Fachhochschule Köln
Cologne University of Applied Sciences

Softwaretechnik

Fomuso Ekellem

WS 2011/12

Weiteren Verlauf der Vorlesung

- **28.11.2011(4 Std) Definitionsphase (nach Balzert 2001)**
- **05.12.2011(4 Std) Entwurfsphase**
- **12.12.2011(2 Std) Implementationsphase**
- **19.12.2011(2 Std) Test-, Abnahme-, Einführungs-, Wartung- und Pflegephase**

- **16.01.2012(2 Std) Produkte und Recht**
- **23.01.2012(4 Std) Projektvorführung**
- **30.01.2012(4 Std) Dokumentationsvorführung**
- **Klausur ??? (Datum ist noch nicht fastgelegt)**

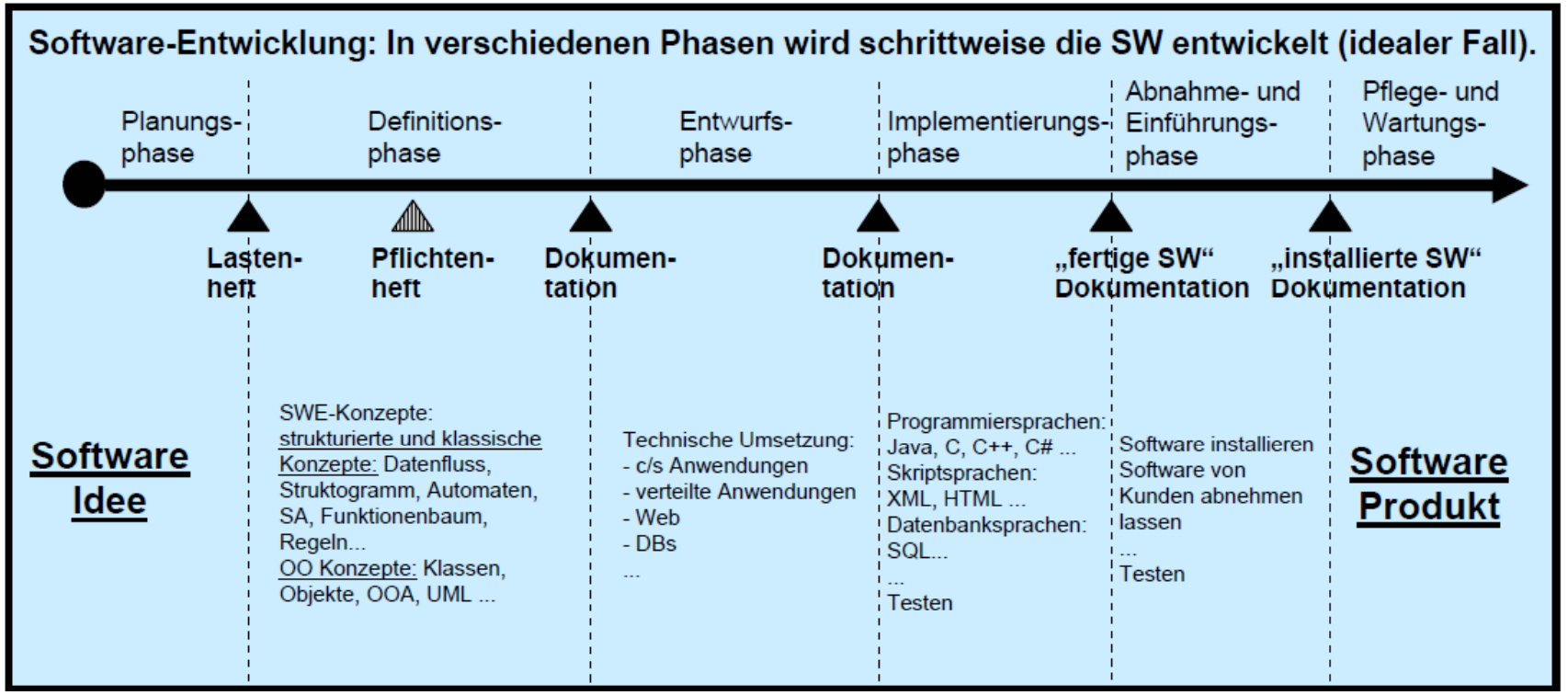
Inhalt

- **Allgemein zu Phasen**
- **Abschluss der Planungsphase**
- **Definitionsphase**
  - Ziele
  - Überblick
  - Aktivitäten
  - Pflichtenheft
  - Produkt-Modell
  - Basiskonzepte zur Systemmodellierung

Inhalt

- **Definitionsphase weiter …**
  - Sichten und Ihre Methoden
  - Kombinationen von Basiskonzepten
  - OOA
  - Strukturierte Analyse/ Real Time Analysis
  - Software-Ergonomie
  - Handbücher und Benutzerunterstützungssysteme

Allgemein zu Phasen in Softwaretechnik mit Dokumentation

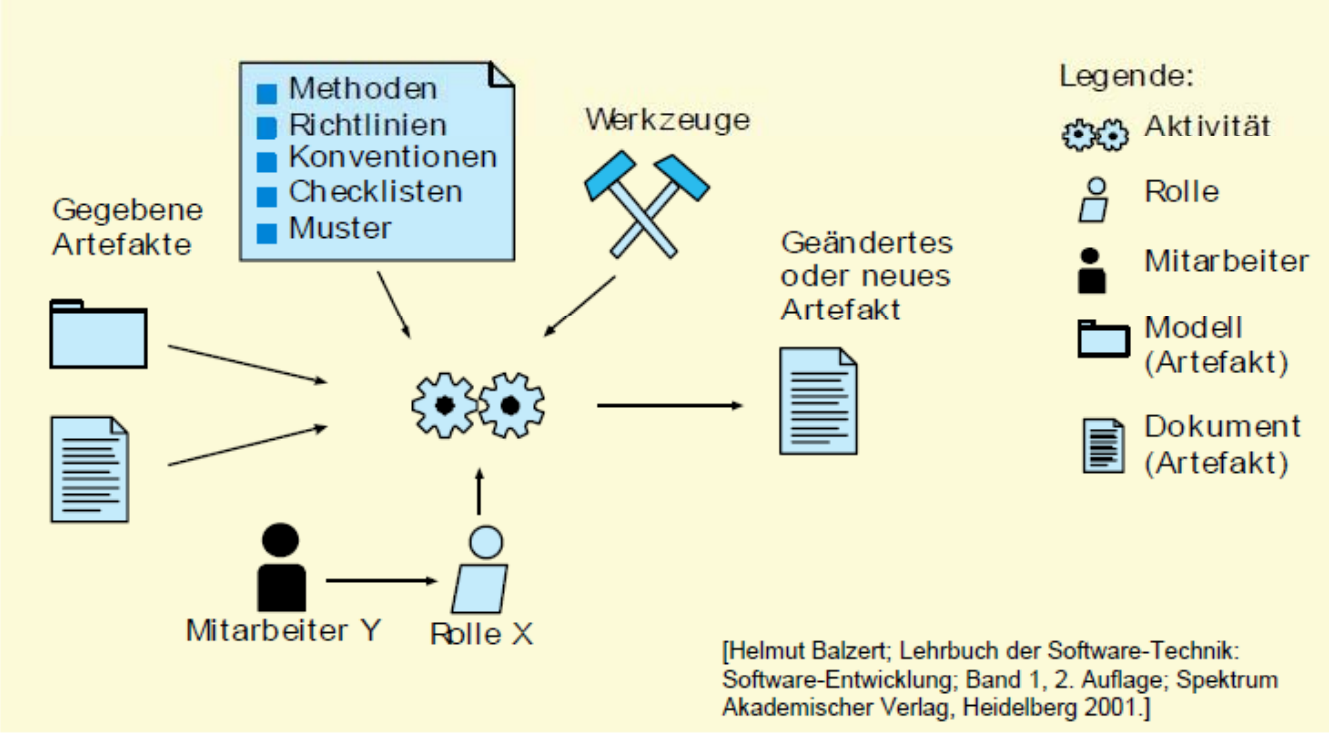


Allgemein zu Phasen

- **Jede Phase zeichnet sich durch folgende Punkte aus:**
  - **Ziele der Phase**
  - **Durchzuführende Aktivitäten**
  - **Aktivitäten/Rollenzuordnung**
  - **Zu erstellende Artefakte**
  - **Zu verwendende Artefakt-Muster**
  - **Zu beachtende Methoden, Richtlinien, Konventionen und Checklisten**
  - **Einzusetzende Werkzeuge und Sprachen**

Allgemein zu Phasen

- **Grafische Darstellung des Ablaufs der einzelnen Aktivitäten in einer Phase im Softwaretechnik sieht wie folgt aus:**



Abschluss der Planungsphase

- Der Abschluss der Planungsphase besteht aus dem Entscheid, mit dem Projekt fortzufahren oder hier abzubrechen. (go oder no-go/stop)

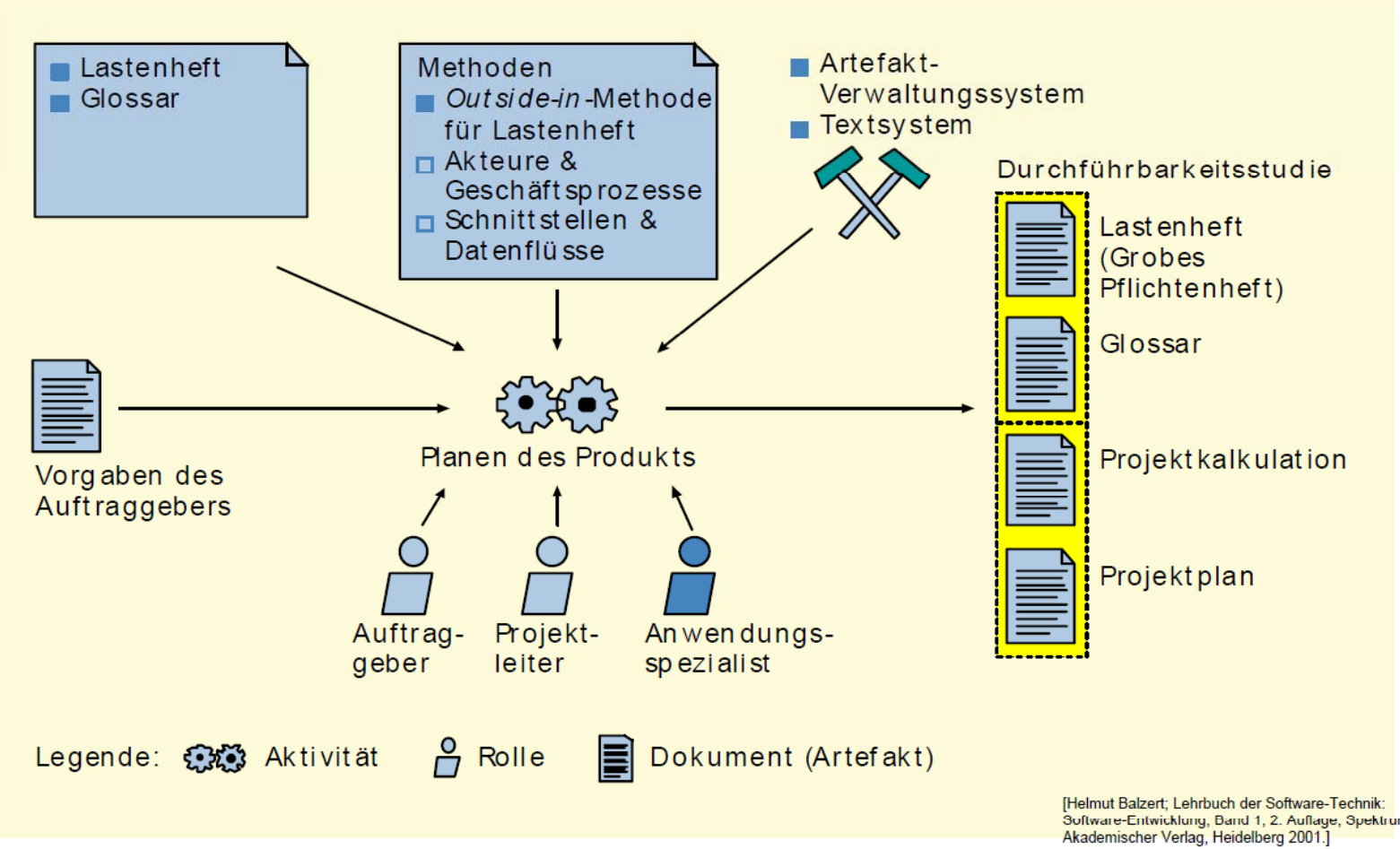
- Die Ergebnisse der Durchführbarkeitsstudie sind:
  - Lastenheft (grobes Pflichtenheft), Glossar
  - Projektkalkulation : Nicht trivial (**Zur Ermittlung der Erstellungskosteneines Software-Produkts werden in der Planungsphase nur einfache Aufwandschätzmethodenangewendet**) z.B.

