize e.g. shared memory
- DB Locking Problems
- Inconsistent Databases



Distribution:

Transactions

- Corba
- EJB
- XTA



Distribution

Glue- Plumbing

- TCP/IP (Transmission Control Protocol/Internet Protocol)
- usually hidden by programming model

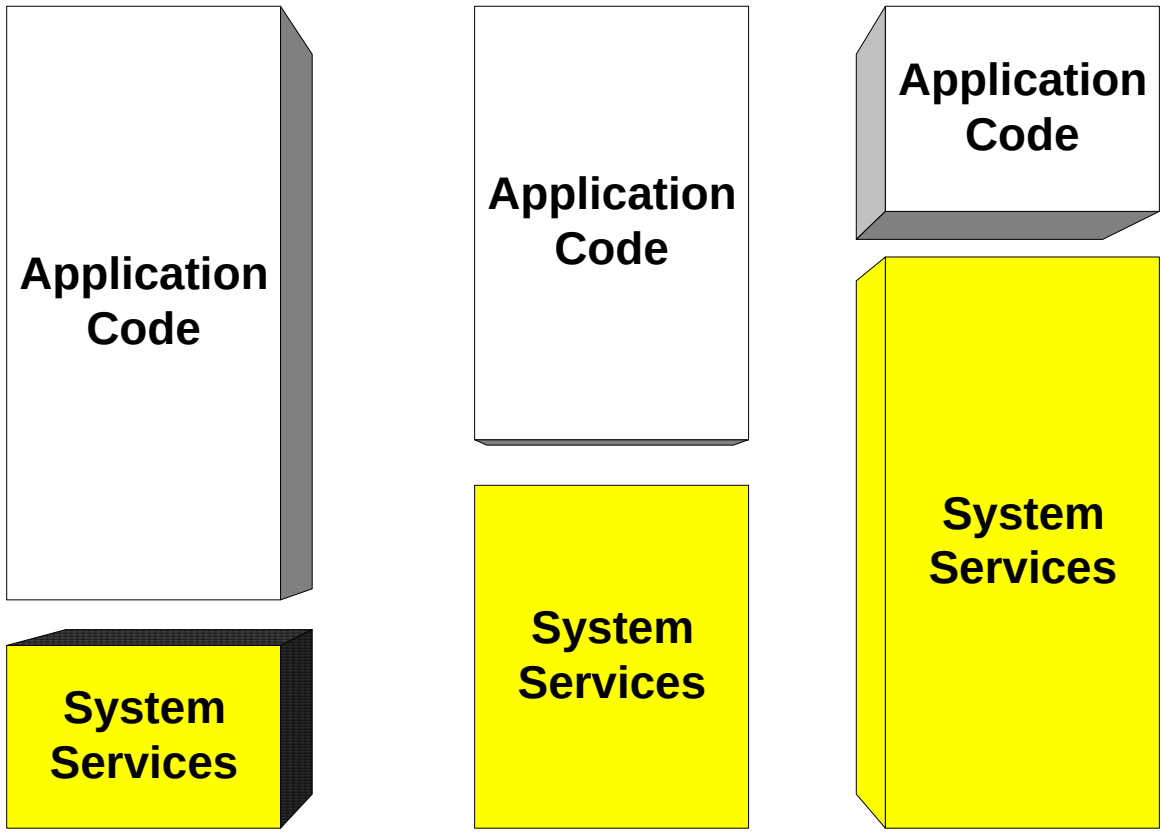
Distribution

Glue- Programming Models

- RPC
- Message Queuing
- peer-to-peer

Distribution

Glue- Communication Levels



Common Buffer | **Common Middleware** | **Common Language**

Distribution

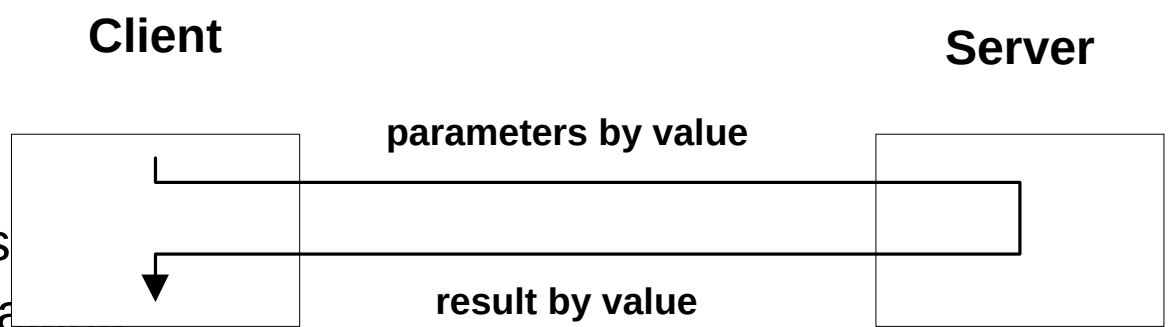
Glue- Types

- Local Glue
  - COM/OLE/DDE
  - OpenDoc
- Distributed Glue
  - DCOM .NET
  - CORBA
  - EJB
  - SOAP

