Von Transaktionsmonitoren zu Applikationsservern

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)

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Überblick (1)

- Die Probleme
  - Bedienung vieler Dialogbenutzer
  - MIPS-Entwicklung: Die Hosts der 70er Jahre ... Die PCs heute
  - Gemeinsamer Zugriff auf Ressourcen – Transaktionen
- Prozeß- und Dialogmodelle
  - Conversational vs. Transactional Programming
  - Und die dahinterliegende Prozeßorganisation
- Transaktionen und verteilte Transaktionen: 2PC
  - ACID Eigenschaften
  - X/Open Transaktionsmanager

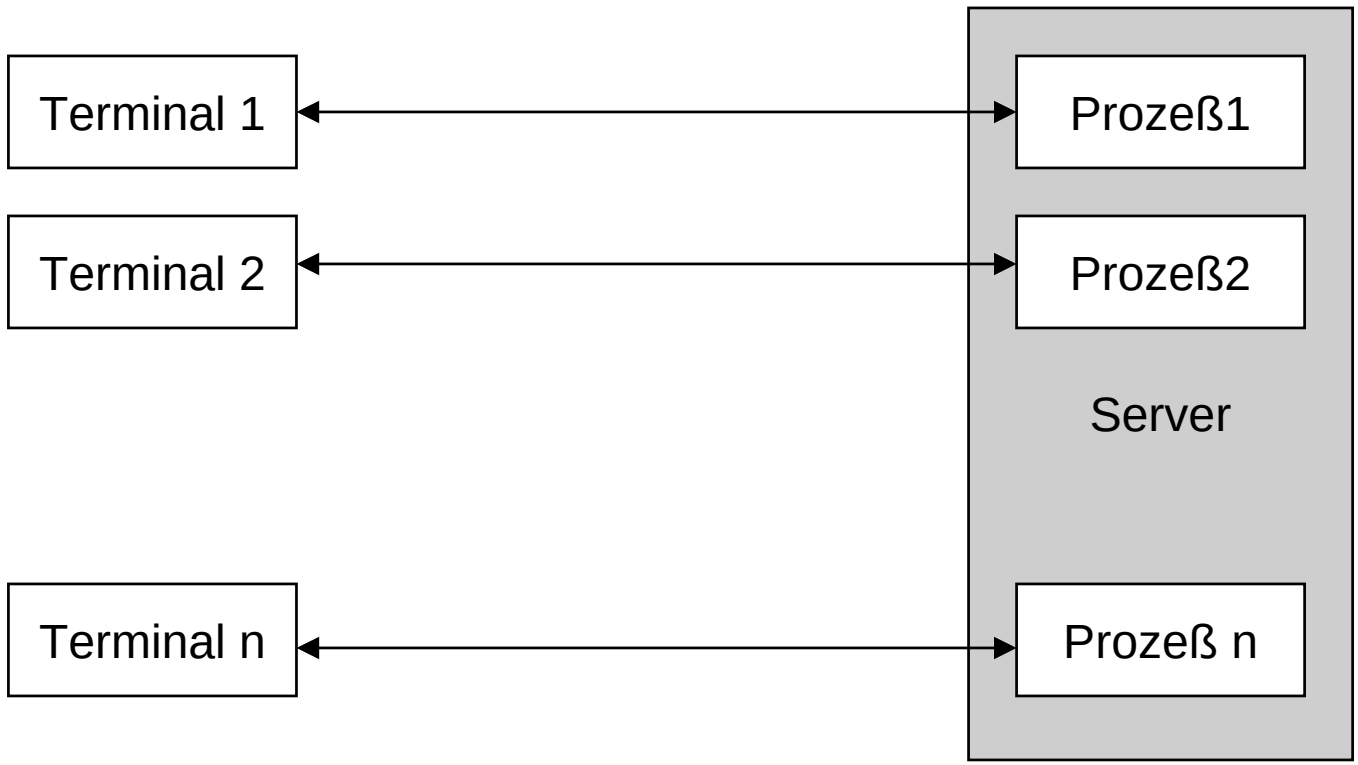
© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Überblick (2)

- CICS als Beispiel für einen Transaktionsmonitor
  - Programmiermodell
  - Dienste
  - Zum Vergleich noch IMS
  - Services von IMS und CICS
- Was leistet im Vergleich dazu ein J2EE Applikationscontainer?

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Bedienung vieler Dialogbenutzer
Ein Prozess pro Benutzer



© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Ein Prozess pro Benutzer das Modell hat Grenzen ...

- Das Modell gibt es in verschiedenen Ausprägungen
- Zum Beispiel auf der AS/400 – mit bis zu ca. 400 Benutzern
- Oder aber auch als Windows Terminalserver mit ca. 20 Benutzern für ca. 4 Prozessoren

- Problem: Prozeßwechsel der Prozessoren sind aufwändig

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

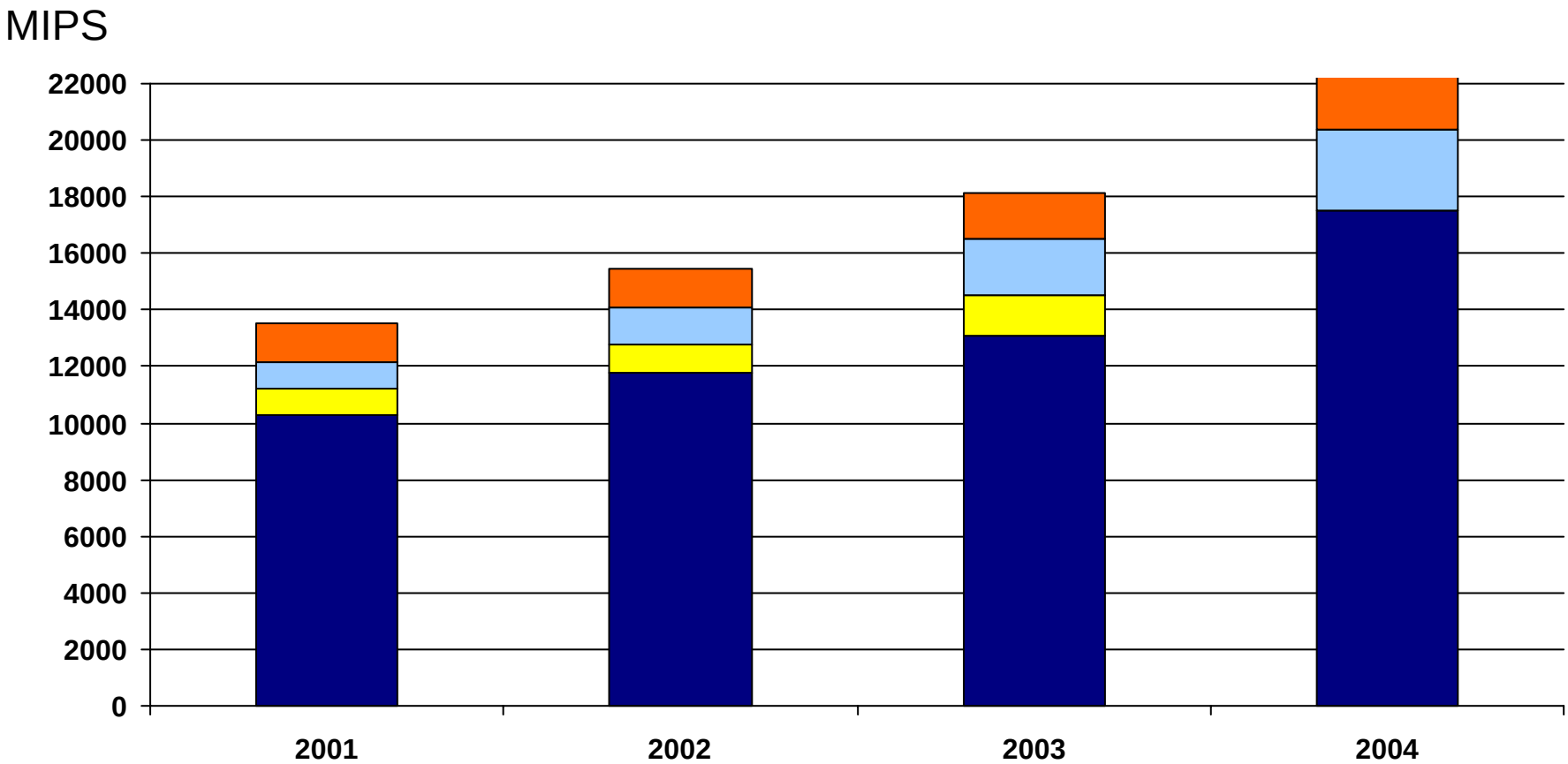
Leistungsdaten eines Mainframes der 70er Jahre ...

- Hardware und Leistung „Großrechner“ der 70er
  - 256 KB Hauptspeicher
  - 0,5 Mips
  - Bediente hunderte Benutzer ..
- Im Vergleich dazu ein Pentium PC (200 MHz) .... Im Jahr 2000 ☺
  - Ca. 200 Mips
  - 256 KB Prozessorcache ☺
  - 128 MB++ Hauptspeicher
- Im Vergleich dazu ein AMD Athlon XP 2500+ (1833 MHz) .... Im Jahr 2004 ☺
  - Ca. 6800 Mips (Quelle c‘t)
  - 512 Prozessorcache ☺
  - 64 bit Prozessor – Adressraum ziemlich groß, ca 19. Trillionen

- Aber Vorsicht – MIPS allein sagen nichts über die Dialog- und IO-Leistung

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Nur als Dimension: Projezierte MIPS Entwicklung Großanwender



© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

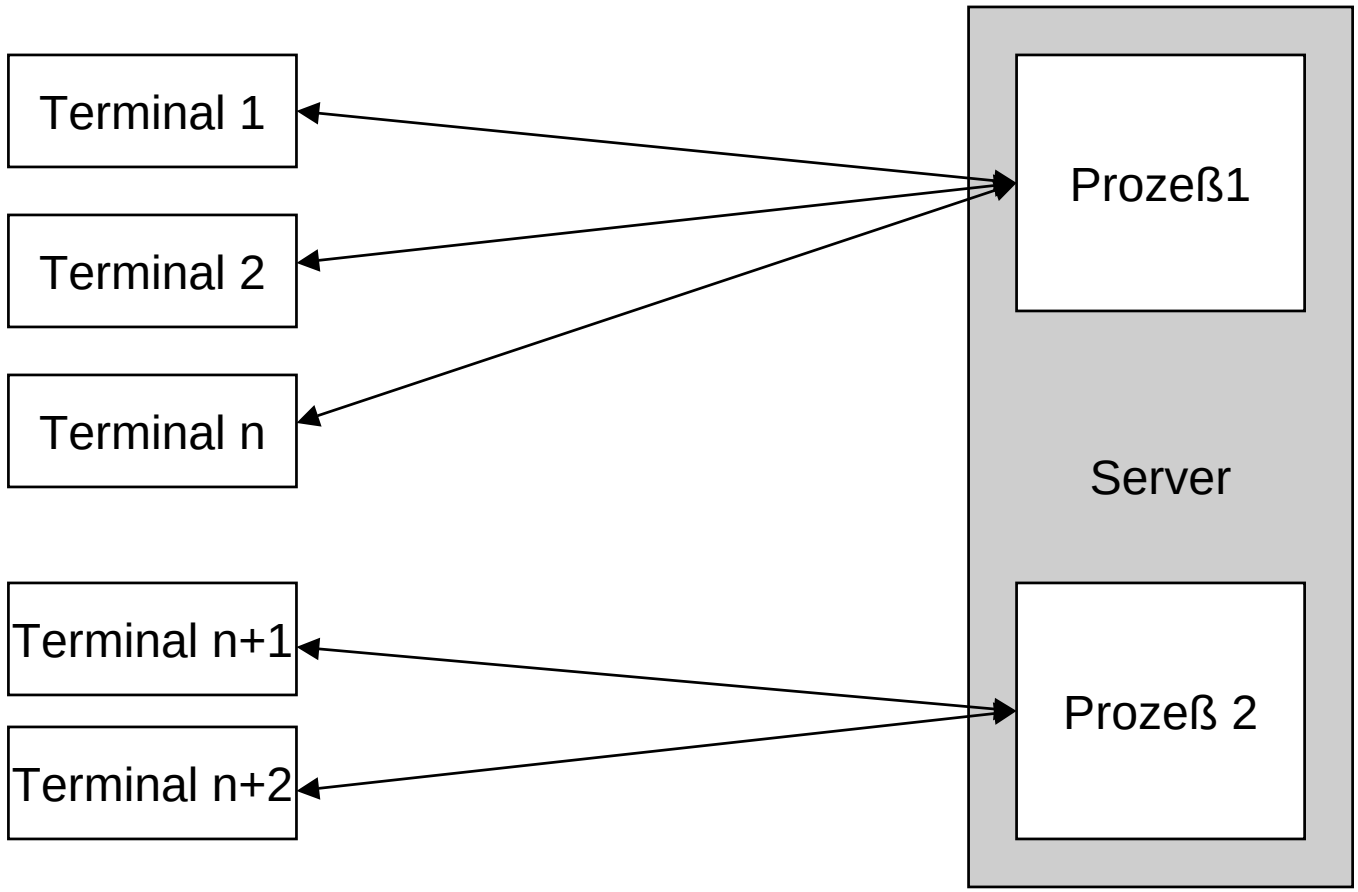
Was war anders und Folgerungen

- Anwendungen waren weniger CPU intensiv
- Datenbanken weitgehend nicht existent – VSAM Files und ähnliches
- Weniger Ressourcen pro Benutzer
- Maschinen waren auf Batch- und nicht Dialogbetrieb ausgelegt

- Man brauchte also ein Konzept für leichtgewichtige Prozesse
- Threads waren damals noch nicht bekannt

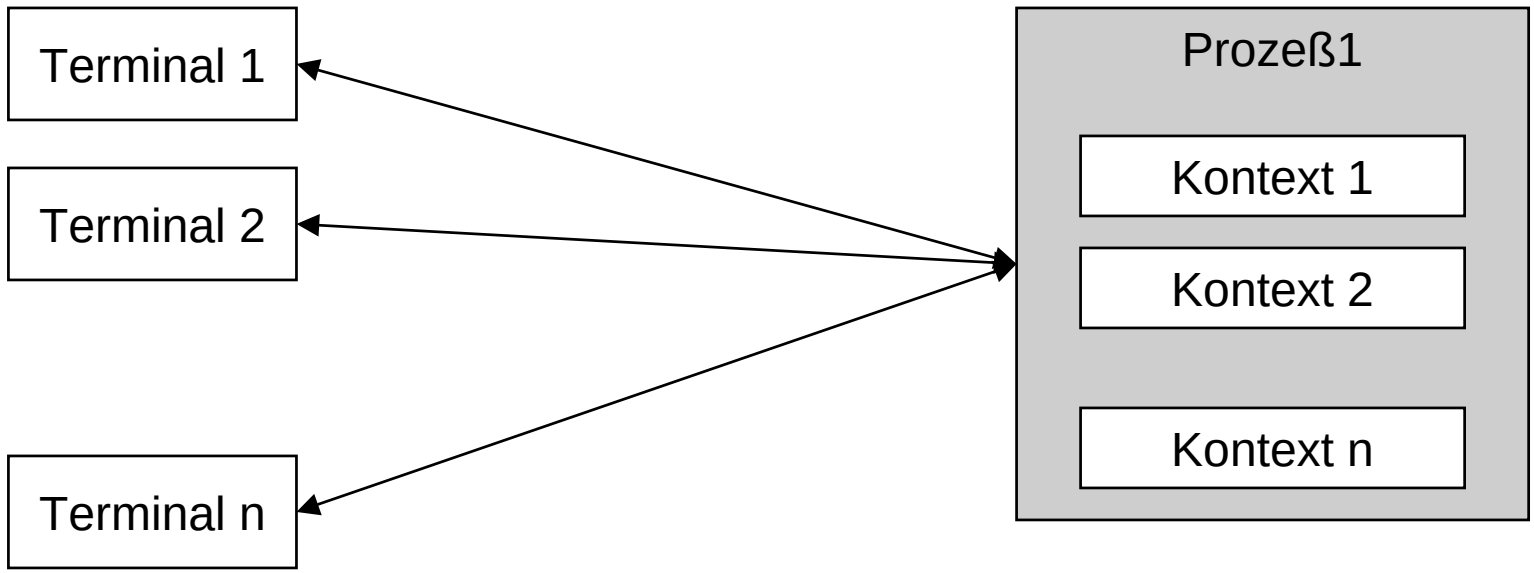
© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Bedienung vieler Dialogbenutzer
Ein Prozess pro Benutzer



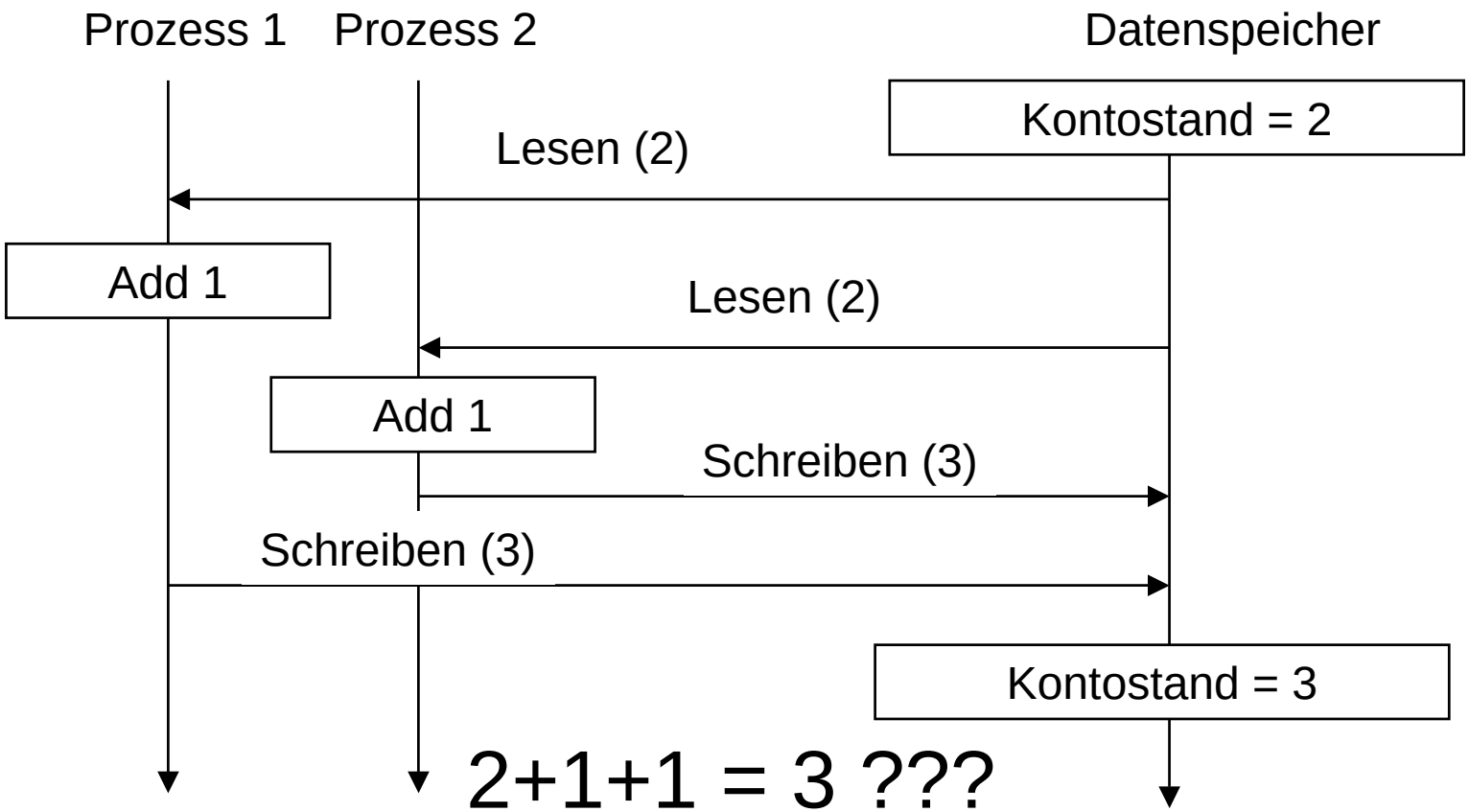
© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Pro Terminal muß es einen Kontext im Prozeß geben ...



© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Problem 2: Gemeinsamer Zugriff auf Datenbanken



2+1+1 = 3 ???