    Die **Wirtschaftlichkeit eines Produktes** ist wie folgt definiert:

    *Gewinn = (Preis – VariableKosten) * Menge – Entwicklungskosten*

  - Projektplan

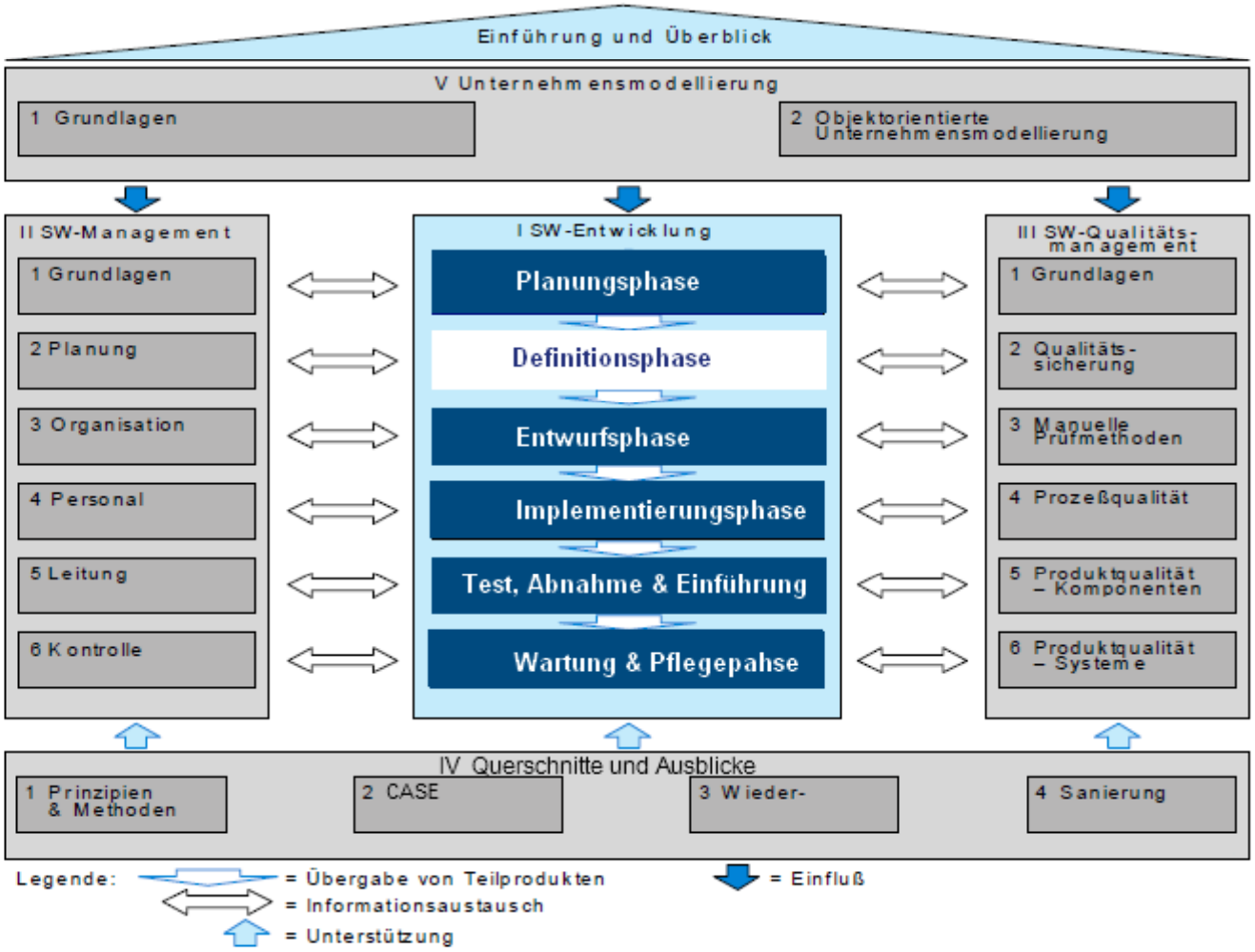
Abschluss der Planungsphase



[Helmut Balzert; Lehrbuch der Software-Technik: Software-Entwicklung, Band 1, 2. Auflage, Spektrum Akademischer Verlag, Heidelberg 2001.]

Definitionsphase

Definitionsphase ist die zweite Phase in der Software-Entwicklung.



Definitionsphase

Ziele

- Definition der Qualitätsziele Vollständigkeit, Konsistenz und Eindeutigkeit sowie von Durchführbarkeit;
- Komplexitätsarten Funktionen, Daten, Algorithmen, zeitabhängiges Verhalten, Systemumgebung und Benutzungsoberfläche bezogen auf eine Anwendung beschreiben können;
- Basiskonzepte und kombinierte Methoden;
- Zusammensetzung der kombinierten Methoden aus Basismethoden;
- Vorgehensweise und Tätigkeiten beim Definitionsprozess;
- Erstellung eines Pflichtenheftes, und die Funktionen.
- Die Aufgaben und Ziele der Software-Ergonomie beschreiben können.
- Elemente der Dialoggestaltung sowie Ein-und Ausgabengestaltung benennen können.
- Die Aufgaben und Ziele der Benutzer-Dokumentation beschreiben können.
- Typen von Benutzer-Handbüchern und Benutzer-Unterstützungssystemen benennen können.

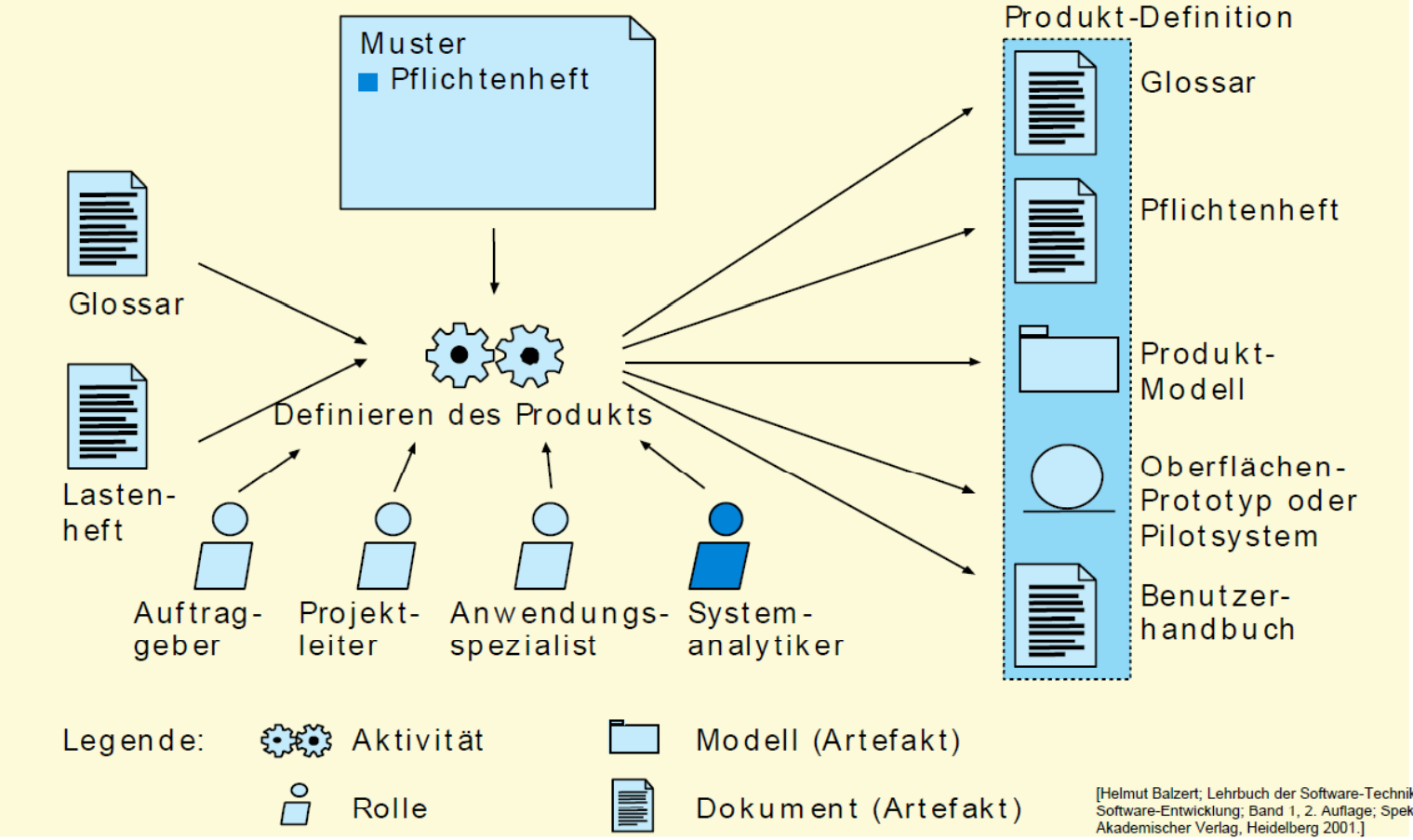
Überblick

- Anforderungen(Requirements) legen die qualitativen und quantitativen Eigenschaften eines Software-Produkts aus Sicht des Auftraggebers fest.
- Anforderungsanalyse(RequirementsEngineering) ist eine systematische Vorgehensweise, um die Anforderungen an das Software-Produkt in einem iterativen Prozess zu ermitteln. RequirementsEngineering deckt auch zum Teil die Systemanalyse.
- In diesem iterativen Prozess stehen verschiedene Basiskonzepte zur Verfügung, um das Produkt zu definieren.
- Dazu werden die Anforderungen ermittelt, analysiert, beschrieben, modelliert, simuliert und verabschiedet.(Planungs- und Definitionsphase)
- Die Anwendung von bestimmten Kombinationen von Basiskonzepten erleichtert die Systemanalyse(Modellierung).
- Im Pflichtenheft werden die verbale Anforderungen (und mit einfachen Grafiken) beschrieben. Das Pflichtenheft ist ein erweitertes Lastenheft.

Aktivitäten in der Definitionsphase

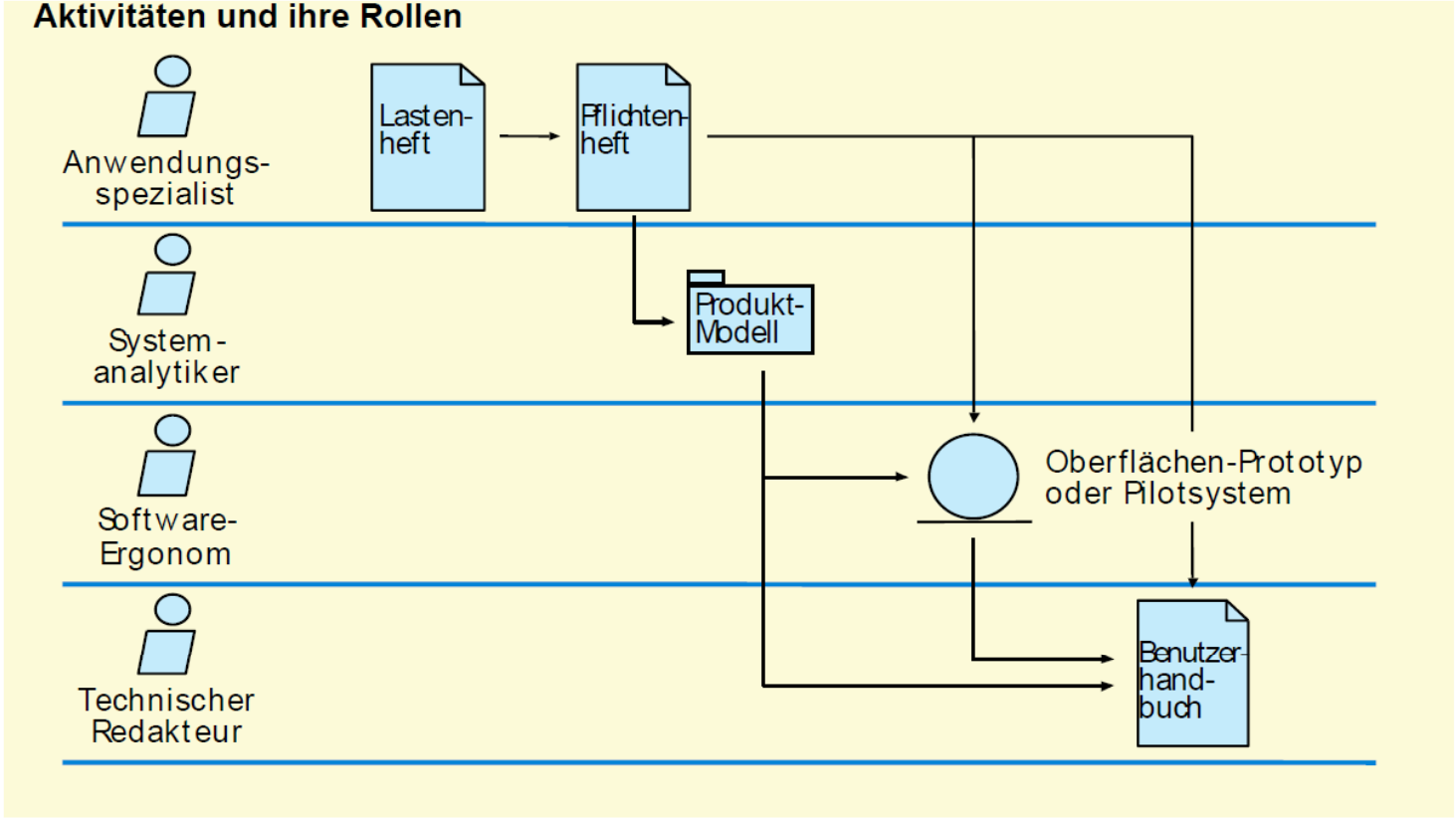
- Soll ein neues Software-Produkt erstellet werden, dann müssen die **Anforderungen** an dieses Produkt in der **Definitionsphase** von den **Anwendungs-Spezialisten** und den **Systemanalytikern** in Zusammenarbeit mit dem **Auftraggeber** und den **Benutzer-Repräsentanten** in Form einer **Produkt-Definition** beschrieben werden.
- Die Systematische Ermittlung, Beschreibung, Modellierung und Analyse der Anforderungen unter Einsatz geeigneter Methoden und Werkzeuge bezeichnet man als **Systemanalyse** (auch requirement engineering).
- Eine **Produkt-Definition** besteht aus mehreren Dokumenten. Ein Dokument davon ist meist das verbal beschriebenes **Pflichtenheft**. Auf der Grundlage des Pflichtenheftes kann ein **Produkt-Modell** erstellt werden.
- Anhand des Produkt-Modells wird ein Konzept für die **Benutzungsoberfläche** erstellt und in Form eines **Oberfläche-Prototyps** oder eines **Pilot-Systems** umgesetzt.
- Unter Berücksichtigung der bisherigen Informationen wird dann ein **Benutzerhandbuch** angefertigt.