Distribution

Distributed Glue

- Messaging Models
  - one way messaging



  - two way mess
  - queued messa

Distribution

Distributed Glue

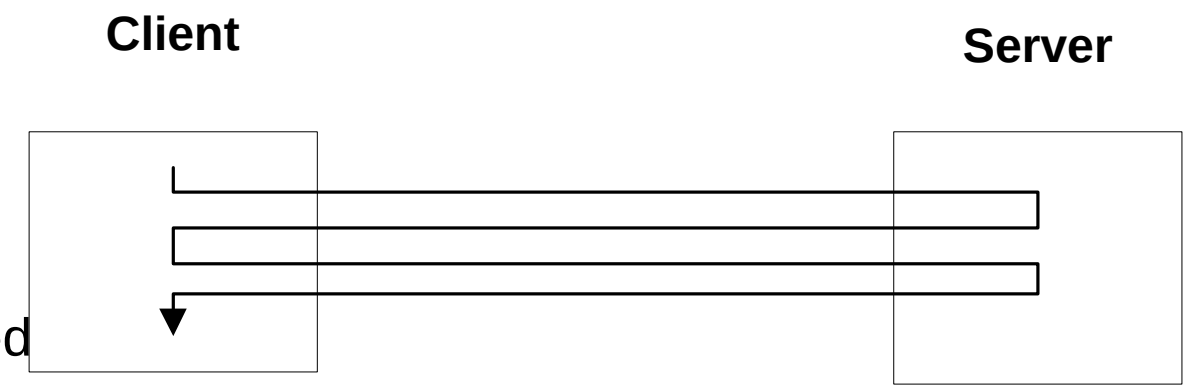
- Messaging Models
  - one way messaging

  - two way me
  - queued messaging

Distribution

Distributed Glue

- Messaging Models
  - one way messaging two way messaging
  - two way messaging



  - queued

Distribution

Distributed Glue

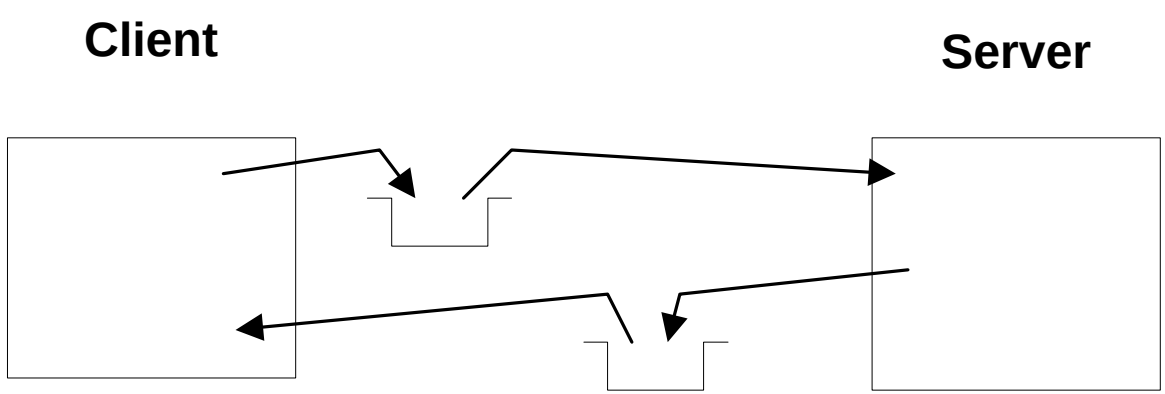
- Messaging Models
  - one way messaging
  - two way messaging

