Vorlesung Software-Management

Sommersemester 2010

Computer-Aided Software Engineering

Prof. Dr. K.-P. Fähnrich / Thomas Riechert

01.06.2010

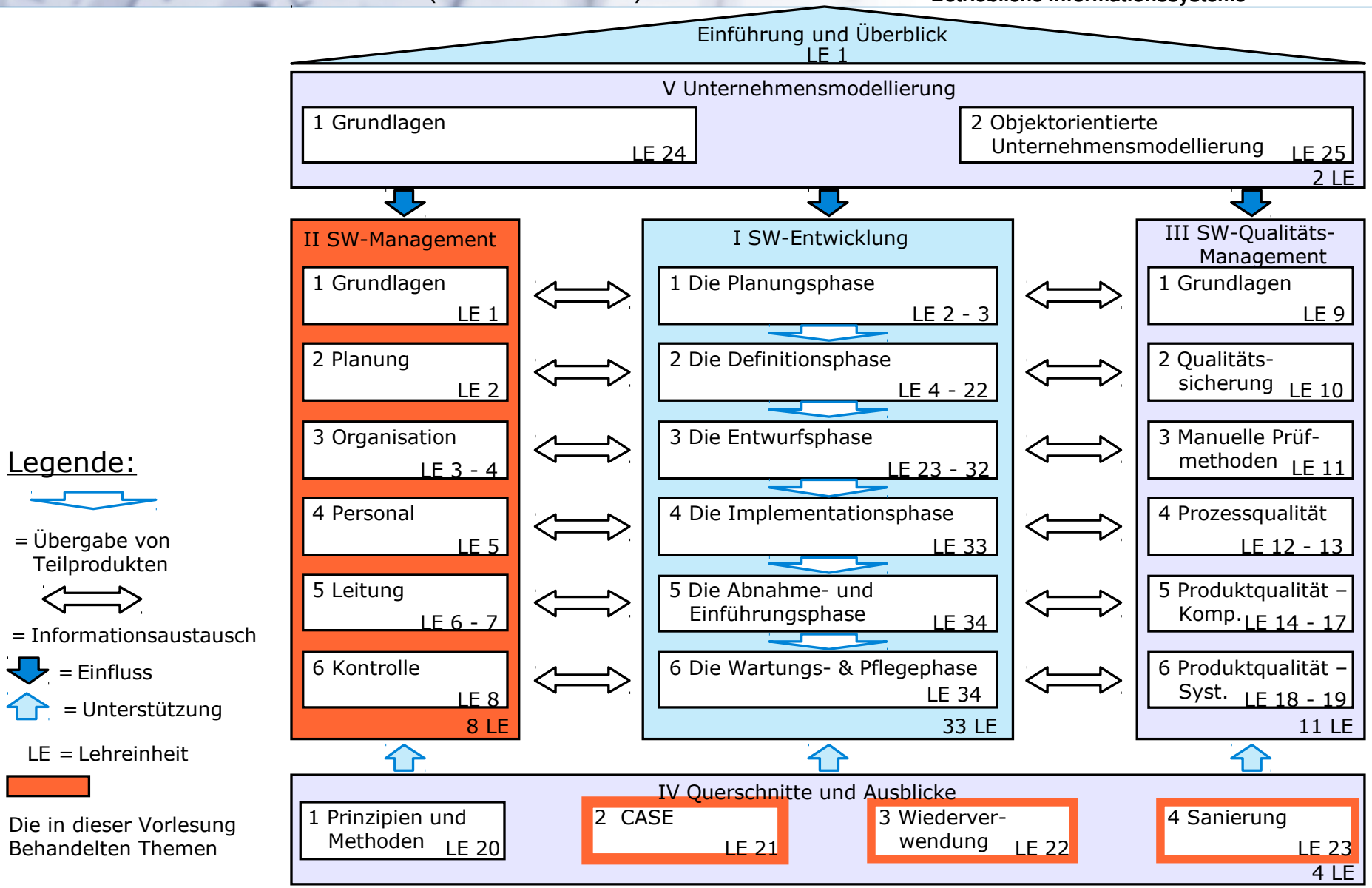
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

(1) Grundlagen
(2) Planung
(3) Organisation: Gestaltung
(4) Organisation: Prozessmodelle
(5) Personal
(6) Leitung
(7) Innovationsmanagement
(8) Kontrolle: Metriken, Konfigurations- und Änderungsmanagement
(9) CASE
(10) Wiederverwendung
(11) Sanierung

Begleitliteratur: Helmut Balzert, Lehrbuch der Software-Technik
Quelle der Grafiken und Tabellen: Helmut Balzert, Lehrbuch der Software-Technik, wenn nicht anders angegeben

Haus der Softwaretechnik (nach Balzert)

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme



Legende:

= Übergabe von Teilprodukten

= Informationsaustausch

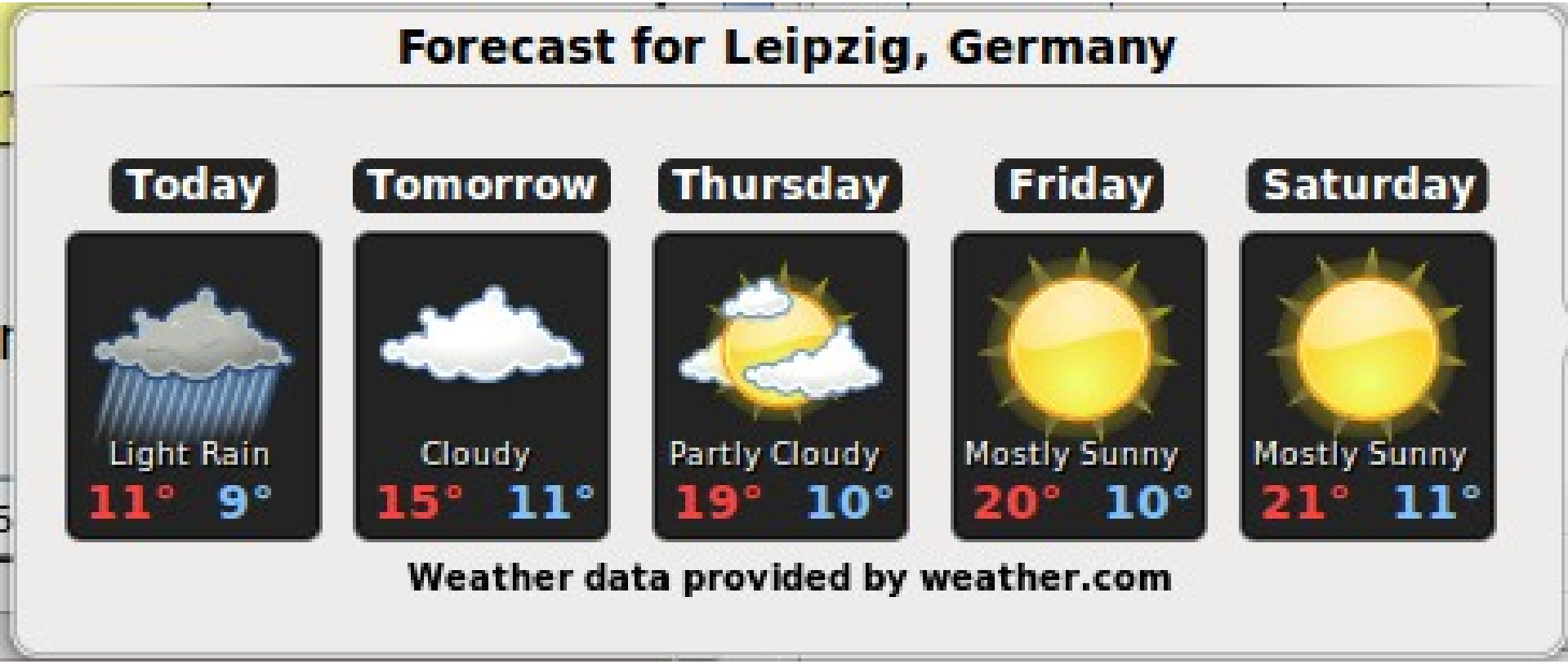
= Einfluss

= Unterstützung

LE = Lehreinheit

Die in dieser Vorlesung Behandelten Themen

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

1. **Einführung**
   - 1.1. Was ist CASE?
   - 1.2. CASE-Werkzeugkategorien
   - 1.3. Ziele von CASE
   - 1.4. Allgemeine Anforderungen an Werkzeuge, Plattformen und Umgebungen
2. **Zur Auswahl von CASE-Umgebungen**
3. **Evaluationsverfahren für CASE**
4. **Kosten/Nutzen von CASE**
5. **CASE-Tools in der Praxis**

   Aus der Diplomarbeit von Herrn Gahlert

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- **CASE**: Computer Aided Software Engineering

CASE befasst sich mit allen computerunterstützten Hilfsmitteln, die dazu beitragen, die SW-Produktivität und die SW-Qualität zu verbessern sowie das SW-Management zu erleichtern.

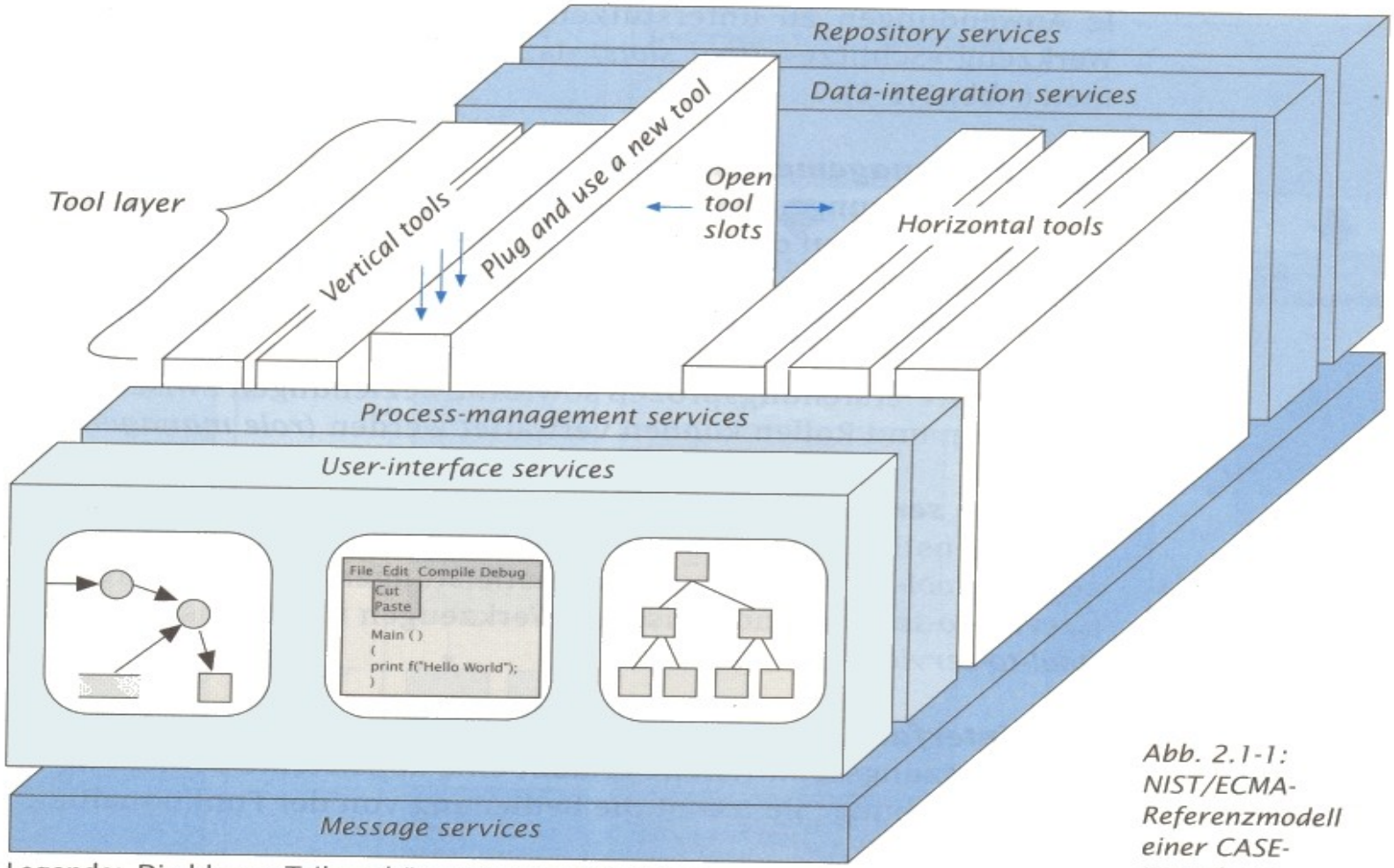
- CASE lässt sich gliedern in:
  - CASE-Werkzeuge:
    - Software-Produkte, die zumindest einzelne bei der SW-Erstellung benötigte Funktionen bzw. Dienstleistungen zur Verfügung stellen.
  - CASE-Plattformen:
    - Stellen Basisdienstleistungen, ein Repository und einen Nachrichtendienst zur Verfügung.
- Beide zusammen ergeben CASE-Umgebungen, auch Software-Entwicklungs-umgebungen (SEU) genannt.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- **Repository services**: Zur Verwaltung von Objekten und ihren gegenseitigen Beziehungen.
- **Data-integration services**: Zur Handhabung der Daten auf einem höheren semantischen Niveau und manchmal der Metadaten.
- **Werkzeuge (tools)**: Stück Software, das nicht Teil der CASE-Plattform ist und das Dienstleistungen der Plattform in Anspruch nimmt.
- **Process-management services**: Zur Kommunikation auf der Ebene der zu erledigenden Aufgaben (role management).
- **Message services**: Zur Kommunikation zwischen den Werkzeugen, Dienstleistungen und zwischen Werkzeugen und Dienstleistungen.
- **User-interface services**: Zur Bedienung der Umgebung.