Aktivitäten in der Definitionsphase



[Helmut Balzert; Lehrbuch der Software-Technik: Software-Entwicklung; Band 1, 2. Auflage; Spektrum Akademischer Verlag, Heidelberg 2001.]

Aktivitäten in der Definitionsphase



[Helmut Balzert; Lehrbuch der Software-Technik: Software-Entwicklung; Band 1, 2. Auflage; Spektrum Akademischer Verlag, Heidelberg 2001.]

Pflichtenheft

- **Pflichtenheft** ist detaillierte verbale Beschreibung der Anforderungen an ein neues Produkt.
- **Funktion eines Pflichtenhefts**
  - **Aufgabe:** Enthält eine Zusammenfassung aller fachlichen Anforderungen, die das zu entwickelnde Software-Produkt aus der Sicht des Auftraggebers erfüllen muss. Grundlage des Vertrages.
  - **Adressaten:** Auftraggeber, Auftragnehmer und Anwendungsspezialisten, Systemanalytiker, Entwerfer, Qualitätssicherer, Benutzerrepräsentant.
  - **Inhalt:** stellen in der Regel Konkretisierung und Detaillierung der Lastenheft-Inhalte dar.
  - **Form:** Vorgegebenes, standardisiertes, grobes Gliederungsschema mit festgelegten Inhalten.

Pflichtenheft

- **Funktion eines Pflichtenhefts weiter…**
  - **Sprache:** Detaillierte verbale Beschreibung mit Nummerierung einzelner Anforderungen. Nummerierung notwendig um sich in anderen Dokumenten und ich späteren Phasen drauf beziehen zu können.
  - **Didaktik:** Das Gliederungsschema ist so aufgebaut, dass das Pflichtenheft gut lesbar ist, und eine leichte Einarbeitung erlauben.
  - **Zeitpunkt:** Erstes Dokument was nach Abschluss der Planungsphase erstellt wird.
  - **Umfang:** Die notwendigen Anforderungen müssen in ausreichender Detaillierung beschrieben werden. Beschreibung des was, nicht des wie.
- Die Gliederung eines Pflichtenhefts könnte die von Lastenheft nach VDI/VDE-Richtlinien sein. Eine Erweiterung dieser Gliederung reicht einfach. Siehe Seiten 19, 20 und 21 von Vorlesung 5. Es gibt auch noch andere Gliederungen…

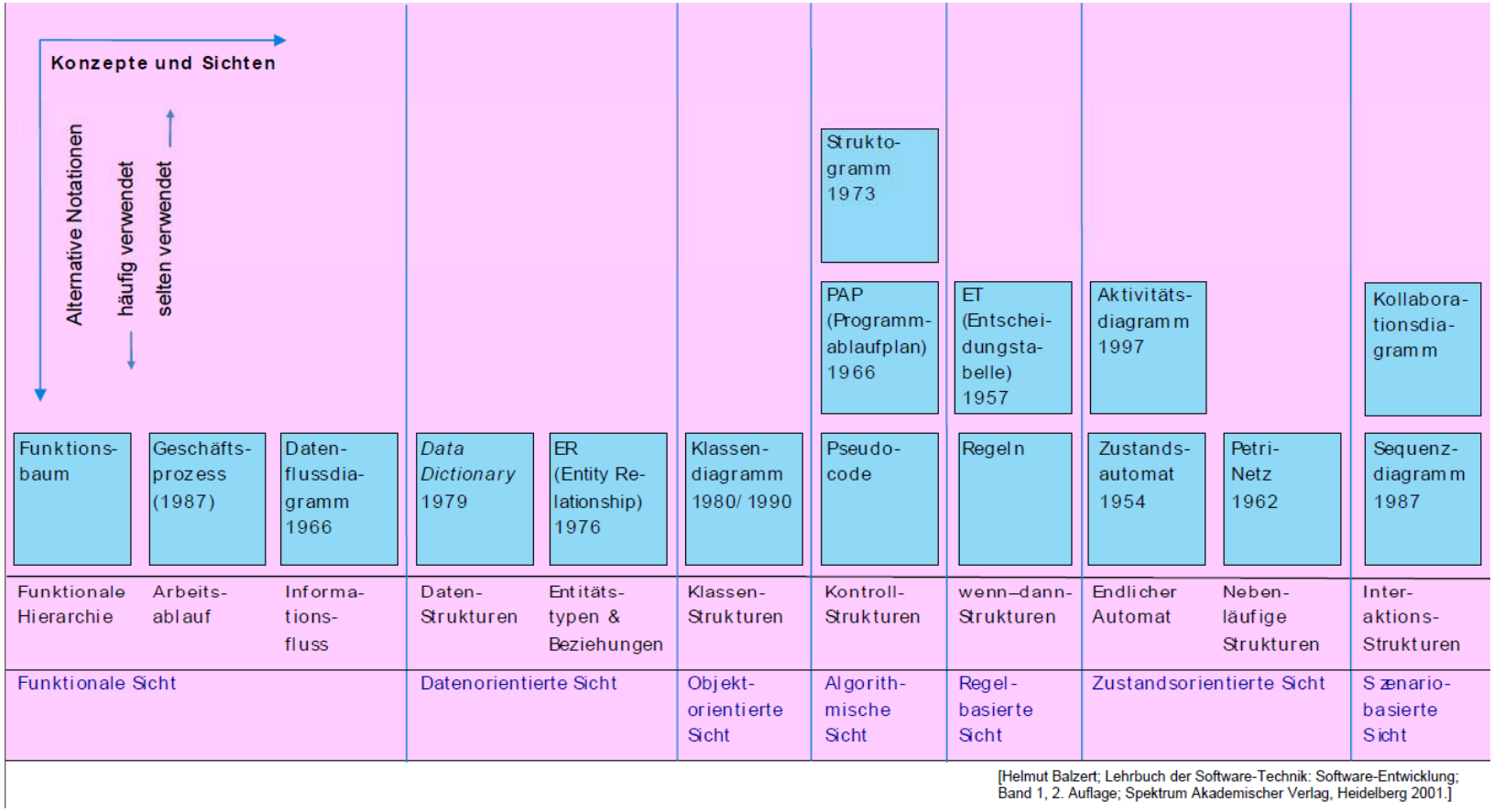
Produkt-Modell

- In der Softwaretechnik ist ein **Modell** eine idealisierte und vereinfachte Darstellung eines Weltausschnitts mit dem ziel dessen Eigenschaften besser studieren zu können.
- **Ist-Analyse**: Neu zu entwickelnde Software löst oft vorhandenes System ab. Ist-Analyse zeigt Schwachstellen des vorhandenen Systems. Diese können dann bei der Definition(Formulierung) der neuen Anforderungen berücksichtigt werden.
- **Anwendungsspezialist** und **Systemanalytiker** kümmern sich aktiv um die Ermittlung der Anforderungen. **Anforderungen** bekommen sie schriftlich. Weitere Infos in der Regel durch Interviews und Befragungen.

Produkt-Modell

- **Produktmodell**
  - **Vollständigkeit**: Beschreibt den Grad in dem das Produkt dem Benutzer alle notwendigen Funktionen und Daten selbst zur Verfügung stellt.
  - **Konsistenz**: Beschreibt den Grad in dem die definierten Anforderungen untereinander widerspruchsfrei sind
  - **Eindeutigkeit**: Beschreibt den Grad in dem die definierten Anforderungen genau eine Interpretation erlauben
- **Produkt-Definition** setzt sich meistens aus verschiedenen Artefakten zusammen. Ist ein juristisches Dokument. Mit ihrer Unterzeichnung sind Anforderungen ans Produkt verabschiedet.

Systemmodellierung



[Helmut Balzert; Lehrbuch der Software-Technik: Software-Entwicklung; Band 1, 2. Auflage; Spektrum Akademischer Verlag, Heidelberg 2001.]

Systemmodellierung

- Unterschiedliche Sichte erfordern unterschiedliche Systemmodelltypen

Verschiedene Sichte der Systemanalyse(Modellierung):

- Funktionale Sicht
- Datenorientierte Sicht
- Objektorientierte Sicht
- Algorithmische Sicht
- Regelbasierte Sicht
- Zustandsorientierte Sicht
- Szenario-basierte Sicht

Systemmodellierung

Typische Gruppierung von Systemmodelle

- **Architekturmodelle** (Use-Case-Diagramm - Geschäftsprozess Diagramme, Funktionales Modell-Funktionsbäume, Datenflussdiagramme)

  Beschreibung der grundlegenden Elemente und der Struktur eines Softwaresystems
- **Objektmodelle** (Klassenmodell, Vererbungsmodell, Interaktionsmode)

  Beschreibung eines Softwaresystems mittels Klassen und Objekten
- **Datenmodelle**

  Beschreibung der Daten und Datenstrukturen eines Softwaresystems
- **Verhaltensmodelle** (Zustandsautomat, Zusicherungsdiagramm)

  Beschreibung des Verhaltens eines Softwaresystems
- **Regelbasierte Modelle**

  Beschreibung der Entscheidungen bei ereignisgesteuerten Softwaresystemen

Systemmodellierung

- Anwendungen lassen sich nach Komplexitätsarten gliedern.

Wir unterschieden 6 Komplexitätsarten:

- Komplexität der Funktionen
- Komplexität der Daten
- Komplexität der Algorithmen
- Komplexität des zeitabhängigen Verhalten
- Komplexität der Systemumgebung
- Komplexität der Benutzungsoberfläche

Basiskonzepte zur Systemmodellierung

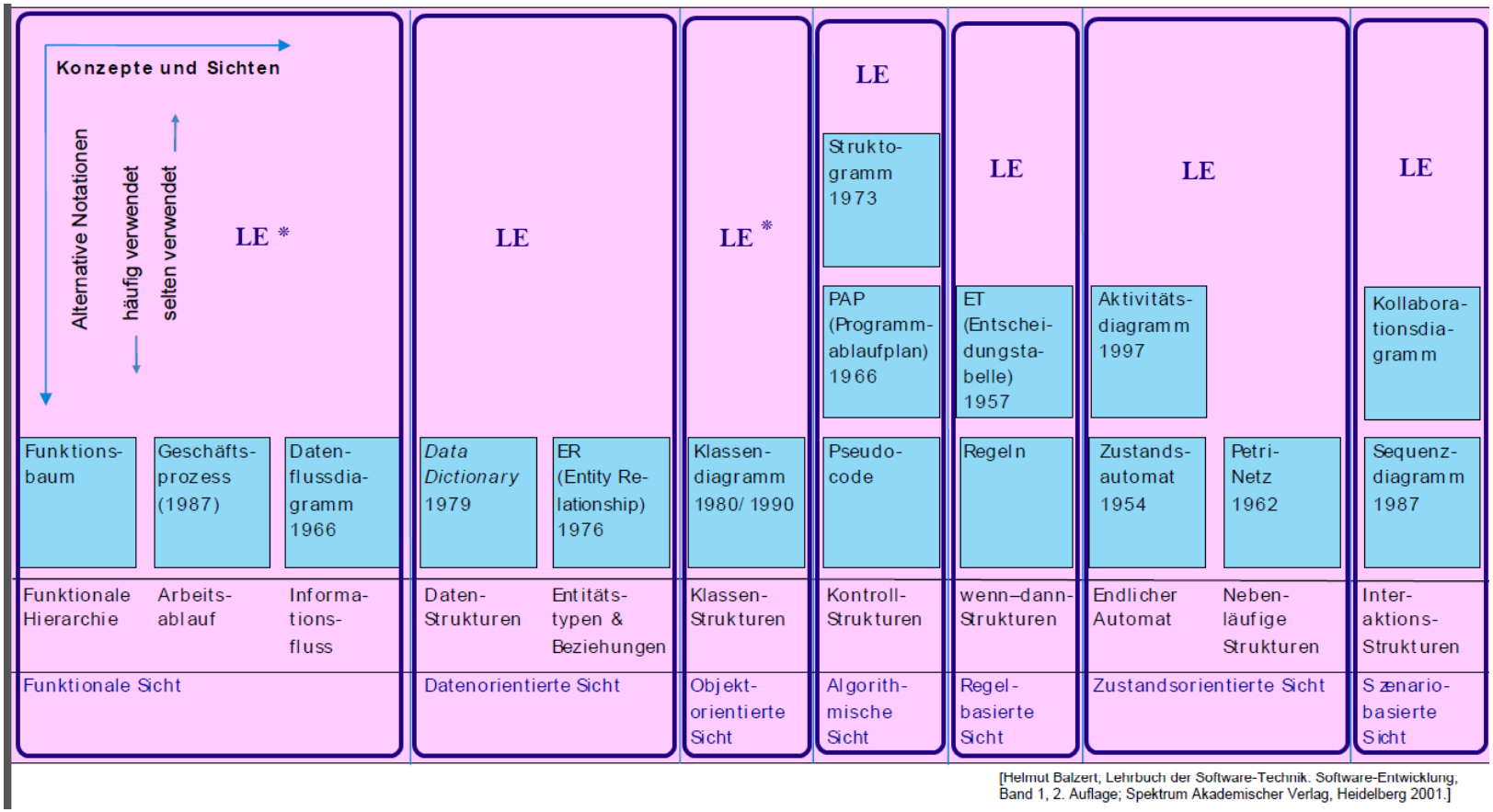
- Zur Beschreibung des Produkt-Modells benutzt man **Basiskonzepte**.