  - queued messaging

Distribution

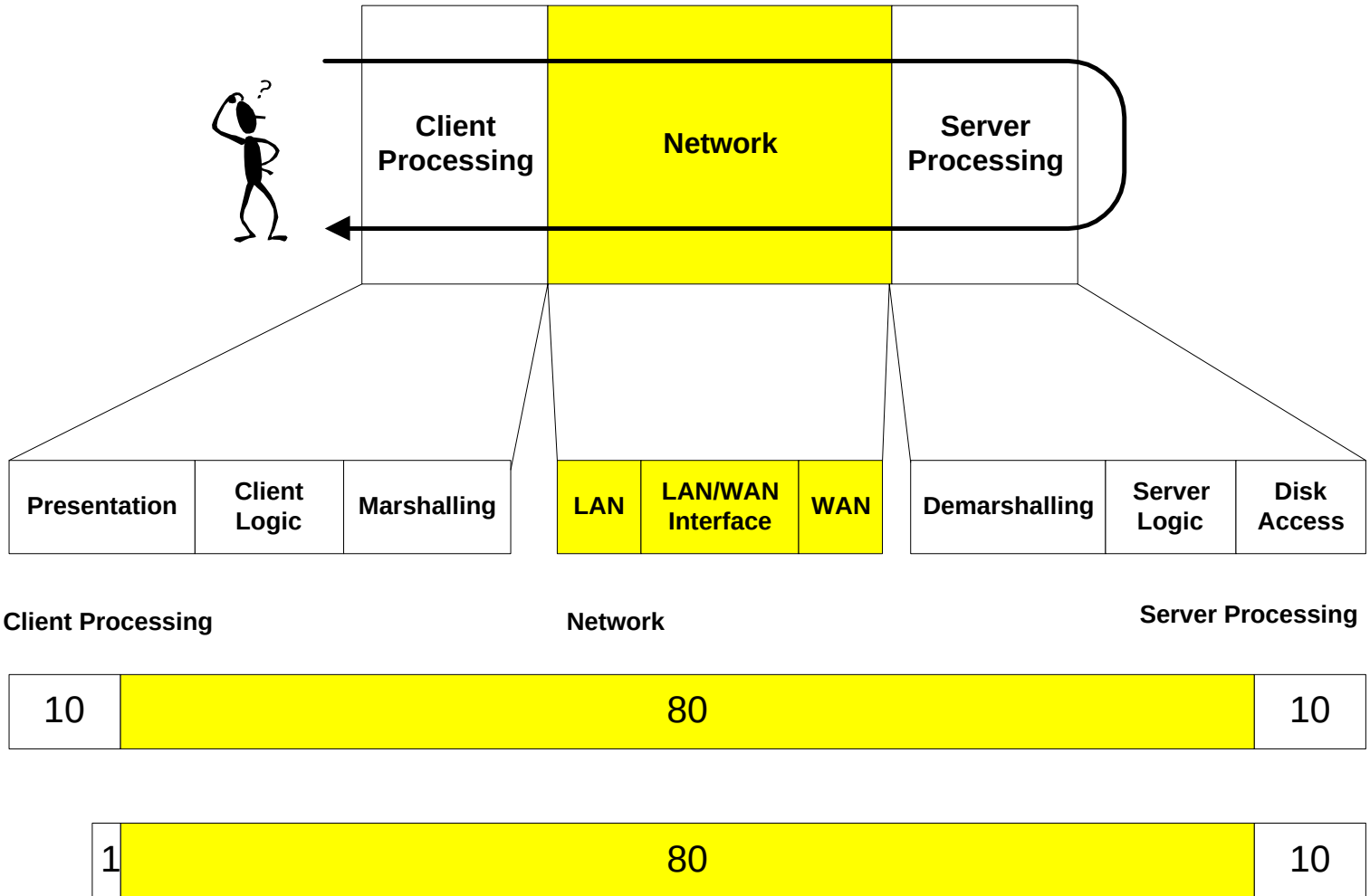
Distributed Glue

- Messaging Models
  - one way messaging
  - two way messaging
  - queued messaging



