Die Bearbeitung aller Übungsaufgaben, die mit diesem Formblatt nachgewiesen wird, ist **Voraussetzung für die Zulassung zur Prüfung**.		**Abgabetermin: vor Weihnachten** bzw. **Mitte Juni**
Name (DRUCKSCHRIFT)	Vorname	Matrikel-Nr.
1.		
2.		
3.		
4.		
5.		
6.		
7.		
Übungsthema	1. Korrektur	2. Korrektur
1.1 Projektmanagement		
1.2 Erkenntnisgewinnende Methoden		
2. Modelldarstellung (V Phasenkonzepte 1)		
3. Phasenkonzepte, Zeitrelationen (V Phasenkonzepte 2)		
4.1 Problemanalyse: Systemabgrenzung		
4.2 Problemanalyse: Seiteneffekte		
5.1 Referenzmodell Unternehmen		
5.2 Referenzmodell EK – VK		
6. Informationsflussmodelle		
7. Statische Objektmodellierung		
8. Klass. Prozessmodellierung		
9. Moderne Prozessmodellierung		
10. Gruppenverarbeitung		

Hinweise zu den Materialien

Die Materialien zu dieser Lehrveranstaltung befinden sich auf meiner Homepage unter „Lehrveranstaltungen und Folien → Software Engineering“.

Diese Materialien stellen weder inhaltlich noch sprachlich ein Skriptum dar, sondern eine Begleitlektüre dieser Lehrveranstaltung:

- inhaltlich: Die Materialien enthalten den Unterrichtsstoff nicht vollständig und gehen teilweise darüber hinaus. Ihre Lektüre ersetzt den Besuch der Lehrveranstaltung nicht.

- sprachlich: Die Materialien sind weitgehend auf Englisch gehalten, entsprechend dem Grundsatz: Die Sprache empfohlener oder verbindlicher Begleitlektüren zu Lehrveranstaltungen an Hochschulen kann eine in der jeweiligen Disziplin übliche internationale Publikationssprache (z.B. Englisch) sein, die von der Unterrichtssprache der Lehrveranstaltung (z.B. Deutsch) abweicht. An der TH Nürnberg kursierende deutsche Übersetzungen dieser Materialien sind und werden von mir nicht autorisiert.

Hinweise zu den Übungen

Neben den fachlichen Lernzielen werden zwei für wissenschaftliches Arbeiten ganz allgemein gültige in den Fokus gestellt, nämlich, die explizite inhaltliche Strukturierung von Texten und den Umgang mit Quellen zu erlernen (vgl. meine Anleitung zum wissenschaftlichen Arbeiten):

- Es wird eine inhaltlich explizit strukturierte Beantwortung der Fragen erwartet, d.h. Sie verwenden jeweils eine geeignete inhaltliche Gliederung mit Nennung der Gliederungspunkte in Form von Überschriften.
  Spätestens bei mehr als fünf bis sieben (oft schon bei weniger) Gliederungspunkten in einer Ebene ist eine neue Gliederungsebene einzuführen, sonst entsteht automatisch eine inhaltlich unstrukturierte Aufzählung. Inhaltlich unstrukturierte Aufzählungen sind aber grundsätzlich nicht akzeptabel.
  Aufzählungspunkte und Spiegelstriche sind keine inhaltliche, sondern nur eine formale Strukturierung (Layoutstrukturierung)!

- Die Angabe der Quellen (Bücher, Zeitschriften, Internet etc.), an denen Sie sich bei der Aufgabenbeantwortung orientieren, ist verbindlich. Dies gilt für wörtliche Zitate ebenso wie für nicht wörtliche (sinngemäße). Die Quellenangaben sind den einzelnen Teilen Ihrer Antworten zuzuordnen, nicht nur summarisch zu nennen.
  Jede zitierte Abbildung erfordert eine spezielle Quellenangabe.
  Zu manchen Fragen werden Sie nur wenig Quellen finden, also: Selbst denken ist auch erlaubt und erwünscht ☺

- Vergleich, Auswahl und Bewertung verschiedener Quellen

Rahmenthema: Geschäftsprozess Auftragsverwaltung
(Bestellung des Kunden, Auftragsannahme, Auftragsbestätigung mit Liefertermin, Lieferschein, Lagerentnahme der bestellten Ware, ggf. Korrektur der Liefermengen, Verpackung der Ware, Versand der Ware mit Lieferschein an den Kunden, Fakturierung, Rechnung an den Kunden, Reklamation des Kunden)

Je nach Ihren speziellen Interessen können Sie auch andere geeignete Rahmenthemen behandeln. Schaffen Sie ggf. Synergieeffekte mit anderen Lehrveranstaltungen.

Graphiken sind stets ausführlich zu kommentieren.

General topic: business process order management
(customer order, order acceptance, order confirmation with delivery date, delivery note, stock requisition of the items ordered with optional changes of quantities delivered, packing of the items, shipment of the items to the customer with delivery note, invoicing, invoice to the customer, customer complaint)

According to your specific interests, you can choose other adequate general topics as well. If possible, benefit from synergy effects with other courses.

Diagrams should always be commented in detail.

ÜA 1 Projektmanagement und erkenntnisgewinnende Methoden

Selbststudium:

1. Projektmanagement: S. 34-39 der Foliensammlung

2. Wiederholung des Informationssystem-Begriffs
Auf meiner Homepage unter "Courses and Slides" - "Grundlagen der Wirtschaftsinformatik" den Foliensatz "Cultural differences and specialties of information systems" auswählen, darin Folien 6-11

Die erste Übungsaufgabe - dazu brauchen Sie keine Vorkenntnisse - zerfällt in zwei Teile:

1.1: Planen Sie eine Arbeitswoche mit Hilfe einer Exceltabelle.
- Zeiten von oben nach unten (Zeilen)
- Wochentage von links nach rechts (Spalten)
- Verwenden Sie 7-9 Tätigkeitsarten, die Sie farbig markieren:
  z.B. Selbststudium, Übungsbearbeitung, Berufstätigkeit; Wegezeiten, Freizeit (Pausen) und Puffer nicht vergessen
- für jede Tätigkeitsart ist eine Wochensumme zu bilden
- die prozentualen Anteile jeder Tätigkeitsart sind in einem Kuchendiagramm für die gesamte Woche darzustellen

1.2 (mit S. 33): Es geht um Requirements Engineering, d.h. um die Analyse der Anforderungen an ein Informationssystem, die ein Anwender (business expert) einem Wirtschaftsinformatiker mitteilt.
Diese Anforderungen werden in den allermeisten Fällen natürlichsprachlich formuliert, ohne jegliche mathematisch-formale Optimierung, die wir für die Modellierung eines Informationssystems aber unbedingt brauchen.

Bei der Übungsaufgabe sollen Sie
1. solche Stellen in einem natürlichsprachlichen Text erkennen und herausfiltern, die nicht oder nur schwer in die formale Sprache eines Modells zu übersetzen sind. Das sind genau die Stellen, bei denen Kommunikationsprobleme auftreten und wir als Wirtschaftsinformatiker beim Anwender nachfragen müssen, damit er seine Angaben präzisiert.
2. diese Kommunikationsprobleme klassifizieren, d.h. zu Kategorien zusammenfassen.

Für unseren Onlinekurs vereinfachen wir diese Übungsaufgabe etwas gegenüber dem Text auf S. 33 und lassen den dort genannten Teil 2 weg.

Ihre Aufgabenlösung soll eine zweispaltige Excel-Tabelle sein:
erste Spalte: Kategorie (mindestens 5 Kategorien)
zweite Spalte: Beispiel aus dem alten Kochrezept (mindestens 10-15 Beispiele)
Diese Tabelle sortieren Sie nach der Kategorie.

Zusammengefasst:

1.1 IT-Projektmanagement
Ihre nächste Arbeitswoche / Ihr Studium als Projekt
oder Rahmenthema: ein Kundenauftrag als Projekt

1.1 IT project management
your next working week / your studies as a project
or general topic: a customer order as a project

1.2 Erkenntnisgewinnende Methoden: Formalisierung, Darstellung
Umsetzung natürlicher Sprache in eingeschränkte natürliche, formale Sprache
(Anfänge der Formalisierung und Mathematisierung, Requirements Engineering):
Diskussion der auftretenden Probleme anhand eines alten Kochrezepts

1.2 Knowledge-gaining methods: formalization, representation
transformation of natural language into restricted natural, formal language
(begin of formalization and mathematization, requirements engineering)
discussion of the occurring problems by means of an old cooking recipe

ÜA 2 Modelldarstellung

Unser erstes großes Thema sind die Phasenkonzepte (software life cycle models, software process models), bei denen es um strukturierte Herstellungsprozesse für qualitativ hochwertige Software geht.
S. 40-62, 209-213

Es gibt darin mehrere wesentliche Erkenntnisse, die Sie unbedingt im Kopf behalten müssen:

1. Fokus: Zusammenspiel von technischen und organisationalen (sozialen, menschlichen) Informationssystemen bei der Bildung von soziotechnischen Informationssystemen - das ist Grundwissen Wirtschaftsinformatik - bei mir auch dargestellt als Schlüssel-Schloss-Prinzip:
- soziale Informationssysteme sind nicht-formal, haben aber formale Anteile
- technische Informationssystem sind rein formal
Diese beiden Ebenen müssen möglichst gut zusammenspielen (wie Schlüssel und Schloss), um ein effektives soziotechnisches Informationssystem zu erzeugen.
Ein Informationssystem im Allgemeinen ist ein informationsverarbeitendes System, egal ob menschlich oder technisch.