  Ein Basiskonzept ist:
  - ein atomares Konzept,
  - konzeptionell langlebig,
  - phasenübergreifend verwendbar und
  - in unterschiedlichen Kontexten einsetzbar.
- Meisten Basiskonzepte sind recht alt
- Abstraktionsniveau der Basiskonzepte unterschiedlich, je nachdem in welchen Phasen das Konzept eingesetzt wird.

Basiskonzepte zur Systemmodellierung

- Vorraussetzung für erfolgreiche Produkt-Definition ist der Einsatz geeigneter Basiskonzepte.
- Neue Entwicklungen haben dazu geführt, dass verschiedene Basiskonzepte in geeigneter Weise zu kombinierten Konzepten für die Systemanalyse zusammengesetzt wurden. Im Mittelpunkt steht heute:
  - OOA (Object Oriented Analysis)
- Als Stand der Technik sind anzusehen:
  - SA (Structured Analysis)
  - RT (Real Time Analysis)

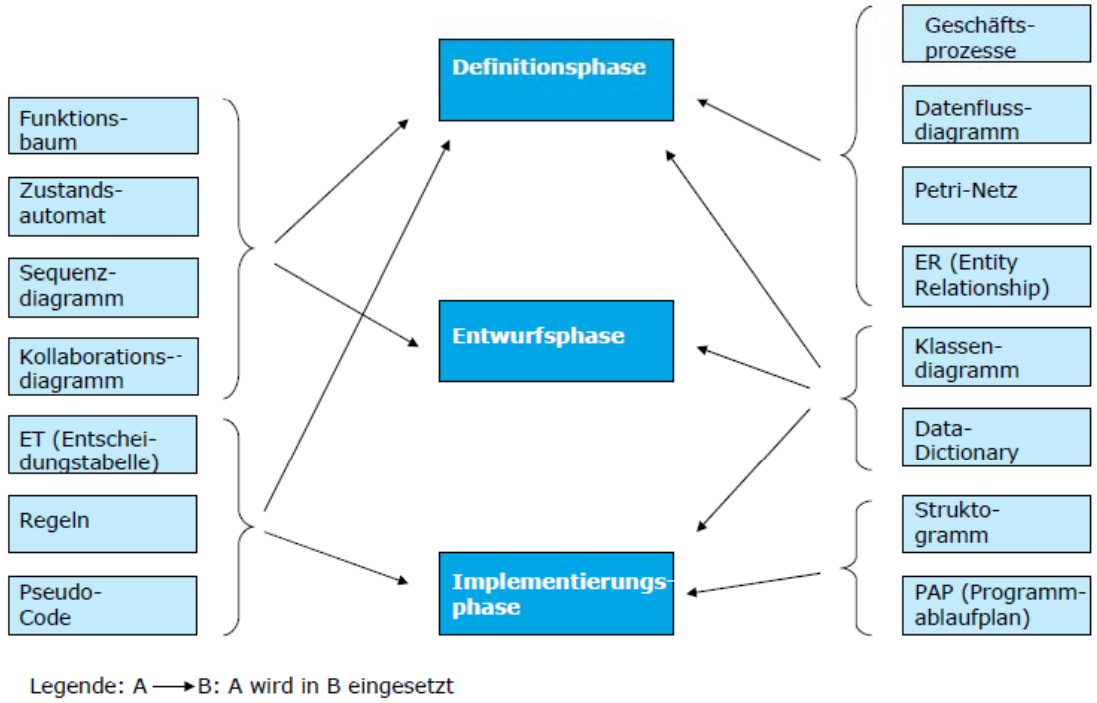
Basiskonzepte zur Systemmodellierung



[Helmut Balzert, Lehrbuch der Software-Technik: Software-Entwicklung, Band 1, 2. Auflage; Spektrum Akademischer Verlag, Heidelberg 2001.]

Basiskonzepte zur Systemmodellierung

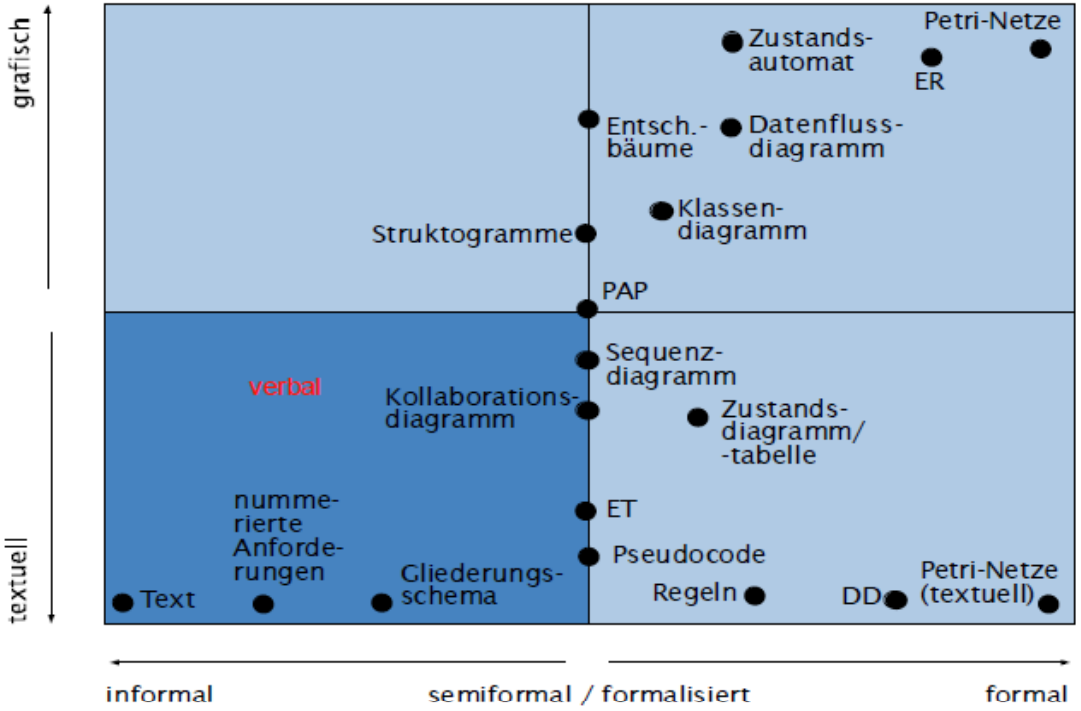
Die verschiedenen Basiskonzeptewerden in verschiedenen Phasen eingesetzt, um verschiedene Anwendungsbereichemit unterschiedlicher Komplexitätzu modellieren.



Legende: A ⟶ B: A wird in B eingesetzt

Basiskonzepte zur Systemmodellierung

- Die Basiskonzepte reichen von informal bis formal und von textuell bis grafisch.

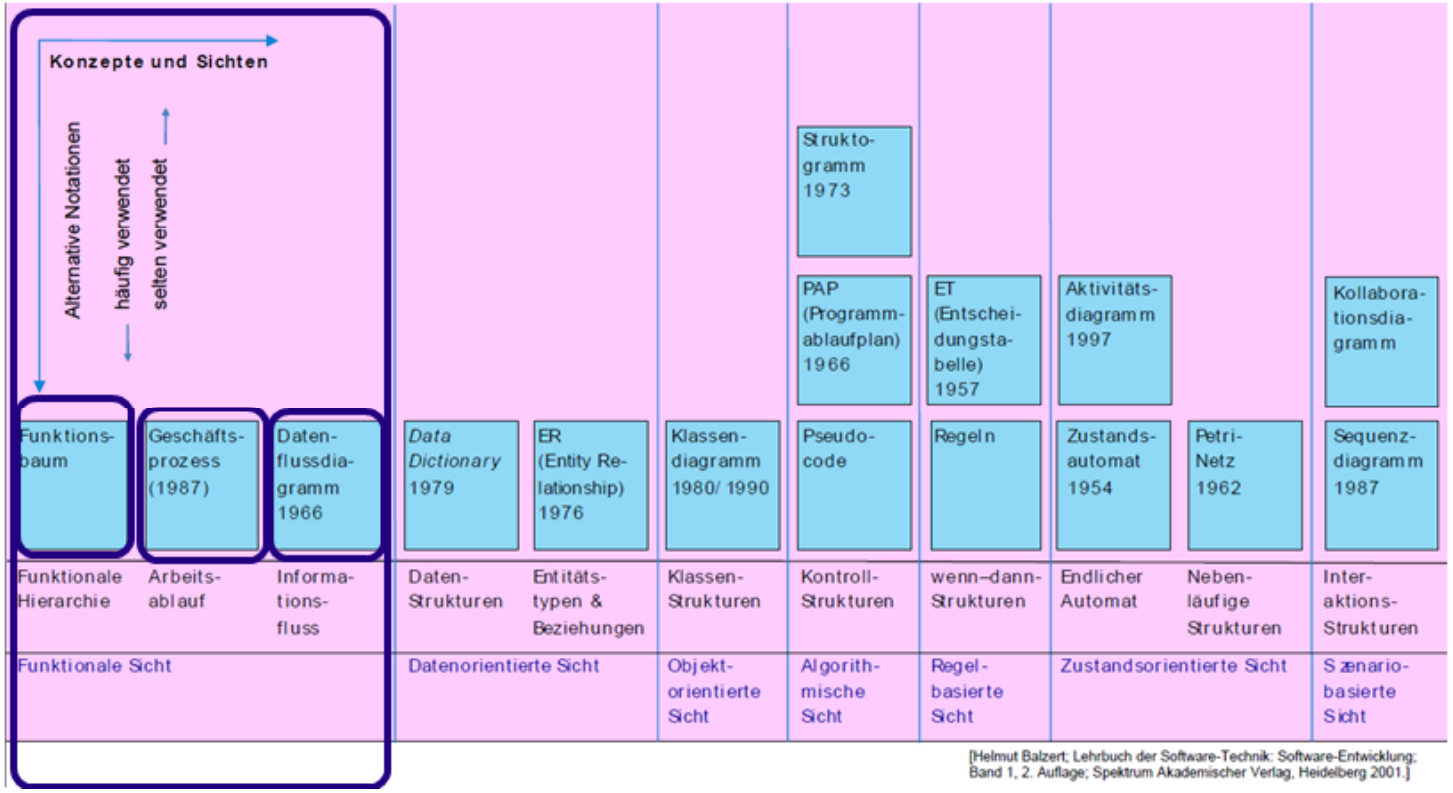


Basiskonzepte zur Systemmodellierung

- **Textuelle Beschreibungen** legen die Anforderungen in Form von natürlich-sprachlichen Texten fest.
- **Grafische Darstellungen** legen Anforderungen durch grafische Symbole und Linien zwischen den Symbolen fest. Diese werden durch textuelle Symbole bezeichnet.
- Unter **formaler Beschreibung** versteht man die Verwendung formaler, textueller Sprachen.
- **Semi-formalisierte Beschreibung** sind stärker formalisiert als die Umgangssprache aber nicht streng mathematisch wie eine formale Sprache.
- **(Informal)Verbale Beschreibungen** sind nochmals gegliedert in:
  - Beschreibungen ohne festgelegte Regeln;
  - Beschreibungen mit nummerierten oder markierten Anforderungen;
  - Beschreibungen mit festgelegten Schemata.

Sichten und Ihre Methoden

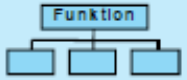
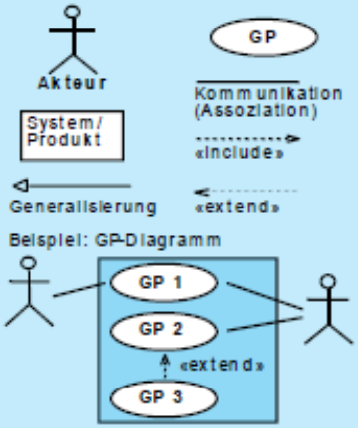
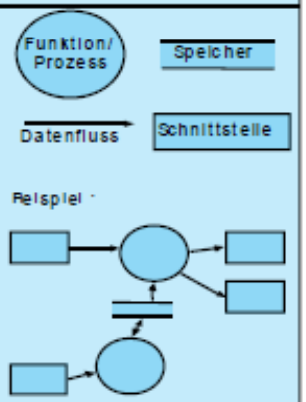
- **Funktionale Sicht**



[Helmut Balzert; Lehrbuch der Software-Technik: Software-Entwicklung; Band 1, 2. Auflage; Spektrum Akademischer Verlag, Heidelberg 2001.]