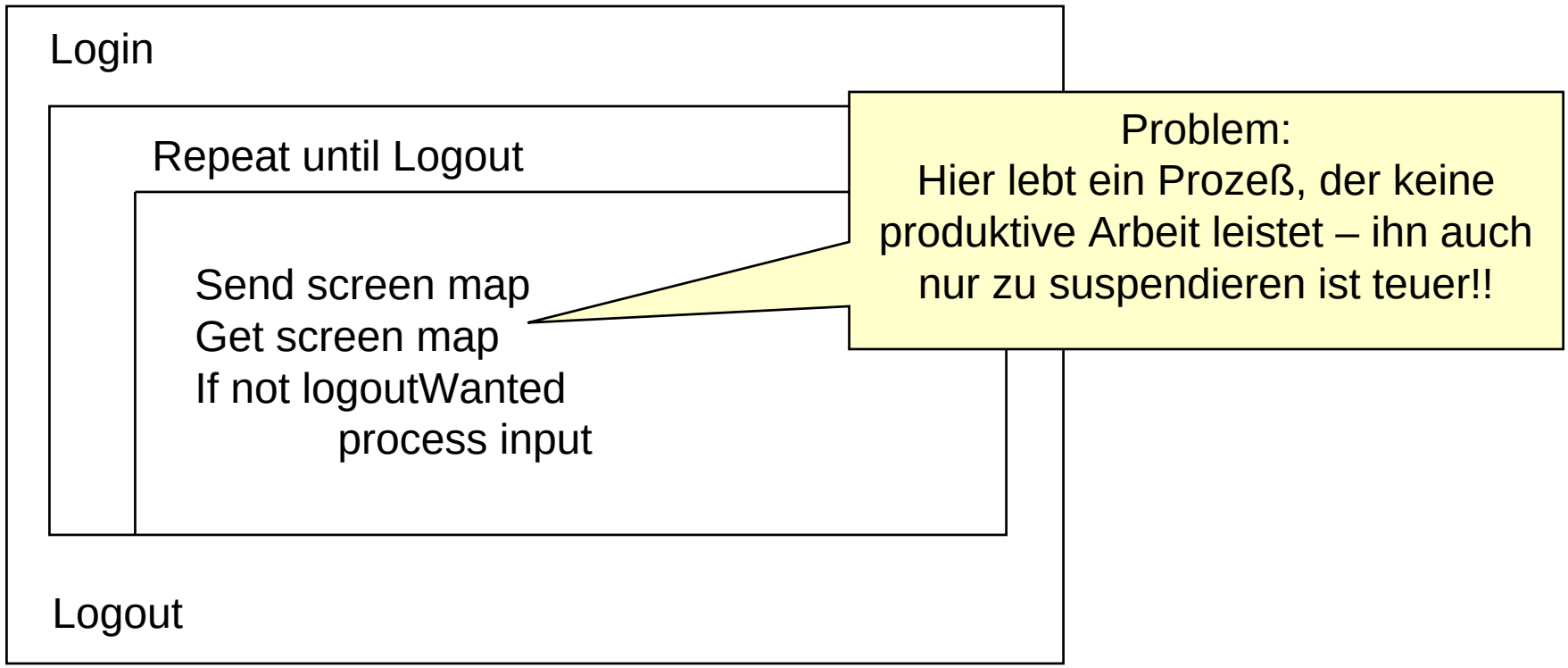
Klassische Race Condition

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Prozeß- und Dialogmodelle

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Prozeß- und Dialogmodelle
Conversational Programming



© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Zu einem Prozeßkontext bei symmetrischem Multiprocessing gehören u.a. ...

- Name des Prozesses
- Bezeichnung der gerade bearbeiteten Aufträge
- Stellung in der Prozeßhierarchie
- Rechnerkernzustand (Register, ...)
- Alarmzustand
- Arbeitszustand
- Rechte
- Betriebsmittelkonten
- Beschreibung zugeordneter Objekte und Betriebsmittel (IO, Controller, Screens, ...)
- Beschreibung des Programmadreßraumes

Quelle: Siegert, Betriebssysteme, Eine Einführung, Oldenbourg Verlag 1988

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Und bei einem Prozeßwechsel passiert folgendes …

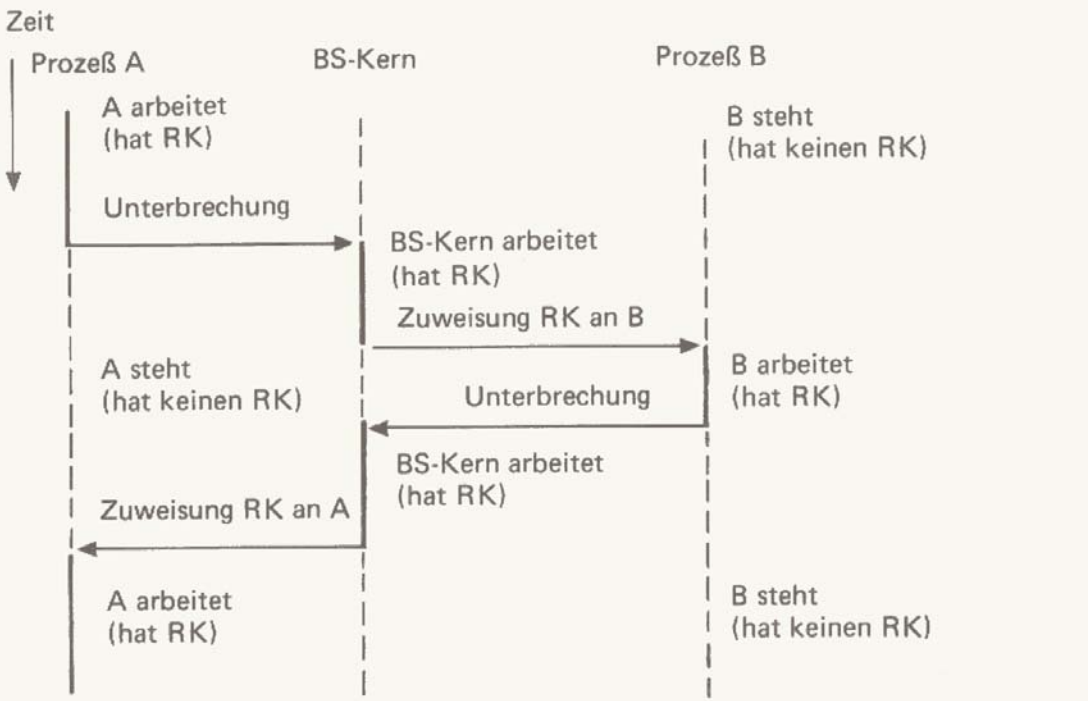
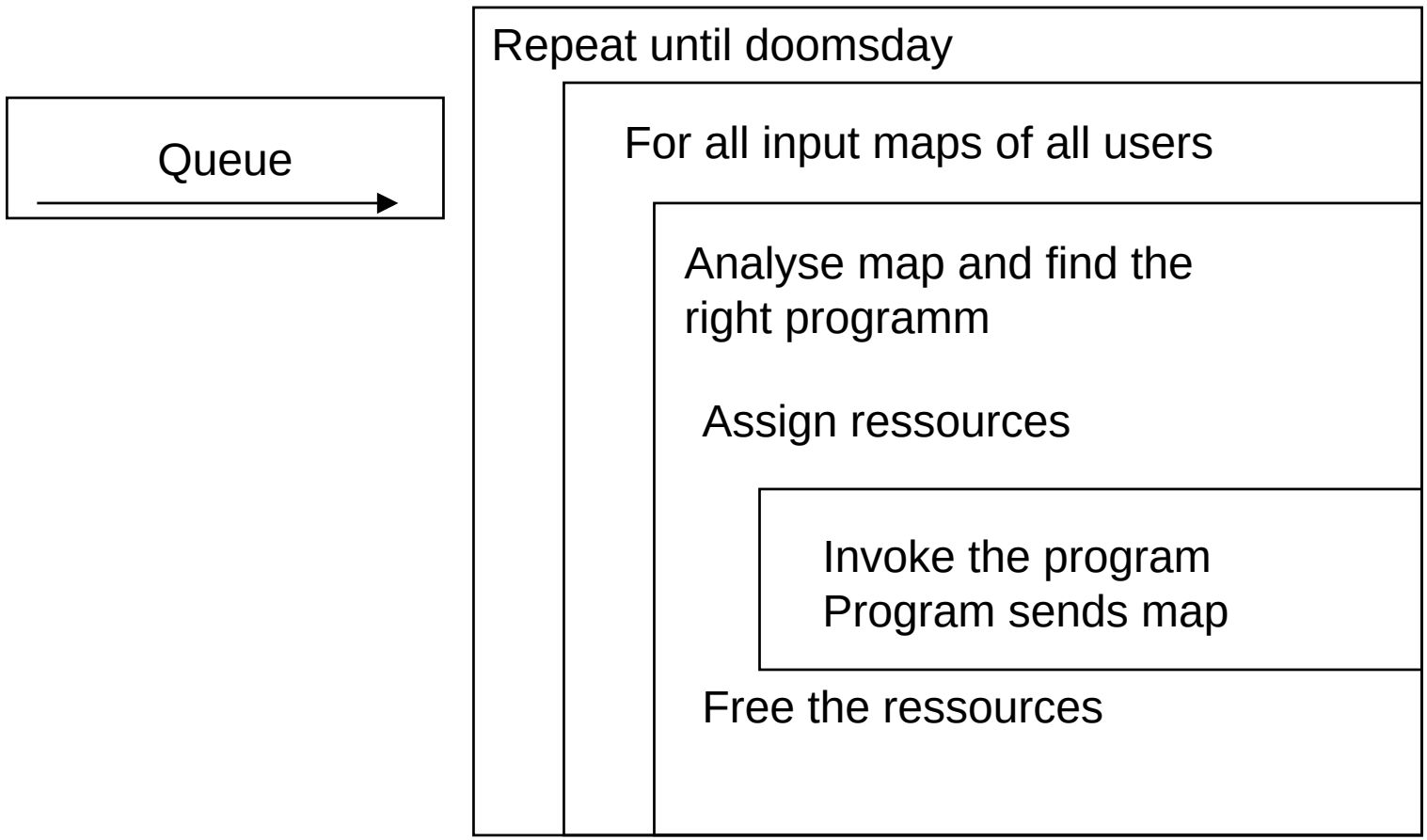


Abb. 3-2: Zeitliche Verschränkung der Prozeßbearbeitung in einem Einprozessorsystem

Quelle: Siegert, Betriebssysteme, Eine Einführung, Oldenbourg Verlag 1988

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Transactional Programming



© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Transactional Programming

the program

analyse input map
e.g. derive virtual keys
find proper action and
subsequent dialog state
perform the action
set new dialog state
send new map

Mindestens das sagt uns. Wir brauchen einen Zustand oder aber auch Kontext ...

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Vorsicht also: Der Begriff Transaktion bedeutet ..

- Transaktion im Sinne von „transactional programming“
  - Man programmiert „eine Transaktion“
- Transaktion im Sinne der Datenbankprogrammierung
  - Ich muß eine Transaktion gegen eine Datenbank fahren

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Transaktionen und verteilte Transaktionen: 2PC

- ACID Eigenschaften
- 2PC

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

ACID Eigenschaften (1)

- Atomicity (Atomarität)
  eine Transaktion wird entweder komplett ausgeführt (commit) oder gar nicht – sie erscheint dann so, als wäre sie nie gestartet worden (rollback).

- Consistency (Konsistenz)
  eine Transaktion führt eine Datenbank von einem konsistenten Zustand in den nächsten über
  (Hinweis: konsistent = technisch konsistent)

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

ACID Eigenschaften(2)

- Isolation:
  Transaktionen, die zeitlich parallel ausgeführt werden erscheinen so, als wären sie isoliert voneinander (hintereinander) ausgeführt worden
  - -> siehe Serialisierbarkeitstheorie, Sperrprotokolle
- Durability:
  Nach dem Commit werden die Effekte dauerhaft, also persistent gespeichert

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Beispiel für verteilte TAs oder TAs mit >1 Ressource ..

- Wenn ein Callcenter-Mitarbeiter von einem Kunden einen Auftrag für eine Waschmaschine bekommen hat, muss Folgendes passieren:
  - (OP1) Auftrag, Rechnung: Es muss ein Auftrag und eine Rechnung erstellt werden. Die Rechnung muss in der Buchhaltung verbucht werden.
  - (OP2) Auslagerung, Lieferschein: Die Verfügbarkeit der Waschmaschine im Lager muss geprüft werden und falls vorhanden, muss die Waschmaschine ausgelagert werden. Der Einfachheit halber nehmen wir an, dass die Waschmaschinen immer lagernd verfügbar sind. Wir können deshalb immer einen Lieferschein erstellen. Die Auslagerung und der Lieferschein benötigen als Daten:
    - die Rechnungsnummer und die Auftragsnummer,
    - die Adresse des Kunden.
  - (OP3) Montageauftrag: Es muss ein Montageauftrag erstellt werden. Der Auftrag weist einen Mitarbeiter an, die Waschmaschine aus dem Lager abzuholen, zum Kunden zu bringen und dort zu montieren.