2. Fokus: Unterscheidung der beiden Hauptphasen und Modellebenen in jedem Phasenkonzept (standardisiert auf S. 52 und S. 61):
- die analytische Phase / Problemanalyse, in der informationsrelevante Modelle unabhängig von IT entworfen werden.
- die synthetische Phase / IT-System-Entwicklung, in der implementationsrelevante Modelle entworfen und Software generiert wird.

3. Fokus: Hier kommen Fokus 1 und Fokus 2 zusammen (S. 52):
Ein Sollkonzept (design of the planned state) besteht in der Wirtschaftsinformatik immer aus zwei Teilen:
- Sollkonzept für ein soziales Informationssystem/Anwendungsbereich (in das ein technisches Informationssystem eingebettet wird)
- Sollkonzept für das technische Informationssystem (das in das soziale Informationssystem/Anwendungsbereich eingebettet wird)
Reine Informatiker verwenden Sollkonzepte nur für technische Systeme, ohne den organisational-sozialen Anteil.

4. Fokus: Sie müssen darüber hinaus die folgenden Termini verstanden haben und definieren können:
gewinnorientierte und nicht-gewinnorientierte Organisationen - profit and non-profit organizations
Abgleich und Harmonisierung von Geschäftszielen und IT-Zielen - business IT alignment
Sicherstellung, dass IT die Unternehmensziele unterstützt - IT governance
Beschreibungsmodell eines real existierenden Zustands/Ablaufs - descriptive model
Designmodell für einen geplanten Zustand/Ablauf - prescriptive model
(Anwender-)Beteiligungs-Strategie - participative strategy
Anforderungsanalyse - Requirements analysis/engineering
Ist-Zustand - current state
Ist-Aufnahme - elicitation (of the survey) of the current state
Ist-Analyse - analysis of the current state
Soll-Zustand - planned state
Soll-Konzept - design of the planned state
Fachkonzpet - business concept

Schauen Sie sich noch weitere Beispiele für alternative Phasenkonzepte an, die in Privatwirtschaft und Industrie verwendet werden (auch sie enthalten implizit immer die Unterscheidung von analytischer Phase und synthetischer Phase: S. 55-60, 209-213

Übung 2
Sie untersuchen einen fiktiven Vorgang, das Entgegennehmen eines Auftrags am Telefon, und beschreiben ihn in einer dreispaltigen Tabelle.
Dazu müssen Sie eine fiktive Firma im Kopf haben, die Sie unter 1. ganz kurz darstellen.
2. In der ersten Spalte einer Tabelle beschreiben Sie diesen Vorgang grob in wenigen Schritten (höhere Abstraktionsebene: Standardablauf)
3. In der zweiten Spalte der Tabelle verfeinern Sie die groben Schritte (aus 2) in mehrere feine Schritte (niedrigere Abstraktionsebene: Details des Standardablaufs).
4. In der dritten Spalte der Tabelle geben Sie mögliche Ausnahmen bei den feinen Schritten (aus 3) an.

Achten Sie darauf, dass die horizontalen Zuordnungen in den Tabellenzeilen eindeutig sind. Jedem Schritt, auch Prozess-Schritt, Aktivität oder Funktion genannt, geben Sie ein Etikett, das aus Substantiv und Verb besteht: z.B. Kundengespräch weiterleiten.
Sie brauchen keine graphische Darstellung abzugeben.

Zusammengefasst:

2. Beschreibung eines Ist-Zustands auf verschiedenen Abstraktionsebenen
   Vorgangsbeschreibung: Entgegennehmen eines Auftrags am Telefon
   aus der Sicht des Auftragnehmers, nicht aus der Sicht des Kunden!
   Ausführung: 1. Offenlegung Ihrer Annahmen über das Unternehmen
   2. höhere Abstraktionsebene: Standardablauf
   3. niedrigere Abstraktionsebene: Details des Standardablaufs
   4. niedrigere Abstraktionsebene: Ausnahmen
   Genaue Zuordnung zwischen 2., 3. und 4
   Darstellung als Tabelle mit drei Spalten für 2., 3. und 4. (ohne Graphik)
   Lernziel 1: Benutzung eingeschränkter natürlicher Sprache
   einerseits so natürlich, dass der Text anwenderseitig verständlich ist, und
   andererseits so formal, dass er als Grundlage eines math. Modells dienen kann
   Lernziel 2: Beachtung von Grenz- / Sonderfällen
   Lernziel 3: Nutzung der Layoutmöglichkeiten eines modernen Textsystems:
   Einrückungen, Absatzgliederung, Fettdruck, Unterstreichung

2. Description of a current state on different abstraction levels
   process description: acceptance of an order at the phone
   text, optionally with a diagram
   requirements: 1. describe your assumptions regarding the company
   2. higher abstraction level: standard process
   3. lower abstraction level: details of the standard process
   4. lower abstraction level: exceptions
   Exact correlations between 2, 3 and 4
   Presentation as table with three columns for 2, 3 and 4

Goal 1: learn to use restricted natural language
on the one hand so natural that a user can understand the text and
on the other hand so formal that it can serve as base of a mathematical model
Goal 2: learn to take critical and special cases into consideration
Goal 3: learn to use the layout features provided by a modern word-processing system:
indentations, paragraph structure, bold type, underline

ÜA 3 Phasenkonzepte und Zeitrelationen

Zeitrelationen zwischen den Teilphasen eines Phasenkonzepts.
S. 70-105 (63-69 überspringen wir)

Es gibt mehrere wesentliche Lernziele:

1. (S. 70-88) Software-Qualität ist in der einen oder anderen Form ein Ziel von allen Projektbeteiligten (Stakeholder).
Ein IT-System zu programmieren, ohne vorher ein detailliertes Sollkonzept erarbeitet zu haben, ist aber völlig unprofessionell und führt zu mangelnder Software-Qualität.
Mangelnde Software-Qualität führt zu einer Kostenexplosion während des Software-Einsatzes.
Während der analytischen Phase zu sparen, ist daher der falsche Weg, um Software-Kosten in Grenzen zu halten.
Dementsprechend ist für die analytische Phase (Problemanalyse) im Verhältnis zur synthetischen Phase (Systementwicklung) ein Zeitanteil - und damit auch ein Kostenanteil - von um die 50 Prozent anzusetzen.

2. (S. 75, 84) Entwurfsfehler, die während der Analysephase gemacht werden und erst während des Probebetriebs oder noch später auffallen, versursachen extrem hohe Kosten bei ihrer Behebung.
Die Beseitigung von reinen Programmierfehlern, die spätestens beim Testen schon im Software-Haus auffallen, ist vergleichsweise billig. Hohe Investitionen in die Analysephase rentieren sich später mit einer wesentlich billigeren Synthesephase.

3. (S. 89-91) Als Wirtschaftsinformatiker erstellen wir IT-Systeme für Menschen (business experts, "user" - dieses Wort sollte man besser gar nicht verwenden). Die müssen später damit arbeiten und müssen uns vorher bei der Erstellung unterstützen. Wenn business experts aus irgendeinem Grund ein Projekt ablehnen und sich dagegen wehren, ist es von vornherein zum Scheitern verurteilt.

4. (S. 94-105) Sie sollten einen Überblick über Ziele und Techniken der analytischen Phase gewinnen.

Zu Übungsaufgabe 3:

Ein Software-Haus soll im Rahmen von zwei getrennten Projekten ein IT-System für einen Serienfertiger und ein anderes IT-System für einen Indiviualfertiger vergleichbarer Größe planen und entwickeln.

Machen Sie sich zunächst die Unterschiede zwischen einem Serienfertiger und einem Individualfertiger klar:
- Ein Serienfertiger produziert wenige verschiedene Waren in einem weitgehend formalisierbaren Produktionsprozess.
- Ein Individualfertiger produziert viele verschiedene Waren je nach aktuellen Kundenwünschen in verzweigten, schwer formalisierbaren Produktionsprozessen.

Ausgehend vom Phasenkonzept auf S. 52 (ohne use und maintenance) sollen Sie die prozentualen Zeitrelationen für die sieben verbleibenden, dort genannten Teilphasen (von elicitation of the current state (Ist-Aufnahme) bis test) abschätzen.

Dafür verwenden Sie eine Tabelle:
vier Spalten:
- geschätzte Prozentangaben für die einzelnen Phasen des Projekts für den Serienfertiger
- kurze Begründung dazu
- geschätzte Prozentangaben für die einzelnen Phasen des Projekts für den Individualfertiger
- kurze Begründung dazu

sieben Zeilen: für jede der sieben Teilphasen eine Tabellenzeile

Grundsätzlich gibt es dafür zwei Argumentationsmöglichkeiten (welcher würden Sie sich anschließen?):

1. Man sagt, dass die Zeitrelationen für beide SW-Entwicklungsprozesse in etwa gleich sind, weil die Komplexitäten in etwa gleich groß sind:
- Die Komplexität beim Serienfertiger liegt darin, dass man einen weitgehend formalisierbaren Produktionsprozess mit sehr vielen Details exakt modellieren muss. Komplexität geht in die Tiefe.
- Die Komplexität beim Individualfertiger liegt darin, dass man verschiedene schwer formalisierbare Prozessvarianten mit unterschiedlichen Parametern modellieren muss. Komplexität geht in die Breite.

2. Die zweite Argumentation sagt, dass der Modellierungsaufwand (analytische Phase) beim Individualfertiger größer ist, da die Ermittlung, Formalisierung, Parametrierung und Modellierung verschiedener Prozessvarianten ungleich länger dauert als die detailgenaue Modellierung eines detaillierten, aber weitgehend festen Produktionsprozesses bei einem Serienfertiger.

Zusätzlich: Wie schätzen Sie das Verhältnis der Gesamtdauern der beiden Projekte ein?