Sichten und Ihre Methoden

- **Übersicht von Funktionale Sicht nach Balzert 2001:**

Basis - konzept	Funktionale Hierachie	Arbeitsablauf		Informationsfluss
Notation	Funktionsbaum	Geschäftsprozess (GP)		Datenflussdiagramm
»Muster« der Notation	Baumhierachie	Netz (UML)	Muster/ Schablone	Netz
Elemente			Geschäftsprozess: Ziel: Kategorie: Vorbedingung: Nachbedingung Erfolg: Nachbedingung Fehlschlag: Akteure: Auslösendes Ereignis: Beschreibung: 1 2 Erweiterungen: 1a Alternativen: 1a	
Methodisches Vorgehen	top-down	Von den Akteuren ausgehend *(outside-in)*		Von den Schnittstellen ausgehend *(outside-in)*
Regeln	■ Zu einer Vater-fkt. nur fachlich eng zusammen-gehörige Fkt. ■ Pro Hierachie-ebene gleiches Abstraktions-niveau	■ GP-Name: Gerundium oder Substantiv-Verb oder Anfang-Ende ■ Zuerst Standardfall, dann Sonderfälle ■ Gemeinsames Verhalten: «include»-Beziehung ■ Erweitertes Verhalten: «extend»-Beziehung		■ Nur Datenfluss beschreiben, keinen Kontrollfluss ■ Schnittstelle ist Quelle/ Senke von Informationen ■ Nur Funktionen/ Prozesse greifen auf Schnittstellen und Speicher zu

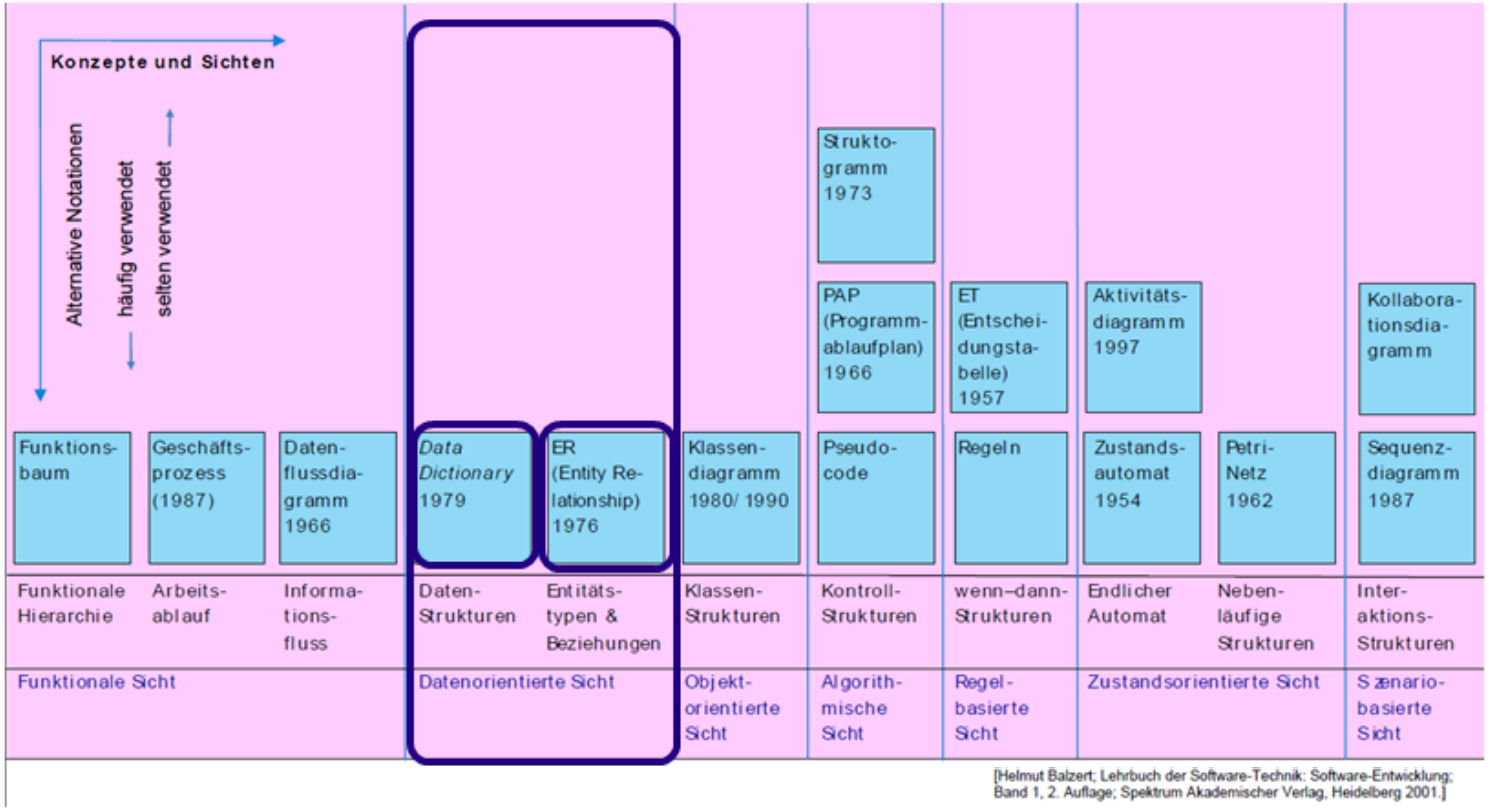
Sichten und Ihre Methoden

Funktionale Sicht

- Die funktionale Sicht auf ein zu entwickelndes Produkt kann durch die hierarchische Gliederung von Funktionen, angeordnet in einem Funktionsbaum, beschrieben werden.
- Durch Geschäftsprozesse können die Arbeitsabläufe von Akteuren mit dem Produkt auf einer hohen Abstraktionsebene beschrieben werden. Die Dokumentation erfolgt in UML durch Geschäftsprozessdiagramme. Zur Spezifikation einzelner Geschäftsprozesse kann eine Geschäftsprozess-Schablone eingesetzt werden.
- Durch DFD kann der Informationsfluss zwischen Funktionen, Speicher und Schnittstellen zur Umwelt dargestellt werden.

Sichten und Ihre Methoden

- **Datenorientierte Sicht**



[Helmut Balzert; Lehrbuch der Software-Technik: Software-Entwicklung; Band 1, 2. Auflage; Spektrum Akademischer Verlag, Heidelberg 2001.]

Sichten und Ihre Methoden

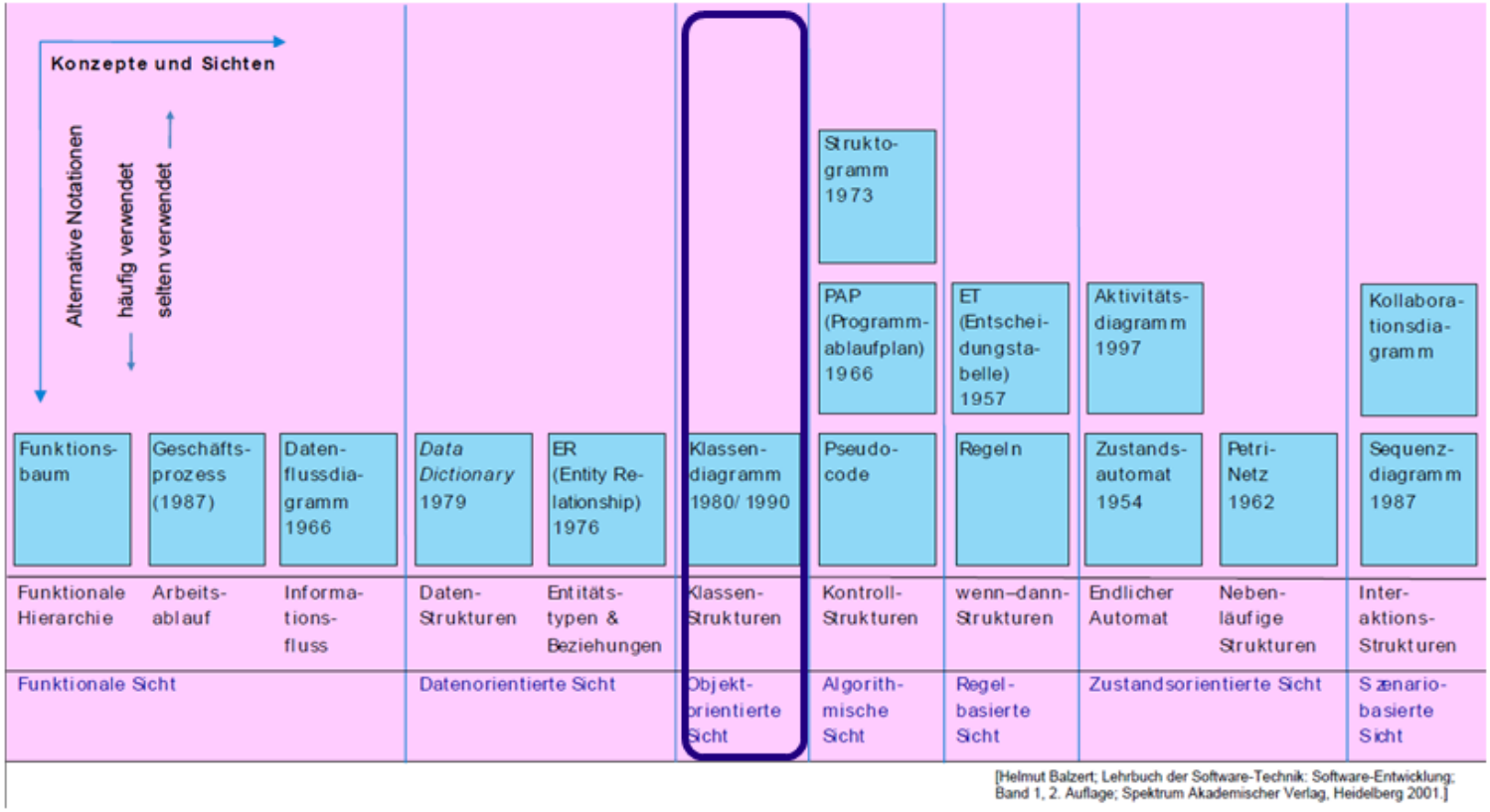
- **Datenorientierte Sicht (Grob)**
  - Das **ER-Modell** erlaubt die Modellierung von permanent zu speichernden Datenstrukturen und ihren Beziehungen zueinander.
  - **Entitäten (entities)** werden zu Entitätstypen (entity types) zusammengefasst. Gleichrangige, fachliche Zusammenhänge zwischen Entitäten werden durch Beziehungen (**Assoziationen, relationships**) beschrieben, die zu Beziehungstypen (relation types) zusammengefasst werden.
  - Durch **Rollen** wird angegeben, welche Funktion eine Entität in der Beziehung spielt.
  - Die Eigenschaften von Entitäten und Beziehungen werden durch Attribute beschrieben.
  - Jede Entität muss durch einen eindeutigen Schlüssel identifizierbar sein.
  - Der Komplexitätsgrad einer Assoziation wird durch die Kardinalität angegeben (Muss- oder Kann-Beziehung,variable oder feste Obergrenze).
  - Eine rekursive Beziehung liegt vor, wenn ein Entitätstyp mit sich selbst in Beziehung steht.
  - Das Ergebnis einer ER-Modellierung bezeichnet man als **konzeptionelles Modell**.

Sichten und Ihre Methoden

- **Datenorientierte Sicht (Grob)**
  - Eine präzise Definition von Datenstrukturen wird durch **Data Dictionary-Einträge** (DD) ermöglicht. DD-Einträge verwenden dabei eine textuelle, modifizierte Backus-Naur-Form zur Definition.

Sichten und Ihre Methoden

- **Objektorientierte Sicht**



[Helmut Balzert; Lehrbuch der Software-Technik: Software-Entwicklung; Band 1, 2. Auflage; Spektrum Akademischer Verlag, Heidelberg 2001.]

Sichten und Ihre Methoden

- **Objektorientierte Sicht**

Objektorientierte SW-Entwicklung basiert auf einer Reihe von Grundkonzepten. Hier werden verschiedene Diagrammarten benutzt, um unterschiedliche Aspekte darzustellen. Als grafische Notation wird UML verwendet.

OO-Konzepte sind gut geeignet für Datenkapselung

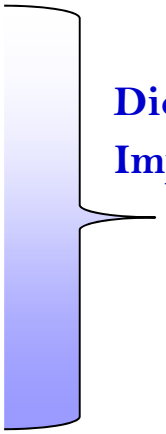
Es ist entscheidend, dass das Konzept der Vererbung sinnvoll eingesetzt wird

Sichten und Ihre Methoden

- **Objektorientierte Sicht**

Konzepte der objektorientierten Programmierung und Modellierung am Beispiel von C++.