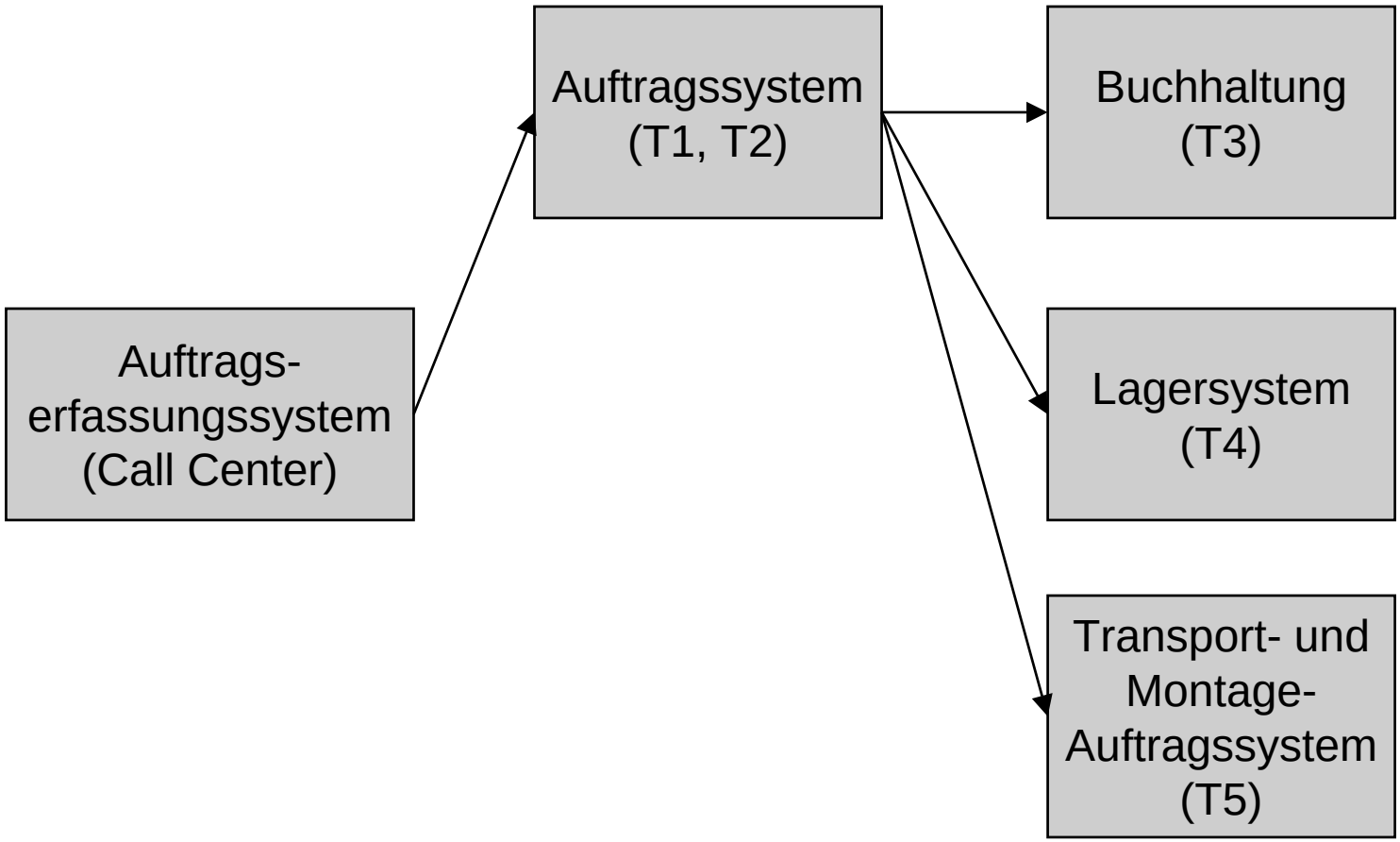
© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Beispiel für verteilte TAs oder TAs mit >1 Ressource ..

- In einem Unternehmen können an einem solchen Vorgang zum Beispiel vier DV-Systeme auf verschiedenen Plattformen beteiligt sein:
  - ein Auftragserfassungssystem,
  - ein Buchhaltungssystem,
  - ein Lager- und Bestandsverwaltungssystem,
  - ein System für Transport- und Montageaufträge.

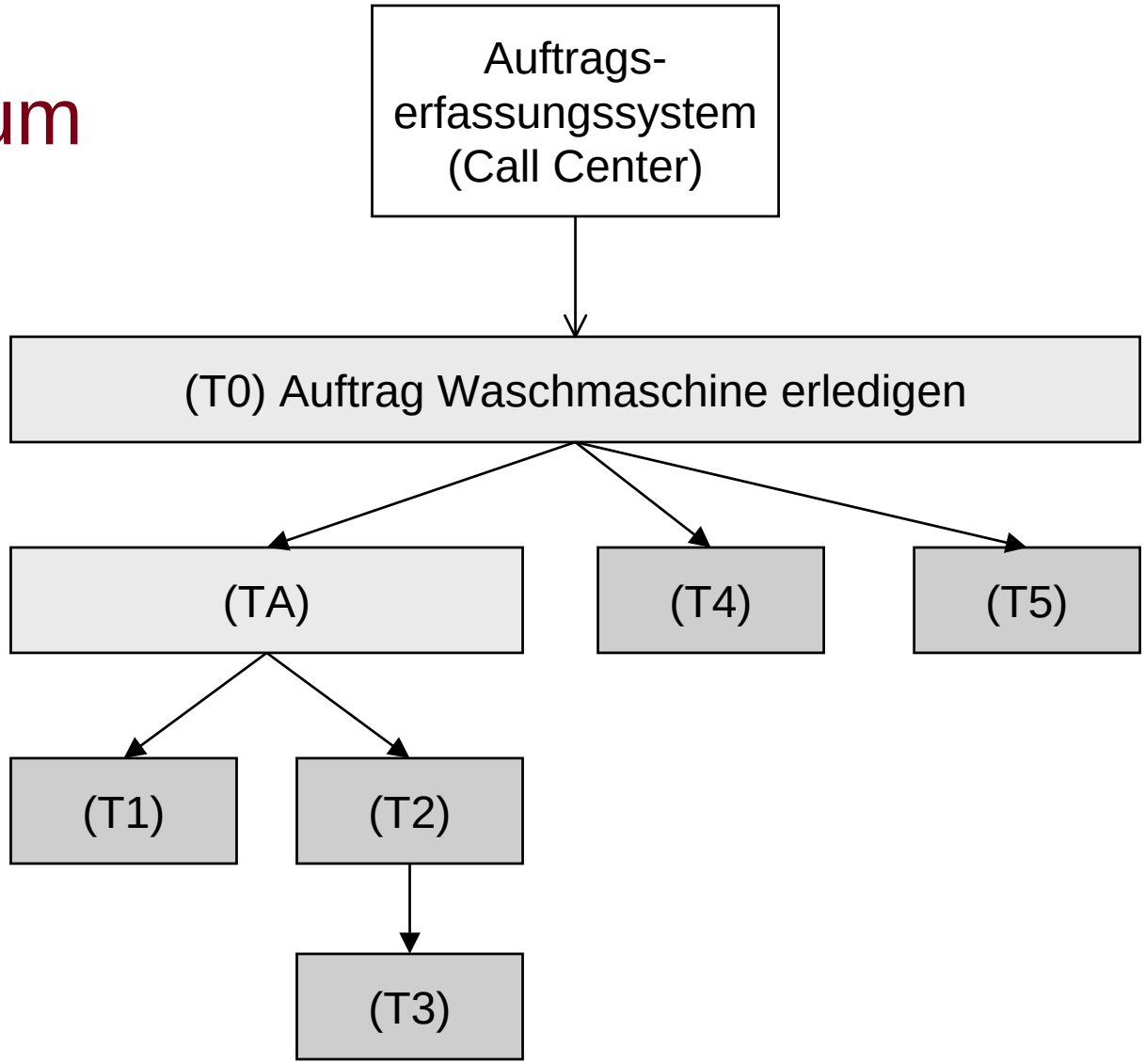
© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Verteilte Transaktion (Beispiel)



© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

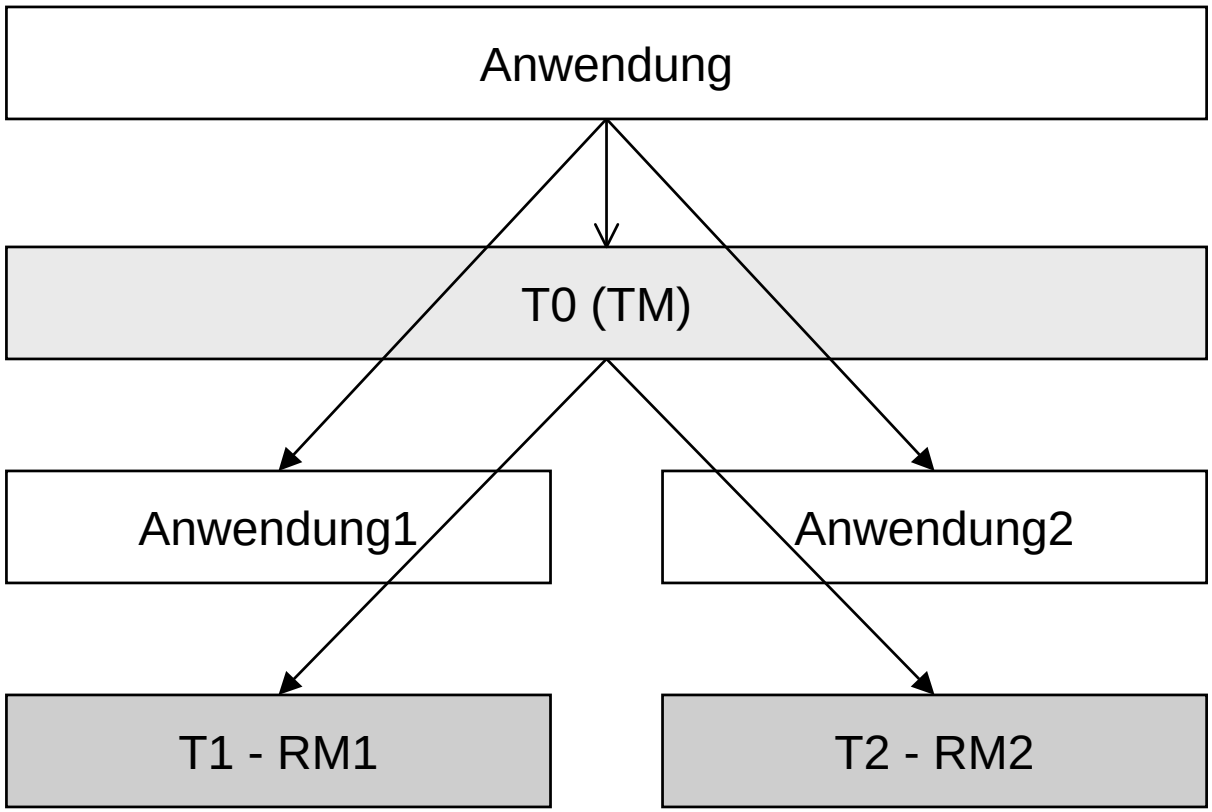
Transaktionsbaum



© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

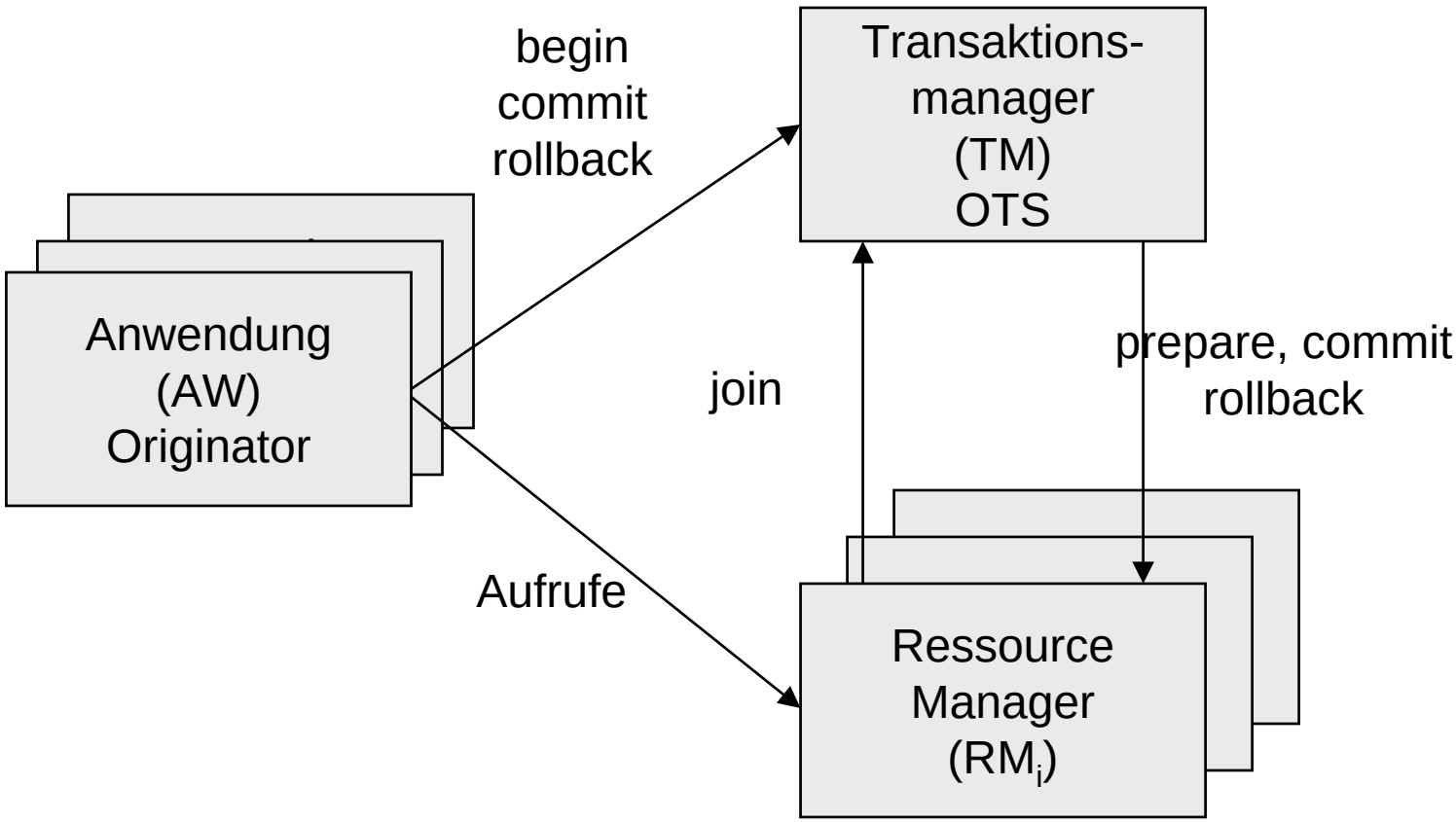
Transaktionsbaum

Transaktions- und Ressourcemanager



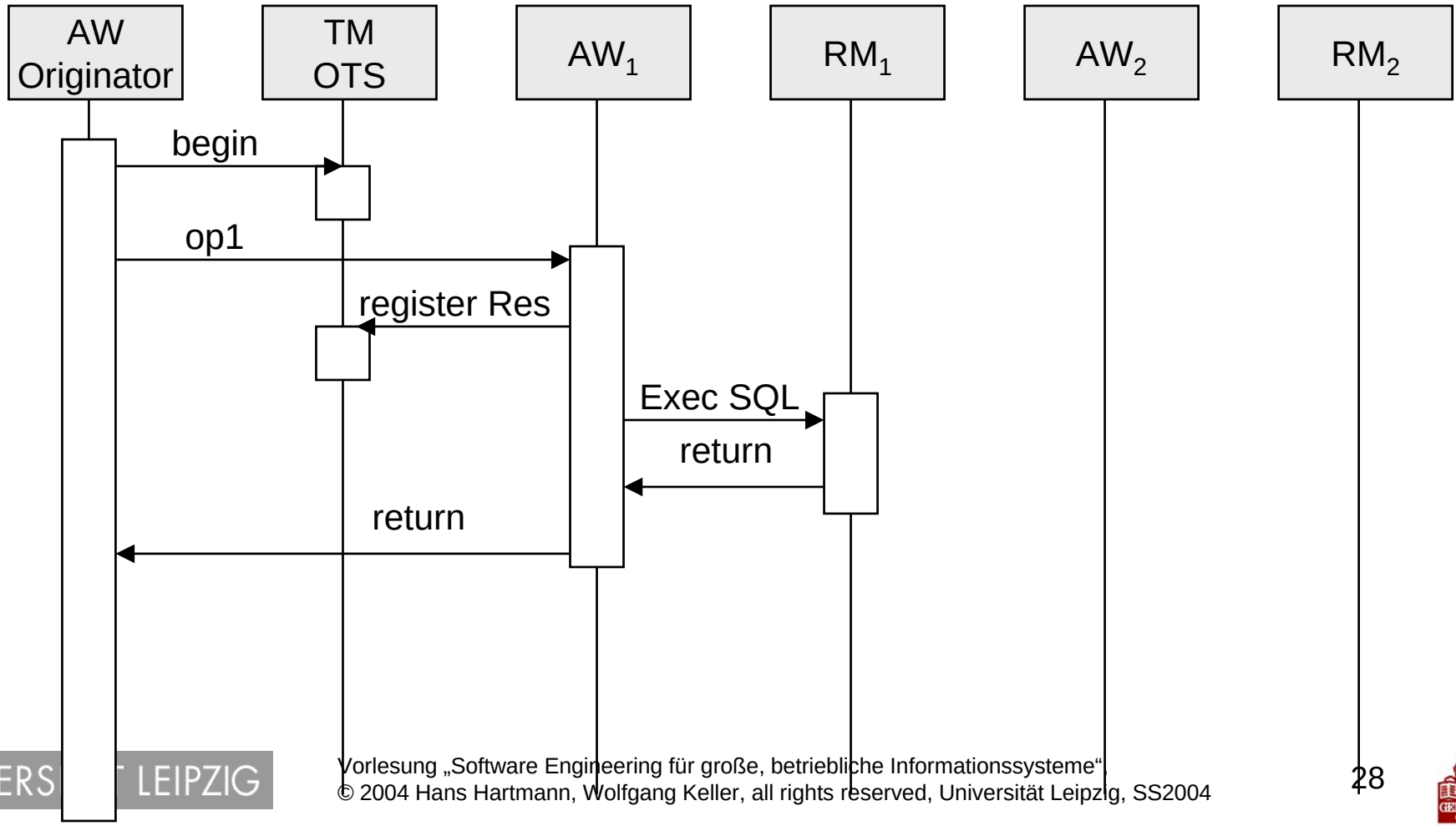
© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

2PC Teilnehmer



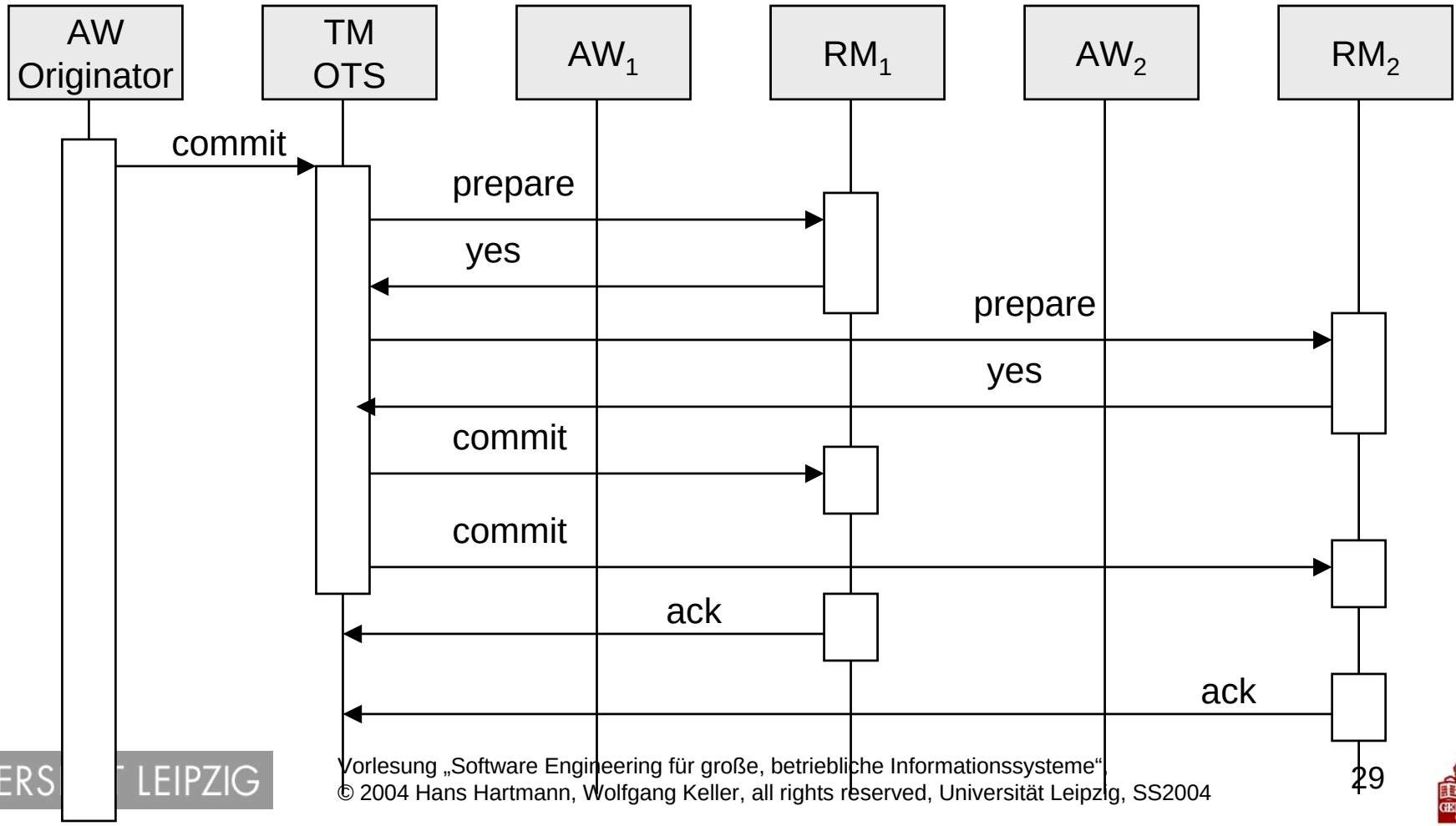
© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