Zusammengefasst:

3. Phasenkonzepte und Entwurfsebenen
Ein SW-Haus soll je ein IT-System (Rahmenthema) für zwei verschiedene Arten von Firmen entwickeln:
Standardfirma (Serienproduktion) und Non-standard-Firma (Individualproduktion)
Vergleichen Sie die zeitlichen Aufwände in den SW-Entwicklungs-Prozessen für beide Firmentypen in tabellarischer Form. Wieviel Zeit würden Sie prozentual für die einzelnen Haupt- und Teilphasen (im Phasenkonzept aus der Vorlesung) ansetzen (Zeitrelationen)?
Begründen Sie Ihre Ansicht kurz.

3. Software life cycle models and design levels
Development of an IT system (general topic) for two different types of companies:
Standard company (mass production) and non-standard company (individual production)
Effective time relations between main phases and subphases; motivation
Verwenden Sie dazu das Phasenkonzept aus der Vorlesung.
Principle: ~65% analytic phase, ~35% synthetic phase.
Use the software life cycle model from the course.

ÜA 4 Problemanalyse: Systemabgrenzung und Seiteneffekte

Es geht um Voruntersuchung und Problemanalyse. Dazu ist nicht viel zu kommentieren, dafür später umso mehr zur Übungsaufgabe 4.

S. 106-109, 111-114, 117-127

In Abschnitt 2 (ab S. 132) geht es um die Voruntersuchung (preliminary examination; oder strategic phase in der ITIL-Terminologie). Man ermittelt in einer Machbarkeitsstudie Fakten, die für die Entscheidung nötig sind, ob man zu einer Problematik überhaupt ein IT-Projekt startet. Diese Voruntersuchung steht also vor (!) dem Phasenkonzept auf S. 107.

2.1 Prinzipien der Voruntersuchung, Arten der Machbarkeit

2.2 Systemabgrenzung (auch schon auf S. 108) und Isolationsproblem. Das brauchen Sie für die Übungsaufgabe 4.

2.3 Vorformalisierung, Formalisierbarkeit und Formalisierungsaufwand (nötig für die Erstellung eines formalen Modells)

In Abschnitt 3 (ab S. 118) sind wir dann mitten im Phasenkonzept aus der letzten Woche.

S. 119 zeigt die Ähnlichkeiten im Ablauf bei der Aufnahme des Ist-Zustands und bei der Planung des Soll-Zustands.

3.2 beschäftigt sich kurz mit Requirements Engineering. Dazu empfehle ich Ihnen das Buch von Chris Rupp und den Sophisten.
Sie können auch meinen auf S. 127 genannten Aufsatz lesen (pdf finden Sie auf meiner Homepage).

In 3.3 und 4 geht es um den Einfluss von Veränderungen in einer Organisation auf die unterstützenden technischen Informationssysteme, die sich deshalb ebenfalls verändern müssen und dadurch bei unsachgemäßer Wartung häufig altern (information systems aging). Wenn Sie das Thema IS Anti-Aging vertiefen wollen, steht Ihnen die auf S. 127 genannte Doppelmasterarbeit auf meiner Homepage zur Verfügung.

Spätestens zur Übungsaufgabe 4 müssen Sie sich für einen bestimmten Unternehmensbereich (Kernbereich, Rahmenthema) entscheiden, den Sie dann auch für die Aufgaben 6-9 brauchen. Am einfachsten ist immer eine Auftragsverwaltung; Sie können aber auch eine Einkaufsverwaltung wählen o. dgl. Eine Auftragsverwaltung ist als Teil des Vertriebs zu verstehen.
Ich gehe im Folgenden davon aus, dass Sie die Auftragsverwaltung als Kernbereich wählen.

Die Aufgabe 4 besteht aus zwei Teilen.

4.1
In dieser Übungsaufgabe geht es um die Abgrenzung eines Kernbereichs (hier: Auftragsverwaltung), der im Zentrum eines Projekts steht, gegenüber den benachbarten betrieblichen Hauptfunktionsbereichen.

Rufen Sie sich dazu zunächst die betrieblichen Hauptfunktionsbereiche ins Gedächtnis, z. B. Vertrieb, Marketing, Personalwesen etc. etwa 10 Stück. Die brauchen Sie im Detail später wieder für Aufgabe 5.1.

Es muss Ihnen bewusst sein, dass Sie betriebliche Hauptfunktionsbereiche nicht wie mathematische Größen ein für allemal und für jede Organisation eindeutig definieren können. Das sieht bei jeder Organisation etwas anders aus.
- Natürlich hat jeder Hauptfunktionsbereich Anteile, die standardmäßig immer zu ihm dazugehören. Bei einer Auftragsverwaltung etwa die Auftragserfassung, der Versand der Auftragsbestätigungen etc.
- Aber jeder Hauptfunktionsbereich hat Randbereiche, die man auch anderen Hauptfunktionsbereichen zuordnen kann.
Beispiel: Die Fakturierung (Erstellung und Verbuchung von Rechnungen) ist so ein Randbereich, den man entweder der Auftragsverwaltung oder der Finanzbuchhaltung zuordnen kann.

Schauen Sie sich mindestens die folgenden Beziehungen zwischen Hauptfunktionsbereichen an und überlegen Sie, welche Randbereiche in ihrer Zuordnung strittig sein könnten, also entweder der Auftragsverwaltung oder einem benachbarten Hauptfunktionsbereich zugeordnet werden könnten:
Auftragsverwaltung - Marketing
Auftragsverwaltung - Personalwesen
Auftragsverwaltung - Produktion
Auftragsverwaltung - Einkauf

Stellen Sie Ihre Ergebnisse in einer Tabelle mit den drei Spalten Kernbereich (Auftragsverwaltung) - Randbereich - angrenzende betriebliche Hauptfunktion dar, wie in der Aufgabenstellung beschrieben.

In der Praxis tauchen solche Probleme auf, wenn Sie für eine Organisation deren Auftragsverwaltung modellieren sollen. Dann müssen Sie diesen Bereich in Zusammenarbeit mit dieser Organisation erst einmal abgrenzen. Nochmal: es ist nicht mathematisch definiert, was zu einer Auftragsverwaltung alles dazugehört, jede Organisation kann da etwas variieren.

4.2
Hier geht es um Seiteneffekte, also überraschende, unerwünschte Auswirkungen eines Fehlers (auslösendes Ereignis) auf andere Bereiche. Das heißt, irgendwo in einem Hauptfunktionsbereich (hier: Auftragsverwaltung) geht etwas schief.
Die Auswirkungen dieses Fehlers bleiben aber nicht auf einen kleinen Bereich beschränkt, sondern:
An anderer Stelle in der Organisation oder/und außerhalb der Organisation gibt es einen Aufschrei, weil der Fehler in der Auftragsverwaltung auch außerhalb Wirkungen - Seiteneffekte - zeigt.
Stellen Sie mögliche Konsequenzen von fünf solchen auslösenden Ereignissen in der Auftragsverwaltung auf die betreffende Organisation selbst (intern) und deren Geschäftspartner (extern) dar. Verwenden Sie die in der Aufgabenstellung beschriebene fünfspaltige Tabelle.

Zusammengefasst:

4. Problemanalyse und Systembegriff (vernetztes Denken)
4.1 Systemabgrenzung (Graphik und Text):
Kernbereich (Auftragsverwaltung) und Randbereiche erkennen und abgrenzen
Tabelle mit drei Spalten: Kernbereich, Randbereich / Grauzone (überlappend, strittig, konkurrierend), angrenzende betriebliche Hauptfunktion
4.2 Seiteneffekte (überraschend, unerwartet, unbeabsichtigt, unerwünscht) ermitteln, die in anderen Bereichen auftreten können.
Tabelle mit fünf Spalten:
1. fünf nicht planbare auslösende Ereignisse (innerhalb des Kernbereichs Auftragsverwaltung) auf BW-Ebene
2.1 unmittelbare (organisationsinterne) Seiteneffekte: angrenzende betriebl. Hauptfunktion
2.2 unmittelbare (organisationsinterne) Seiteneffekte: Seiteneffekt
3.1 mittelbare (organisationsexterne) Seiteneffekte: Bereich (z. B. bei Kunden, Lieferanten)
3.2 mittelbare (organisationsexterne) Seiteneffekte: Seiteneffekt

4. Problem analysis and the term „system“ (networked thinking)
4.1 system delimitation:
Recognize and delimit kernel area (order management) and marginal areas
Table with three colums: kernel area, marginal area / grey zone (overlapping, competing), neighboring main function area
4.2 Find possible side effects (surprising, unexpected, unintended, undesired) which can occur in other areas.
Table with five columns:
1. five unforeseeable initiating events (within the kernel area order management) on business level
2.1 direct side effects (within the organization): neighboring main function area
2.2 direct side effects (within the organization): side effect
3.1 indirect side effects (outside the organization): area
3.2 indirect side effects (outside the organization): side effect

ÜA 5 Referenzmodelle

Referenzmodelle sind Modelle, die Sie kennen oder in Büchern/Internet finden und die zu einem Modell, das Sie erstellen sollen, analog (ähnlich) sind. Referenzmodelle können Sie analogisch auf ein aktuelles Modellierungsprojekt übertragen.

Eine Modellierung basiert nämlich immer auf zwei Quellen (in komplexerer Form auf S. 133 dargestellt):
1. Auf der Beobachtung eines Organisationsbereichs und Gesprächen mit den zugehörigen Beschäftigten: Brauchen Sie besonders für Organisationsspezifika.
2. Auf der Einbeziehung von Referenzmodellen: Brauchen Sie besonders für die Teile eines Organisationsbereichs, die sich zwischen verschiedenen Organisationen kaum unterscheiden.

