

# Software Management

## (Schwerpunkt)

### 9. CASE

Prof. Dr. K.-P. Fähnrich

13.06.2007

# Übersicht der Vorlesung

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

# Gliederung

## 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

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

- **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.

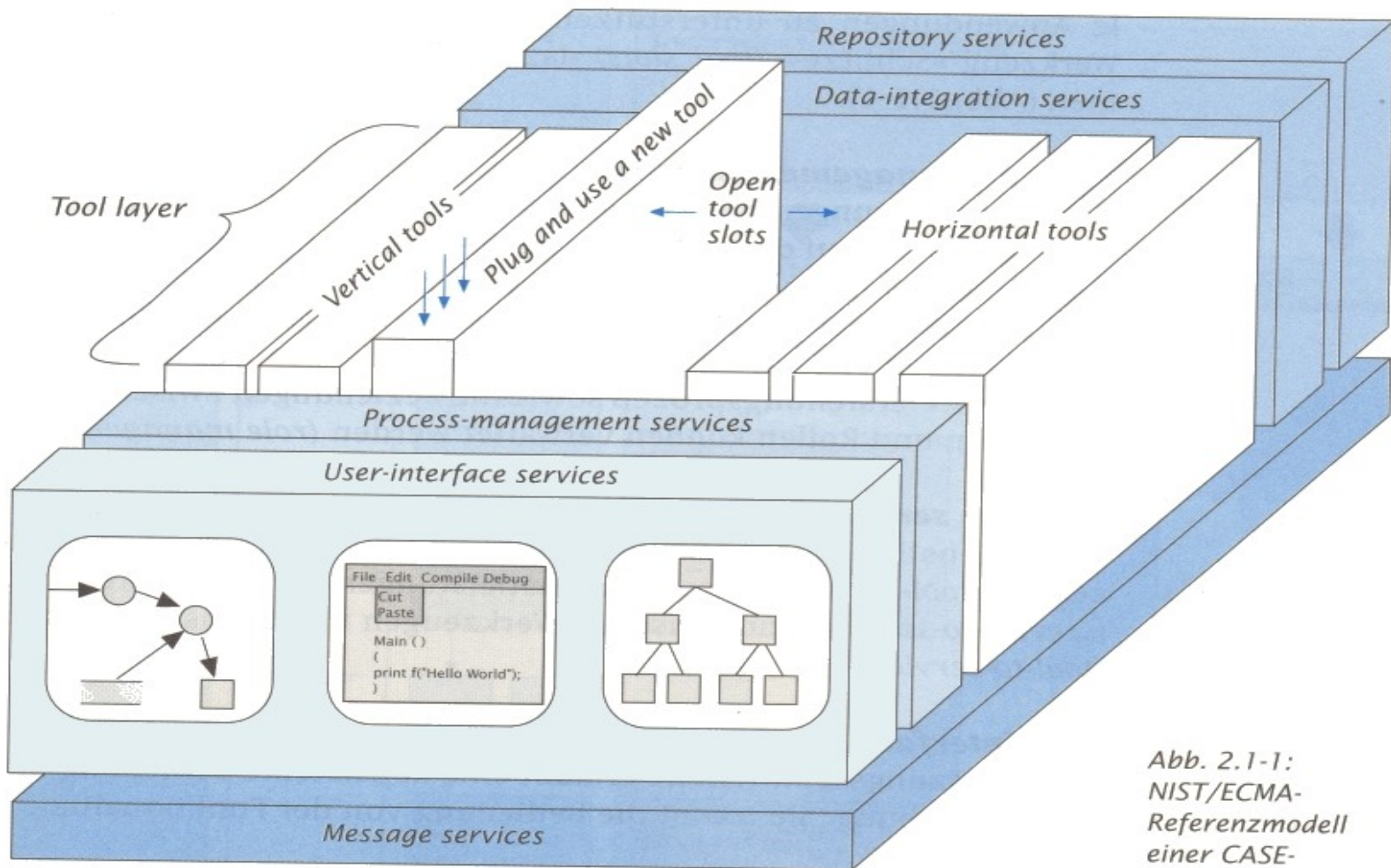
- CASE lässt sich gliedern in:
  - CASE-Werkzeuge:
    - Software-Produkte, die zumindest einzelne bei der SW-Erstellung benötigte Funktionen bzw. Dienstleistungen zur Verfügung stellen.
  - CASE-Plattformen:
    - Stellen Basisdienstleistungen, ein Repository und einen Nachrichtendienst zur Verfügung.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

## Dienstleistungsgruppen

- **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-WerkzeugKategorien

## NIST/ECMA- Referenzmodell



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

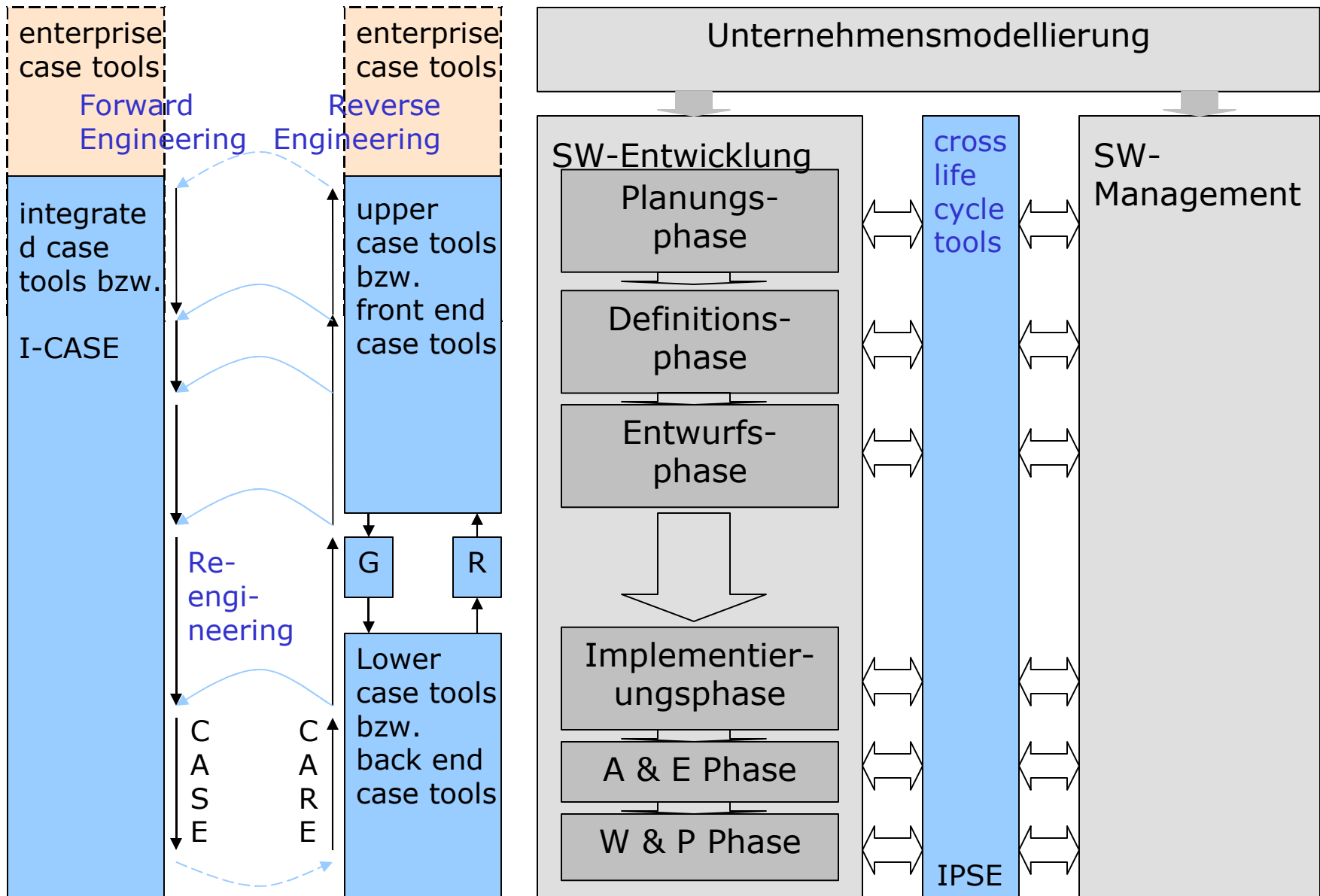
Abb. 2.1-1:  
NIST/ECMA-  
Referenzmodell  
einer CASE-  
Umgebung  
(»Toaster«-Modell)  
/Chen, Norman  
92, S. 19/

- 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





## Globale Ziele

- 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.