- Grundlagen der objektorientierten Systemanalyse
- C++ - Grundbegriffe
- Klassen, Objekte, Methoden
- Abstraktionen, Generalisierungen, Schnittstellen
- Vererbung und Polymorphie
- Konzepte der Parallelverarbeitung



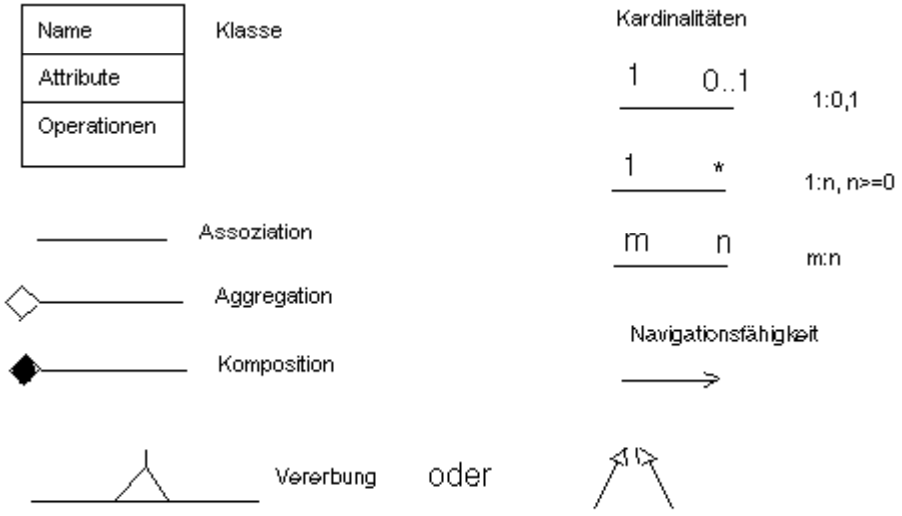
Dies werden wir in der Implementationsphase sehen.

Wichtigste OO-Konzepte sind : Objekte, Klassen, Attribute, Operationen, Assoziationen, CRC-Karten, Vererbung, Pakete, Botschaften, Szenarios

Sichten und Ihre Methoden

- **Objektorientierte Sicht**

Elemente eines Klassendiagrams



Sichtbarkeit

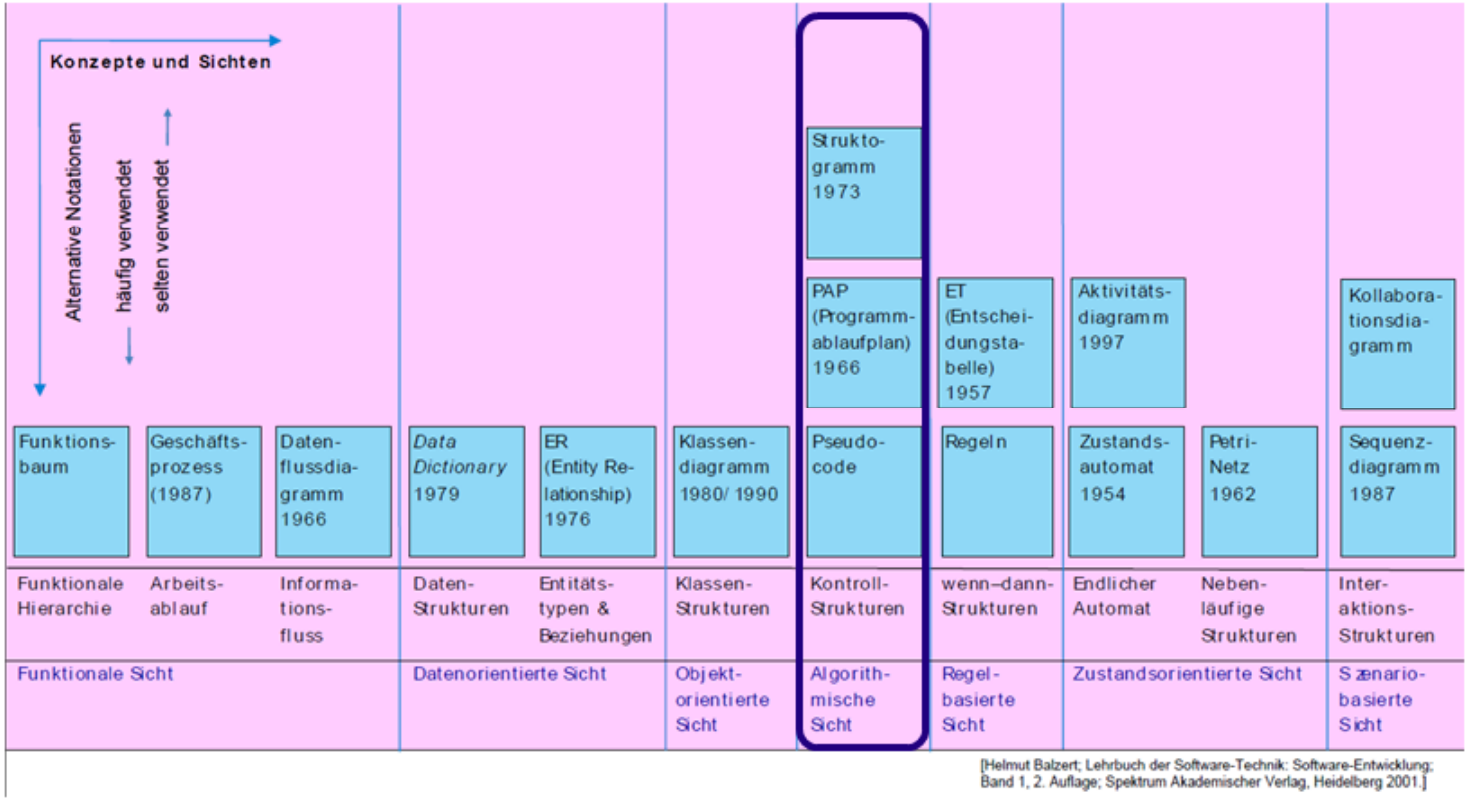
+ public
# protected
- private

Aufbau einer Klasse

KlassenName
Sichtbarkeit Attribut:Typ=Initialwert
Sichtbarkeit Methode(Parameter):Returntyp

Sichten und Ihre Methoden

- **Algorithmische Sicht**



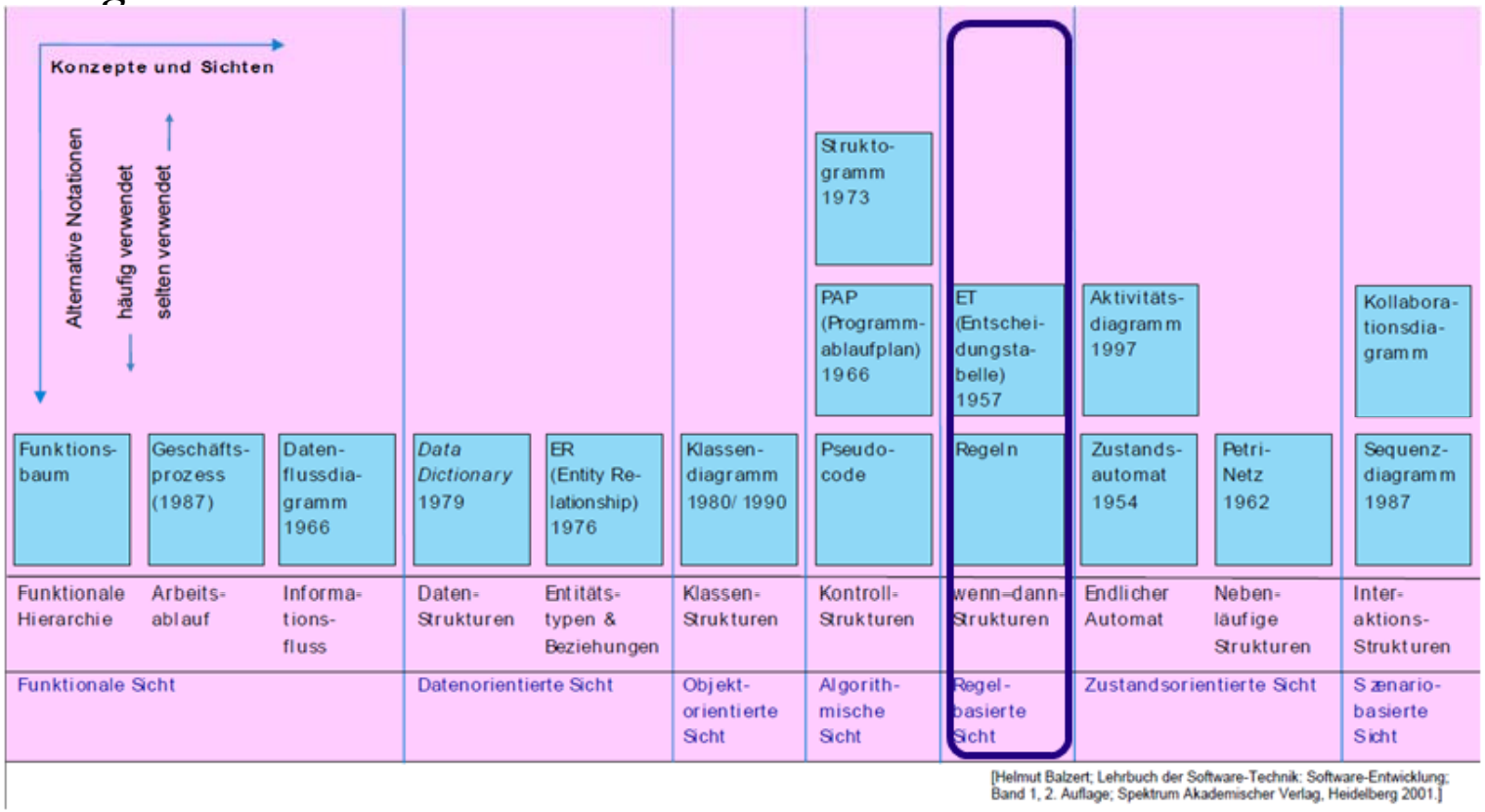
[Helmut Balzert; Lehrbuch der Software-Technik: Software-Entwicklung; Band 1, 2. Auflage; Spektrum Akademischer Verlag, Heidelberg 2001.]

Sichten und Ihre Methoden

- **Algorithmische Sicht (Grob)**
  - Kontrollstrukturen legen innerhalb eines Algorithmus fest, in welcher Reihenfolge, ob und wie oft Anweisungen ausgeführt werden sollen. Die **strukturierte Programmierung** erlaubt nur **lineare Kontrollstrukturen**. Es lassen sich vier verschiedene Typen unterscheiden: die **Sequenz**, die **Auswahl**, die **Wiederholung** und der **Aufruf**. Alle Typen lassen sich beliebig miteinander kombinieren und ineinander schachteln.
  - Eine textuelle Darstellungsform (Pseudo-Code) und zwei grafische (Struktogramme und PAP).