→ CASE-Werkzeug Kategorien

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme



Legende: Die blauen Teile gehören zur CASE-Plattform
Vertical tools: Werkzeuge, die den gesamten Lebenszyklus begleiten, wie das Konfigurationsmanagement
Horizontal tools: Phasenorientierte Werkzeuge, z.B. SA-Werkzeug

Abb. 2.1-1: NIST/ECMA-Referenzmodell einer CASE-Umgebung (»Toaster«-Modell) /Chen, Norman 92, S. 19/

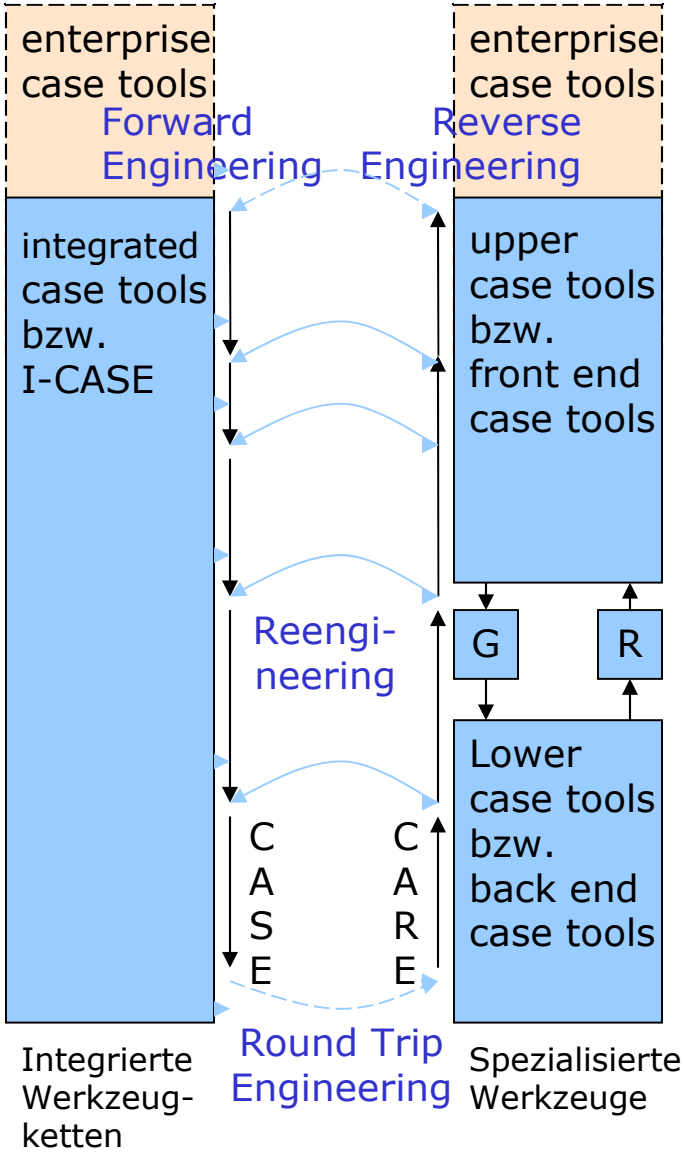
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

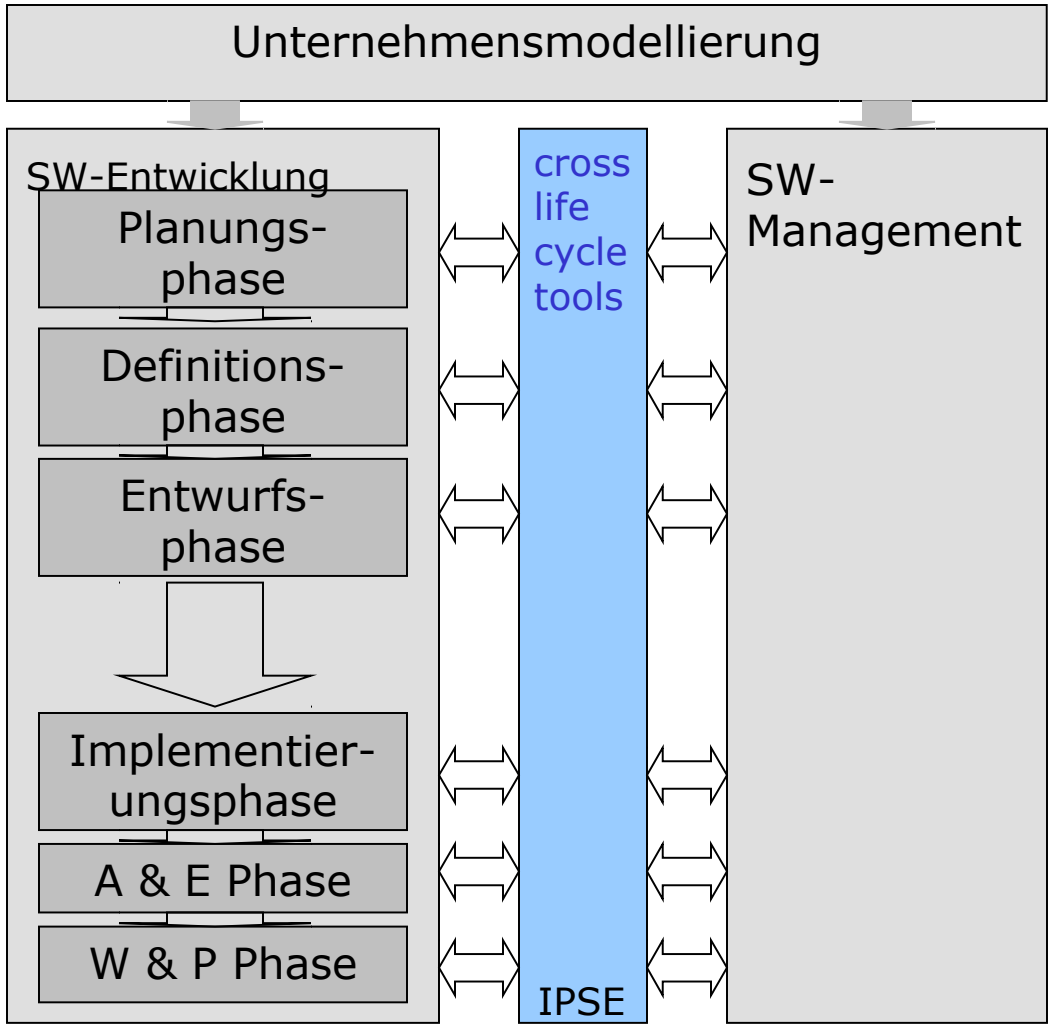
- Beim **Forward Engineering** ist das fertige Software-System das Ergebnis des Entwicklungsprozesses.
- Zum **Reverse Engineering** gehört das Extrahieren von Konstrukten und das Erstellen oder Synthetisieren von Abstraktionen.

  Reverse Engineering kann in jeder Entwicklungsphase gestartet werden. Das vorhandene SW-System ist der Ausgangspunkt der Analyse.
- **Re-Engineering** umfasst die Überprüfung und den Umbau des vorhandenen Systems, so dass eine Wiederherstellung in neuer Form erreicht wird.
- **Round Trip Engineering** bedeutet, an einer beliebigen Stelle als Ausgangspunkt beginnen und an einer beliebigen Stelle enden können (Einsatz von Forward- und Reverse Engineering).

CASE-Werkzeugkategorien

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme





IPSE = integreted project support environment
G= Anwendungsgeneratoren (*application generators*)
R = redesign-Transformator

[Balzert98, S. 595]

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Die globalen Ziele des Einsatzes von CASE sind
  - die Erhöhung der Produktivität,
  - die Verbesserung der Qualität und
  - die Erleichterung des Software-Managements.

- Diese Ziele lassen sich weiter untergliedern in:
  - technische Ziele,
  - wirtschaftliche Ziele und
  - organisatorische Ziele.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Technische Ziele

- Qualität der Dokumentation verbessern.
- Wiederverwendbarkeit erleichtern.
- Verwalten von Konfigurationen und Änderungen.
- Hinweise auf mögliche Schwachstellen (Metriken).
- Generatoren einsetzen, um Arbeitsschritte zu eliminieren.
- Methoden einsetzen, die Konsistenz- und Redundanz-überprüfungen ermöglichen.

Wirtschaftliche Ziele

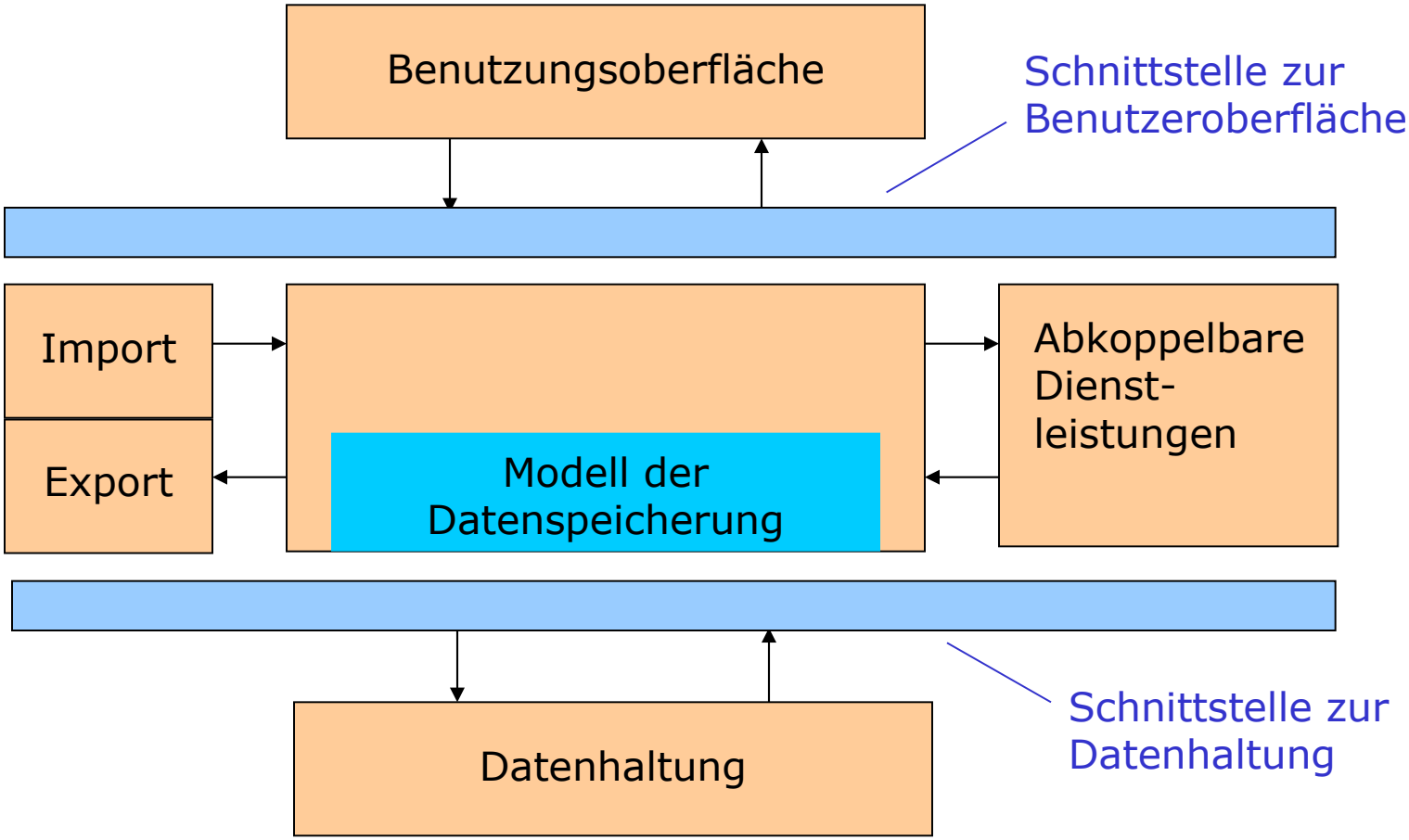
- Effektivität erhöhen.
- Produkte schneller entwickeln.
- Qualität erhöhen.
- Wartungsaufwand reduzieren.
- Personenanbhängigkeit verringern.
- Änderungen noch kurz vor Markteinführung ermöglichen.