## Ziele von CASE

### Technische Ziele

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

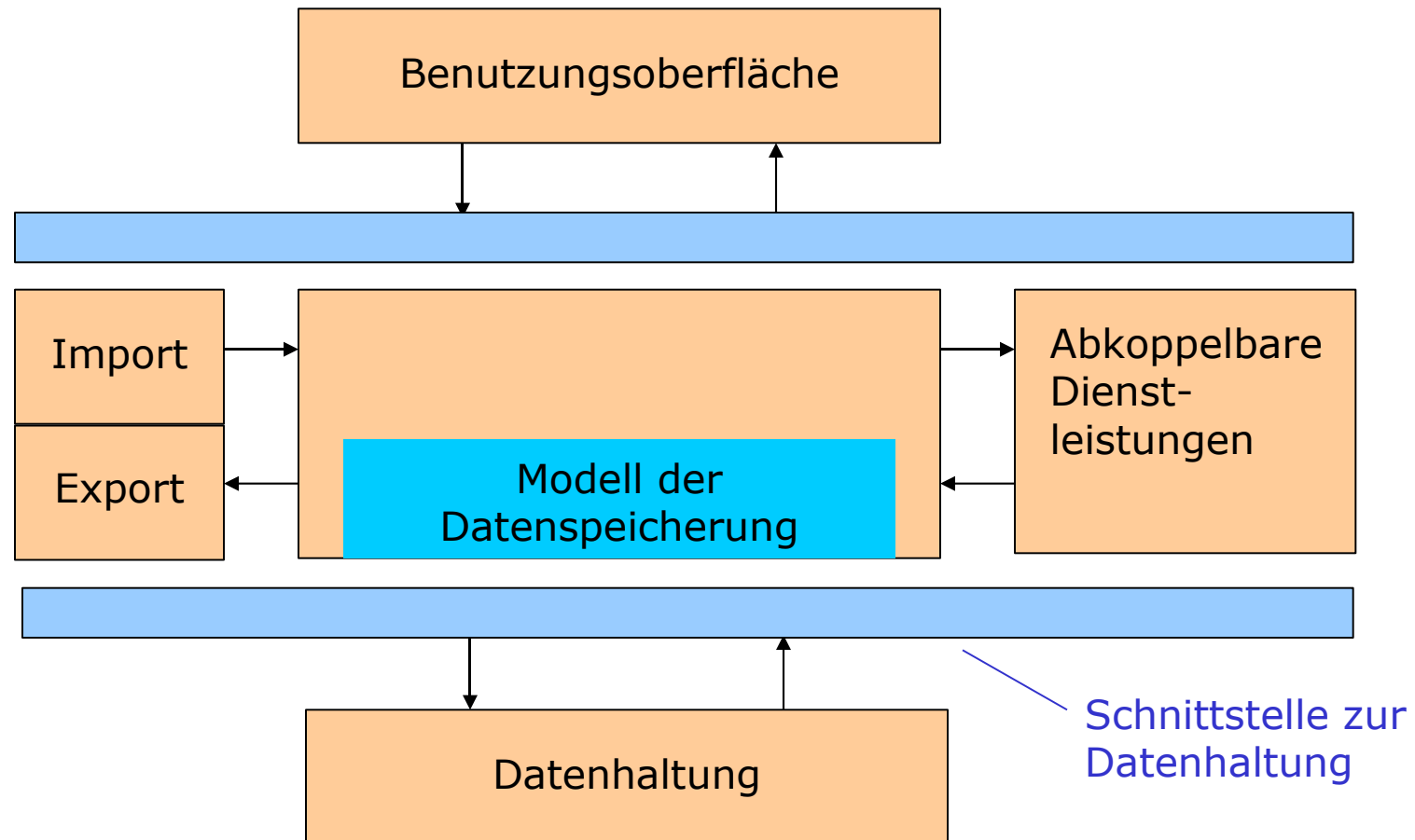
- Effektivität erhöhen.
- Produkte schneller entwickeln.
- Qualität erhöhen.
- Wartungsaufwand reduzieren.
- Personenabhängigkeit verringern.
- Änderungen noch kurz vor Markteinführung ermöglichen.

### Wirtschaftliche Ziele

### Organisatorische 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



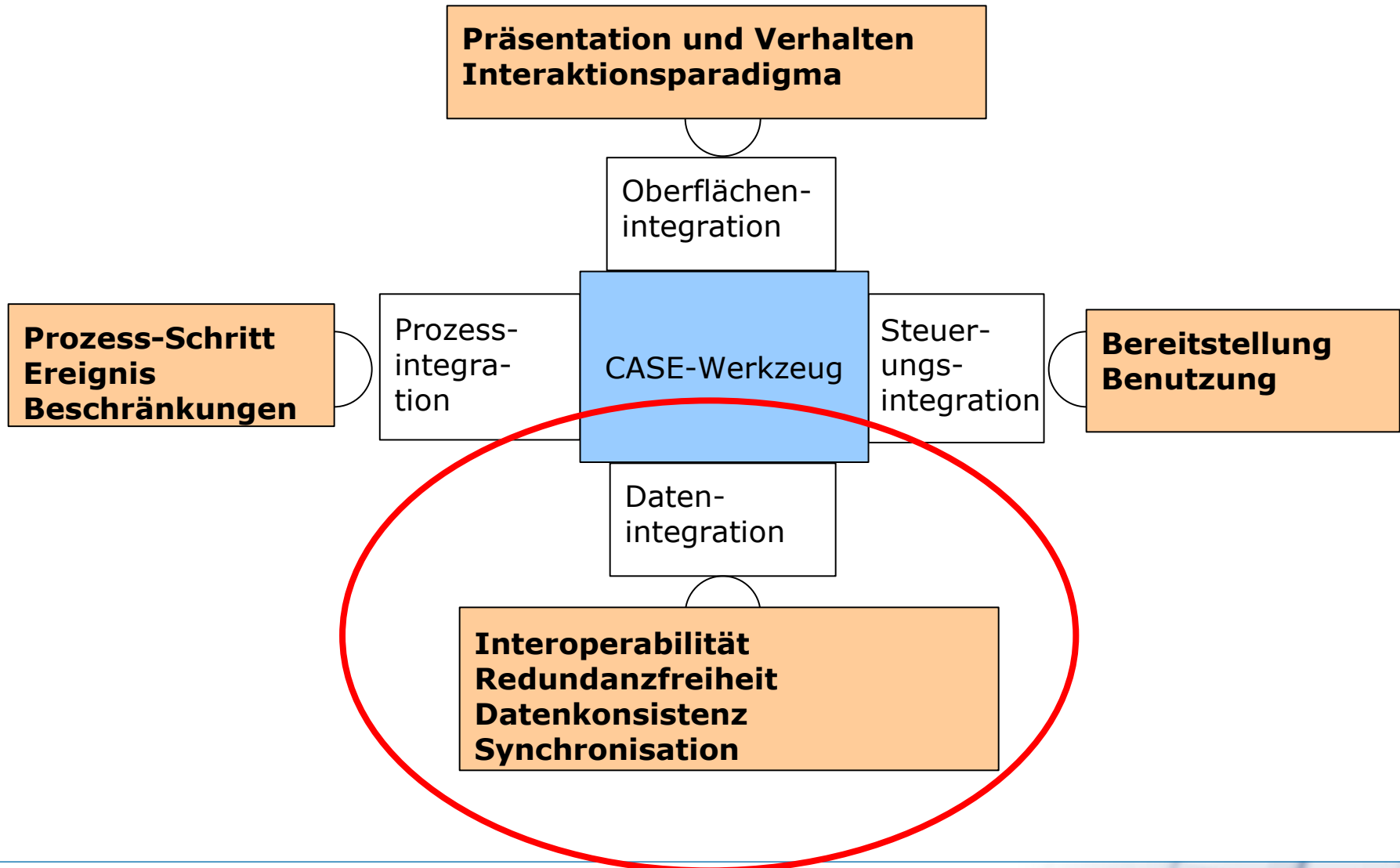
# Allgemeine Anforderungen an CASE-Werkzeuge (1)