Sichten und Ihre Methoden

- **Regelbasierte Sicht**

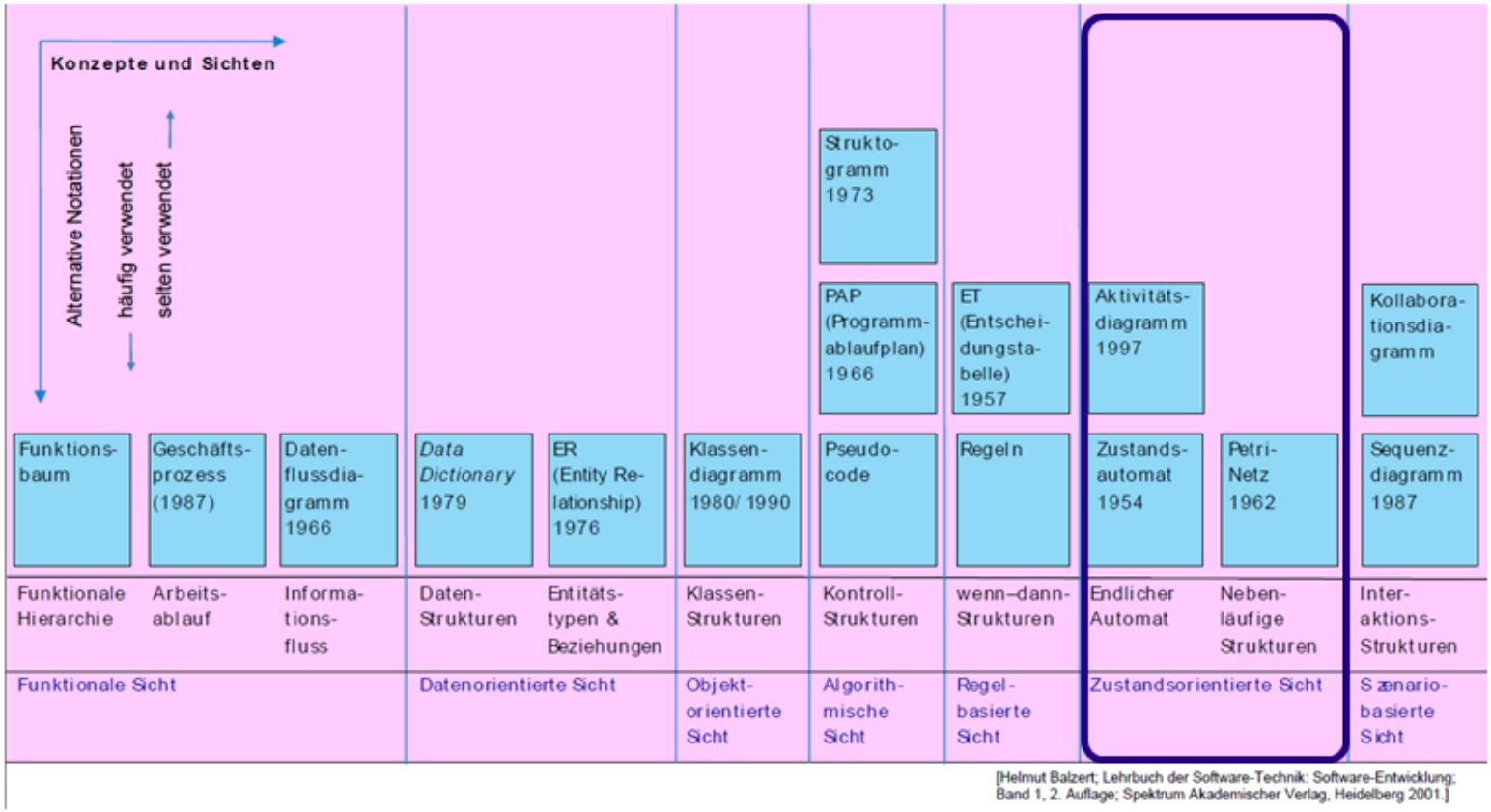


Sichten und Ihre Methoden

- **Regelbasierte Sicht (Grob)**
  - **Regeln** erlauben die Beschreibung von Anforderungen an ein neues System in der Wenn-Dann-Form. Werden Regeln exakt entsprechend einer gegebenen Syntax und Semantik formuliert, dann können sie auch maschinell durch regelbasierte Expertensysteme abgearbeitet werden. Ein Regelinterpreter liest die gespeicherten Regeln und prüft sie ausgehend vom vorgegebenem Ziel (Rückwärtsverkettung, backward chaining) oder ausgehend von den vorgegebenen Fakten (Vorwärtsverkettung, forward chaining).
  - **Entscheidungstabellen** erlauben eine tabellarische oder grafische Darstellung (Entscheidungsbäume) von durchzuführenden Aktionen in Abhängigkeit von Bedingungen**. (Entscheidungstabelle sind viel mehr Kontrlstrukturen)**
  - Erfordert eine Problembeschreibung eine Kombination von Entscheidungstabellen (möglich sind Sequenz, Verzweigung, Schleife und Schachtelung), dann geschieht dies durch einen **Entscheidungstabellen Verbund**

Sichten und Ihre Methoden

- **Zustandsorientierte Sicht**



[Helmut Balzert; Lehrbuch der Software-Technik: Software-Entwicklung; Band 1, 2. Auflage; Spektrum Akademischer Verlag, Heidelberg 2001.]

Sichten und Ihre Methoden

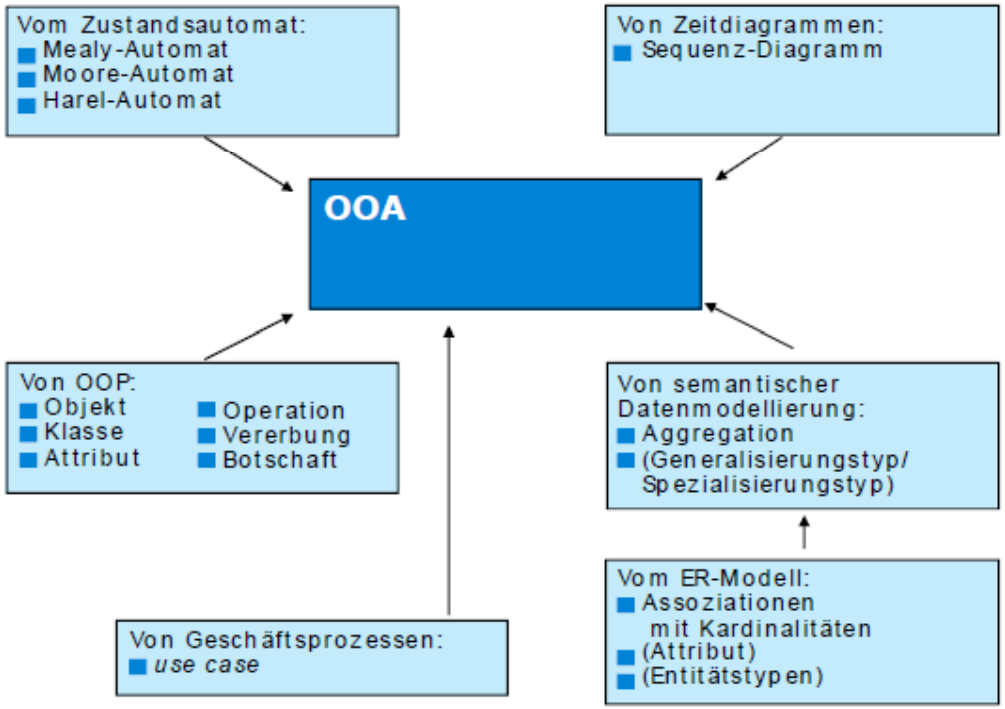
- **Zustandsorientierte Sicht (Grob)**
  - Viele Systeme und Geräte zeigen ein Verhalten, das von der bis dahin durchlaufenen Historie abhängt. Das aktuelle Verhalten wird durch den internen Zustand bestimmt, der durch vorausgegangene Eingaben oder Ereignisse erreicht worden ist. Zur Modellierung solcher Systeme eignen sich **Zustandsautomaten** (finite state machine), auch endliche Automaten (finite automaton, sequential machine) genannt.
  - Das **Aktivitätsdiagramm** ist ein Sonderfall des Zustandsdiagramms. Es wird in der UML zur Beschreibung von Geschäftsprozessen und von Operationen verwendet
  - **Petri-Netze** ermöglichen die Modellierung, Analyse und Simulation nebenläufiger Systeme. In Abhängigkeit von dem zu modellierenden System können verschieden mächtige Klassen von Petri-netzen eingesetzt werden. Es werden einfache Petri-Netze (Bedingungs/Ereignis-Netze, Stellen/Transitions-Netze) und höhere Petri-Netze (Prädikat/Transistions-Netze, zeitbehaftete Petri-Netze) unterschieden.

Kombinationen von Basiskonzepten

- Verschiedenen Basiskonzepte werden auch in Kombination eingesetzt.
- Die wichtigsten Kombinationen von Basiskonzepte führen zu den Konzepten
  - Objektorientierter Analyse (OOA),
  - Strukturierte Analyse (SA) und
  - Strukturierte Analyse Real-Time(SA/RT).

OOA-Objektorientierte Analyse

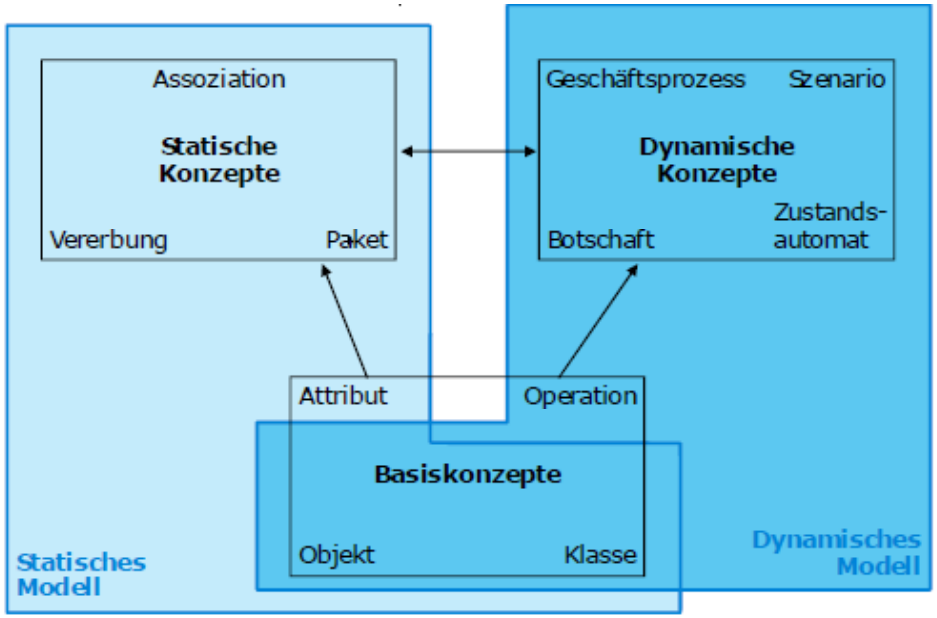
- **Entstehung der OOA(Balzert 2001)**



OOA-Objektorientierte Analyse

- **Ziel der OOA**

  Ziel der OOA ist es, die fachliche Lösung eines neuen Software-Produkts mit Hilfe objektorientierter Konzepte zu modellieren. • Die entstehende Spezifikation besteht aus einem statischen und einem dynamischen Modell.



OOA-Objektorientierte Analyse

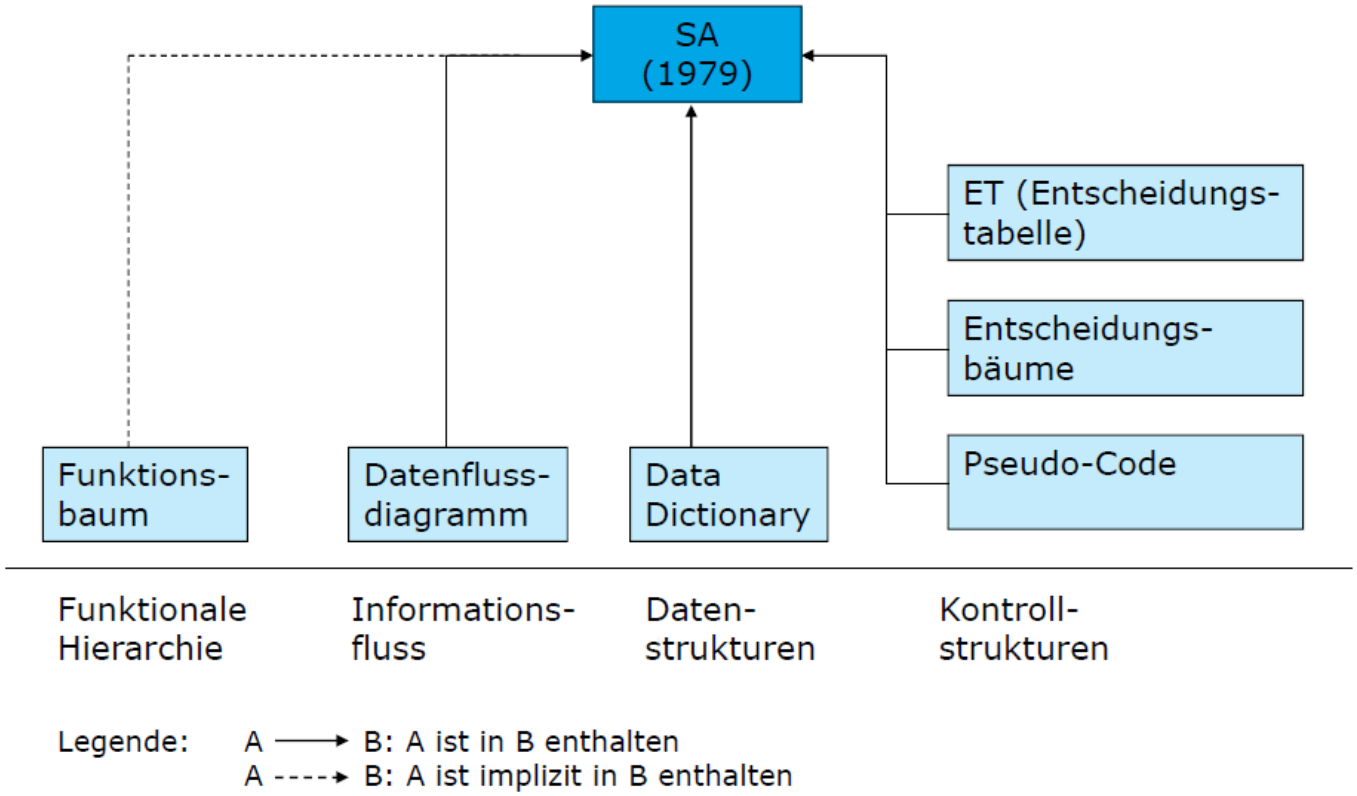
- Die OOA verwendet die Basiskonzepte der Objektorientierung - Objekt, Klasse, Attribute, Operationen – ergänzt um die statischen Konzepte Assoziation, Vererbung, Paket und die dynamischen Konzepte Botschaft, Geschäftsprozess, Zustandsautomat und Szenario um die **fachliche** Problemlösung zu modellieren. Dabei entstehen ein **statisches** und ein **dynamisches** Modell, die sich gegenseitig beeinflussen und ausbalanciert sein sollen.
- **Muster** (patterns) sind bewährte generische Lösungen für immer wiederkehrende Probleme, die in bestimmten Situationen auftreten. OOAMuster werden für die Analyse und Konstruktion von OOA-Modellen benutzt. Wichtige OOA-Muster sind: Liste, Exemplartyp, Baugruppe, Stückliste, Koordinator, Rollen, wechselnde Rollen, Historie, Gruppe und Gruppenhistorie.
- Das Erstellen eines OOA-Modells auf der Grundlage schriftlicher Informationen, z. B. eines Pflichtenheftes, oder mündlicher Informationen, z. B. durch Interviews, gehört zu den schwierigsten Aufgaben der Software-Technik.
- Für jedes objektorientierte Konzept stehen einheitlich aufgebaute **Checklisten** zur Verfügung, die die Konstruktion und Analyse unterstützen