Organisa-torische Ziele

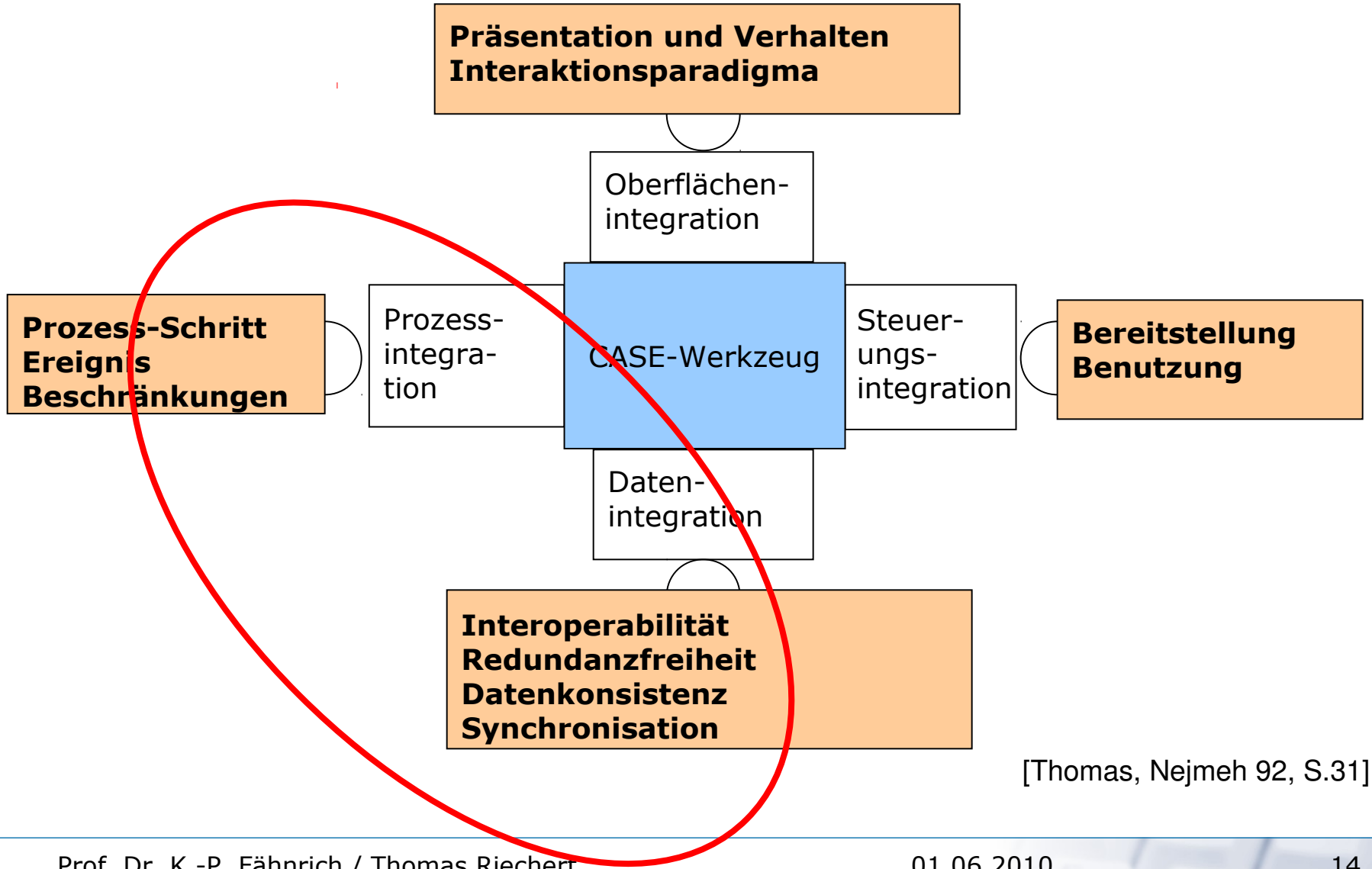
- Unterstützung des gewählten Prozessmodells.
- Verbesserung der Entwicklungsmethoden und –verfahren.
- Erhöhung der Standardisierung.
- Jederzeit Information über den Ist- und Soll-Zustand.
- Flexible Anpassung an geänderte Rahmenbedingungen.

Notwendige Architektur eines CASE-Werkzeugs

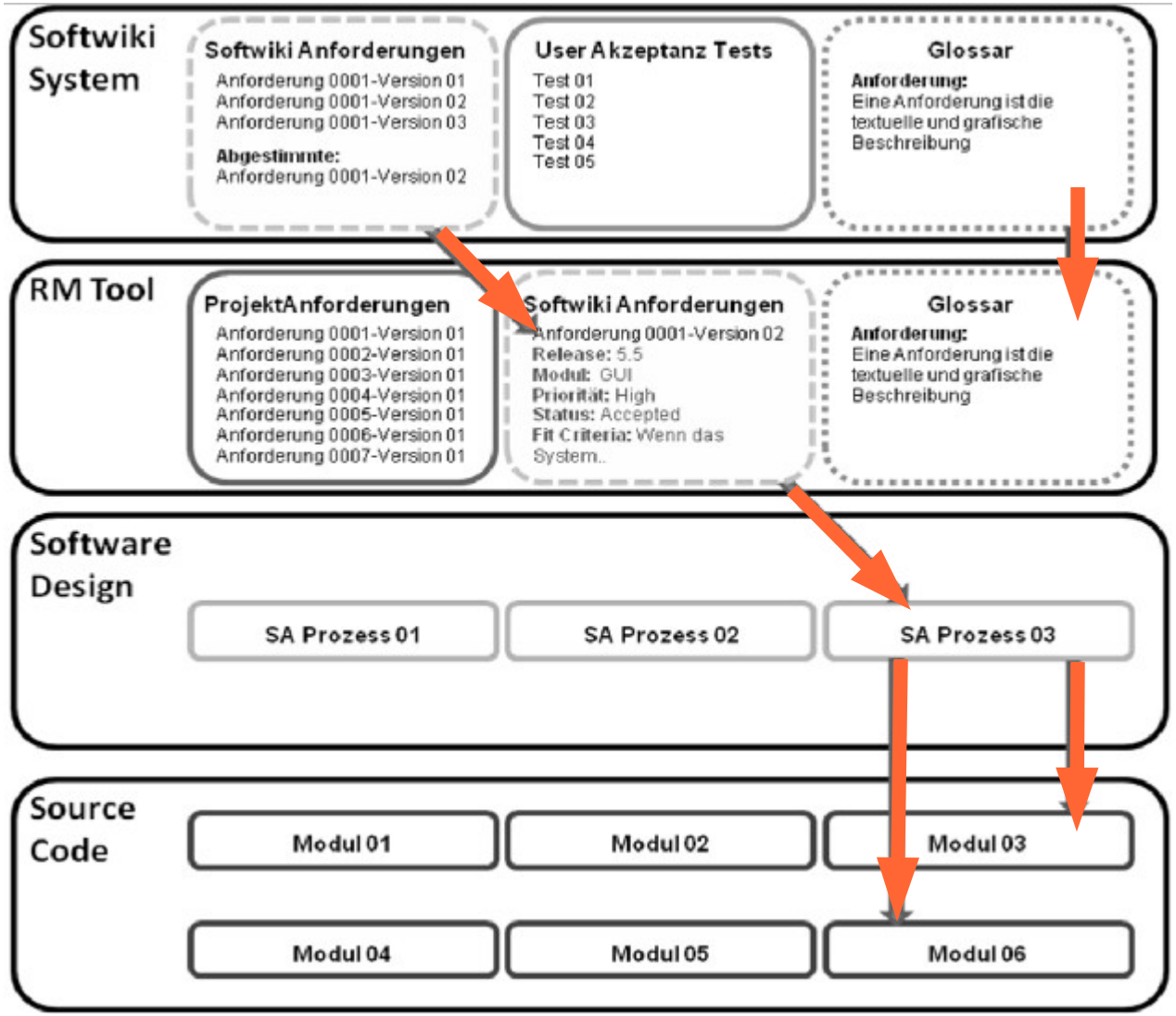
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme



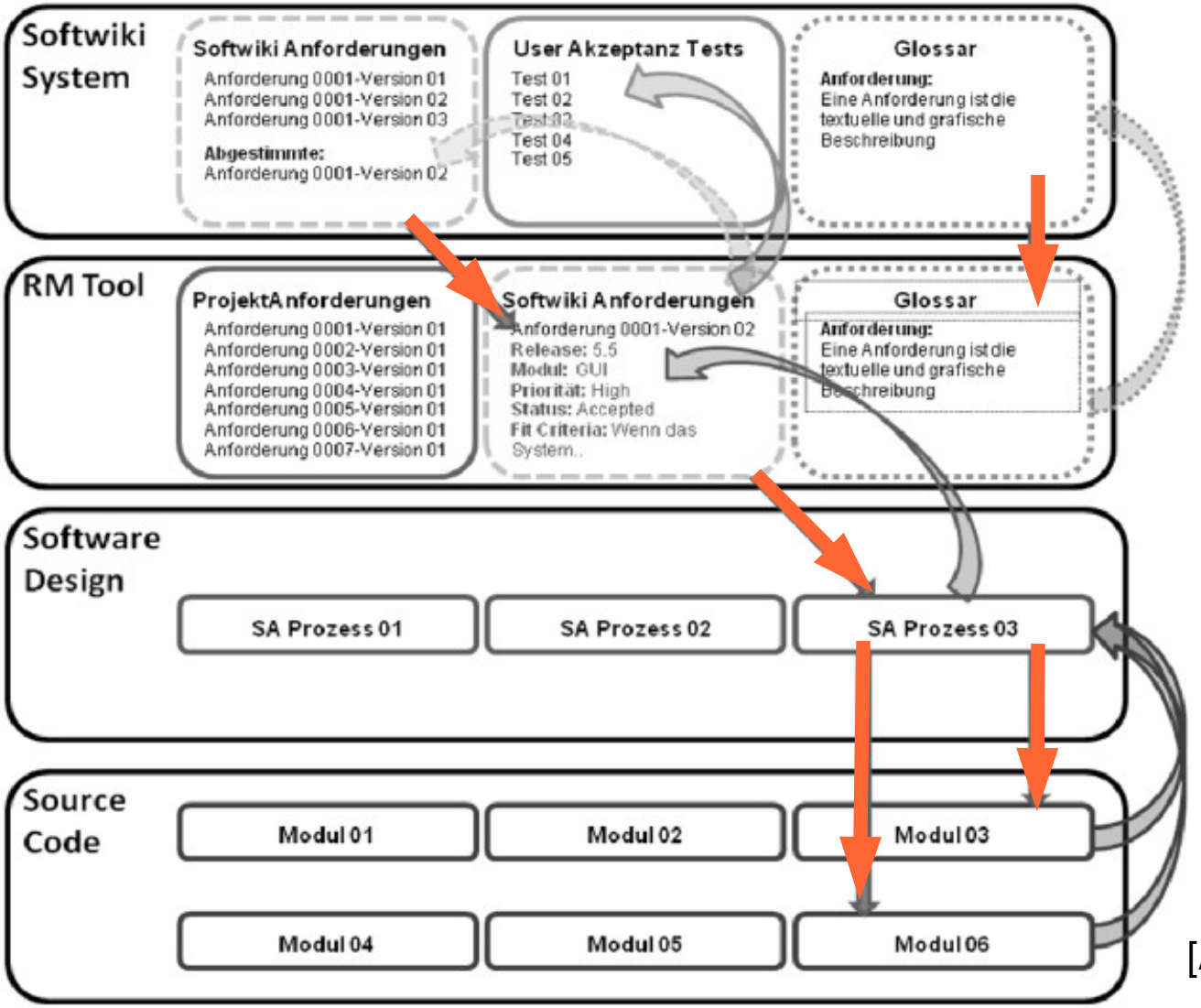
[Balzert98, S.604]