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 effizienzsteigernder 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)

1. Übernahme von Hilfs- und Routinearbeiten
2. Intuitive Bedienung
3. Bereitstellung von Export- und Import-Schnittstellen
  - Drei Möglichkeiten:
    1. Definierte Austauschformate,
    2. Veröffentlichung von Klassen des Metamodells,
    3. Offenlegung der Speicherstruktur des CASE-Werkzeugs.
4. Integrierbarkeit in CASE-Plattformen

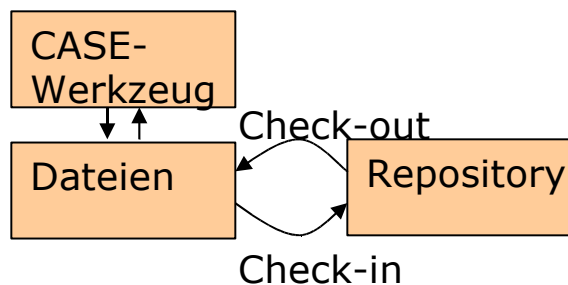
# Integritätsbeziehungen und Beziehungseigenschaften



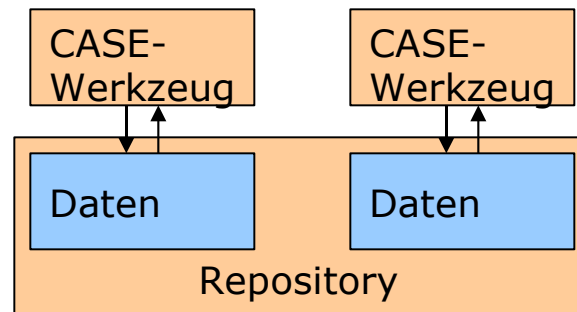
# Basisanforderungen an CASE-Plattformen (1)

## 1. Integrationsfähigkeit von CASE-Werkzeugen

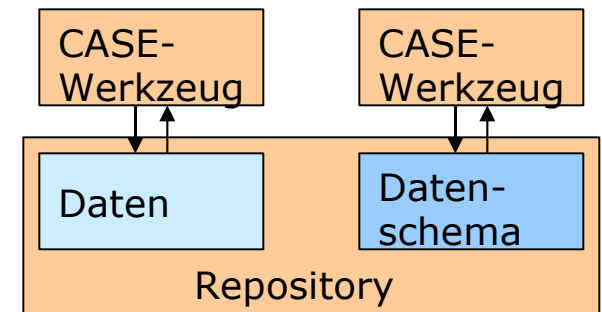
- Grade der Daten-Integration [Rammig, Steinmüller 92]:
  - Black-box-Integration,
  - Grey-box-Integration,
  - White-box-Integration.
- Realisierungsmöglichkeiten eines Repository:
  - Dateisystem,
  - DBS.



**Black-Box-Integration**



**Grey-Box-Integration**



**White-Box-Integration**

# Basisanforderungen an CASE-Plattformen (2)

## 1. Offenheit von CASE-Plattformen

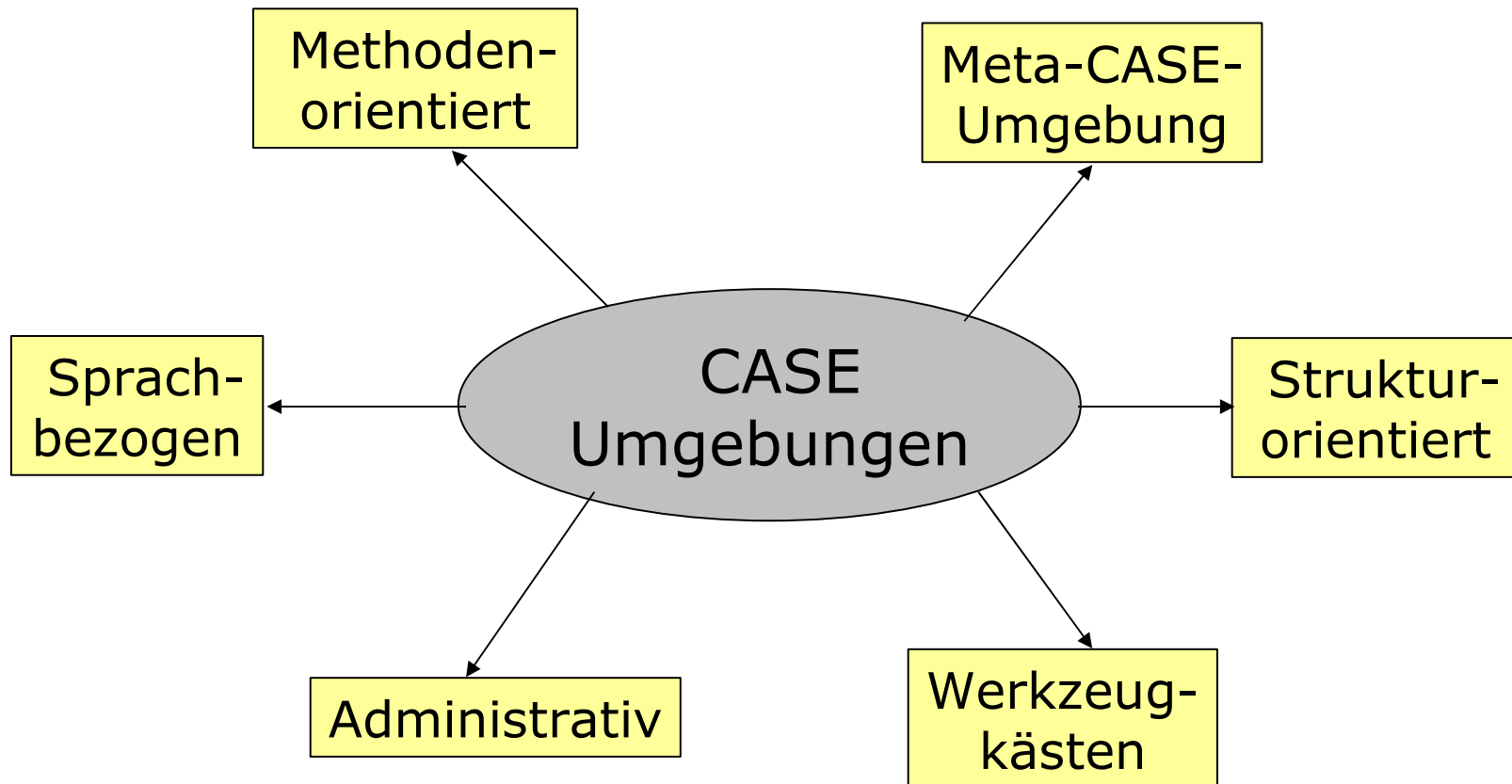
- Eigenschaften einer offenen CASE-Plattform:
  - Definierte Import-/Export-Schnittstellen.
  - Möglichkeit des Einbettens oder Entfernens von CASE-Werkzeuge in die Plattform.
  - Anpassbarkeit.