Bei Referenzmodellen ist zu beachten:
1. Ein Modell ist nicht von vornherein ein Referenzmodell, sondern wird dadurch zu einem Referenzmodell, dass Sie es als solches verwenden.
Beispielsweise können (Daten-, Datenfluss-, (Geschäfts-)Prozess-)Modelle der Verkaufsseite eines Unternehmens als Referenzmodelle für dessen Einkaufsseite dienen (siehe ÜA).
2. Es gibt Referenzmodelle auf höherer Abstraktionsebene, z.B. das generische Datenmodell auf S. 152, das aus Oberbegriffen (umbrella terms) besteht.
3. Häufig besteht keine 100%ige Analogie zwischen einem Referenzmodell und einem neu zu erstellenden Modell, sondern nur eine Teilanalogie (partial analogy). Aber auch Teilanalogien sind sehr nützlich.

Wir brauchen nur einen kleinen Teil der Folien.
S. 131: Beispiele für analogisches Denken in der Wirtschaftsinformatik
S. 151-152: Analogien bei Datenmodellen
S. 153: Teilanalogien

Natürlich können Sie den Rest der Folien zu Referenzmodellen durcharbeiten, wenn Sie das Thema interessiert. Dann empfehle ich Ihnen aber gleichzeitig den Aufsatz „Analogisches Denken als Erkenntnisstrategie zur Modellbildung in der Wirtschaftsinformatik", zitiert auf S. 155, auf meiner Homepage.

Die ÜA 5 ist umfangreich – wie alle noch folgenden

5.1 Zweck dieser Aufgabe ist, dass Sie eine Grundvorstellung (organisationales Referenzmodell) von drei häufigen Unternehmenstypen im Kopf haben. Das ist wichtig, wenn Sie es als Informatiker mit einem Kundenunternehmen zu tun bekommen, das Sie noch nicht kennen. Dann müssen Sie in etwa wissen, welche betrieblichen Hauptfunktionsbereiche Sie dort antreffen werden, um von den Beschäftigten im Unternehmen ernst genommen zu werden und mit ihnen effektiv kommunizieren zu können.

Also eine Tabelle mit vier Spalten:
- Name des Hauptfunktionsbereichs (schauen Sie in ein modernes BWL-Buch!)
- Produzierendes Unternehmen
- Dienstleister
- Händler

In den Tabellenzeilen stehen die etwa 10 Hauptfunktionsbereiche.

Jede Tabellenzelle enthält eine genauere Beschreibung eines Hauptfunktionsbereichs (darunter liegende Abstraktionsebene).
WICHTIG:
- Aus Ihrer Erfahrung kennen Sie Mischtypen, z.B. den Frisör, der einerseits Dienstleister ist, andererseits auch Handel treibt. Bei dieser Aufgabe geht es nur um die drei Reinformen, nicht um Mischtypen.
- Klären Sie in Ihren Übungsgruppen die Frage, ob und inwieweit ein Dienstleister einen Einkauf und eine Lagerwirtschaft hat.
- Überlegen Sie sich weiterhin: Welche Entsprechung hat der Hauptfunktionsbereich Produktion eines produzierenden Unternehmens bei einem Dienstleister? Mit anderen Worten: In welchem Hauptfunktionsbereich erfolgt die Wertschöpfung bei einem Dienstleister.
- Der Rest ist relativ trivial.

5.2 In diesem Teil geht es um die spiegelbildliche Ähnlichkeit der Einkaufsseite und der Verkaufsseite eines Unternehmens.

5.2.1 Die Datenmodelle von Einkauf und Verkauf
Zeichnen Sie die Strukturen auf S. 152 ab, übersetzen Sie die Bezeichnungen der Entitätstypen ins Deutsche und versuchen Sie, die Modelle im Detail zu verstehen.
Mehr ist nicht zu tun. Damit beschäftigen wir uns im nächsten Sommer in Datenbanken genau.

5.2.2 Die Modelle der Datenflüsse zwischen Einkauf und Lieferant bzw. zwischen Kunde und Verkauf.
Zeichnen Sie die beiden (gleichen!) grafischen Modelle in eines: Zwischen zwei Kreisen verlaufen beschriftete Datenflusspfeile.
Der linke Kreis enthält Einkauf bzw. Kunde, der rechte Lieferant bzw. Verkauf.
Bringen Sie die Datenflusspfeile in eine zeitlich einigermaßen sinnvolle Abfolge beginnend mit
- Angebotsanforderung
- Angebot
- Auftrag
etc.
WICHTIG: Die Datenflusspfeile werden mit Datenflussbezeichnungen beschriftet, z.B. Kundendaten, Auftragsdaten, Rechnungsdaten und nicht mit Funktionsbezeichnungen, z.B. Kundendaten übermitteln, Rechnungsdaten senden.
Auf Konsistenz von 5.2.2 und 5.2.3 achten!

5.2.3 Die Modelle des Einkaufsprozesses und des Verkaufsprozesses
Der Einkaufsprozess in unserem Unternehmen ist verzahnt mit dem Verkaufsprozess bei einem Lieferanten, ebenso wie der Verkaufsprozess in unserem Unternehmen verzahnt ist mit dem Einkaufsprozess bei einem Kunden.
Stellen Sie die beiden verzahnten Prozesse in einem (!) Modell mit zwei Swimlanes (Tabellenspalten) dar.
Die linke Swimlane enthält Einkauf bzw. Kunde, die rechte Lieferant bzw. Verkauf.
Sie wechseln ständig zwischen den beiden Sichten.
Es beginnt mit:
Links: Angebotsanforderung senden
Rechts: Angebotsanforderung erhalten
Rechts: Angebot erstellen
Rechts: Angebot senden
Links: Angebot erhalten

Links: Angebot prüfen
etc.
WICHTIG: Funktionen werden mit Substantiv + Verb bezeichnet.

Zusammengefasst:

5. Referenzmodelle, Analogie
5.1 Entwickeln Sie ein Referenzmodell der Unternehmensstruktur (bspw. aus Organigramm)
für produzierende Unternehmen, Dienstleister, Handel:
zehn Hauptfunktionsbereiche und die darunter liegende Abstraktionsebene
Stellen Sie die Hauptfunktionsbereiche der drei Unternehmenstypen
parallel in einer 4-spaltigen Tabelle dar.
5.2 Stellen Sie die Analogie der Verkaufs- und Einkaufsseite eines Unternehmens dar
(Datensicht, Datenflusssicht, Prozesssicht).
Hinweis zur Prozesssicht: zwei Swimlane-Spalten Einkauf / Kunde und Lieferant / Verkauf.

5. Reference models, analogy
5.1 Design a reference model of the vertical structure of a company (from organization chart)
for producing companies, service companies, trading companies:
ten main function areas and the abstraction level below
Present the main function areas of the three company types in parallel in a table with 4 columns.
5.2 Present the analogy of the sales view and purchase view of a company
(data view, information flow view, process view).
Hint to process view: two swimlane columns purchase / customer and supplier / sales.

ÜA 6 Informationsflussmodelle

Wir kommen zur zweiten Hälfte dieses Kurses, in der wir uns mit der konkreten Modellierung bestimmter Modell-Aspekte und der Wiedergabe/Darstellung dieser Modelle (Modell-Repräsentation, Modell-Darstellung) mithilfe bestimmter Modell-Notationen (graphische Symboliken) beschäftigen werden.

An dieser Stelle beginnen auch die Abschnitte, bei denen ich nicht jedes Detail erklären werde, sondern Sie ganz gezielt zum Selbststudium auffordere. In späteren Semestern, in FWPF, bei der Bachelorarbeit und natürlich im Berufsleben geht es nicht ohne Selbststudium. Je früher Sie beginnen, sich daran zu gewöhnen, umso besser ist es.

S. 156: Modelle sind mentale Größen, die wir als Menschen im Kopf haben. Wenn wir so ein Modell mithilfe einer bestimmten Modell-Notation aufzeichnen, erhalten wir eine Modell-Repräsentation/Darstellung, alltagssprachlich wiederum Modell genannt.
Jede Modell-Notation verwendet Konten (nodes) und Kanten (arcs).
Jede Modell-Repräsentation/Darstellung muss übersichtlich gestaltet sein!!!

Umfangreiche Modell-Repräsentationen sind zu komplex, um von anderen Menschen verstanden zu werden. Deshalb zerlegt (decomposition) man sie in kleinere:
- horizontal gemäß verschiedenen Modellaspekten (S. 157-158) und
- vertikal gemäß verschiedenen Abstraktionsebenen (S. 159)
Alle Teilmodelle müssen untereinander kompatibel sein (S. 160)!!!

In einem Meta-Modell beschreiben wir alle möglichen Grundkomponenten (basic elements) eines bestimmten Modellaspekts, und zwar unabhängig von einer bestimmten Modell-Notation/Symbolik (notation, graphical representation).

Nachdem sich solche Notationen ständig ändern, ist es für mich ein wichtiges Lernziel, dass Sie von speziellen Notationen unabhängig werden und leicht von einer zu einer anderen Notation umzuschalten lernen.
Wir werden zwar Modelltypen der UML behandeln, aber auch sie ist nicht die Spitze aller denkbaren Modellnotationen, sondern eine unter mehreren.

S. 157-158: Übersicht über die vier essentiellen Modellaspekte (egal, ob Sie prozedural oder objektorientiert denken).
Funktionsstrukturmodelle sind trivial: Sie beschreiben nur statisch die Teilfunktionen (Bestandteile) einer Funktion, ohne über deren Ablauf etwas auszusagen (S. 161).
Die drei anderen Modellaspekte werden wir ausführlich behandeln, und zwar in folgender Reihenfolge:
- Informationsflussmodelle
- Datenmodelle (genauer im SoSe in Datenbanken)
- Steuerflussmodelle (so übersetzt man control flow ins Deutsche, und eben nicht Kontrollfluss, wie es teilweise sogar Informatiklehrbücher falsch formulieren).