2PC Beispiel



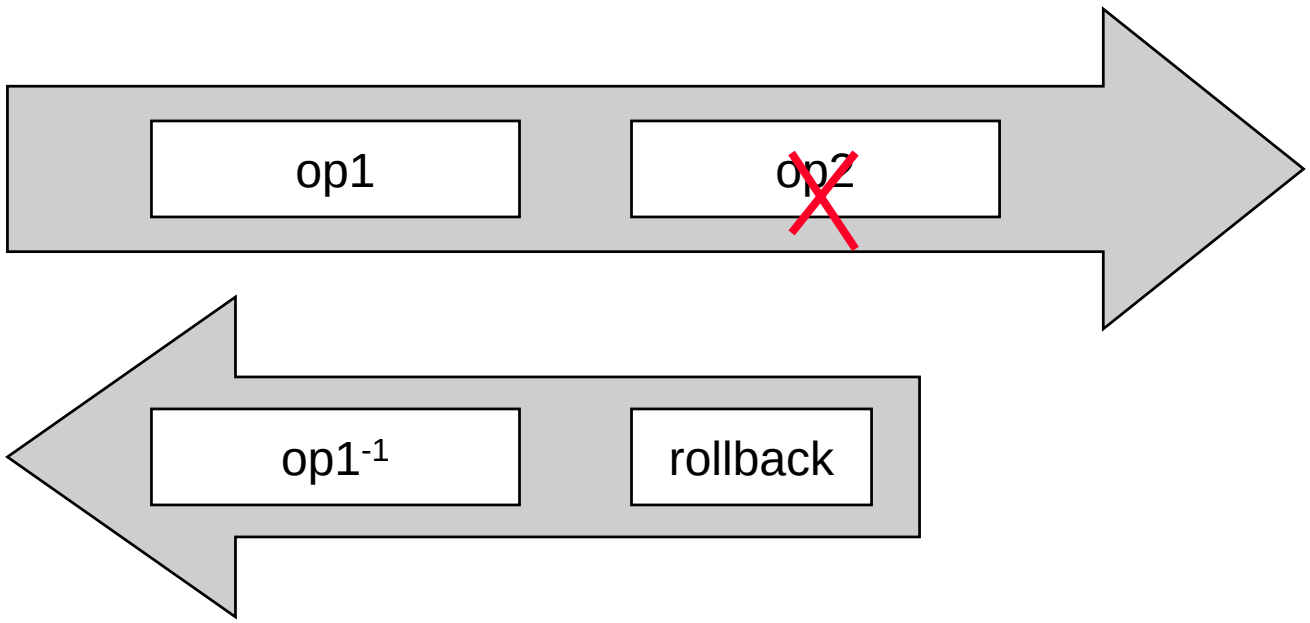
© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

2PC Beispiel



© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Als anderes Modell gibt es auch noch „Soft Rollback“



© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

TP-Monitore – ganz allgemein (1)

- Ein „transaction processing monitor“ ist eine Art von Middleware.
- Er nimmt einen Strom von Request entgegen von vielen Clients
- Er bietet „failover“ Management, wenn einer der Server versagt und steuert Transaktionen im Auftrag des Clients
- Er bietet Übersetzung von Kommunkationsprotokollen an

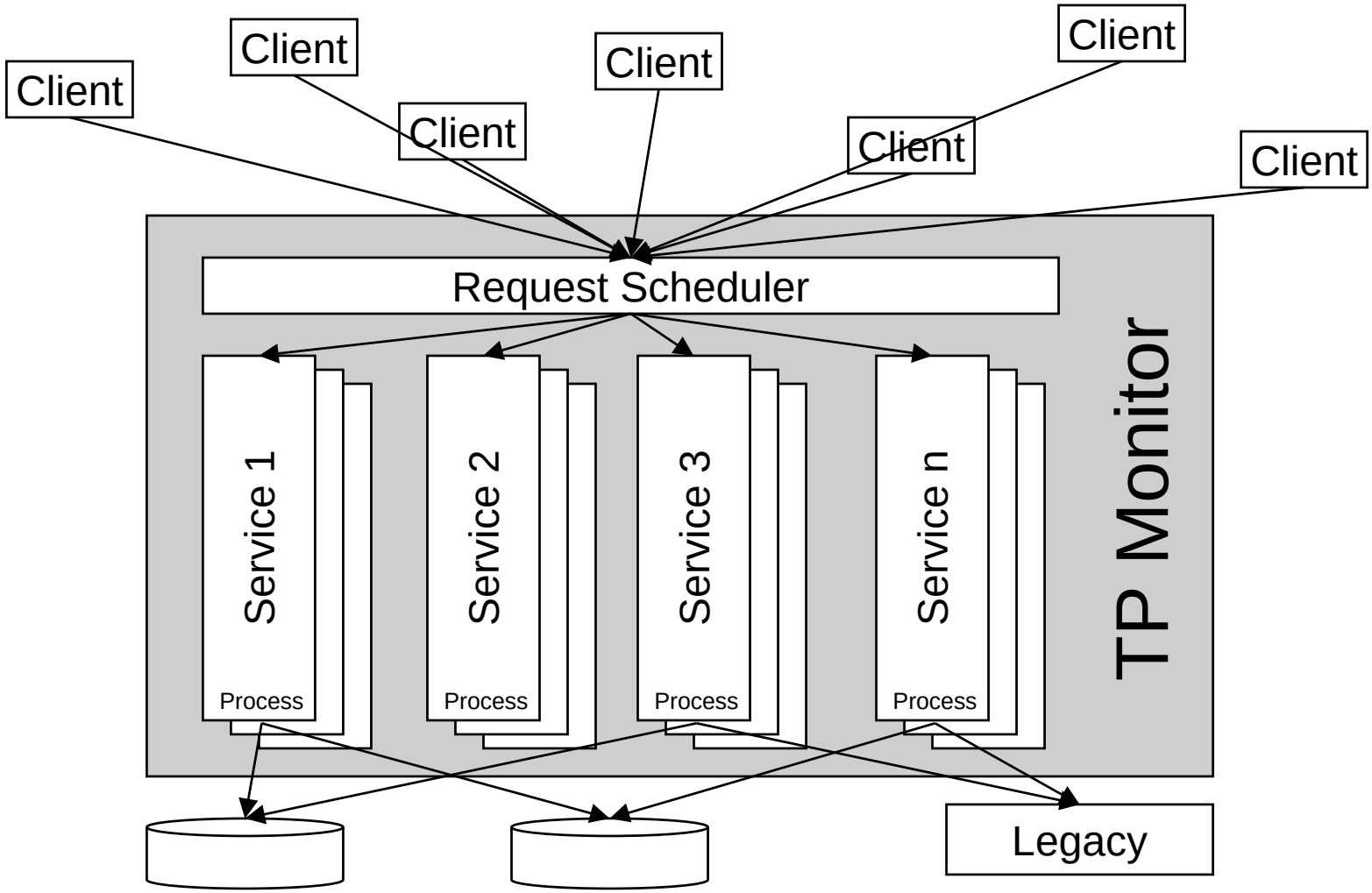
© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

TP-Monitore – ganz allgemein (2)

- Er konsolidiert Requests und Antworten darauf (Responses) ziwschen Clients und vielen heterogenen Servern
- Er bietet darüber hinaus umfangreiche Funktionen zu seiner eigenen Steuerung an (Service Monitoring)

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

TP Monitor - allgemein



© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

CICS als Beispiel für einen Transaktionsmonitor

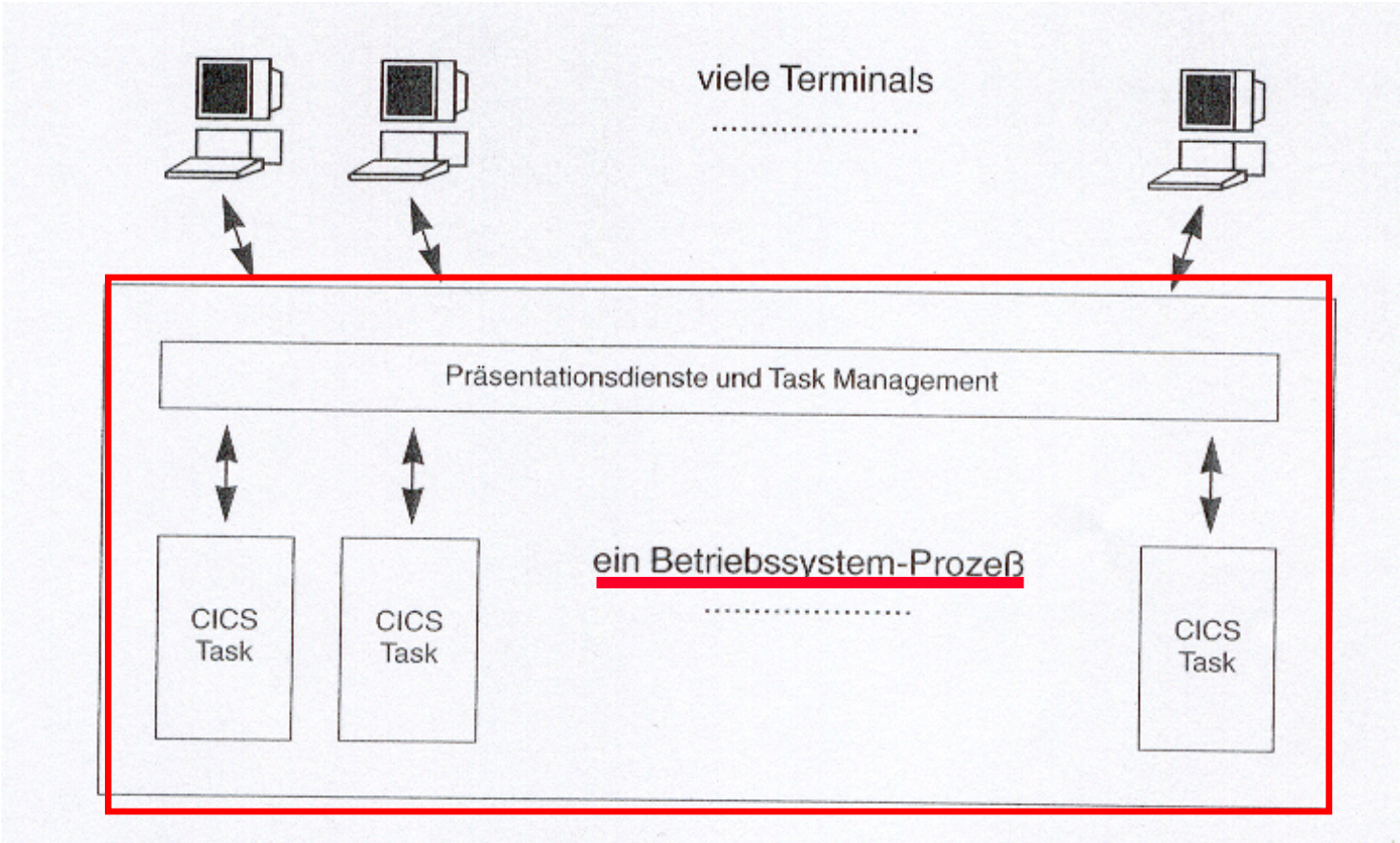
© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