Distribution

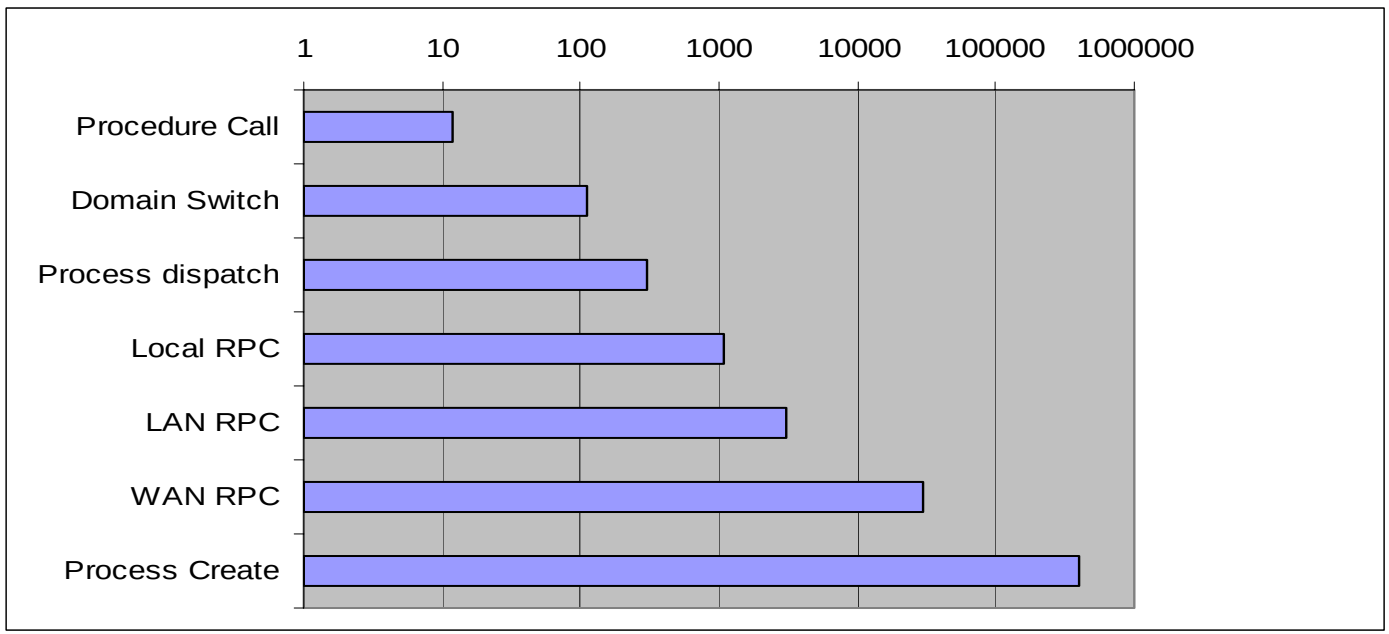
Performance



Distribution

Performance

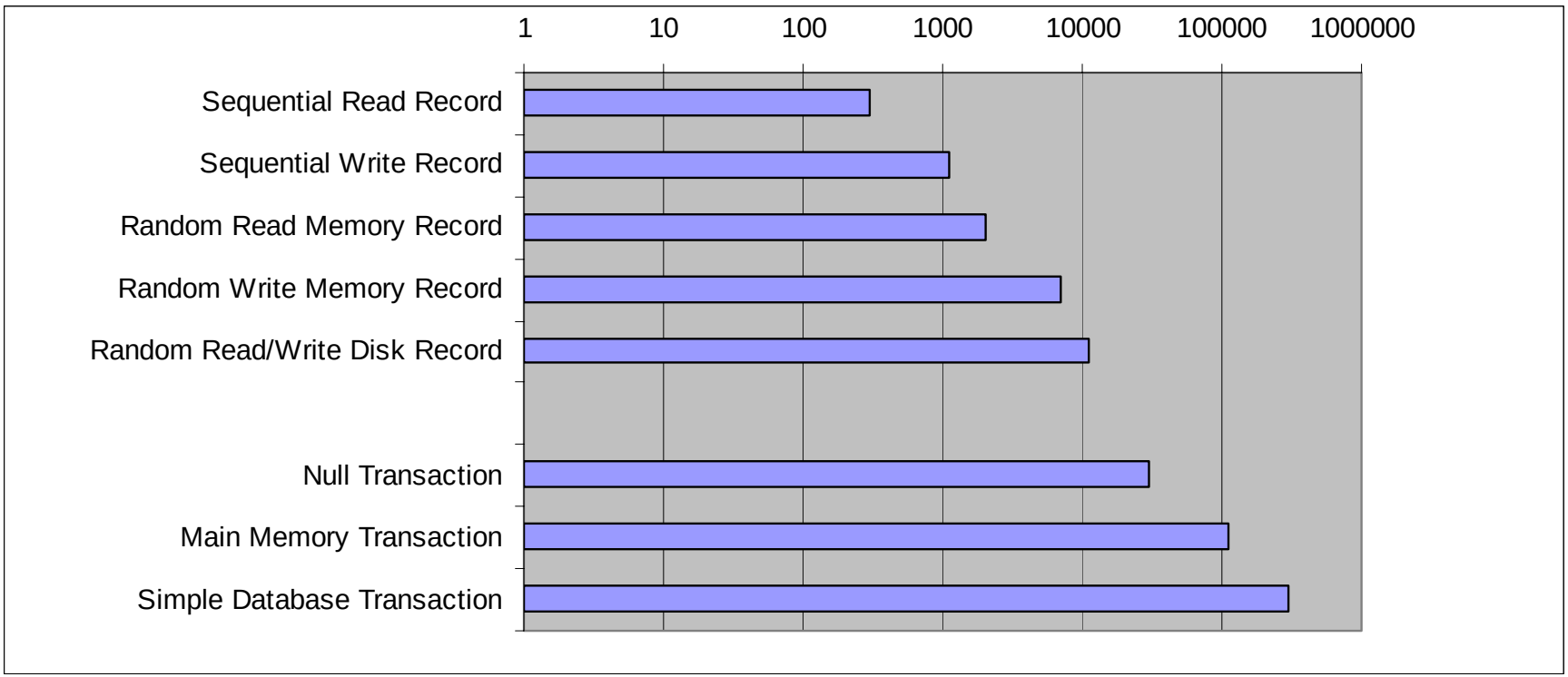
- Typical Operations



Distribution

Performance

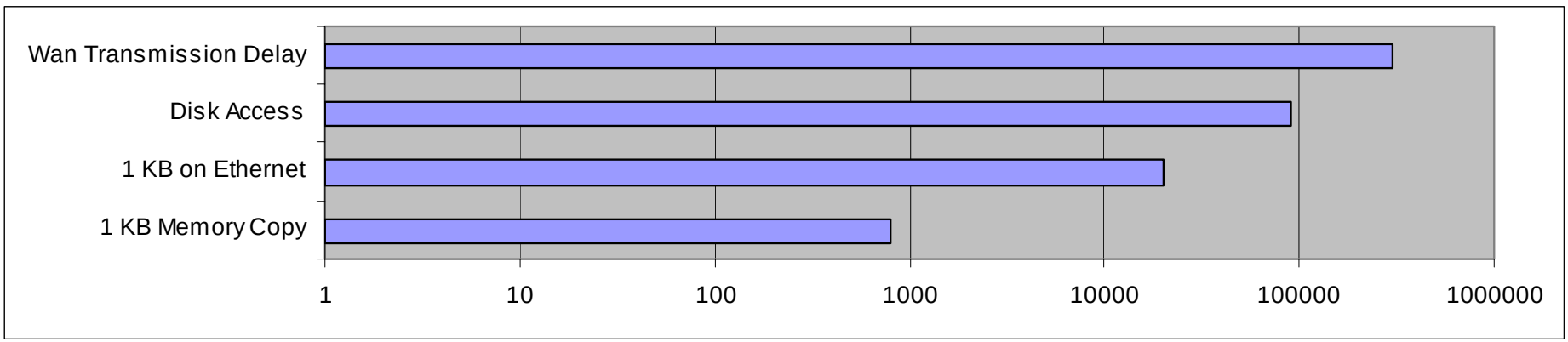
- Typical Operations



Distribution

Performance

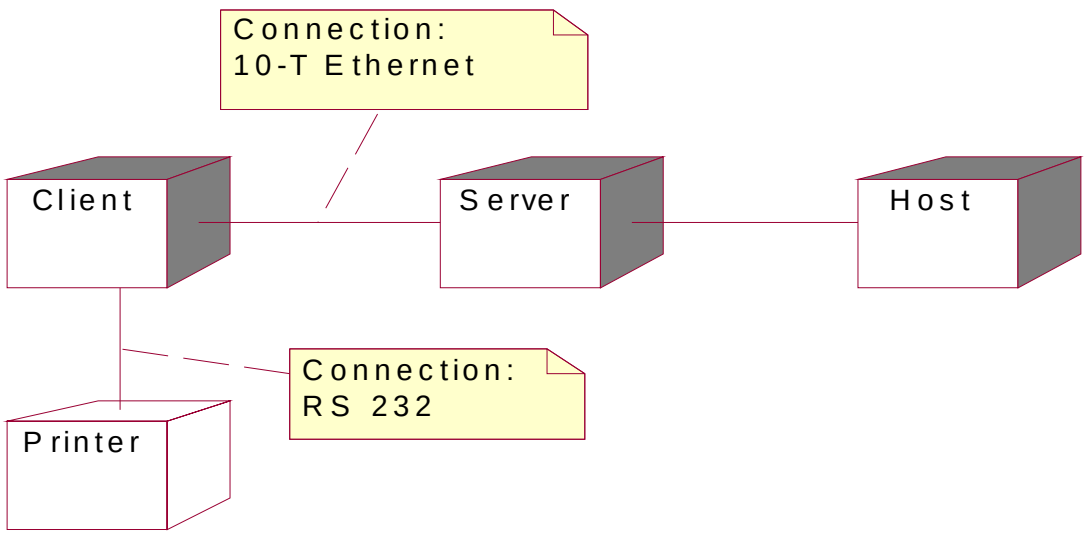
- Typical Operations



Distribution

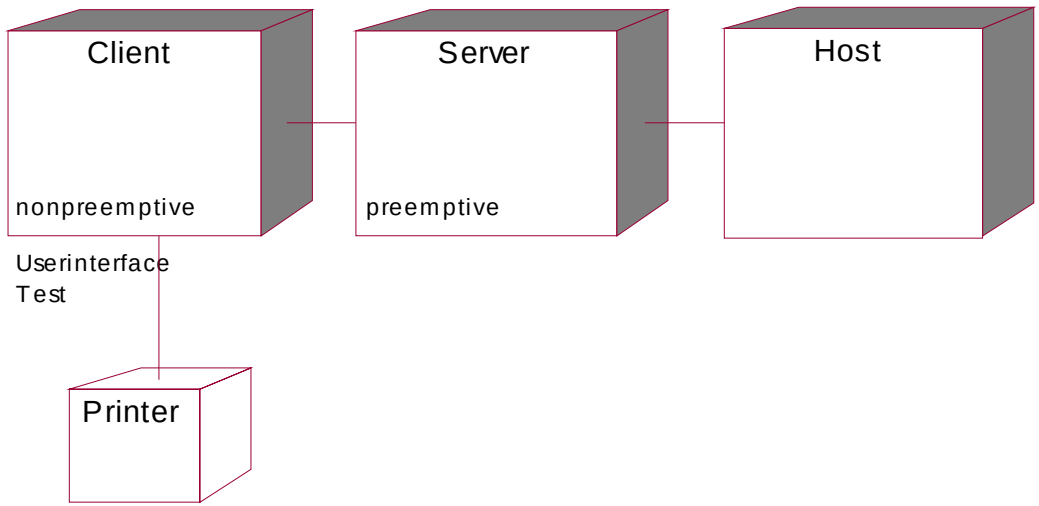
Modeling in UML

- Node: A node is computational resource having at least memory and often processing