S. 162-164 Wichtiges zu Informationsflussmodellen und ihren Notationen.

S. 214-221 Beispiele für Informationsflussmodell-Darstellungen

Bemerkungen zur Notation mit Structured Analysis

1. Structured Analysis ist ein Top-Down-Modellierungsansatz, in dem man ausgehend von groben Modellen immer feinere Modelle erstellt (S. 159). Dabei wird immer nur e i n e Funktion auf der nächst niedrigeren Abstraktionsebene in einer eigenständigen Grafik in ihre Teilfunktionen verfeinert (zoom-in).
Dabei können auch externe Schnittstellen und Datenflüsse verfeinert werden.

2. Eine Modell-Darstellung enthält grundsätzlich nur eine Abstraktionsebene.

3. Structured Analysis hat vier Modellkomponenten:
- Funktionen (Kreise, Ovale)
- Speicher (Rechtecke, oben und unten fett, z.B. Datenbank, Papierausdruck)
- externe Schnittstellen (Rechtecke mit einer offenen Seite)
- Datenflusspfeile

4. Structured Analysis hat keine Symbolik für Steuerflüsse.
Nur durch die Anordnung von Funktionssymbolen kann/soll man einen Ablauf andeuten.

5. Funktionen werden mit Substantiv + Verb bezeichnet.
Auf der Ebene der betrieblichen Hauptfunktionsbereiche verwendet man deren Namen ohne Verb.
Funktionen werden immer aus der Sicht der betrachteten Organisation bezeichnet (nicht aus der Sicht von Kunden oder Lieferanten).
In der Wirtschaftsinformatik umfassen Funktionen betriebliche Aufgaben, die von Menschen und/oder IT ausgeführt werden. D.h., Menschen werden als Teile von Funktionen betrachtet. (Anders in der Informatik: Hier sind die Funktionen nur IT-Funktionen, die von außenstehenden Menschen bedient werden.)

6. Datenflusspfeile können nur eine Funktion und einen Speicher (Datenbank) sowie eine Funktion und eine externe Schnittstelle verbinden, in der Regel in Input- und Output-Richtung aus der Sicht der beteiligten Funktion.

7. An Datenflusspfeilen stehen nur Bezeichnungen für Datenflüsse, keine Bezeichnungen für Funktionen!

8. Funktionen werden nicht mit Datenflusspfeilen verbunden. Datenflüsse zwischen Funktionen werden über einen zentralen Datenspeicher (z.B. unternehmensweite Datenbank) abgewickelt.

9. Externe Schnittstellen und Speicher, die mit einer Funktion verbunden sind, dürfen bei der Verfeinerung (zoom-in) der Funktion nicht wegfallen, sondern müssen auf die nächst niedrigere Abstraktionsebene übernommen werden.

10. Die Datenflüsse von und zu externen Schnittstellen laufen in der Realität über die unternehmensweite Datenbank. Man modelliert in Structured Analysis aber direkte Datenflüsse, um sichtbar zu machen, mit welchen Schnittstellen der Fokus der Ebene 0 Daten austauscht.

Zu Übungsaufgabe 6

Ich achte bei der Korrektur dieser ÜA (auch in der Klausur!) genau auf die exakte Kompatibilität zwischen je zwei Abstraktionsebenen!!!

Gehen Sie aus von dem Funktionsbereich, den Sie in 4.1, teilweise auch in 5.2 modelliert haben – Ihr Rahmenthema. Das kann bspw. eine Auftragsverwaltung sein.
Modellieren Sie diesen Funktionsbereich mit der Notation von Structured Analysis.

6.1 Abstraktionsebene 0 (Kontextdiagramm): Betten Sie diesen Funktionsbereich als black box (Kreis in der Mitte des Diagramms) in die Datenflüsse von und zu dessen externen Schnittstellen ein.
Zeichnen Sie auch die Organisationsgrenze ein. Überlegen Sie dazu, dass der betrachtete Funktionsbereich Teil einer Organisation ist (die in der Regel alle typischen betrieblichen Hauptfunktionsbereiche enthält, und nicht nur den von Ihnen betrachteten Funktionsbereich).
Externe Schnittstellen Ihres Funktionsbereichs können organisationsintern (betriebliche Hauptfunktionsbereiche) oder organisationsextern (Geschäftspartner) sein!

6.2 Abstraktionsebene 1: Verfeinern Sie (zoom-in) das Kontextdiagramm aus 6.1:
Verwenden Sie 5 bis 7 Teilfunktionen auf gleicher Abstraktionsebene, eine organisationsweite Datenbank (liegt in der Mitte des Diagramms) und die externen Schnittstellen (liegen am Rand des Diagramms) aus 6.1. Datenflüsse verbinden Teilfunktionen mit der Datenbank und Teilfunktionen mit externen Schnittstellen. Keine direkten Datenflüsse zwischen den Funktionen.
Nummerieren und ordnen Sie die Teilfunktionen (nebeneinander, in einer Diagonale, im Kreis) entsprechend ihrer Reihenfolge in einem fiktiven Geschäftsprozess des in 6.1 beschriebenen Funktionsbereichs.
Zeichnen Sie die Organisationsgrenze und die Grenze des Funktionsbereichs ein.
Achten Sie genau auf Konsistenz zwischen dem Diagramm in 6.1 und dem in 6.2.

6.3 Abstraktionsebene 2: Verfeinern Sie (zoom-in) eine einzige Teilfunktion aus 6.2.
Zeichnen Sie die Organisationsgrenze, die Grenze des Funktionsbereichs und die Grenze der verfeinerten Teilfunktion aus 6.2 ein.
Achten Sie genau auf Konsistenz zwischen dem Diagramm in 6.2 und dem in 6.3.

6.4 Überführen Sie das Diagramm aus 6.3 eins zu eins in ein UML-Anwendungsfalldiagramm.
Mitarbeiter sind Teile der Funktionsbereiche, stehen also innerhalb.
Dazu gibt es keine weiteren Hinweise: Selbststudium! Beispiel S. 221.

Zusammengefasst:

6. Methoden und Konzepte der Funktionsstruktur- / Informationsflussmodellierung
   Informationsflussdiagramm in Structured Analysis für das Rahmenthema
   Wissens-, Daten-, Beleg-(, Material-, Geld-)Flüsse
   sprachliche Unterscheidung zwischen Speichern, Funktionen und Flüssen
   1. Kontextdiagramm (aus 4.1 und 5.2 ableiten)
      organisationsinterne / -externe Schnittstellen
   2. Erste Abstraktionebene:
      Verfeinerung (vollständig!) des Systems (Fokus) aus 1. mit 5-7 Teilfunktionen
   3. Zweite Abstraktionsebene: Verfeinerung *einer* Teilfunktion aus 2. (vgl. Aufgabe 2)

4. Umsetzung von 3. in UML-Anwendungsfalldiagramm mit systeminternen Akteuren

6. Methods and concepts of function structure modeling and information flow modeling
   information flow diagram in Structured Analysis for the general topic
   knowledge, data, paper (, material / goods, money) flows
   textual distinction between stores, functions and flows
   1. Context diagram (to be derived from 4.1 and 5.2)
      interfaces inside and outside of the organization
   2. First abstraction level:
      refinement (complete!) of the system (focus) from 1 with 5 to 7 subfunctions
   3. Second abstraction level: refinement of one single subfunction from 2 (cf. problem 2)
   4. Transformation of 3. into a UML use case diagram with system internal actors

ÜA 7 Statische Objektmodelle

Wir kommen zu einem einfacheren Thema, das Sie schon aus den Kursen zur objektorientierten Programmierung kennen sollten und das die Schnittstelle zu Datenbanken im kommenden SoSe bildet. Termini wie Klasse, Instanz, Methode setze ich also voraus.

Auf S. 157 betrachten wir die linke obere Tabellenzelle.
Statische Objektmodelle, Klassenmodelle, Daten(struktur)modelle, diese Termini laufen alle auf dasselbe hinaus. Man beschreibt die Struktur von unbewegten (statischen) Daten in einem Speicher, einer Datenbank. (Vorige Woche hatten wir im Gegensatz dazu Modelle für in Bewegung befindliche, „fließende" (dynamische) Daten, also Daten- oder Informationsflussmodelle.)

S. 165-170: Metamodelle und Notationen
S. 222-223: stark vereinfachte Beispiele (nicht als Vorlage für die Übungsaufgabe nehmen, da erwarte ich viel mehr von Ihnen!)

Was Sie sich von den Folien hauptsächlich merken müssen, sind die drei Kategorie-Dimensionen der Beziehungen/Assoziationen in statischen Objektmodellen:
- die numerische Klassifikation (Multiplizität, Kardinalität)
- die syntaktische Beschreibung (Primärschlüssel-Fremdschlüssel-Beziehung)
- die semantische Interpretation (einfache, kompositionelle, taxonomische Beziehungen)

Wenn Sie davon noch nicht alles im Detail verstehen, ist es nicht so schlimm, weil wir dieses Thema im kommenden Sommer sehr ausführlich behandeln werden.
Im Moment geht es nur darum, dass Sie genügend wissen, um die Übungsaufgabe korrekt bearbeiten zu können.

Als Notationen zur Modelldarstellung/repräsentation verwendet man vorzugsweise (UML-)Klassendiagramme, Datenstrukturdiagramme, Entity-Relationship-Diagramme. Nur Klassendiagramme sind dafür geeignet, auch Methoden zu modellieren.

Zu Übungsaufgabe 7

Bleiben Sie möglichst bei dem Anwendungsbereich, den Sie in Aufgabe 6 modelliert haben.
Erstellen Sie dazu ein statisches Objektmodell mit der Notation des UML-Klassendiagramms.
Nicht mehr als 8 Klassen, damit das Modell übersichtlich bleibt!
Schauen Sie sich zu den Beziehungsstrukturen nochmal ÜA 5.2.1 an.