[Thomas, Nejmeh 92, S.31]



[Auer et al. 2009]



[Auer et al. 2009]

Allgemeine Anforderungen an CASE-Werkzeuge (1)

1. Weitgehend „methodentreue“ Unterstützung
   - Bei der Auswahl von CASE-Werkzeugen ist darauf zu achten, dass sie die ausgewählten Methoden nicht zu stark verfälschen.
2. Bereitstellung der notwendigen Basisfunktionalität.
3. Bereitstellung effizienzverbessernder Funktionen.
   - Der produktive Ansatz wird verbessert durch Spezialfunktionen, Makrorekorder, Speicherung des aktuellen Zustandes, ...
4. Bereitstellung qualitätssteigernder Funktionen.
   - Um die Qualität zu verbessern müssen CASE-Werkzeuge:
     - erstellte Produkte oder Teilprodukte laufend überprüfen,
     - geeignete QS-Protokolle erstellen,
     - auf mögliche Schwachstellen hinweisen und
     - fehlerhafte Eingaben verhindern.

Allgemeine Anforderungen an CASE-Werkzeuge (2)

5. Übernahme von Hilfs- und Routinearbeiten

6. Intuitive Bedienung

7. Bereitstellung von Export- und Import-Schnittstellen

   - Drei Möglichkeiten:

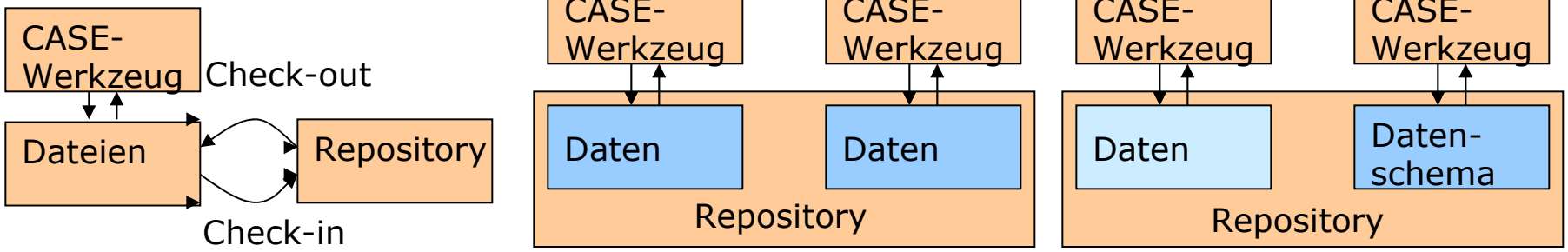
     1. Definierte Austauschformate,
     2. Veröffentlichung von Klassen des Metamodells,
     3. Offenlegung der Speicherstruktur des CASE-Werkzeugs.

8. Integrierbarkeit in CASE-Plattformen

1. Integrationsfähigkeit von CASE-Werkzeugen

- Grade der Daten-Integration [Rammig, Steinmüller 92]:
  - Black-box-Integration, Grey-box-Integration, White-box-Integration.
- Realisierungsmöglichkeiten eines Repository:
  - Dateisystem, DBS.

[Balzert98, S.608]



Black-Box-Integration **Grey-Box-Integration** **White-Box-Integration**

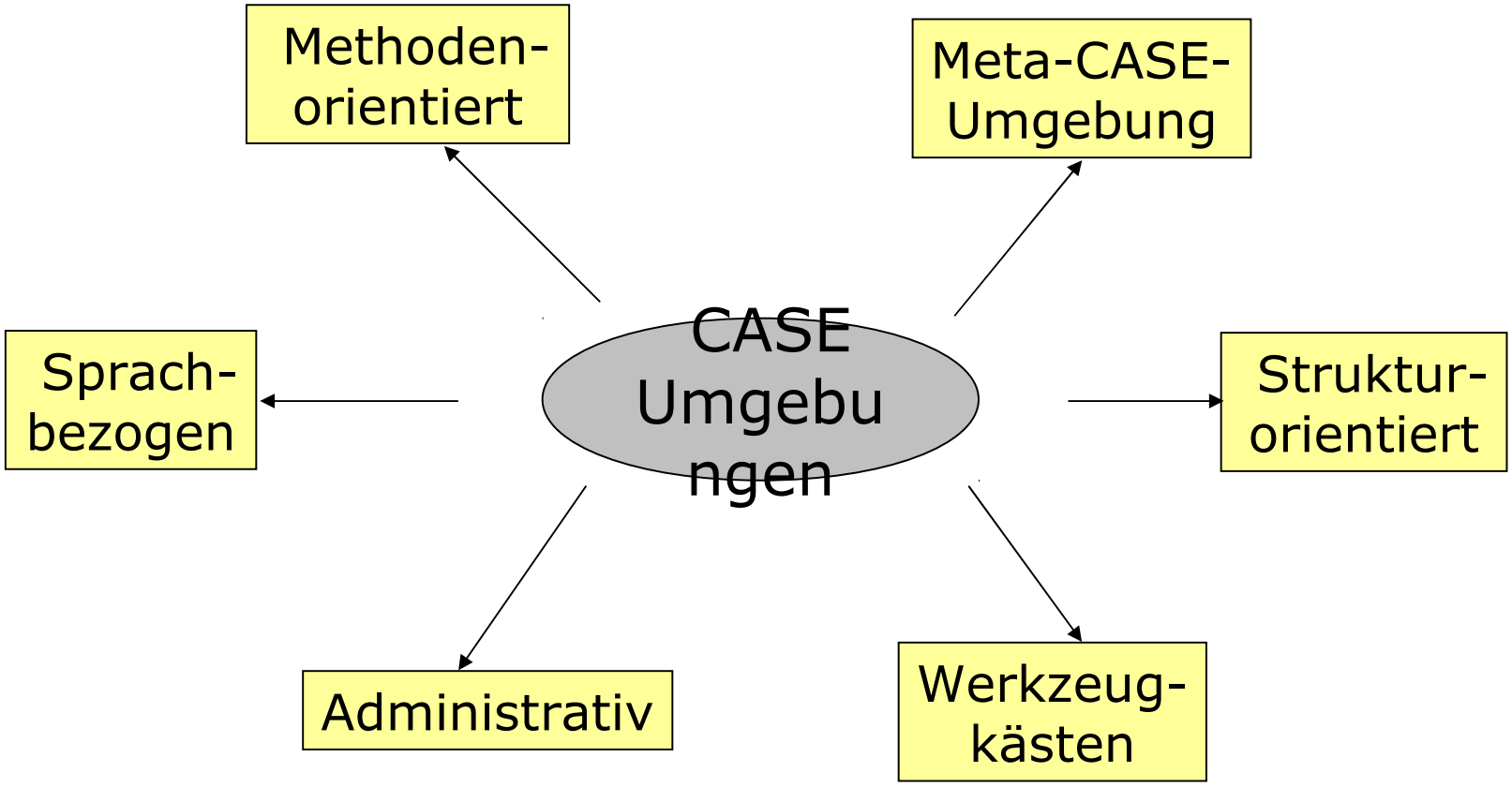
2. Offenheit von CASE-Plattformen

- Eigenschaften einer offenen CASE-Plattform:
  - Definierte Import-/Export-Schnittstellen.
  - Möglichkeit des Einbettens oder Entfernens von CASE-Werkzeuge in die Plattform.
  - Anpassbarkeit.

3. Multiprojekt- und Interprojekt-Fähigkeit

- Multiprojekt-Fähigkeit: Verwaltung mehrerer parallel oder zeitlich versetzt laufender Entwicklungen.
- Interprojekt-Fähigkeit: separate Verwaltung projektspezifischer Informationen.
- Mögliche technische Realisierung: Zentrales Repository.

Klassifikation von CASE-Umgebungen



1. Vollständigkeit

- Eine vollständige CASE-Umgebung unterstützt die zwei Tätigkeitsgruppen:
  - Tätigkeiten, die bei fast allen Prozessmodellen auftreten,
  - Prozessmodellspezifische Tätigkeiten.
- Eine CASE-Umgebung heißt partiell vollständig, wenn sie einen Teilbereich der Software mit semantisch integrierten Hilfsmitteln unterstützt.

2. Inkrementeller Einsatz

- Die überwiegende Mehrzahl der CASE-Umgebungen sind nur partiell vollständig und nicht inkrementell einführbar.
- Viele Anwender stellen sich eigene, nicht-integrierte CASE-Werkzeuge aus verschiedenen Werkzeuggruppen zusammen.

Heutige Situation: Methoden und CASE

- Erkenntnisse:
  - Analyse und Entwurf sind Bestandteile jeder Software-Entwicklung
  - Sprachen der 4. Generation reichen im Allgemeinem nicht aus
- Konfigurationsmanagement hat höhere Bedeutung gewonnen.
- Qualitätssicherung hat an Stellenwert zugenommen.
- Prozessmodelle sind wichtiger geworden.
- CASE-Umgebungen unterstützen die Objektorientierte- Software-Entwicklung.
- Das Forward Engineering wird ergänzt durch das Reverse Engineering, das Re-Engineering, das Round Trip Engineering und das Enterprise Engineering.