capability. Components are deployed on Nodes. Example: Processor or Device
- Connection: Association between Nodes for communication



Modeling

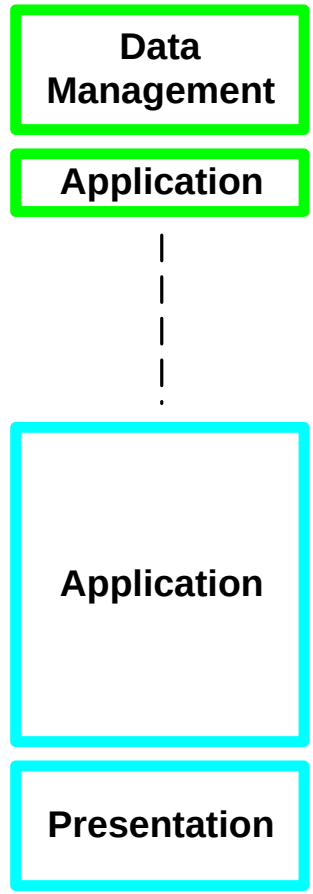
- Deployment: Assignment of Components to Nodes



Distribution

Beispiel: Phoenix

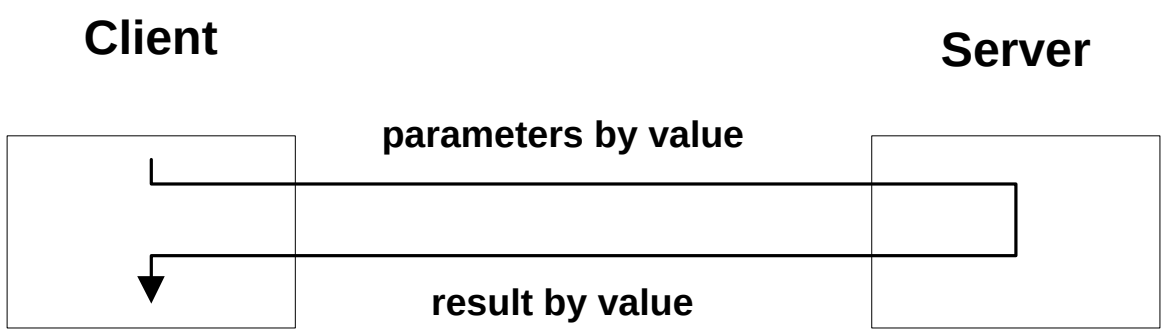
- Server
  - runs under CICS/TP
  - Language: C
- Partitioning:
  - Remote Data Access
  - Distributed function?
    (7 Functions on server side)
- Client
  - Language: Smalltalk