CICS heißt

- **C**ustomer **I**nformation **C**ontrol **S**ystem
  - Beinhaltet Unterstützung für „transactional programming“
  - Und für 2PC – also Unterstützung für Transaktionen mit mehr als einer „managed ressource“

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

CICS Prozessorganisation



Quelle: Brössler et al.; Softwaretechnik, Hanser 2000

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

CICS typisch – wie es verwendet wurde und wird

Typical Scenario

CICS tasks are usually started by someone at a 3270 terminal (or its equivalent) pressing one of the 3270 "transmit" keys (ENTER or one of the PFkey). They are usually ended by a CICS application program explicitly ending the task. A typical scenario:

- A user hits Enter.
- CICS remembers the transaction code left behind by the previous task (or CICS picks it out of the screen data).
- CICS verifies that this terminal user is authorized for that transaction code.
- CICS schedules the initial program for that transaction code, *the task is born*. Any passed COMMAREA is placed in the "linkage section".
- That (or subsequent) program "maps" the terminal data to see what has been asked of it.
- If appropriate, other programs are invoked.
- VSAM or DB2 data is retrieved and/or written.
- A program sends a screen image back to the terminal.
- A program returns control to CICS. It identifies which transaction to run the next time that terminal operator responds and the data to pass to the initial program. *This ends the task*.

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Unter CICS Programmierern bekannte Eigenschaften

- Alle CICS Tasks eines CICS laufen im selben Prozeß
- Damit sind sie nicht gegeneinander geschützt
  - Man kann durch eine Endlosschleife eine ganze CICS Instanz aufhängen
  - Man kann durch Programmierfehler die Daten anderer Tasks zerstören oder das komplette CICS abschießen

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

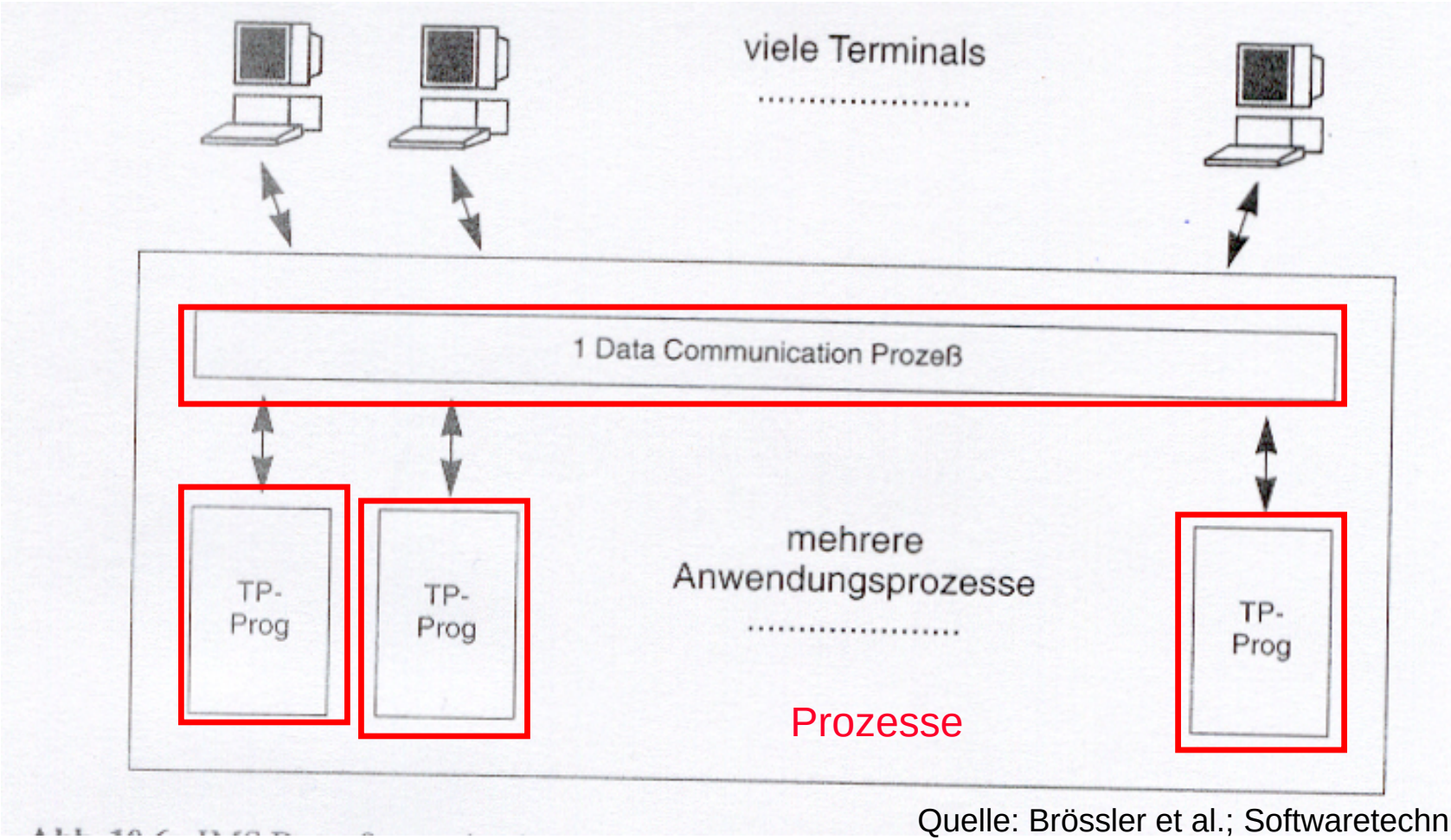
Aufbau eines CICS „Anwendungsprogrammes“

```
CICS-DIALOGPROGRAMM.
    EXEC CICS RECEIVE
        MAP(Maskenname)
        INTO(Masken-Datenstruktur)
    END-EXEC
    PERFORM Zustand-Info-herstellen
    PERFORM Eingabe-Verarbeiten-Ausgabe-Aufbereiten
    PERFORM Zustand-Info-sichern
    EXEC CICS SEND
        MAP(Maskenname)
        FROM(Masken-Datenstruktur)
    END-EXEC
    EXEC CICS RETURN
    END-EXEC.
```

Quelle: Brössler et al.; Softwaretechnik, Hanser 2000

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Zum Vergleich: IMS Prozeßmodell



Quelle: Brössler et al.; Softwaretechnik, Hanser 2000

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Eigenschaften von IMS

- Jedes „Anwendungsprogramm“ läuft bei IMS in einem oder mehreren Prozessen
- Damit besteht Speicherschutz
  - Ein fehlerhaftes Programm kann das IMS-System nicht abschießen
  - Ein fehlerhaftes Programm kann aber sehr wohl die Daten einer anderen Session des selben Programmes „zerschießen“
  - Aber nicht die anderer „Programme“

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Und zum Vergleich die IMS „Programmschleife“

```
IMS-DIALOGPROGRAMM.
   PERFORM VERARBEITUNG UNTIL END-OF-MESSAGES.
   GOBACK.
VERARBEITUNG.
   PERFORM GET-INPUT-MESSAGE
   IF NOT END-OF-MESSAGES
      PERFORM Zustand-Info-herstellen
      PERFORM Eingabe-Verarbeiten-Ausgabe-Aufbereiten
      PERFORM Zustand-Info-sichern
      PERFORM INSERT-OUTPUT-MESSAGE
   END-IF.
GET-INPUT-MESSAGE.
   CALL 'CBLTDLI' USING  DLI-GU IO-PCB
                         INPUT-MESSAGE-IO-AREA
*  keine Nachrichten in der Eingabe-Queue mehr vorhanden ?
*  -> Flag für Programmabbruch setzen.
   IF STATUS-CODE OF IO-PCB = 'QC'
      SET END-OF-MESSAGES TO TRUE
   END-IF.
```

Quelle: Brössler et al.; Softwaretechnik, Hanser 2000

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Welche Services bieten IMS und CICS also (1)

- Unterstützung für performantes „transaktionales“ Programmieren auf Maschinen, die für Batch-Processing ausgelegt sind
  - Zuordnung von Terminalsessions zu Programmen
- Unterstützung für leichtgewichtige Prozesse
- Halten von Kontext über Dialogschritte (CICS: COMMON AREA)
  - Da gibt es Limits, z.B. 32kByte Größenbeschränkung – Beeinflusst den Programmierstil
- DLL-artiges, performantes Nachladen von Unterpogrammen
  - Und wie schaut das mit richtigen DLLs aus? Gibt es auch seit zOS.