Modellieren Sie die Beziehungen zwischen den Klassen unter Angabe der drei oben genannten Kategorie-Dimensionen.
Achten Sie auf eine übersichtliche Anordnung der Klassen.

Ordnen Sie die einzelnen Angaben in jeder einzelnen Klasse folgendermaßen (in dieser Reihenfolge!):
1. Klassenattribute (zwei Stück)
2. Klassenmethoden (zwei Stück)
3. Instanzattribute (nur Schlüsselattribute und die wichtigsten einfachen Attribute)
4. Instanzmethoden (vier Stück)

Trennen Sie diese vier Gruppen durch Querstriche.

Ich korrigiere nur in dieser Weise geordnete Klassendiagramme, denn Klassenmethoden gehören zu Klassenattributen, Instanzmethoden zu Instanzattributen.

Klassenattribute und -methoden kennen Sie aus der Programmierung als „static".
Klassenattribute sind z. B. Anzahl der Instanzen einer Klasse, der Primärschlüsselwert der aktuell aufgerufenen Instanz, der nächste freie Primärschlüsselwert bei einem inkrementellen Schlüssel.
Klassenmethoden sind z. B. Instanz anlegen (Konstruktor), Instanz löschen, Instanz suchen.

Instanzattribute sind diejenigen Attribute, die uns im SoSe laufend begegnen werden, also für einen Kunden z. B. Kd_ID, Kd_Name, Kd_PLZ, Kd_Ort, Kd-Land, Kd_Kreditlimit, Kd_Bonität

Instanzmethoden beziehen sich auf Instanzattribute, z. B. Kd_Name holen (get).
So etwas gibt es teilweise auch in Datenbanken als attached procedures.

Zusammengefasst:

7. Methoden und Konzepte der statischen Objektmodellierung
   UML-Klassendiagramm (aus SA-Diagramm abgeleitet) möglichst 3NF (→ DB)
   mit Fremdschlüsseln und Kardinalitäten/Multiplizitäten an den 1:n-Pfeilen
   Redundanzen beachten: Auftragsbestätigung, Lieferschein, Rechnung
   Referenzdatenmodelle aus 5.2 beachten

7. Methods and concepts of static object modeling
   UML class diagram (derived from SA diagram), if possible 3NF (→ DB)
   with reference keys and cardinalities/multiplicities on the one-to-many arrows
   Note redundancies: order confirmation, delivery note, invoice
   Pay attention to the reference data models from 5.2

ÜA 8 Klassische Prozessmodelle

Für den Rest dieses Kurses werden wir uns mit Steuerfluss-(control flow)-Modellen, Ablaufmodellen, Prozessmodellen, Verhaltensmodellen, behavior models befassen. Alles mehr oder weniger dasselbe.
Auf S. 157 betrachten wir die rechte untere Tabellenzelle.
Das betrifft zunächst die ÜA 8 und 9.
Folien: S. 171-176, 177-198, 224-237
Für besonders Interessierte: S. 199-207, 242-267

Ich werde nicht jedes Notationsdetail erklären, sondern verweise Sie auf Selbststudium.

Wie jede Modelldarstellung mit einem Diagramm bilden auch grafische Darstellungen von Steuerfluss-Modellen immer Netzwerke mit Knoten (nodes) und Kanten (arcs).
In der Wirtschaftsinformatik verwendet man nur activity-on-node-Diagramme: die Symbole für Funktionen und Ereignisse bilden die Knoten. Auf den Kanten liegen die Zustände.
In der reinen Informatik verwendet man auch Activity-on-arc-Diagramme: die Symbole für Zustände bilden die Knoten im Netzwerk, die Funktionen liegen auf den Kanten. Bspw. Zustandsübergangsdiagramme (state transition), Petri-Netze (S. 224, 229).

Im Prinzip handelt es sich bei Prozessmodellen um total triviale Dinge. Erfahrungsgemäß machen jedoch einige Aspekte die größten Schwierigkeiten:

1. Es gibt genau drei verschiedene grundlegende Steuerflussvarianten (Kanten im Diagramm; S. 173):
- die zeitliche Abfolge (Sequenz)
- die ein-, zwei-, mehrseitige bedingte Verzweigung (IF/XOR, CASE)
- die Wiederholung/Iteration (LOOP) als kopfgesteuerte (abweisende, DO WHILE) Schleife bzw. als fußgesteuerte (annehmende, UNTIL) Schleife
Mit diesen drei Modellkomponenten lässt sich jeder mathematisch einigermaßen sinnvolle, wohldefinierte Ablauf darstellen. Den Böhm-Jacopini-Aufsatz (S. 176) dazu sollten Sie alle kennen!
Hinzu kommt zusätzlich noch die parallele Verarbeitung (AND).

2. Steuerflusslinien von unten nach oben, d. h. Rücksprünge und Sprünge (GOTO), sind generell verboten!

3. Bei jeder bedingten Verzweigung, jeder Fallunterscheidung und jeder Schleife gehört zum BEGIN IF/CASE/LOOP ein grafisch eindeutig erkennbarer, expliziter END IF/CASE/LOOP (S. 195, 198).

4. Die Schachtelung von bedingten Verzweigungen, Fallunterscheidungen und/oder Schleifen geschieht nach dem LIFO-Prinzip (last in, first out), d. h. die Steuerfluss-Modellkomponente, die als letzte geöffnet wurde, wird als erste geschlossen (S. 180-188).

5. Die konsistente 1:1-Übersetzung eines gegebenen Ablaufs in verschiedene Diagrammtypen. (Denken Sie an die Konsistenz zwischen verschiedenen Abstraktionsebenen bei Structured Analysis!)

6. Im Symbol für die Abfrage einer bedingten Verzweigung werden keine Operationen außer Vergleichsoperationen durchgeführt. Die Belegung der Variablen, die man dafür braucht, muss vorher erfolgen.

7. UML-Sequenzdiagramme.

Zu 1, 2, 3 und 4: Jedes Grundschulkind kann Abläufe intuitiv modellieren, allerdings im GOTO-Stil, und den müssen Sie als echte Informatiker spätestens jetzt ablegen! Ziel ist, dass Sie lernen, strukturierte (wohlgeformte) Diagramme zu entwerfen!

Wir behandeln folgende Notationen:
- Programmablaufplan (flow chart) S. 225
- Nassi-Shneiderman-Diagramm, S. 225
- Hierarchy plus input process output (HIPO) S. 226
- Jackson-Baum, S. 225, 227 – brauchen Sie später für ÜA 10
- Ereignisgesteuerte Prozesskette (event-driven process chain) S. 228-229
- Aktivitätsdiagramm (UML)
- Sequenzdiagramm (UML) S. 231

Nur am Rande schauen wir uns an:
- Business Process Modeling Notation (BPMN) S. 229-230
- Netzplantechnik (network modeling technique) S. 232-233
- Entscheidungsbäume (decision trees) S. 234-237

Zu Übungsaufgabe 8

Sie legen sich einen Ablauf/Geschäftsprozess zurecht, der zumindest Teile des Diagramms aus 6.2 abdeckt. Dieser Prozess muss mindestens eine bedingte Verzweigung und mindestens eine Schleife enthalten.
Diesen Prozess modellieren Sie konsistent mit mehreren Notationen. Konsistent bedeutet, dass Sie die entworfenen Diagramme durch Austausch der Symbole 1:1 ineinander übersetzen können.

8.1 Programmablaufplan: Die Notationselemente finden Sie auf S. 190-191, 225.
Das alte, unstrukturierte Schleifensymbol mit Rücksprung zum Schleifenanfang dürfen Sie nicht verwenden.

8.2 Struktogramm (Nassi-Shneiderman): Die Notationselemente finden Sie auf S. 190-191, 201.
Diese Notation hat keine Steuerflusslinien und zwingt daher als einziges von vornherein zu einem strukturierten (goto-freien) Modellentwurf.
Bei der Case-Anweisung müssen Sie an der Spitze des auf der Spitze stehenden Dreiecks eine waagrechte Linie ergänzen.

8.3 Jackson-Baum: Die Notationselemente finden Sie auf S. 225. Ein schlechtes Beispiel steht auf S. 227.
Diese Notation ist gewöhnungsbedürftig:
- Man liest sie von links nach rechts. Zur Konsistenzprüfung drehen Sie dieses Diagramm um 90° im Uhrzeigersinn und legen es neben den Programmablaufplan.
- Der Jackson-Baum kann (im Gegensatz zu allen anderen Notationen) in ein und demselben Diagramm mehrere Abstraktionsebenen enthalten. Verfeinert wird von oben nach unten.
- Der Jackson-Baum kennt ursprünglich nur kopfgesteuerte Schleifen. Deshalb muss eine fußgesteuerte Schleife aus Schleifenkörper und gleich anschließend kopfgesteuerter Schleife mit dem gleichen Schleifenkörper zusammengesetzt werden. Wir halten uns nicht daran, sondern vermerken die Schleifenart zusammen mit der Schleifenbedingung neben dem

Schleifen-Sternchen. Unter dem Kasten mit dem Schleifen-Sternchen liegt aufgeschlüsselt der Schleifenkörper.
- Bedingungen werden in einem Kasten formuliert, neben dem der Bedingungs-Kringel steht. Darunter folgt links der Ja-Zweig, rechts der Nein-Zweig mit je einem eigenen Kasten; darunter werden die beiden Zweige aufgeschlüsselt.