Distribution

Beispiel: Phoenix

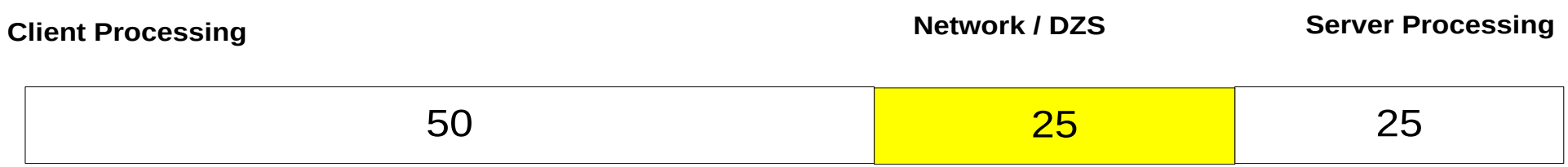
- Middleware DZS (Optimized WAN DB Access)
  - SQL Interface
  - extensions
  - One Way Messaging
  - Simple Messaging (synchronous)

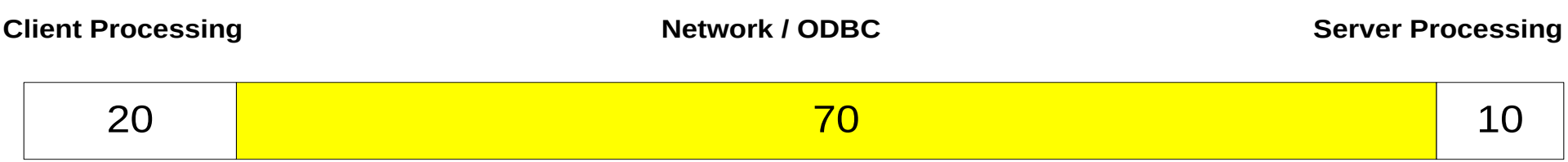


Distribution

Beispiel: Phoenix

- Performance
  - Static SQL from Client side
  - Clustering SQL Requests
  - Compression
  - Compiled Procedures