## 2. 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



# Basisanforderungen an 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 Allgemeinen 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.

## Heutige Situation: Werkzeuge/Plattformen/Umgebungen

- 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.

## 1.7. CASE – Trends

- 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

# Gliederung

## 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

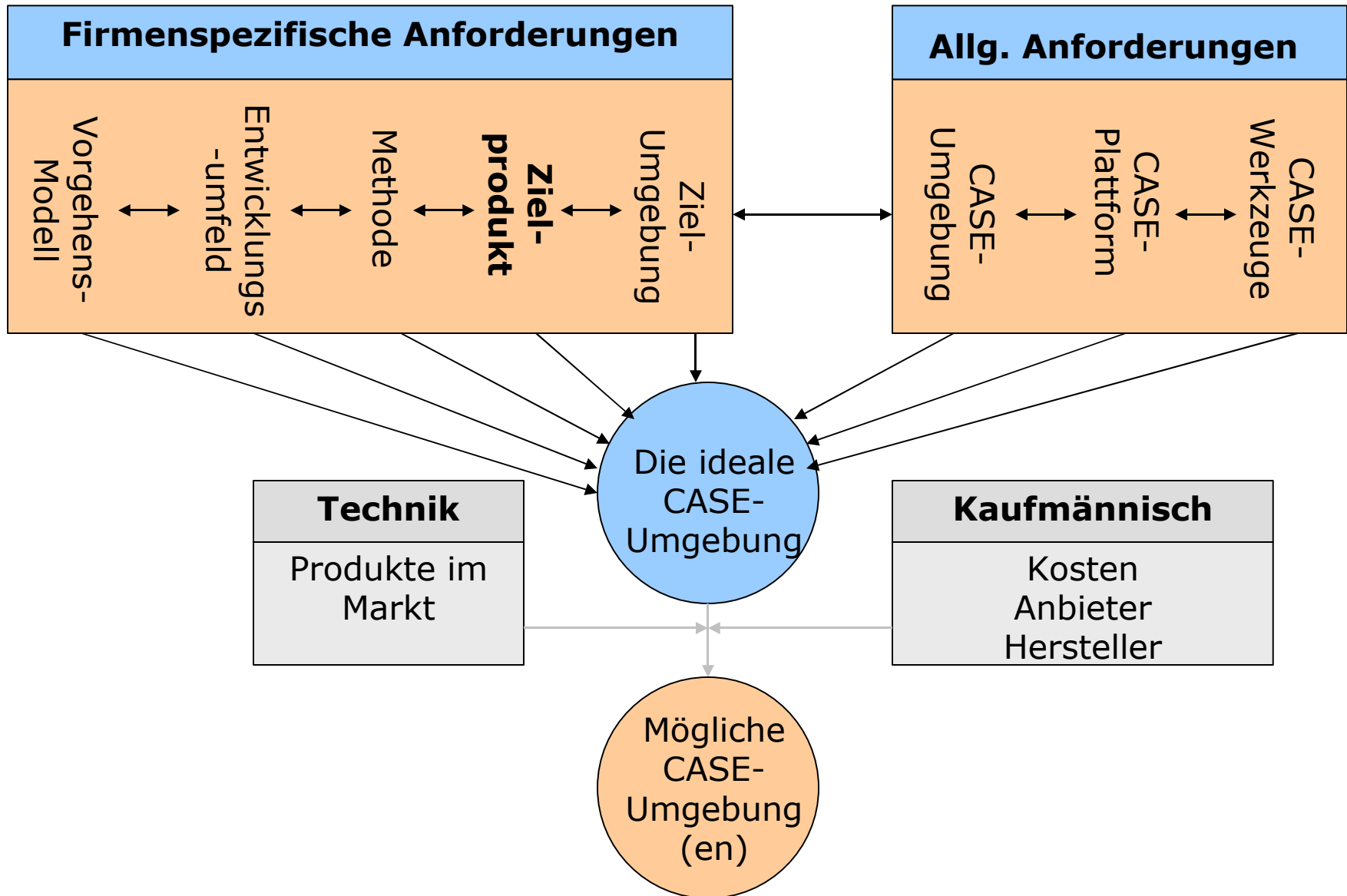
## 3. Evaluationsverfahren für CASE

## 4. Kosten/Nutzen von CASE

## 5. CASE-Tools in der Praxis

Aus der Diplomarbeit von Herrn Gahlert

# Bestimmende Faktoren der CASE-Auswahl



## 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.



# Gliederung

## 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

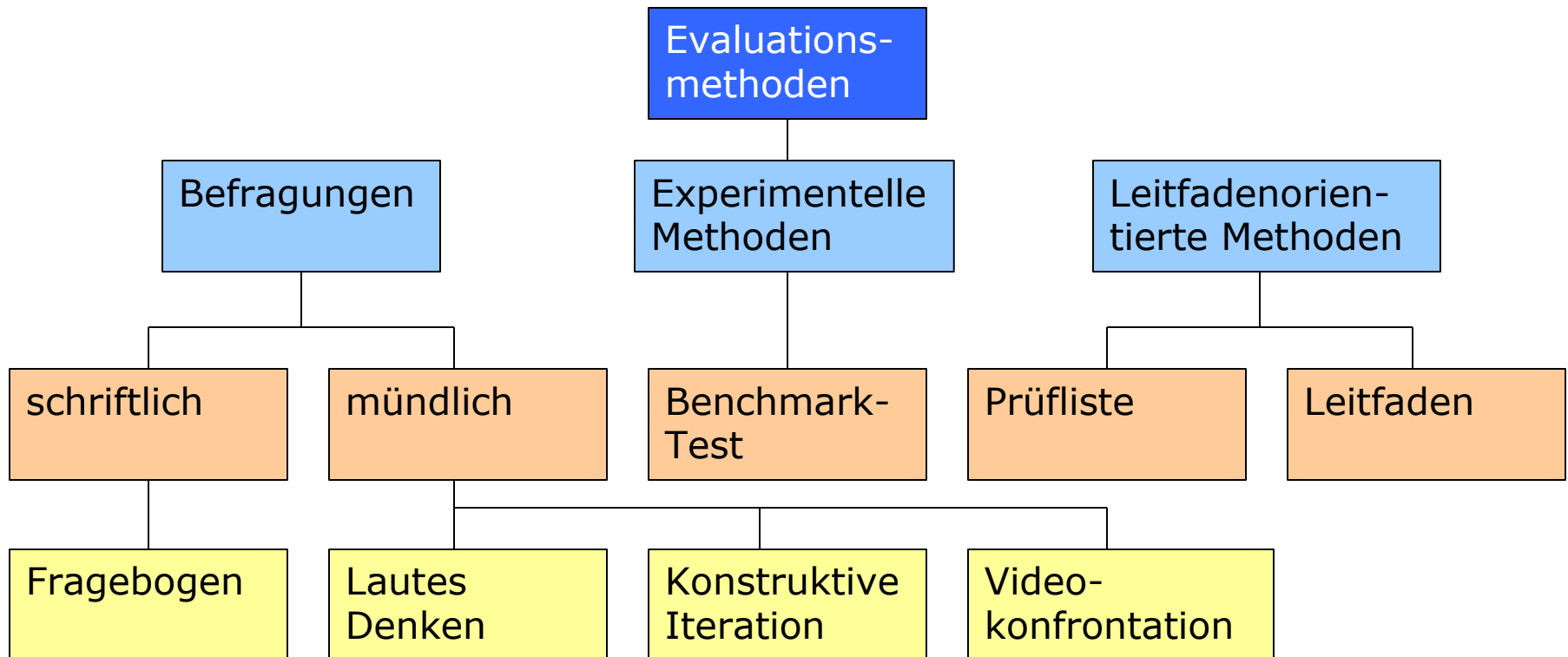
## 5. CASE-Tools in der Praxis

Aus der Diplomarbeit von Herrn Gahlert

## Zu evaluierende Bereiche

- 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



Überblick über Evaluationsmethoden [Oppermann et al. 92]

## Besonderheiten einer CASE-Evaluation

- 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.