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Welche Services bieten IMS und CICS also (2)

- Transaktionsklammer (Ressorce Manager) über mehr als eine transaktionsgesicherte Ressource .. Also zum Beispiel
  - DB2
  - IMS-DB
  - Und von jedem auch mehr als eine Instanz

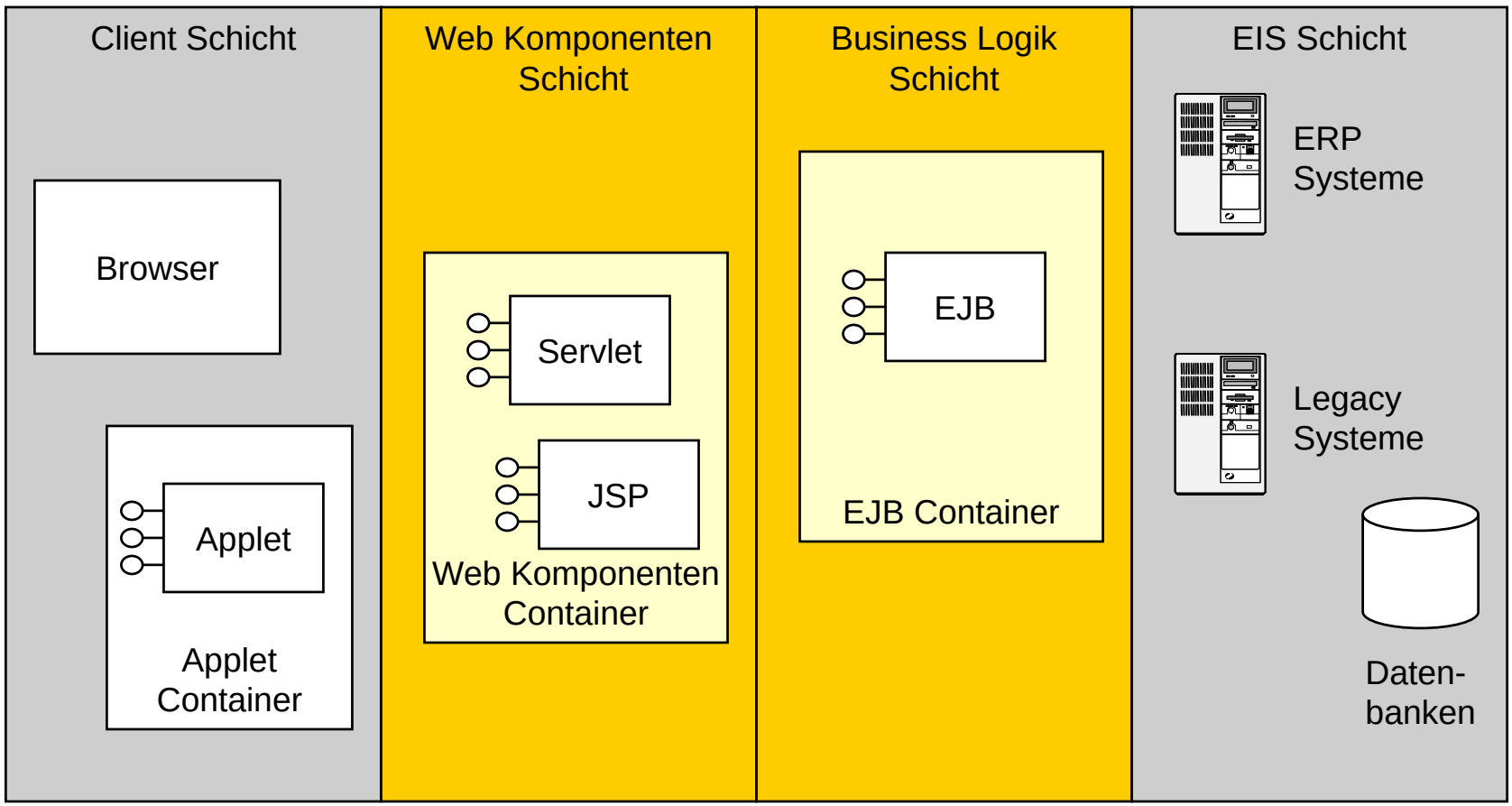
- Es gibt aber auch Gefahren
  - Schlecht programmiertes CICS kann sogar den SYSPLEX beeinträchtigen, also bis zur Sicherungsmaschine durchschlagen.

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Was leistet im Vergleich dazu ein J2EE Applikationsserver?

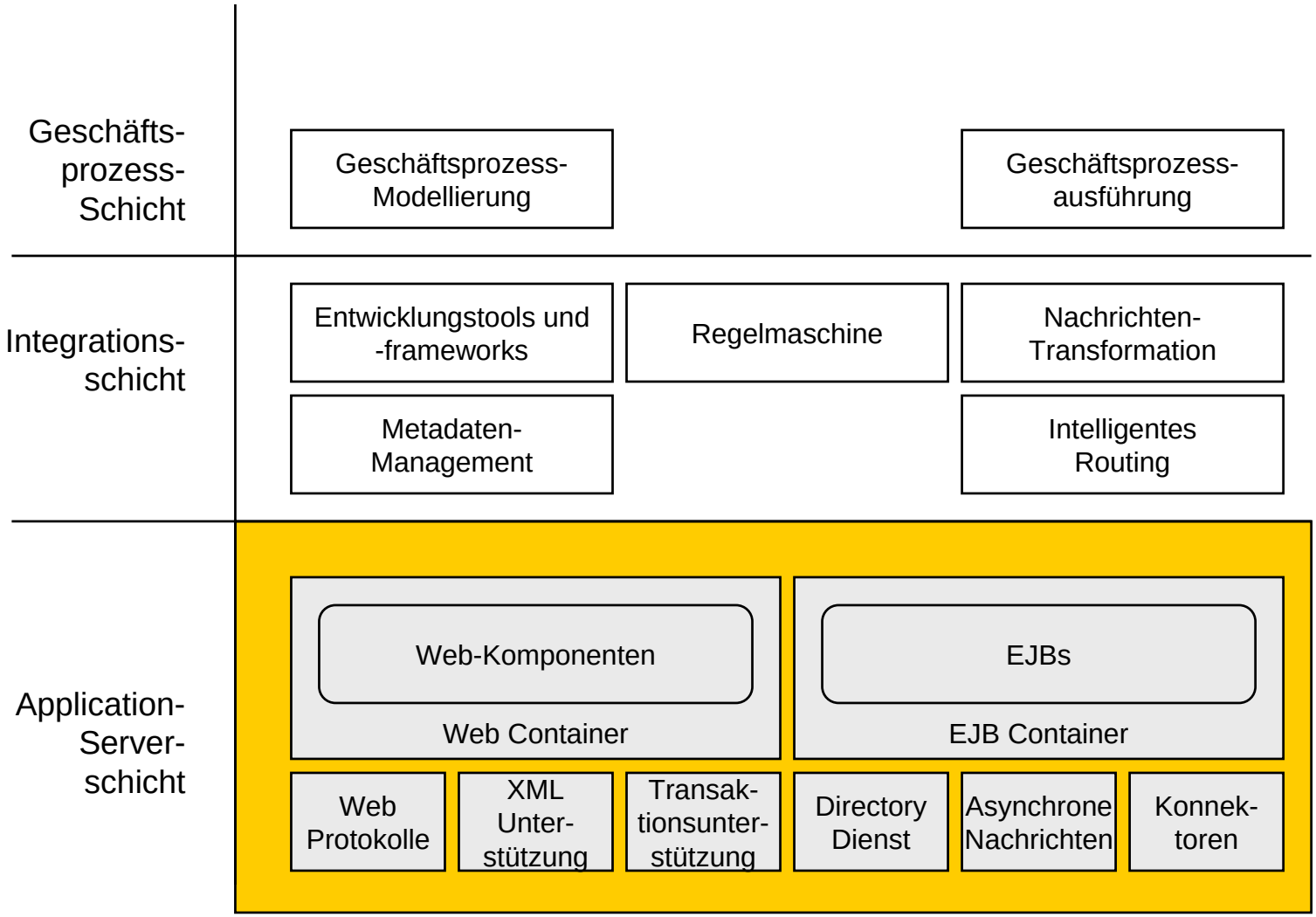
© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

J2EE Applikationsserver
Wo finde ich den?



© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

J2EE Applikationsserver
Wo finde ich den?



© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

J2EE Applikationsserver
Wesentliche Dienste

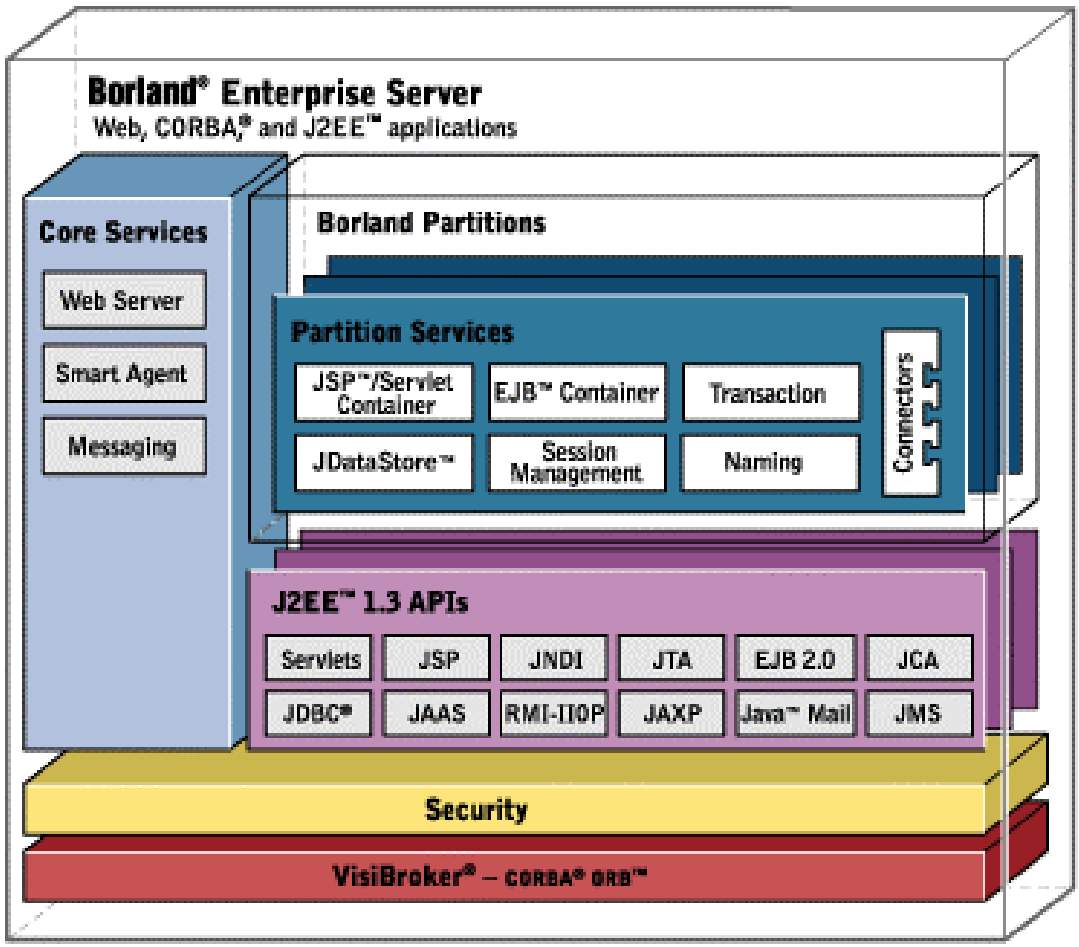
- Ressource Manager für TA – Steuerung
- Container für Programme
  - JSP, EJB, Servlets
- Persistenzdienste (CMP und BMP)
- Load Balancing
- Security
- Messaging
- Name Services
- Backend Connectivity (Konnektoren)

Bei CICS/IMS so nicht direkt vorhanden ...

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

J2EE Applikationsserver

Selbstbeschreibung eines Repräsentaten ...



Quelle:
Borland

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004

Zusammenfassung

- Namen und Programmiersprachen haben sich geändert
- Produkte sind besser geworden – die Frameworks sind umfangreicher
- Herausforderungen sind ähnlich geblieben
  - Performantes Bedienen von Requests vieler Clients
  - Transaktionssteuerung
  - Weitere Anwendungsdienste
- Kenntnis „der alten Eisen“ immer nocht ganz nützlich – weil man sie immer wieder integrieren muß

© 2004 Hans Hartmann, Wolfgang Keller, all rights reserved, Universität Leipzig, SS2004