8.4 HIPO
Ein HIPO-Diagramm zerfällt in drei Spalten. Links stehen die Inputdaten, rechts die Outputdaten und in der Mitte der Ablaufplan mit den Funktionen.
Alle Inputdaten werden von links mit Pfeilen mit den Funktionen in der Mitte verbunden, die Outputdaten von rechts.
Datenquellen und Datensenken können Digitalspeicher, Papier oder Mensch sein. Symbole: Trommel, Endlospapiersymbol, Smiley oder Strichmännchen (UML use case Diagramm).
Für die Prozessmodellierung in der Mitte können Sie verschiedene Notationen verwenden. Am besten eignen sich Programmablaufplan und Aktivitätsdiagramm, weil man sie vertikal in die Länge ziehen kann und so bei den Datenflüssen zu Funktionen im parallel laufenden Ja- und Nein-Zweig einer bedingten Verzweigung kein Chaos entsteht. Auf S. 226 sehen Sie ein schlechtes Beispiel, weil in der Mitte ein Nassi-Shneiderman-Diagramm verwendet wird.

Zusammengefasst:

8. Methoden und Konzepte der Prozess- / Ablaufmodellierung
   (mindestens eine Schleife und eine Bedingung)
   Als Grundlage für diese Aufgabe dient das SA-Diagramm aus 6.3
   Konsistente Modellierung (8.1 bis 9.6) ein und desselben Ablaufs als
   1. Programmablaufplan
   2. Struktogramm
   3. Jackson-Baum
   4. HIPO-Diagramm

8. Methods and concepts of process modeling
   (at least one loop and one condition)
   starting point is the SA diagram from 6.3
   consistent modeling (8.1 to 9.6) of one and the same process as
   1. flow chart
   2. Nassi-Shneiderman diagram
   3. Jackson tree
   4. HIPO diagram

ÜA 9 Moderne Prozessmodelle

Der theoretische Vorspann ist der gleiche wie für ÜA 8, also weiterhin Prozessmodelle.
Es geht jetzt darum, die vier Teilaufgaben von ÜA 8 in konsistenter (!) Form weiterzuführen (außer Teilaufgabe 9.7). Sie bleiben also bei dem Prozess aus ÜA 8.

Wichtig: Zur Konsistenzprüfung beginnen Sie Ihre Bearbeitung mit dem ggf. korrigierten (!) Programmablaufplan (8.1) aus ÜA 8. Den legen Sie auf jeden Fall an den Anfang Ihrer Lösung von ÜA 9.

Die Teilaufgabennummerierung setzt sich aus ÜA 8 fort. D. h. 9.1 bis 9.4 gibt es nicht, ÜA 9 beginnt mit 9.5.

9.5 Ereignisgesteuerte Prozesskette EPK, event-driven process chain EPC (S. 190-191). Die üblichen Notationselemente für Prozessmodelle werden um das Sechsecksymbol für Ereignisse erweitert.

Wichtig:
1. Der IF heißt in der EPK XOR. Hinter dem Verzweigungssymbol, in dem XOR steht, folgen Ja-Zweig und Nein-Zweig, jeweils beginnend mit einem Ereignissymbol, in dem die jeweilige Bedingung steht. Die Bedingung wird also nicht an/in das XOR-Symbol geschrieben. Als ENDIF führt ein zweites XOR-Verzweigungssymbol die beiden Zweige wieder zusammen. (Die Darstellung auf S. 190 ist zu ungenau)

2. Die EPK hat seltsamerweise keine eigene Notation für Schleifen. Man markiert sie heute mit einer eckigen Klammer um den Schleifenkörper, an der außen neben einem Sternchen die Schleifenbedingung steht (S. 191 in der Spalte des PAP – control flow chart).

3. Es ist nicht nötig, dass zwischen zwei Funktionen immer ein Ereignis steht (das ist die klassische Notationssyntax). Das bläht ein Diagramm nur auf und hat keinen Informationswert. Also lassen Sie inhaltsleere Ereignisse weg. Ereignisse brauchen Sie aber auf jeden Fall für Prozess-Beginn, -Ende und Bedingungen bei Verzweigungen.

4. Vorsicht: Eine Funktion kann niemals einem Ereignis entsprechen!

9.6 UML-Aktivitätsdiagramm (S. 190-191): Hat fast die gleiche Symbolik wie der Programmablaufplan.
Unterschiede:
- ENDIF bzw. ENDCASE: Die vorangehenden Zweige werden in einer Raute zusammengeführt. Der PAP hat hier nur einen Knoten.
- LOOP: Wird mit einer eckigen Klammer um den Schleifenkörper markiert, an der außen neben einem Sternchen die Schleifenbedingung steht. Die abgeschrägten Rechtecke des PAP fallen weg. Also genauso wie bei der EPK.
Die alte unstrukturierte Symbolik mit dem Rücksprung auf S. 191 dürfen Sie nicht verwenden!!!

9.7 UML-Sequenzdiagramm: Dieser Diagrammtyp ist primär auf der Programmierebene einzusetzen, obwohl Sie im Internet auch damit modellierte Geschäftsprozesse finden. Die sehen dann sehr seltsam aus. Das tut man also nicht!

Jetzt können Sie einen anderen Prozess als bisher betrachten (ggf. unter Verwendung Ihres Klassendiagramms aus ÜA 7). Konsistenz mit früheren Teilaufgaben spielt hier keine Rolle mehr.
Ihr Diagramm sollte ein Dialog-Programm darstellen (also Benutzeraufrufe und Benutzereingaben enthalten).
Verwenden Sie auf jeden Fall wieder eine Bedingung und eine Schleife.
Schauen Sie sich das Beispiel auf S. 231 sehr genau an (Kunden neu anlegen, wenn sie in der Datenbank noch nicht vorhanden sind)! Dort fehlt die loop-Bedingung „solange Kunden überprüft werden sollen".

Zum Grundverständnis:
Sie können sich die Idee des Diagramms ungefähr so vorstellen, wie ein EPK-Diagramm, das auf verschiedene vertikale Swimlanes (die Funktionsbereiche repräsentieren) auseinandergezogen wird (S. 228).
Beim Sequenzdiagramm haben Sie keine Swimlanes, sondern eine vertikale Linie („Lebenslinie") für jede Klasse, auf die der Steuerfluss mittels Methodenaufruf zugreift.
Der Steuerfluss wird also auf die angesprochenen Klassen auseinandergezogen.
Manche Autoren verstärken die Lebenslinien, wenn auf eine Klasse zugegriffen wird. Das bringt wenig, und darauf achte ich nicht.

Die Steuerflusslinie, die Sie aus PAP, EPK und Aktivitätsdiagramm kennen, muss genau wie dort ohne Unterbrechung (!!!) durch das Diagramm laufen:

- beginnt bspw. bei der Klasse Main mit einem Methodenaufruf der Klasse X, symbolisiert durch einen horizontalen durchgezogenen Pfeil bis zur Lebenslinie der Klasse X. Am Pfeil steht die aufgerufene Methode und die Übergabeparameter in Klammern
- läuft die Lebenslinie der Klasse X ein kleines Stück nach unten

- springt zu der aufrufenden Klasse (bspw. Main) zurück, symbolisiert durch einen horizontalen gestrichelten Pfeil, an dem die Rückgabe-Parameter stehen
- läuft dort ein kleines Stück nach unten

- die nächste Methode wird aufgerufen usw.

Weiterhin ist wichtig:
1. Main steuert das gesamte Programm. Dieses Prinzip führt auch zu einer wesentlich leichteren Fehlersuche.

2. Der Benutzerdialog sollte vollständig über Main laufen.
Beachten Sie für Ihr Methodenverständnis: Der Benutzer kann einem Programm nicht einfach irgendeinen Befehl geben, sondern das Programm muss den Benutzer zu einer Eingabe auffordern. Im Diagramm ruft Main sozusagen eine Methode des Benutzers auf!!!

3. IFs und LOOPs werden auf „Karteikarten" dargestellt und können geschachtelt werden. Das Prinzip sehen Sie auf S. 180.
Natürlich können Sie auch Methodenaufrufe schachteln, aber nicht zu viele (max. 4) und nur solange das Diagramm übersichtlich bleibt. Sonst wieder zurück zu Main.
Das heißt, es gibt insgesamt drei Modellkomponenten, die Sie schachteln können.

4. Zu einer Methode darf und muss es nur einen einzigen Rücksprung geben.

Zusammengefasst:

9. Methoden und Konzepte der Prozess- und dynamischen Objektmodellierung
   (mindestens eine Schleife und eine Bedingung)
   5. ereignisgesteuerte Prozesskette
   6. UML-Aktivitätsdiagramm
   7. UML-Sequenzdiagramm (unter Verwendung des Klassendiagramms aus 7.)

9. Methods and concepts of process modeling and dynamic object modeling
   (at least one loop and one condition)
   5. event-driven process chain
   6. UML activity diagram
   7. UML sequence diagram (using the class diagram from 7.)

ÜA 10 Gruppenverarbeitung – Break point analysis

Es geht um den eigentlich ganz simplen Gruppenverarbeitungs-Algorithmus, mit dem man statistische Auswertungen mit Zwischensummen auf verschiedenen Ebenen sowie einer Gesamtsumme erstellen kann.

Warum bespreche ich dieses Thema mit Ihnen? Weil dieser Algorithmus für Datenbanken im SoSe wesentlich ist. Wenn man ihn verstanden hat, tut man sich bei SQL an einer Stelle sehr viel leichter: bei Aggregatfunktionen (SUM, AVG, MAX, MIN, COUNT), die auf Gruppen operieren, also auf bestimmten Teilmengen einer vorher definierten Tabelle.
Was kann Standard-SQL: Auswertungen von Aggregatfunktionen auf nur einer einzigen Ebene. Die zugrundeliegenden Daten werden dabei ebenso wenig ausgegeben wie eine Gesamtsumme.