# Gliederung

## 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

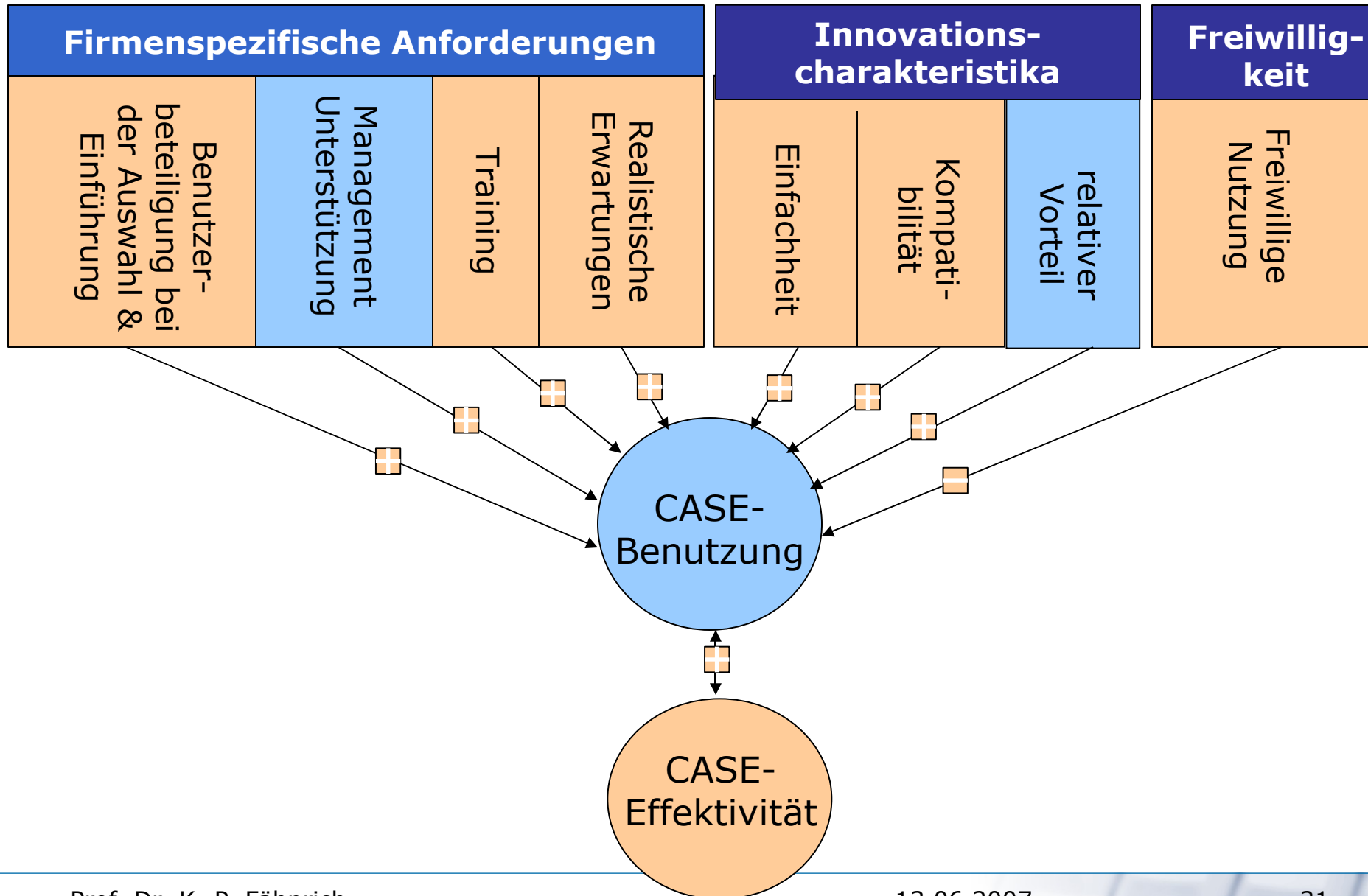
Aus der Diplomarbeit von Herrn Gahlert

## 4. Kosten/Nutzen von CASE

### 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.

# Einflussfaktoren auf die CASE-Benutzung



## 4. Kosten/Nutzen von CASE

### 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

# Gliederung

## 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

# Diplomarbeit Tobias 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

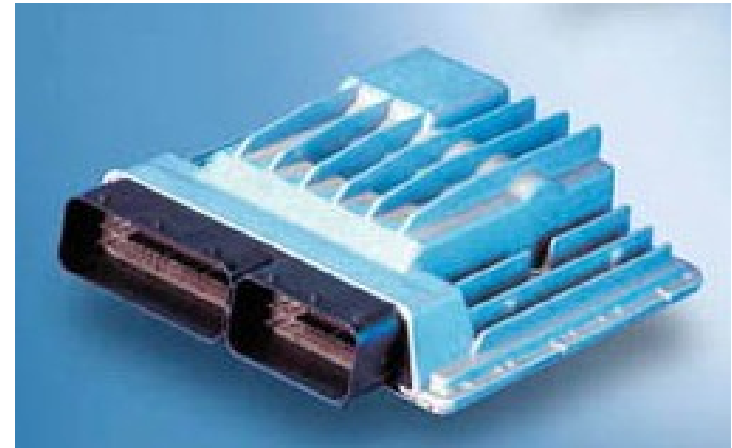
## Motivation

- 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

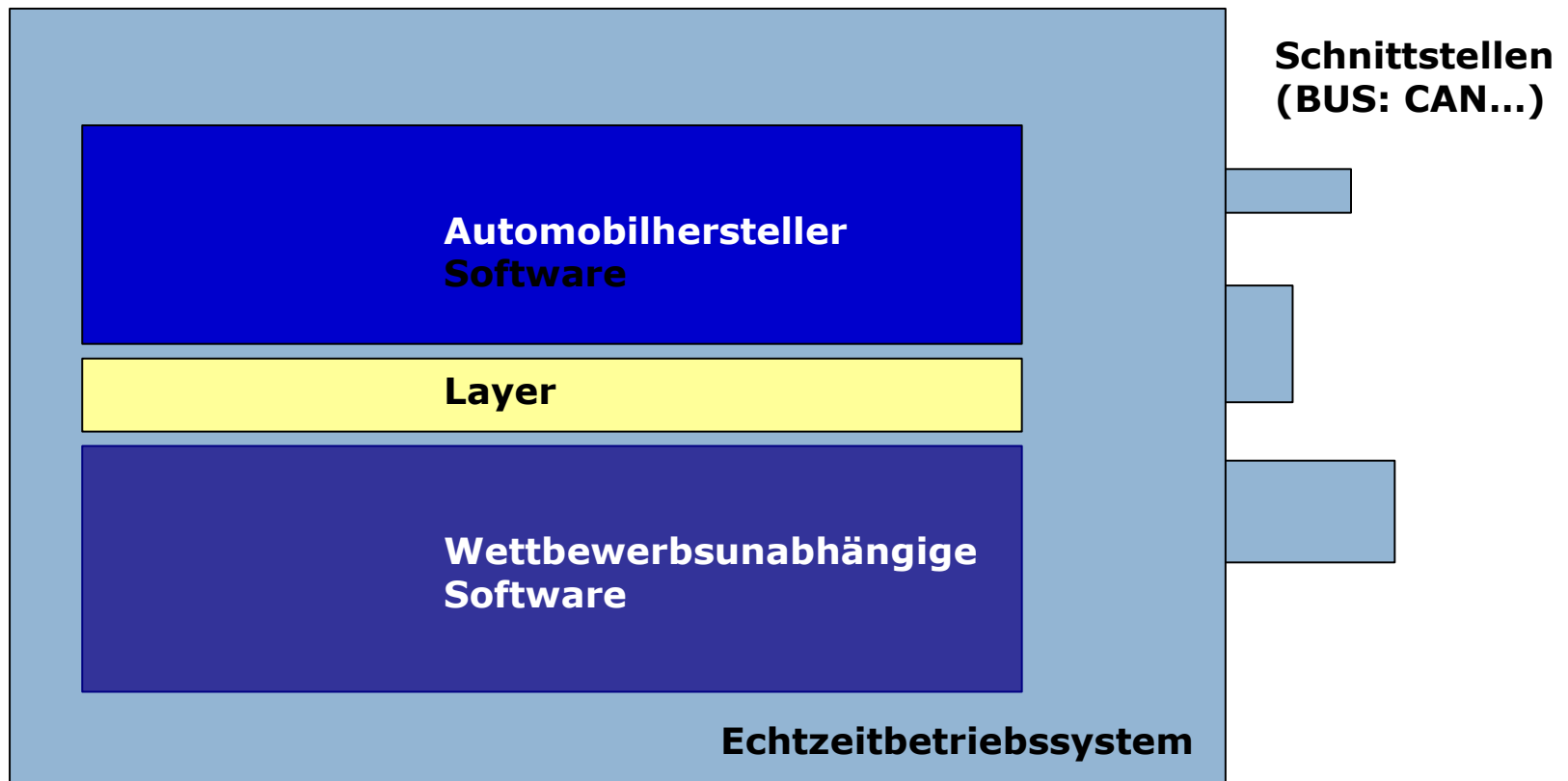


## Eingebettete Systeme (2)

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

# Eingebettete Software

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



- Warum nicht Objekt Orientiert?
  - Keine Compiler für Eingebettete Systeme  
(keine Standardplattform)
  - (noch) wenig Wiederverwendung wegen fehlender  
standardisierter Plattformen
  - Ressourcen begrenzt
    - Speicherplatz