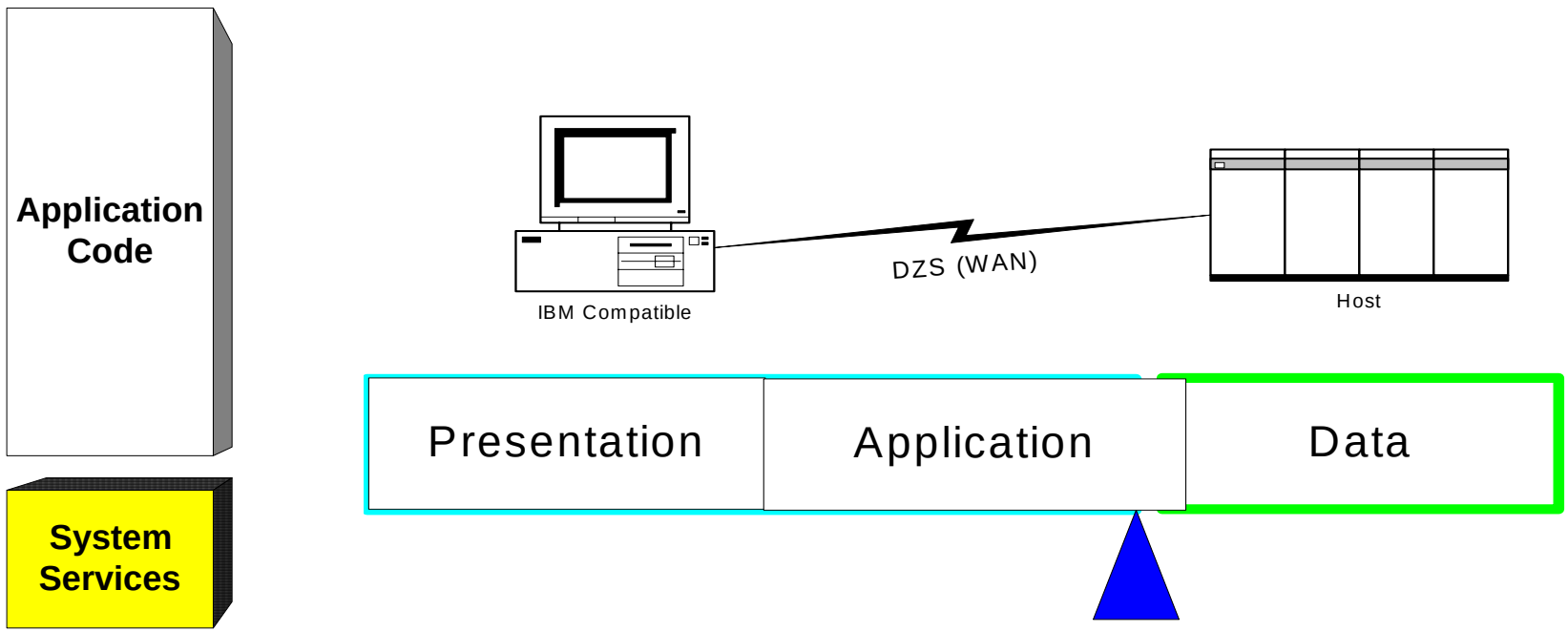




Distribution

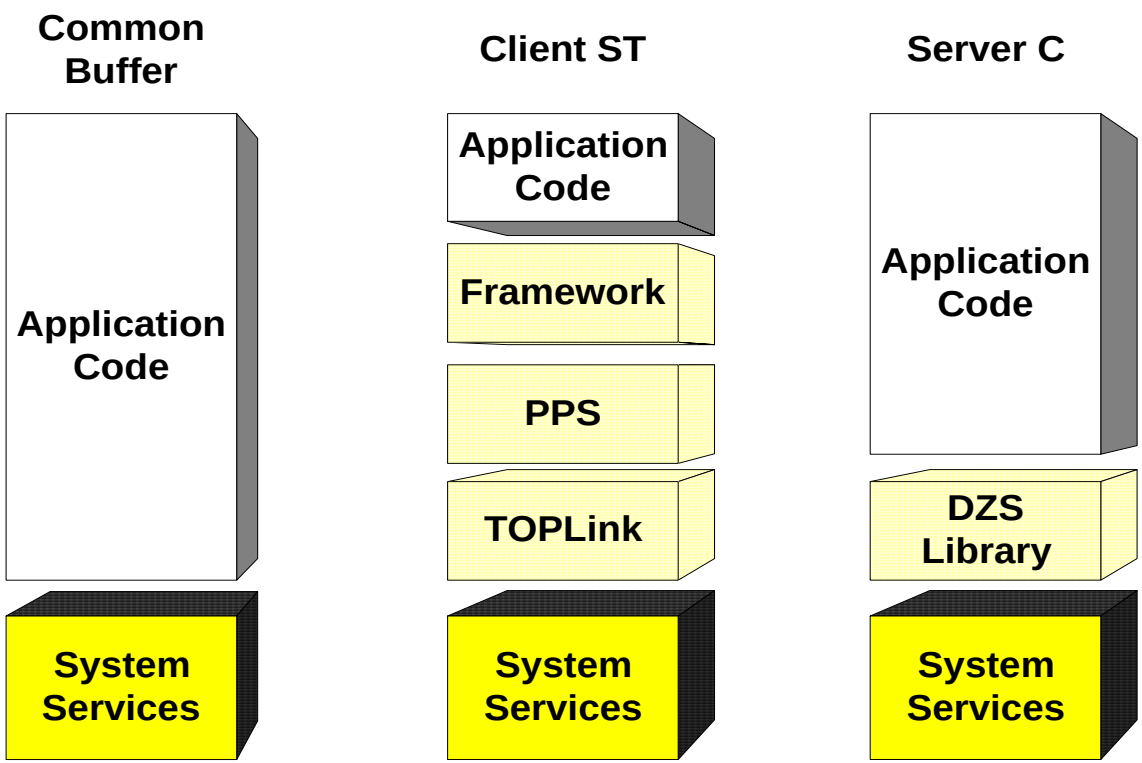
Beispiel: Phoenix

- Communication Level = Common Buffer



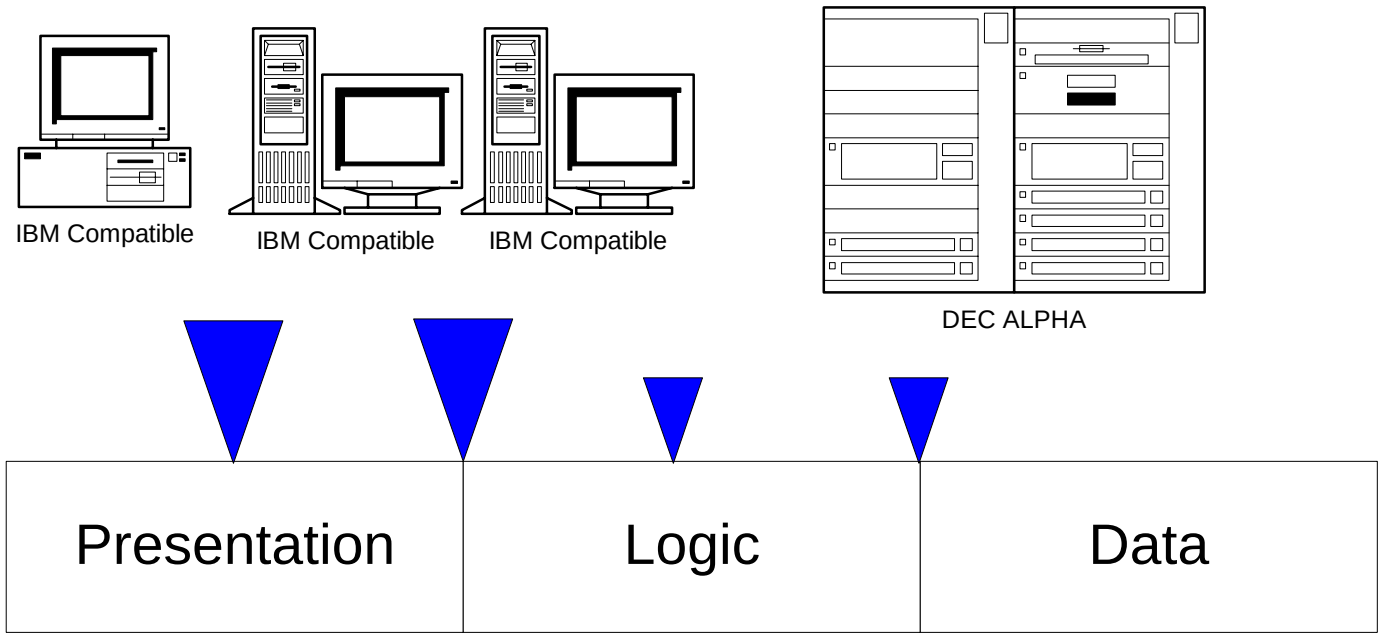
Distribution

Beispiel: Phoenix



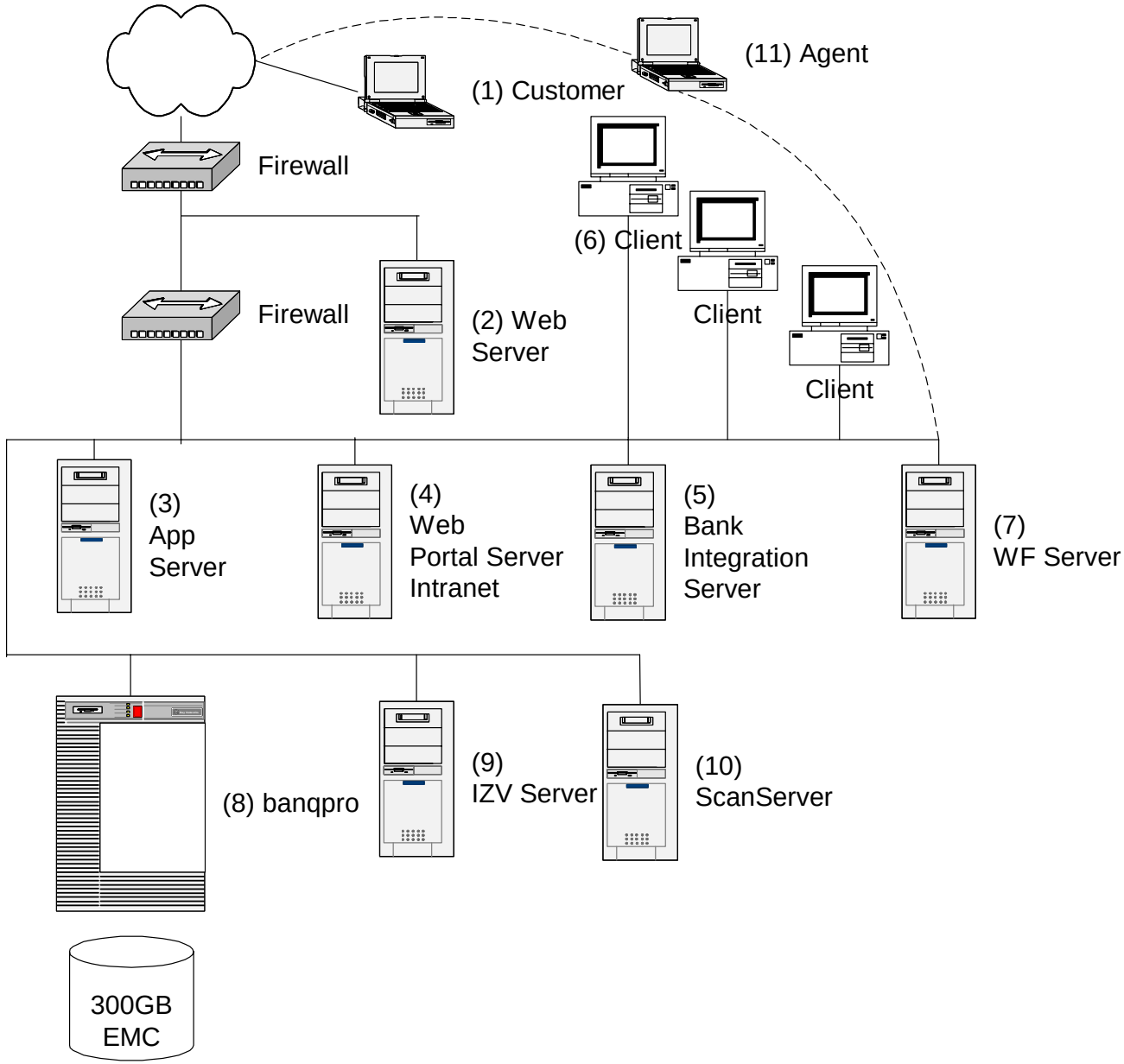
Distribution:

Beispiel: Generali Bank



Distribution:

Generali Bank



Was ist wichtig?

- Technische Architekturen ändern sich schnell
- Technologiegenerationen von 3-5 Jahren.
  - Beispiele: Internet, CORBA, Componentware, DCOM, Java haben vor 10 Jahren bei Architekturüberlegungen noch keine Rolle gespielt
  - Damit muss sich die Anwendungs- und Systemarchitektur ändern.
- Wichtig ist daher die Facharchitektur
  - sie ändert sich langsamer
  - fachliche Konstruktionsprinzipien sind lange gültig