- Verbesserung der Qualität von CASE-Werkzeugen.
- Einfachere, aber noch nicht intuitiv und individualisierbare Bedienung.
- CASE-Werkzeuge sind preiswerter geworden.
- Zunehmende Objektorientierte-Implementierung von CASE-Werkzeugen und -Plattformen.
- Generierung von mehr, aber immer noch nicht ausreichend viel Code.
- Stagnierung der Standardisierung von Schnittstellen, Metamodellen und Plattformen.
- Die Realisierung einer vollständigen CASE-Umgebung ist gescheitert.
- CASE-Werkzeuge und –Umgebungen unterstützen heute vorwiegend die Realisierung von kaufmännischen und technischen Anwendungen.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Software-Anwendungssysteme werden weniger umfangreich als heute:
  - Kleinere CASE-Werkzeuge und -Umgebungen
  - Abnahme der Mitarbeiterkoordination
  - CASE muss das Schnittstellenmanagement unterstützen
- Software-Anwendungssysteme werden internet- und intranetfähig sein:
  - Die Internet- und Intranet-Konzepte müssen unterstützt werden
- CASE-Werkzeuge bestehen aus kleinen Komponenten [Microtool 96].
  - Einzelne, eigenständige Komponenten
  - Eigene Datenhaltung
  - Partielle Referenz auf andere Werkzeuge
  - Auswahl des für den Anwender besten Werkzeugs

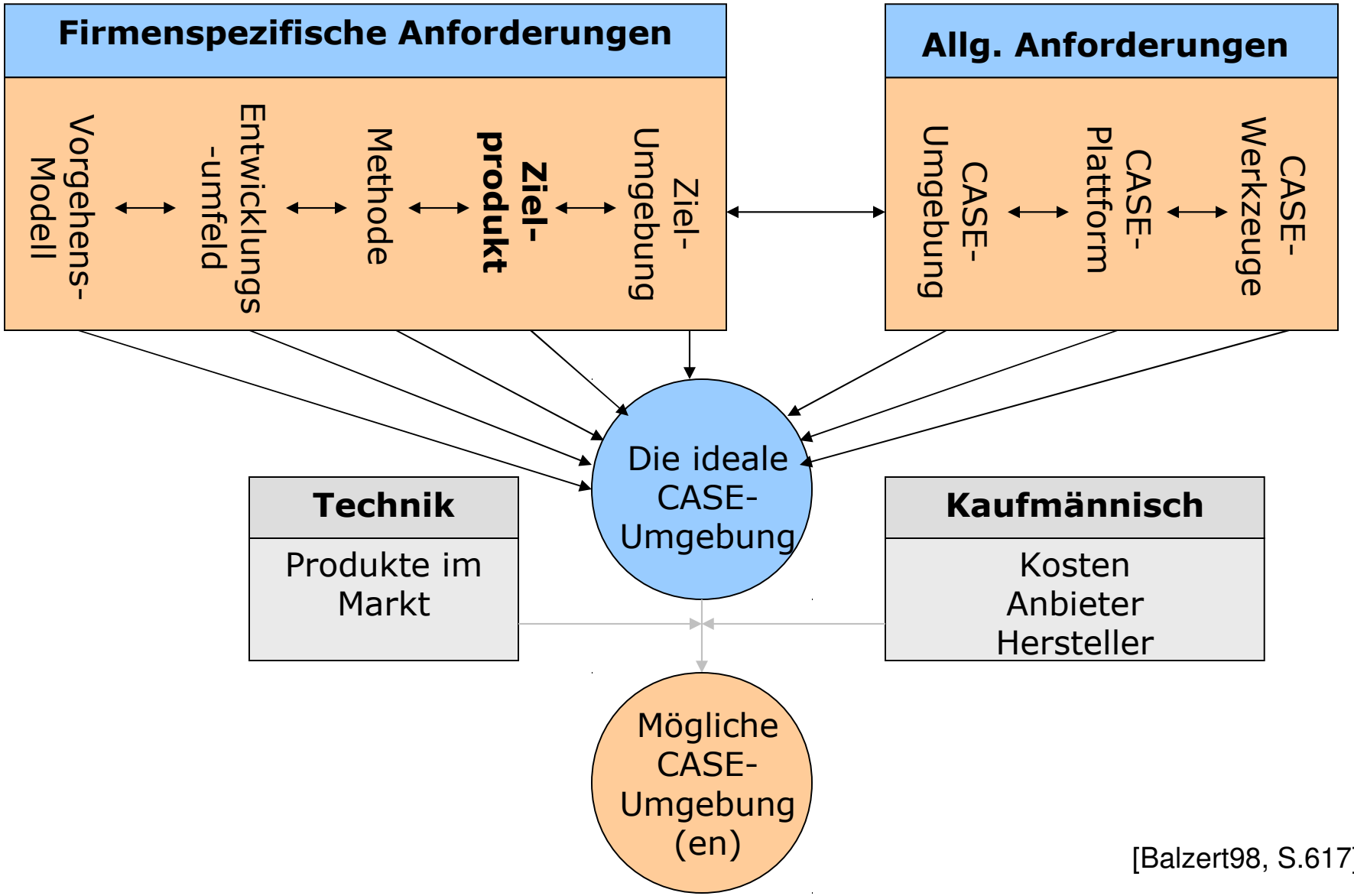
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

1. **Einführung**
   - 1.1. Was ist CASE?
   - 1.2. CASE-Werkzeugkategorien
   - 1.3. Ziele von CASE
   - 1.4. Allgemeine Anforderungen an Werkzeuge, Plattformen und Umgebungen
2. **Zur Auswahl von CASE-Umgebungen**
3. **Evaluationsverfahren für CASE**
4. **Kosten/Nutzen von CASE**
5. **CASE-Tools in der Praxis**

   Aus der Diplomarbeit von Herrn Gahlert

Bestimmende Faktoren der CASE-Auswahl

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme



[Balzert98, S.617]

Zehn Schritte bis zur Entscheidung

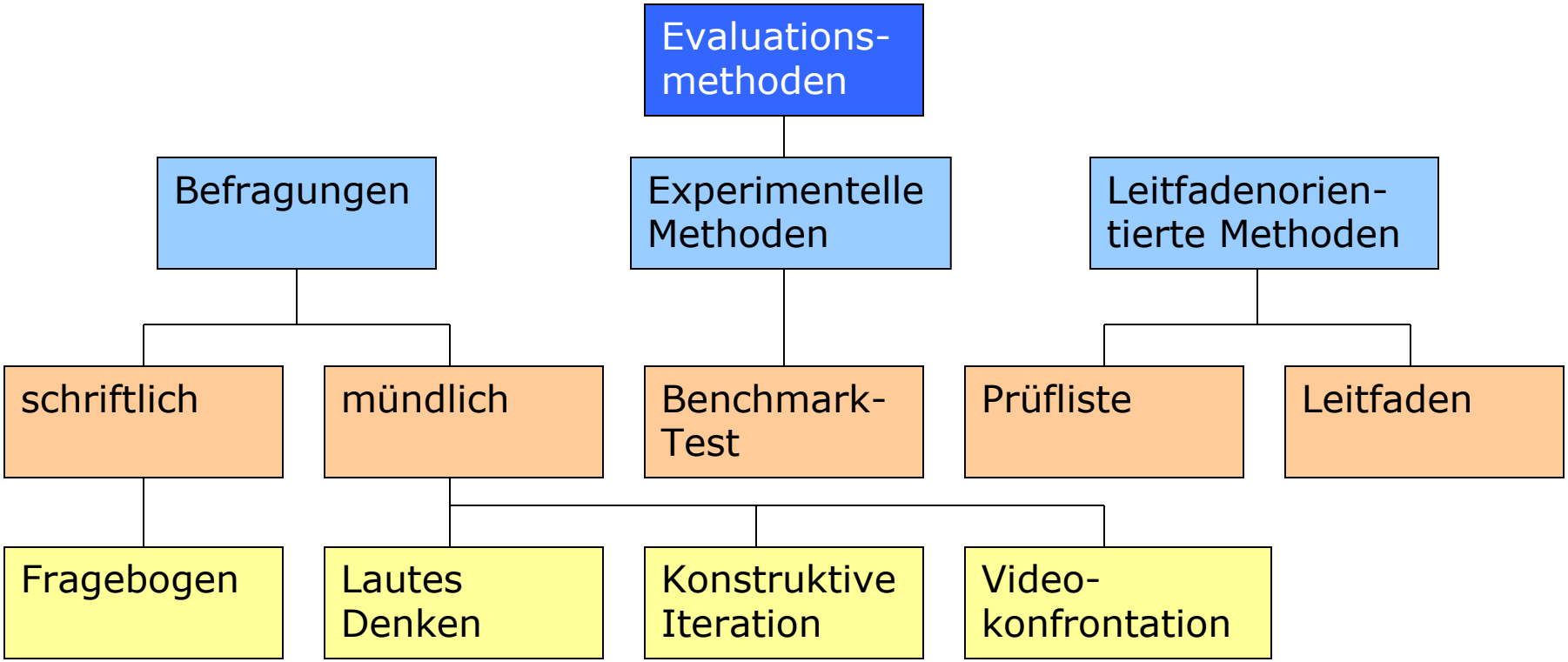
1. Aufstellung eines Kriterienkatalogs.
2. Gewichtung der Kriterien.
3. Vorauswahl der im Markt angebotenen CASE-Umgebungen.
4. Bildung von drei Gruppen:
   a. ausgeschiedene Umgebungen (KO-Kriterien),
   b. engere Wahl,
   c. offene Fragen.
5. Versand eines Fragebogens an die Anbieter der letzten beiden Gruppen und Auswertung des Fragebogens.
6. Quervergleich der 3 bis 5 Umgebungen mit höchster Punktzahl.
7. Evaluation der ausgewählten CASE-Umgebungen.
8. Besuch des Anbieters oder Herstellers sowie Referenzinstallationen.
9. Testinstallation einer oder mehrerer CASE-Umgebungen.
10. Endgültige Entscheidung und Einführung.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

1. **Einführung**
   - 1.1. Was ist CASE?
   - 1.2. CASE-Werkzeugkategorien
   - 1.3. Ziele von CASE
   - 1.4. Allgemeine Anforderungen an Werkzeuge, Plattformen und Umgebungen
2. **Zur Auswahl von CASE-Umgebungen**
3. **Evaluationsverfahren für CASE**
4. **Kosten/Nutzen von CASE**
5. **CASE-Tools in der Praxis**

Aus der Diplomarbeit von Herrn Gahlert

- Funktionalität:
  - Bereitstellung der notwendigen Basisfunktionalität,
  - Bereitstellung effizienzsteigernder Funktionen,
  - Bereitstellung qualitätssteigernder Funktionen,
  - Übernahme von Hilfs- und Routinearbeiten,
  - Bereitstellung von Import-/Export-Schnittstellen.
- Benutzungsoberfläche.
- Qualität.



Überblick über Evaluationsmethoden [Oppermann et al. 92]

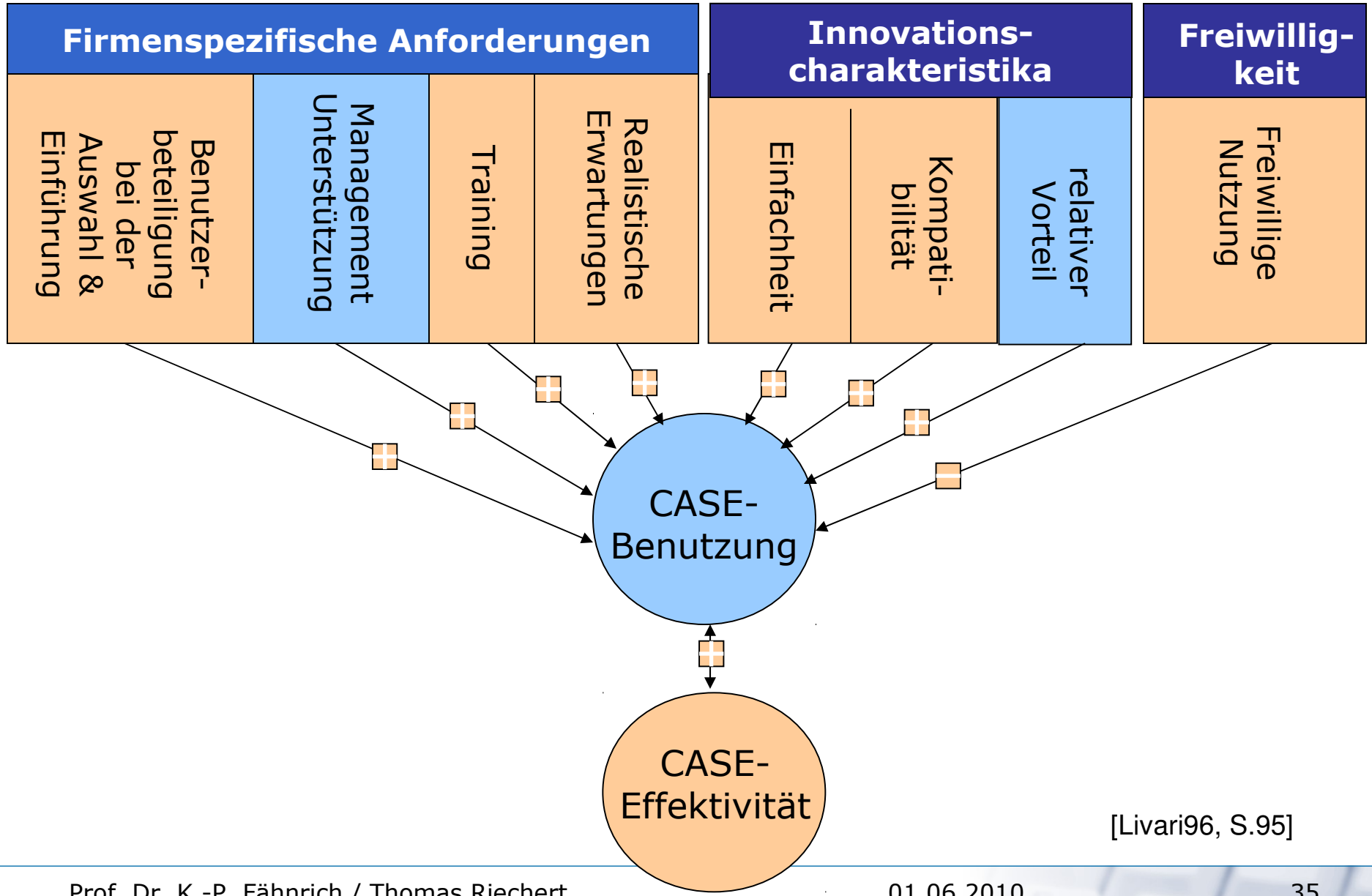
- Besonderheiten:
  - die Benutzer sind Software-Experten;
  - die Funktionalität ist gegen die gewählten Methoden zu evaluieren;
  - es sind Werkzeuge, CASE-Plattformen und deren Zusammenspiel zu überprüfen.
- Bewährt hat sich folgendes Verfahren:
  - Erstellen einer Standardaufgabe, die alle Aspekte der Methode abdeckt;
  - durchführen der Standardaufgabe;
  - durch die Durchführung der einzelnen Schritte prüfen, ob die Basisfunktionalität vorhanden ist, und welche effizienzsteigernden Funktionen angewandt werden können.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Nutzen von CASE