    - Laufzeit



- Definition: Anforderungen legen die qualitativen und quantitativen Eigenschaften eines Produktes aus der Sicht des Auftraggebers fest.
- Ziele:
  - Grundlagen für das Produktmodell festlegen

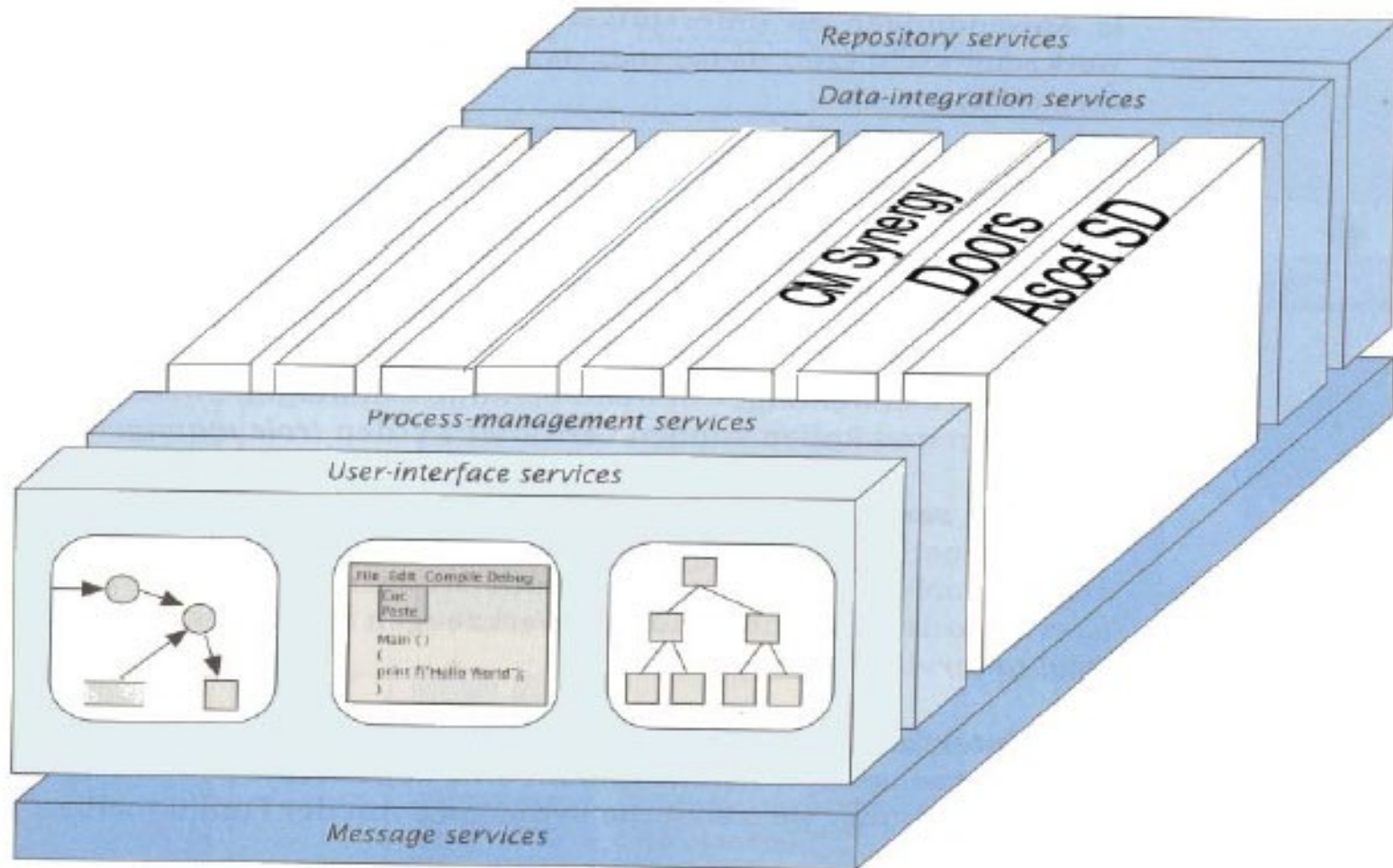
# Konfigurationsmanagement

- 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

## Beispiele für CASE Tools

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

## CASE Plattformen



- 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

## Ascet SD - Automatische Codegenerierung

- Grafische Modellierung und Codegenerierung
- Codegenerator für unterschiedliche Chips
- Simulation
- „Objektorientiert“
- C-Code nah
  
- Nachteil: Modelle teilweise zu komplex
  
- Teile:
  - Betriebssystem Editor
  - Modellierung
  - Zustandsautomaten
  - Experimentierumgebung

## Ascet SD (2)

The screenshot displays the Ascet SD control software interface. The main window is titled "Experiment: >E01\_TipEx Hardware: >H01\_TipEx Working base: >M01\_Project.WB1c". The interface includes several control panels and measurement windows.