Strukturierte Analyse/ Real Time Analysis

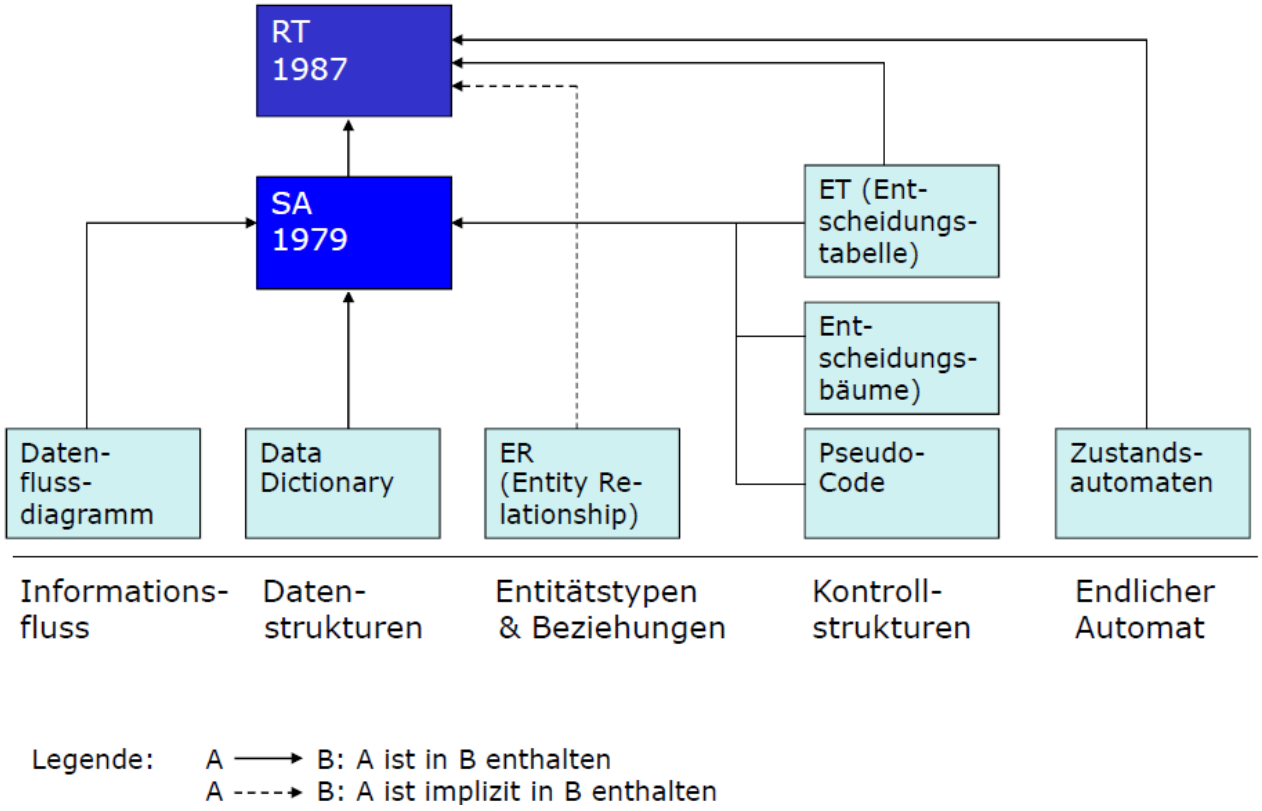
- Die Strukturierte Analyse (SA, structured analysis) besteht aus einem Hierarchiemodell, das die einzelnen DFD als Baumanordnet. Wurzel des Baumes ist das Kontextdiagramm. Blätter des Baumes sind DFD, die nicht weiter verfeinert werden können. Prozesse dieser DFD werden durch Mini-Spezifikationen (MiniSpecs) beschrieben. Innerhalb eines SA-Modells, d. h. von der Baumwurzel bis zu den Baumblättern, muss die Datenintegrität (balancing) sichergestellt sein, d. h., die DFD müssen zwischen Kind- und Elterndiagramm ausbalanciert sein.

- Real-Time Analysis (RT) erweitert SA um die Möglichkeit, Prozesse zu aktivieren und zu deaktivieren. Außerdem können Zeitspezifikationen beschrieben werden. Um dies zu ermöglichen, gibt es neben den atenflüssen auch Kontrollflüsse, die Ereignisse repräsentieren. DFD werden zu Flussdiagrammen verallgemeinert, die zusätzlich Kontrollflüsse enthalten können. Die Prozess-Steuerung der Prozesse, die sich auf einem Flussdiagramm befinden, erfolgt durch einen zugeordnete Kontrollflussspezifikation (Cspec).

SA und seine Basiskonzepte



SA/RT und seine Basiskonzepte



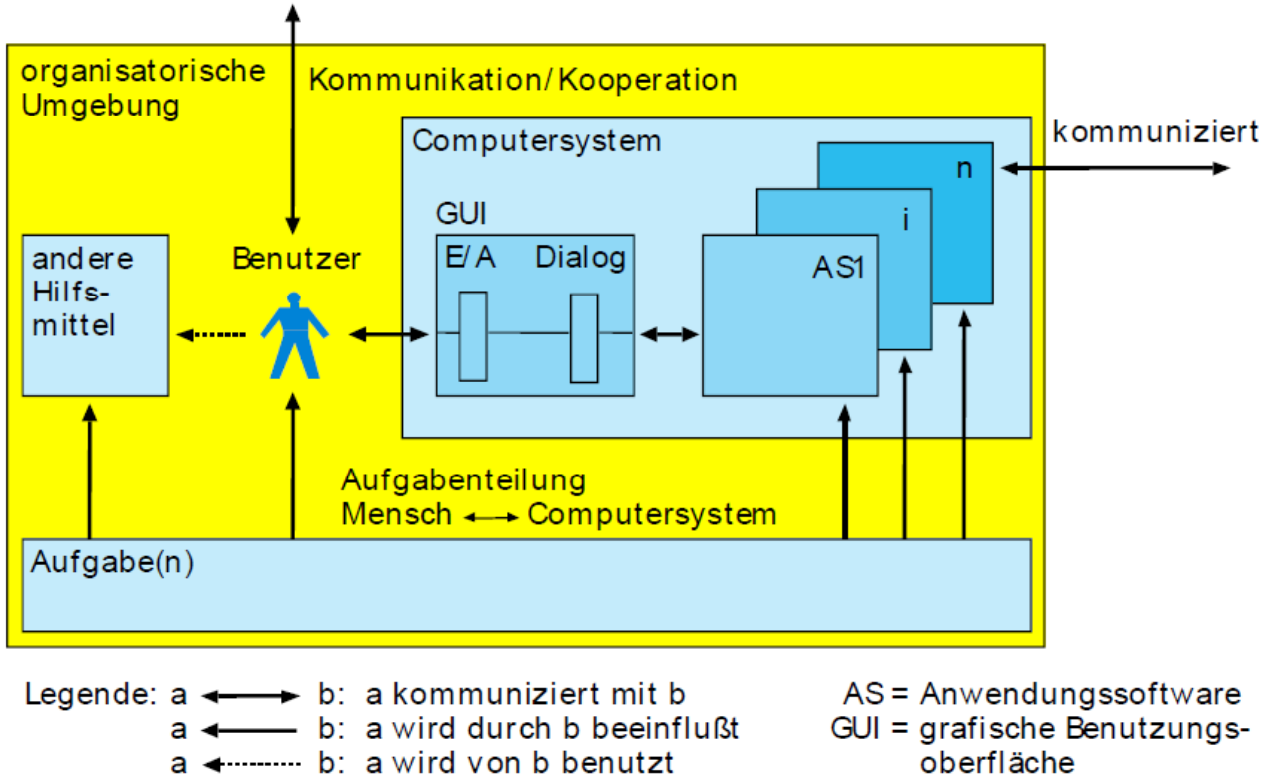
Vergleich SA un SA/RT

- Defizite der Strukturierten Analyse
  - Strukturierte Analyse verzichtet bewusst auf die Beschreibung der Initialisierung und Terminierung eines Systems
  - Zeitanforderungen können in SA nicht festgelegt werden
- Realtime-Analysis erweitert Strukturierte Analyse so, dass ...
  - ereignisgesteuerte Systeme mit
  - Zeitanforderungen und
  - komplexen Prozessaktivierungen
  - modelliert werden können.
- SA/RT ist
  - Industriestandard.
  - Erste Werkzeuge waren schon ein Jahr nach Vorstellung der Methode (1987) verfügbar.

Software-Ergonomie

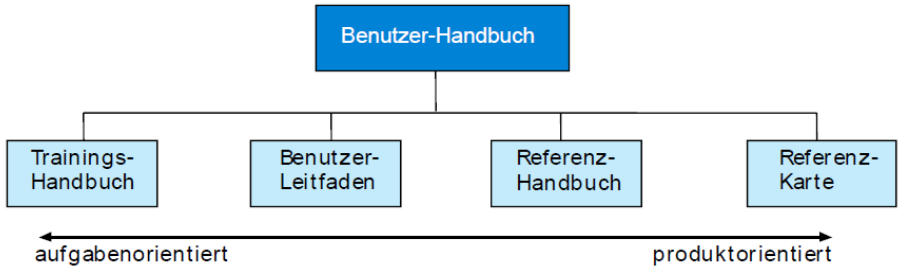
- Die Software-Ergonomie hat das Ziel, dem Benutzer in seinem Nutzungskontext ein gebrauchtaugliches Software-Produkt zur Verfügung zu stellen. Gebrauchstauglichkeit gliedert sich in die Kriterien: Effektivität, Effizienz und Zufriedenstellung.
- Generell ist davon auszugehen, dass jeder Benutzer eine individuelle Lernkurve bei der Benutzung eines Computersystems durchläuft. Grob vereinfacht lassen sich Anfänger, Gelegenheitsbenutzer und Experten unterscheiden, die verschiedene Anforderungen an die Software-Ergonomie haben.
- Das „look and feel“ einer Arbeitsoberfläche wird im wesentlichen durch die GUISysteme bestimmt, die eine Benutzungsoberfläche realisieren und die Kommunikation mit den Anwendungen abwickeln. Eine neuartige Schnittstelle ergibt sich für den Entwickler beim Einsatz eines Web-Browsers als Benutzungsoberfläche eines Software-Produkts

Software-Ergonomie



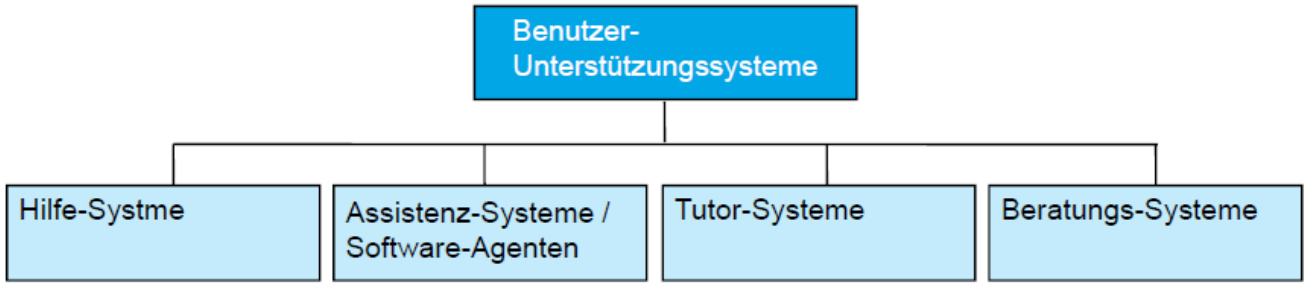
Handbücher und Benutzerunterstützungssysteme

- Zu jedem Software-Produkt gehört eine adäquate, vollständige und fehlerfreie Dokumentation. Die für den Endbenutzer bzw. Anwender des Software-Produktes bestimmte Dokumentation bezeichnet man als Benutzer-Handbuch.
- Es gibt verschiedene Handbuchtypen in Abhängigkeit davon, ob eine produktorientierte Gliederung – Referenz-Handbuch, Referenz-Karte (quick reference) – oder eine aufgabenorientierte Gliederung –Trainings-Handbuch (tutorial), Benutzer-Leitfaden (user guide) – im Vordergrund steht.
- Wichtige Gestaltungsziele für den Benutzer-Leitfaden sind die leichte Navigation, das leichte Erlernen und gute Lesbarkeit.



Handbücher und Benutzerunterstützungssysteme

- Gute Benutzer-Unterstützungssysteme (user support system) beschleunigen die Einarbeitung, reduzieren die Schulungs- und Trainingskosten, erleichtern den Umgang mit den Software-Systemen, unterstützen den Benutzer bei der Problemlösung und übernehmen die Erledigung von „Lehrlings-Arbeiten“.
- Alle Systeme überlappen sich in Teilbereichen. Anzustreben sind daher integrierte Unterstützungssysteme.



Alle Systeme überlappen sich in Teilbereichen. Anzustreben sind daher integrierte Unterstützungssysteme.