- Beeinflussung der Software-Entwicklung durch CASE [Herzwurm, Hierholzer, Kunz 93]:
  - Trotz anfänglicher Akzeptanzprobleme findet mit CASE eine Konsolidierung der betrieblichen Software-Entwicklung statt.
  - CASE bietet ein deutliches Produktivitätssteigerungspotenzial.
    - Der Produktivitätszuwachs liegt zwischen 30 und 600%.
  - Die Qualität des Entwicklungsprozesses und der Software-Produkte verbessert sich spürbar.
- Aber ein Jahr nach der CASE-Einführung wurden
  - 70% der CASE-Werkzeuge niemals,
  - 25% nur durch eine Gruppe und
  - 5% flächendeckend verwendet.



[Livari96, S.95]

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Kosten von CASE

- Dem Nutzen von CASE stehen folgende Kosten gegenüber:
  - Auswahl- und Evaluationskosten,
  - Installationsaufwand,
  - Erlernen der Bedienung,
  - Wartungskosten.

- [Herzwurm, Hierholzer, Kunz 93] Produktivität stieg bis um 600%-Uni Köln: CASE-Tools auf dem Prüfstand, in: Computer Zeitung
- [Ramming, Steinmüller 93] Ramming F.J., Steinmüller B., Frameworks und Entwurfsumgebungen, in: Informatikspektrum, 15, 1992, S. 33-43
- [Microtool 96] Microtool, Eine bedarfsgerechte SEU aus Komponenten – die Architekturlösung von Microtool, Microtool GmbH
- [Oppermann et al. 92] Oppermann R., Murchner B., Reiterer H., Koch M. Software-ergonomische Evaluation- Der Leitfaden EVADIS II, Walter de Gruyter
- [Wassermann 90] Wassermann A.I.. Tool Integration in Software Engineering Environments, in Software Engineering Environments, Springer-Verlag
- [Auer et al. 2009] Sören Auer, Kim Lauenroth, Steffen Lohmann, and Thomas Riechert, Agiles Requirements Engineering für Softwareprojekte mit einer großen Anzahl verteilter Stakeholder, volume XVIII of Leipziger Beiträge zur Informatik, 2009.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

1. **Einführung**
   - 1.1. Was ist CASE?
   - 1.2. CASE-Werkzeugkategorien
   - 1.3. Ziele von CASE
   - 1.4. Allgemeine Anforderungen an Werkzeuge, Plattformen und Umgebungen
2. **Zur Auswahl von CASE-Umgebungen**
3. **Evaluationsverfahren für CASE**
4. **Kosten/Nutzen von CASE**
5. **CASE-Tools in der Praxis**

   Aus der Diplomarbeit von Herrn Gahlert

- Thema: Konzeption und prototypische Umsetzung eines Werkzeuges für das Anforderungsmanagement einer Softwareschnittstelle.
  - Konfigurations- und Änderungsmanagement
  - CASE- Unterstützung
    - DOORS
    - Synergy-CM
    - Tau-Modelling (Rational Rose)

- BMW AG, München
- Abgabe: Juni 2004

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Entwicklung im Automobilbereich zunehmend im Bereich Elektronik und Software

- Vielzahl von Steuergeräten
  - → Möglichkeit neuer Funktionen (Vernetzung)
  - → Erhöhung Sicherheit
  - → Erhöhung Komfort

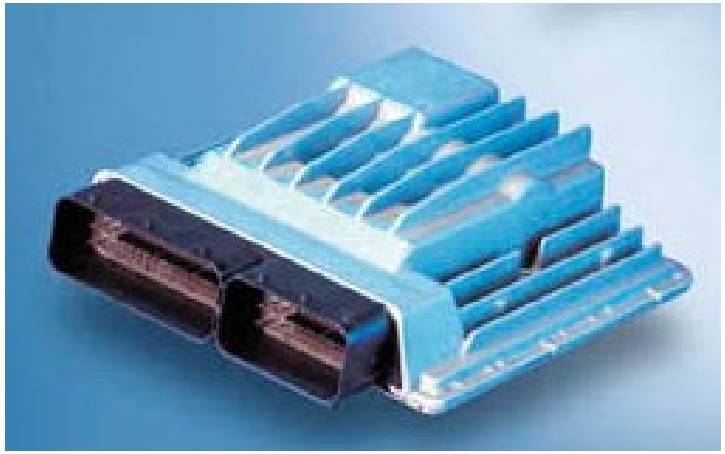
- Aber: 55% der Ausfälle[1] eines Autos durch Software verursacht.
  - → Steigerung der Qualität nötig! Wie?

- Eingebettete Systeme:

  „*An embedded system is a combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a specific function*“ *[2]*

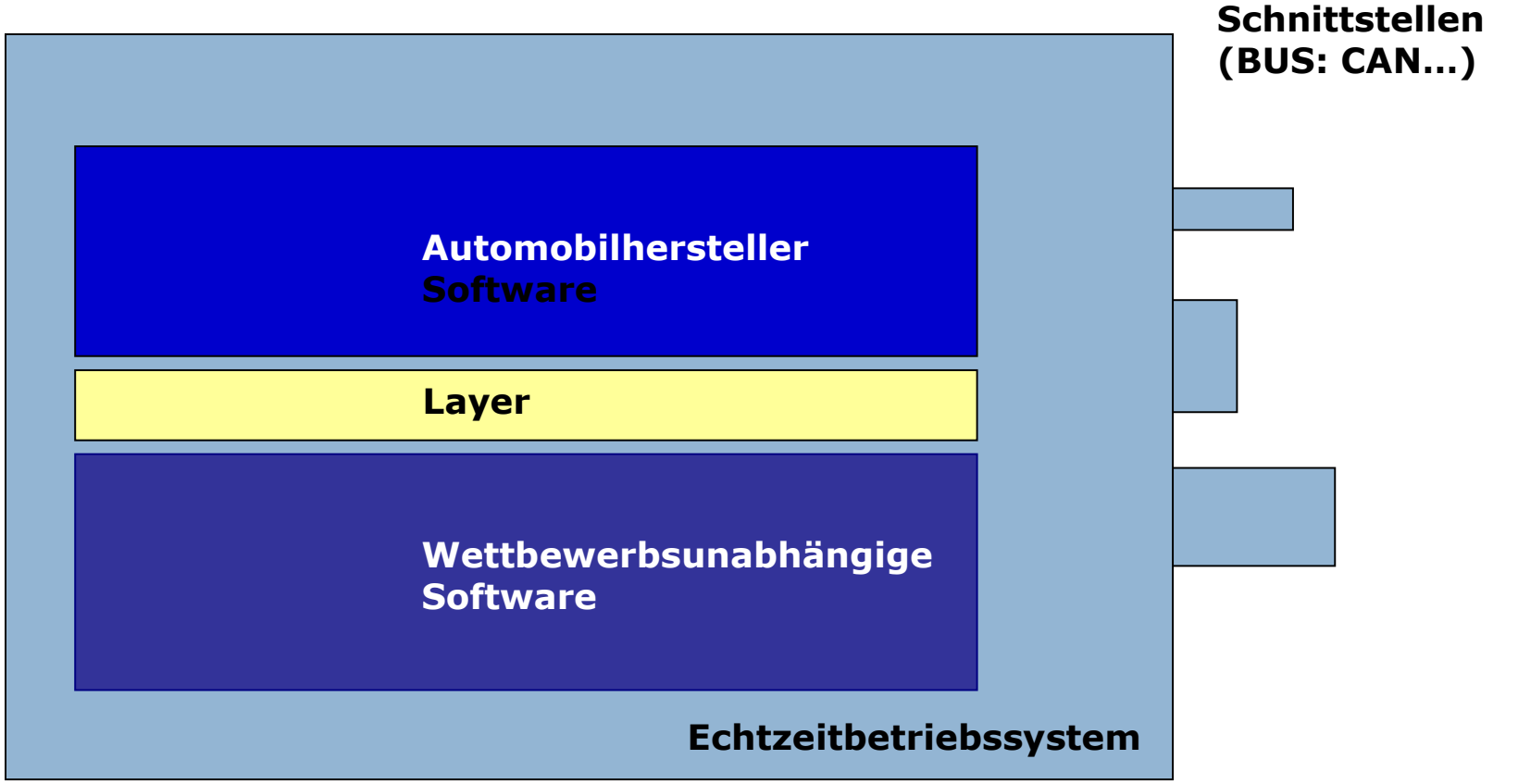
Rechnersystem zur Funktionsausführung ohne direkte Benutzerbedienung

- Steuerung physikalischer Prozesse



- Spezielle Anforderungen:
  - Harte Echtzeitbedingungen - Echtzeit-BS
  - Fehlertoleranz
  - Betriebssicherheit bei Fehlverhalten
  - Systemtest an Simulationsmodellen
  - Extreme Umgebungsbedingungen

- Eingebettete Software
    - C als Programmiersprache
- Aufbau als Schichtmodell darstellbar



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Warum nicht Objekt Orientiert?
  - Keine Compiler für Eingebettete Systeme

    (keine Standardplattform)
  - (noch) wenig Wiederverwendung wegen fehlender standardisierter Plattformen
  - Ressourcen begrenzt
    - Speicherplatz
    - Laufzeit

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Definition: Anforderungen legen die qualitativen und quantitativen Eigenschaften eines Produktes aus der Sicht des Auftraggebers fest.