**Control Panels:**

- enable:** A red button labeled "enable".
- reset:** A green button labeled "reset".
- nameText:** Two green buttons labeled "nameText".
- PMs\_SMD1\_Da:** A slider control with a value of 100.000.
- PMs\_SMD1\_Db:** A slider control with a value of 0.000.
- UP\_K\_SMD1\_Low:** A slider control with a value of 0.500.
- PStep\_SMD1\_D:** A slider control with a value of 2.000.
- in:** A digital display showing 78.000.
- OUT:** A digital display showing 50.7.

**Measurement Windows:**

- Measure Window [1]:** A table showing measurement data:
 

enable	true
reset	false
in	78.000
out	50.732
in	78.000
out	50.732
- Calibration Window [1]:** A table showing calibration parameters:
 

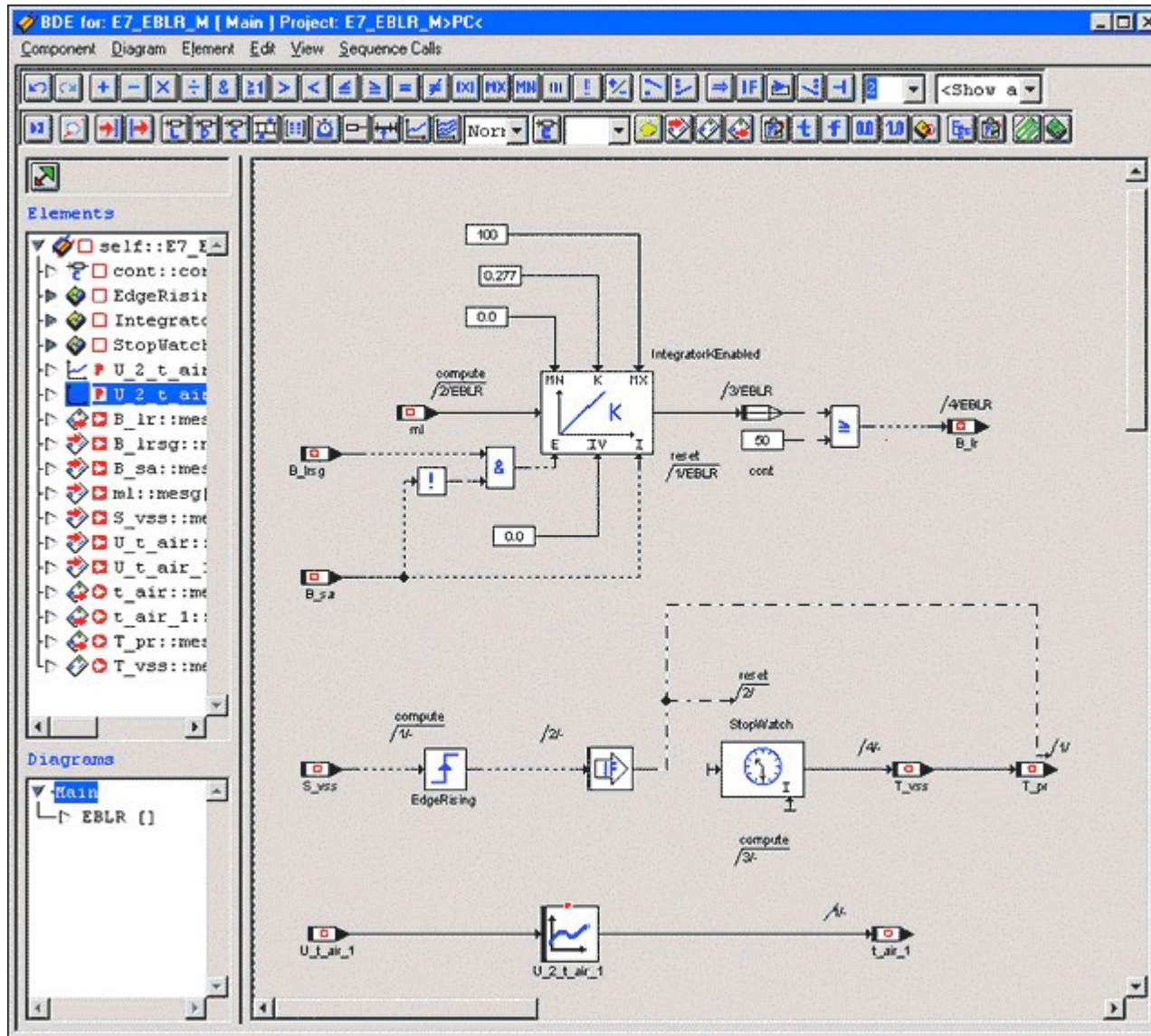
UP_V_SMD1_Low	0.000
UP_K_SMD1_Low	0.500
PMs_SMD1_Out	0.000
PMs_SMD1_Out	100.000
PStep_SMD1_D	2.000
variable_SMD1_D	True
PMs_SMD1_Da	1.000
PMs_SMD1_Db	0.000
UP_V_SMD1_Low	0.000