- Ziele:
  - Grundlagen für das Produktmodell festlegen

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

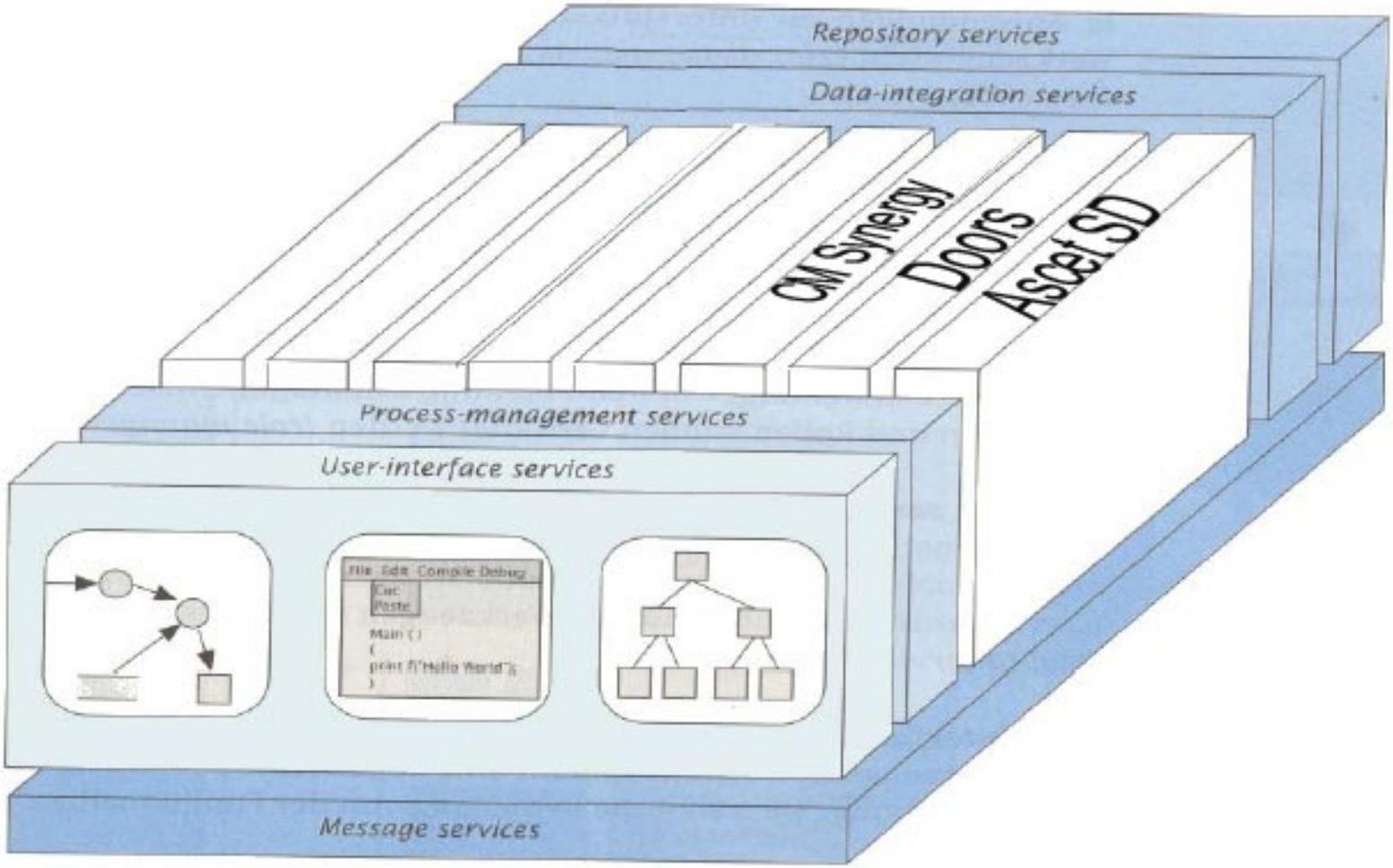
- Definition Konfiguration:
  - Gesamtheit zusammenpassender Software-Elemente

    → KID

- Definition Konfigurationsmanagement
  - Identifikation und Verwaltung von Konfigurationen einer Software zu bestimmten Zeitpunkten
  - Steuerung/Verfolgung von Änderungen

- Ziele:
  - Sicherstellen der Sichtbarkeit, Verfolgbarkeit und Kontrollierbarkeit eines Produktes
  - Überwachen der Konfigurationen

- Text-Editoren
- Testhilfen
- Modul-Bibliotheken
- Editoren für
  - Datenflussdiagramme
  - Entity-Relationship-Diagramme
  - Petri-Netze….


Repository services
Data-integration services
CM Synergy
Doors
Ascet SD
Process-management services
User-interface services
Message services

- Doors von der Firma Telelogic –
- Dynamic Object Oriented Requirements System

- Tool für das Anforderungsmanagement
  - Funktionalität: Verwalten von Anforderungen

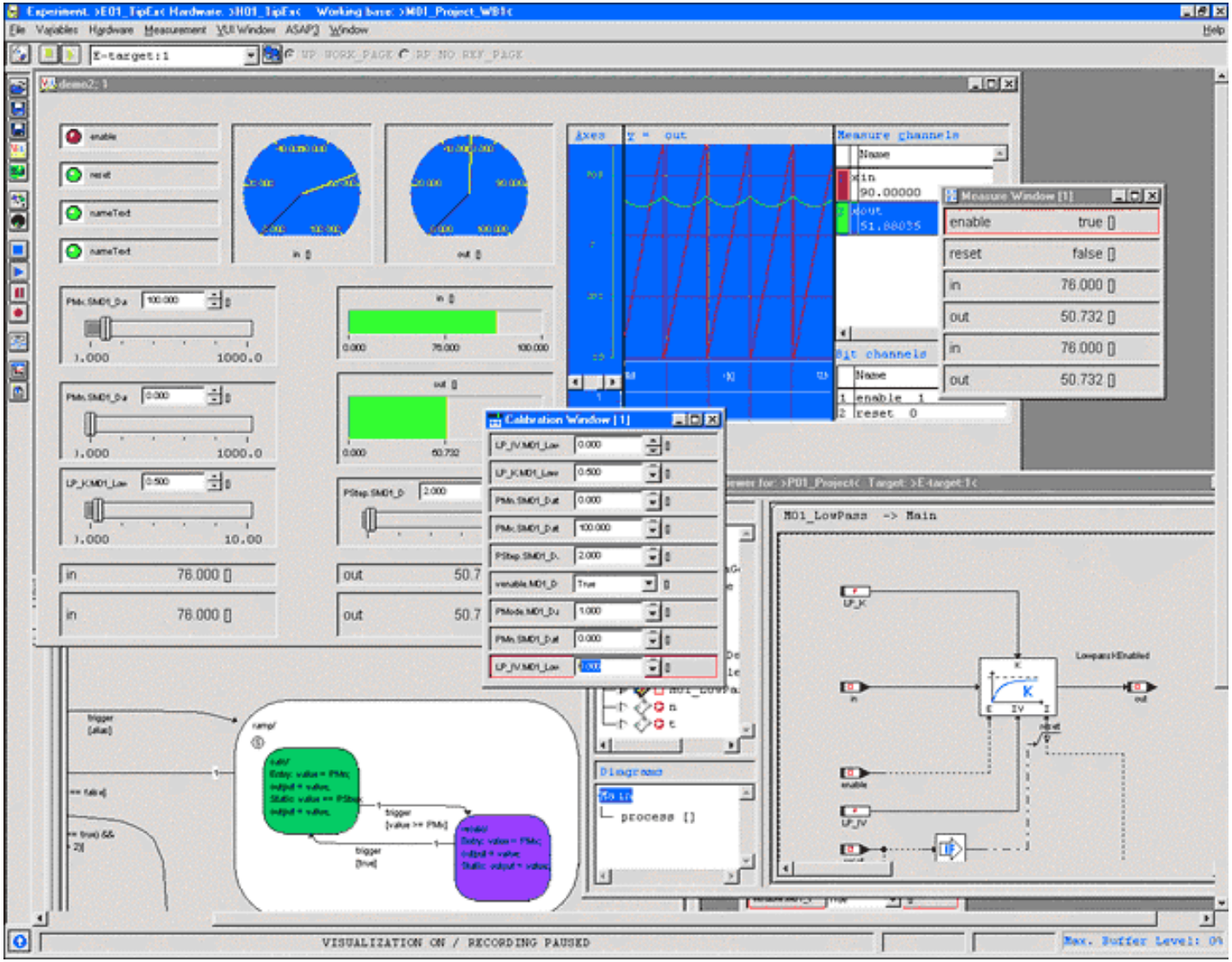
  - Client Server Architektur
  - Baselines
  - Verlinkung
  - Doors/net
  - Ole Einbettung möglich
  - DXL

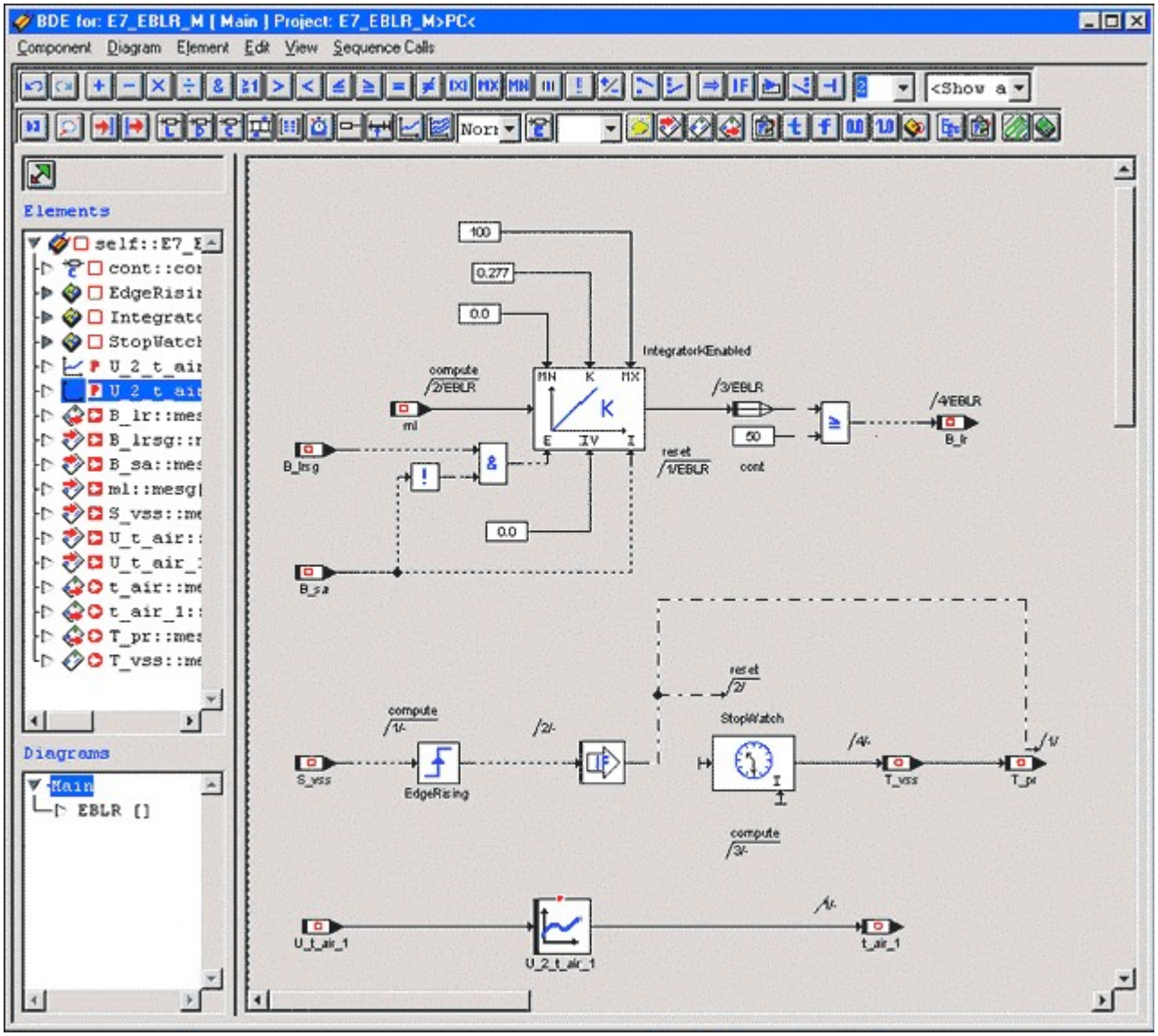
  → Einordnung als cross life cycle tool

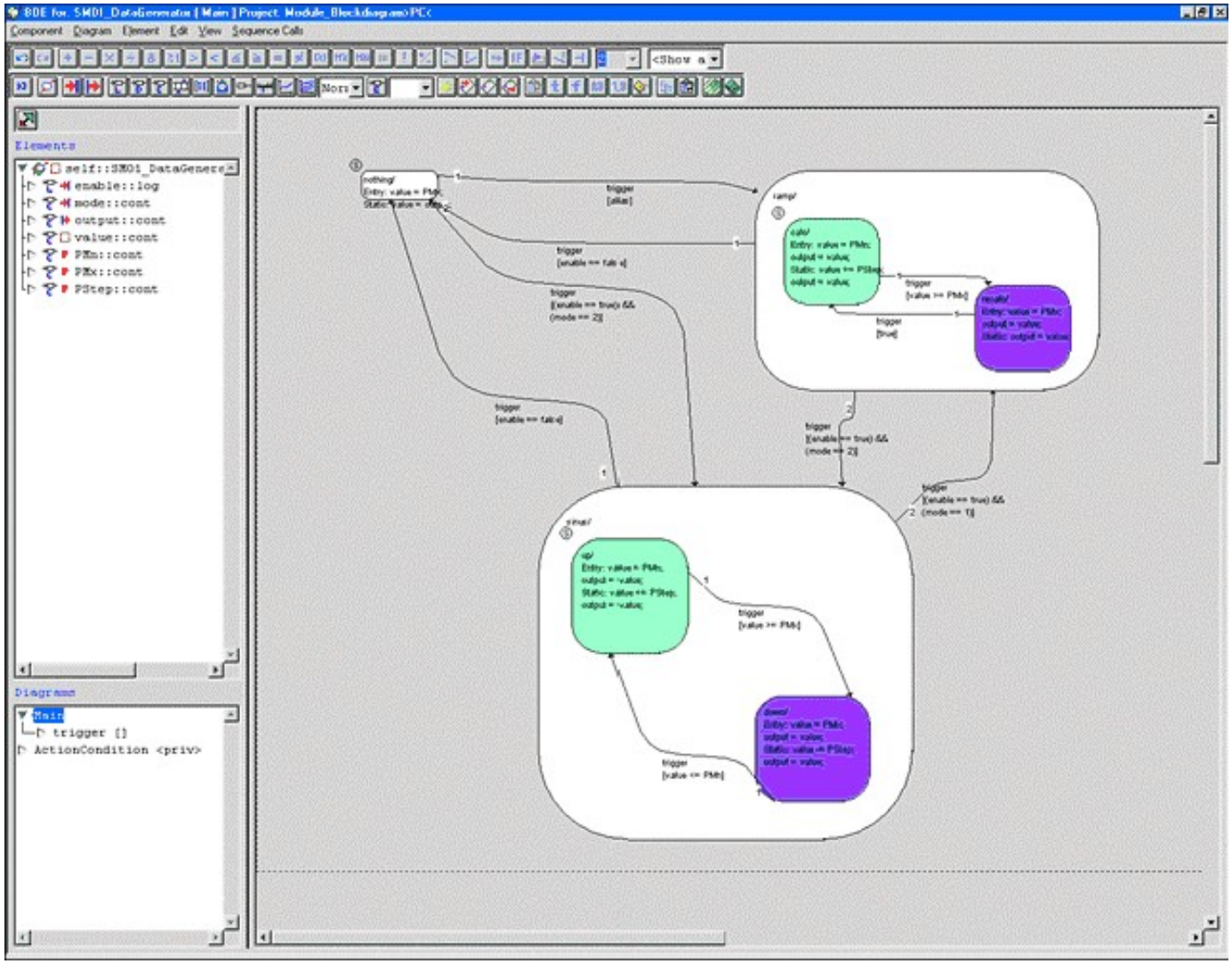
- Grafische Modellierung und Codegenerierung
- Codegenerator für unterschiedliche Chips
- Simulation
- „Objektorientiert“
- C-Code nah

- Nachteil: Modelle teilweise zu komplex

- Teile:
  - Betriebssystem Editor
  - Modellierung
  - Zustandsautomaten
  - Experimentierumgebung


Measure Window [1]
enable true
reset false
in 76.000
out 50.732
in 76.000
out 50.732
Calibration Window [1]
M01_LowPass -> Main
VISUALIZATION ON / RECORDING PAUSED

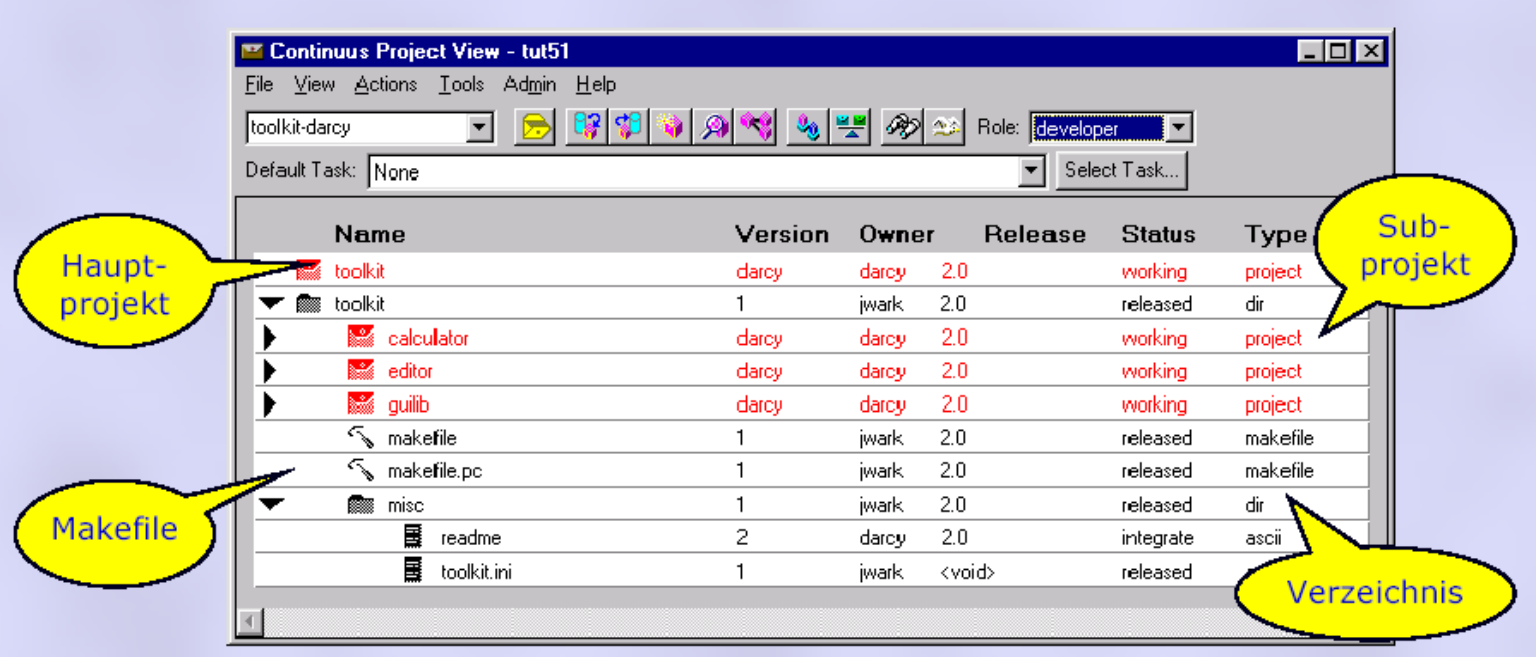

BDE for: E7_EBLR_M [Main] Project: E7_EBLR_M>PC<
Component Diagram Element Edit View Sequence Calls
Elements
Diagrams
Main
EBLR []
100
0.277
0.0
IntegratorKEnabled
compute /2/EBLR
ml
/3/EBLR
50
cont
/4/EBLR
B_lr
B_lrsg
reset /1/EBLR
B_sa
reset /2/
compute /1/
StopWatch
/2/
/4/
S_vss
EdgeRising
T_vss
T_pr
compute /3/
U_t_air_1
U_2_t_air_1
t_air_1


Component Diagram Element Edit View Sequence Calls
Elements
Diagrams
Main
trigger []
ActionCondition <priv>
nothing/
ramp/
run/
up/
down/
trigger
[alive]
[enable == false]
[enable == true) &&
(mode == 2)]
[value >= PMx]
[true]

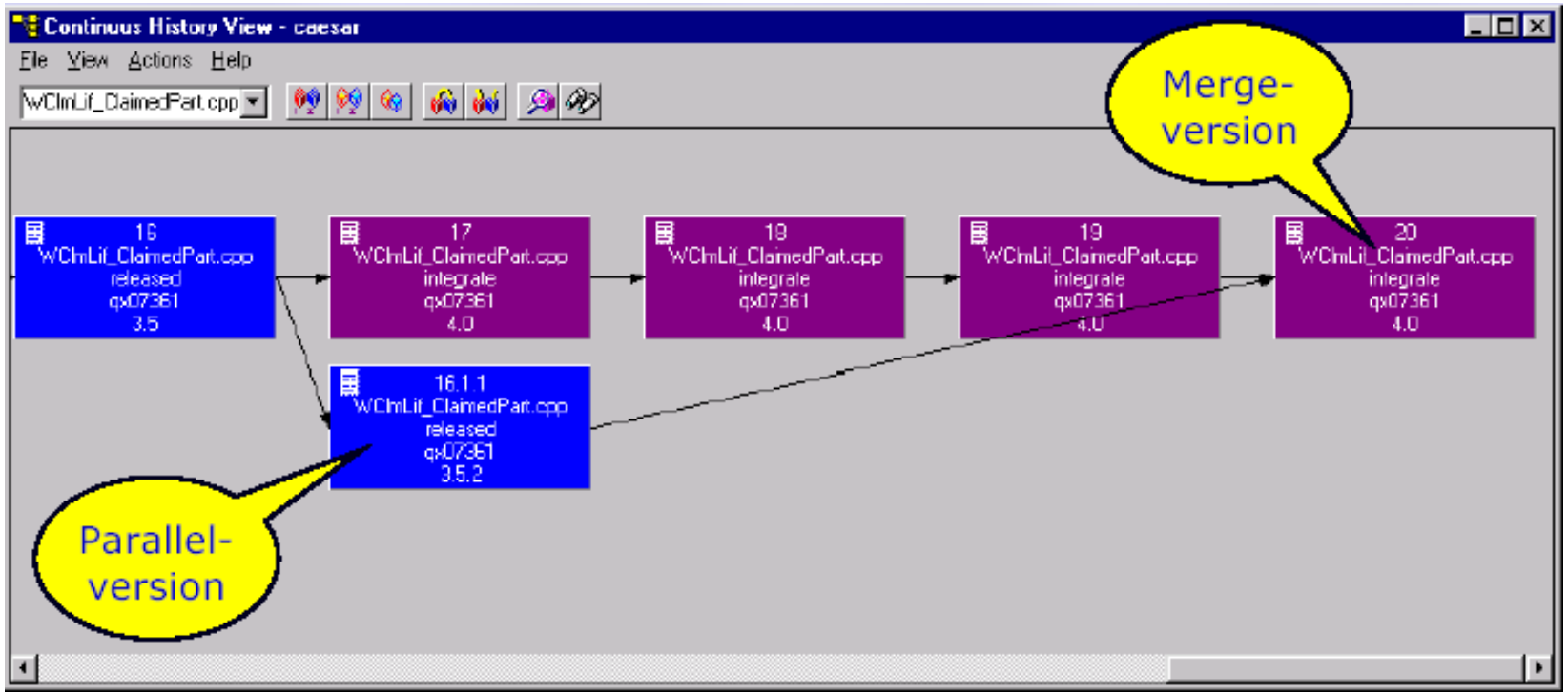
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- CM Synergy– Change Management
  - Taskbasiertes Arbeiten
    - Vorteil: Releaseplanung aufgabenbasiert
  - Dateiversionen in zentralem Repository (Client-Server)
  - Rechteverwaltung
  - Mehrere Projekte möglich
  - (räumlich) Verteiltes Entwickeln möglich

- Projekt/Dateiauswahl



- Versionenhistorie



- Datenhaltung – Integration?
  - Integration selten möglich – Repository als Menge der Daten
  - „Verlinkung“ zwischen den Tools nicht standardmäßig vorhanden

- Frontend – Integration?
  - Integration über den möglichen Aufruf des nächsten CASE-Tools
  - Oft Eigene Tools nötig
  - Jedes Programm ist eigenständig

[1] Mercer03: Mercer Management Consulting, Automobil Elektronik, Problemfelder, Herausforderungen und Lösungsansätze, 2003

[2] Michael Barr, „Programming Embedded Systems in C and C++“, O’Reilly & Associates, Inc., Beijing, Cambridge, Köln, 1999

[3] Helmut Balzert, Lehrbuch der Software-Technik