**Block Diagram:**

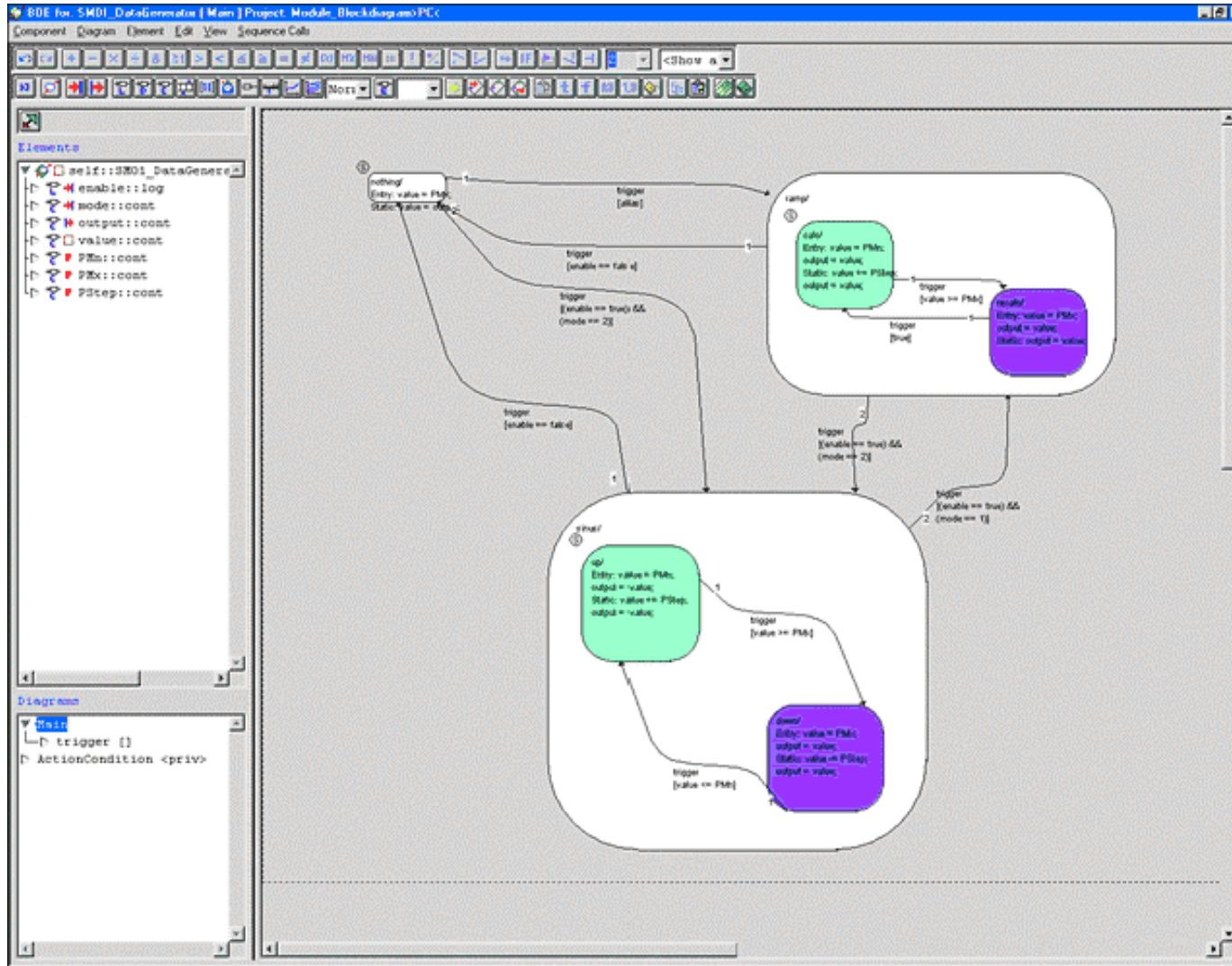
The block diagram shows a control loop. A "Main" block contains a "LowpassEnabled" block and a "K" gain block. The "K" block is connected to a "V" block. The "V" block is connected to an "out" block. The "out" block is connected to a "trigger" block. The "trigger" block is connected to a "name" block. The "name" block is connected to a "trigger" block. The "trigger" block is connected to a "trigger" block. The "trigger" block is connected to a "trigger" block.

At the bottom of the interface, it says "VISUALIZATION ON / RECORDING PAUSED" and "Max. Buffer Level: 09".

## Ascet SD (3)







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

## CM Synergy (2)

- Projekt/Dateiauswahl

The screenshot shows the 'Continuous Project View - tut51' window. The window title is 'Continuous Project View - tut51'. The menu bar includes 'File', 'View', 'Actions', 'Tools', 'Admin', and 'Help'. The toolbar contains various icons for project management. The 'Role' dropdown is set to 'developer'. The 'Default Task' is 'None'. The main area displays a table of projects and files.

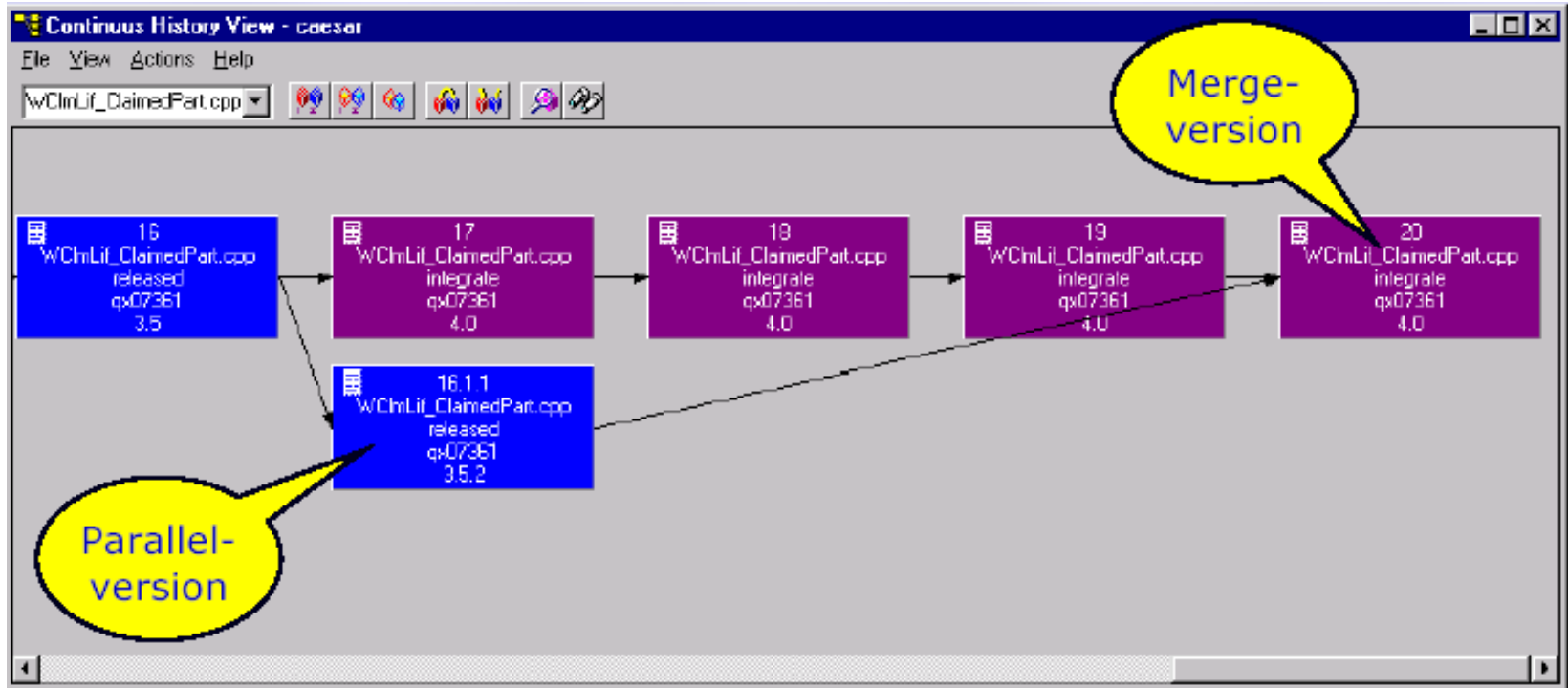
Name	Version	Owner	Release	Status	Type
toolkit	darcy	darcy	2.0	working	project
toolkit	1	jwark	2.0	released	dir
calculator	darcy	darcy	2.0	working	project
editor	darcy	darcy	2.0	working	project
guilib	darcy	darcy	2.0	working	project
makefile	1	jwark	2.0	released	makefile
makefile.pc	1	jwark	2.0	released	makefile
misc	1	jwark	2.0	released	dir
readme	2	darcy	2.0	integrate	ascii
toolkit.ini	1	jwark	<void>	released	

Callouts in the image:

- Haupt-projekt**: points to the 'toolkit' project row.
- Sub-projekt**: points to the 'calculator', 'editor', and 'guilib' project rows.
- Makefile**: points to the 'makefile' and 'makefile.pc' rows.
- Verzeichnis**: points to the 'misc' directory row.

## CM Synergy (3)

- 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

## Quellen

- [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] Balzert