Was kann Gruppenverarbeitung? Deutlich mehr: Grunddaten ausgeben, gleichzeitig Zwischensummen auf mehreren Ebenen und die Gesamtsumme ausgeben. Alles in einer einzigen Anwendung, wofür man in Standard-SQL sehr aufwändig mit zusätzlichen temporären Tabellen arbeiten müsste, wenn es überhaupt ginge.

Gruppenbildung heißt, dass Gruppen, also Daten mit gemeinsamen Merkmalen, zusammengefasst ausgewertet werden. Bei SQL brauchen Sie sich nicht darum zu kümmern, wie der SQL-Interpreter zusammengehörige Daten findet. Bei der Gruppenverarbeitung – bei der Sie ja keine Datenbank-Funktionalität haben –, müssen Sie die Daten eigens nach bestimmten Sortierkriterien vorsortieren, damit ähnliche Daten, die eine Gruppe bilden sollen, in der zugrundeliegenden Datei und beim sequenziellen Lesen unmittelbar hintereinander erscheinen.

Gruppenverarbeitungen gibt es einstufig, zweistufig, dreistufig etc.
Eine n-stufige Gruppenverarbeitung gibt Zwischensummen auf n Ebenen sowie eine Gesamtsumme aus. Sie braucht n+1 Sortierfelder.

Erstes Beispiel – einstufige Gruppenverarbeitung (S. 238):
Aus einer Datei, die Daten zu Kunden-Lieferscheinen enthält, sind Kunden-Sammelrechnungen zu erstellen.
Die Datei enthält vereinfacht nur drei Datenfelder: KdID, LiefDat, LiefWert
Auf jeder Kunden-Sammelrechnung (vereinfachte Form) soll Folgendes erscheinen:
- Kundenüberschrift (KdID) und Kundensumme – Gruppenvorlauf GV und Gruppennachlauf GN
- die einzelnen Lieferschein-Angaben (LiefDat, LiefWert) – Satzverarbeitung

Am Ende soll die Gesamtsumme über alle Kunden-Sammelrechnungen erscheinen.
Kunden-Sammelrechnungen bedeutet, dass jeder einzelne Kunde als eine Gruppe zu behandeln ist.
Die Datei ist also nach KdID und LiefDat zu sortieren, bevor man eine Gruppenverarbeitung darauf anwenden kann.

Der Ablauf einer Gruppenverarbeitung lässt sich am besten mit einem Jackson-Baum darstellen: S. 239. Dort sehen Sie eine zweistufige Gruppenverarbeitung. Eine einstufige erhalten Sie, wenn Sie die unterste Ebene (OGN, OGW, OGV) weglassen.
In der Ebene darüber schreiben Sie GN statt UGN und GV statt UGV.
Die Bedingung lautet: „GW oder 1. Satz oder EOF".

So einen Algorithmus brauchen wir für das Beispiel.

Was die einzelnen Module tun, sehen Sie auf S. 241.

Gruppenwechsel: An welcher Stelle „bemerkt" der Algorithmus, dass eine neue Gruppe beginnt? Wenn er den ersten Satz einer neuen Gruppe liest (nicht, wenn er den letzten Satz der alten Gruppe liest!). Das ist genau die Gruppenwechselbedingung GW, die unmittelbar nach dem Lesen eines Satzes abgefragt werden muss.
Im Beispiel liegt ein Gruppenwechsel vor, wenn die KdID „wechselt".
Bei einem Gruppenwechsel muss zunächst die alte Gruppe abgeschlossen werden (Gruppennachlauf GN) und dann die neue Gruppe begonnen werden (Gruppenvorlauf GV).

Grenzfall 1. Satz: Nach dem Lesen des ersten Satzes ist die Gruppenwechsel-Bedingung erfüllt. Man braucht keinen Gruppennachlauf GN, nur einen Gruppenvorlauf.
Grenzfall EOF: Nach dem Lesen von End-of-File ist die Gruppenwechsel-Bedingung erfüllt. Man braucht nur einen Gruppennachlauf GN, aber keinen Gruppenvorlauf GV und keine Satzverarbeitung.
(Die Kenntnis der Symbole für logisches Nicht und logisches Oder setze ich voraus.)

Zweites Beispiel – zweistufige Gruppenverarbeitung (S. 238, 239):
Aus einer Datei, die Daten zu Kunden-Lieferschein-Positionen enthält, sind wiederum Kunden-Sammelrechnungen zu erstellen.
Die Datei enthält vereinfacht nur vier Datenfelder: KdID, LiefDat, ArtID, LiefPosWert
Auf jeder Kunden-Sammelrechnung (vereinfachte Form) soll Folgendes erscheinen:
- Kundenüberschrift (KdID) und Kundensumme – OGV und OGN
- Lieferscheinüberschrift (LiefDat) und Lieferscheinsumme – UGV und UGN
- die einzelnen Lieferschein-Angaben (ArtID, LiefPosWert) – Satzverarbeitung

Am Ende soll die Gesamtsumme über alle Kunden-Sammelrechnungen erscheinen.
Es gilt:
Obergruppe OG ist der Kunde (KdID), Untergruppe UG der Lieferschein (LiefDat).
Die Datei ist nach KdID, LiefDat und ArtID zu sortieren.

Für die zweistufige Gruppenverarbeitung muss man sich nur zweierlei klarmachen:
- Die Module für Untergruppe UG und Obergruppe OG müssen im Ablauf geschachtelt werden: OGV, UGV, UGN, OGN oder anders geschrieben: UGN, OGN, OGV, UGV.
- Ein Untergruppenwechsel liegt auch dann vor, wenn nur die Obergruppe wechselt. Bspw. zwei verschiedene Kunden haben am gleichen Tag eine Lieferung bekommen; der späteste Liefertag des vorhergehenden Kunden ist der erste Liefertag des nachfolgenden Kunden. Daher die Bedingung „OGW oder UGW".

Jacksonbaum für eine dreistufige Gruppenverarbeitung: S. 240

Nun zur ÜA 10: 10.3 lassen Sie weg und bearbeiten nur 10.1 und 10.2
10.1 Datenfelder: KdID, ArtID, LiefSchNr, PosBetrag
10.2 Datenfelder: KdGrID, KdID, ArtID, LiefSchNr, PosBetrag

Für beide Teilaufgaben:
1. Layout: Sie entwerfen ein mehrspaltiges Layout für die Auswertung. In der ersten Spalte nennen Sie das Modul, das die jeweilige Zeile erzeugt. Dann können Sie gut mit Farben arbeiten.
2. Jackson-Baum mit der Angabe, welches Datenfeld welche Gruppenebene repräsentiert.
3. Modulbeschreibung mit sprechenden Datenfeldnamen.

Zusammengefasst:

10. Gruppenverarbeitungs-Algorithmen (jeweils Layout, Jackson-Baum, Modulbeschreibung)
   1. Zweistufiger Gruppenwechsel (Kunde, Artikel, LieferscheinNr, Positionsbetrag)
      mit Gruppensummenbildung:
      eine Sammelrechnung pro Kunde aus Lieferpositionen ableiten
   2. Wie 1., aber dreistufig, zusätzlich Kundengruppe
   3. Rechnungsdruck mit Übertrag (d.h. eine Rechnung kann mehr als eine Seiten brauchen)
      bestehend aus
      einstufigem Gruppenwechsel und unabhängigem Seitenwechsel
      Input: KundenID, ArtikelID, LieferMenge, LieferWert

10. Breakpoint analysis (layout, Jackson tree, module description)
   1. Two-level breakpoint analysis (customer, item, delivery note number, position amount)
      with calculation of group sums:
      derive a collective invoice per customer from the delivery positions
   2. Like 1., but three-level, customer group in addition
   3. Invoice printing with amount brought forward (that is, an invoice can need
      more than one page) consisting of
      one-level breakpoint analysis and independent paging
      Input: CustomerID, ItemID, quantity of delivered item, amount of delivered item

Zusatzaufgaben

11. Erkenntnisgewinnende Methoden: Mathematisierung, Kommentierung
    Konstruktionsbeschreibungen (Dreiecke): SSS, WSW, SsW mit Voraussetzungen

12. Erkenntnisgewinnende Methoden: Mathematisierung, Kommentierung
    Erläuterung des algebraischen Ansatzes von zehn Textaufgaben
    Bezug Semantik (Aufgabenstellung) – Syntax (Lösung)

13. Modellbildung und Modelldarstellung in der Informatik
    Fachkonzept eines Taschenrechnerprogramms für die 4 Grundrechnungsarten

14. Codierung und Kommentierung
    Erstellung eines speziellen (programmiersprachabhängigen) DV-Konzepts mit
    semantischem Kommentar und aussagekräftigen Variablen- und Prozedurnamen

Übung 1.2

Bei der Umsetzung eines natürlichsprachlichen Textes in die formale Sprache eines mathematischen Modells treten grundsätzlich verschiedene Schwierigkeiten auf.

1. Erläutern Sie diese Situation am Beispiel des folgenden Textes:
Beschreiben Sie die einzelnen Schwierigkeiten anhand von Beispielen.
Eine mathematische Modellierung des Textes ist nicht durchzuführen.

2. Können Sie sich weitere Schwierigkeiten vorstellen, die nicht im folgenden Text, aber in anderen Texten vorkommen können? Nennen Sie solche anhand kurzer Beispiele!

3. Klassifizieren Sie die möglichen Schwierigkeiten aus 1 und 2.
Ergebnis: dreispaltige Tabelle:
Erste Spalte: Kategorie
Zweite Spalte: Beispiel aus dem Kochrezept
Dritte Spalte: Beispiele aus anderen Texten

Fleischbrühe

Zutaten:
2 Pfd. Fleisch, 4-5 l Wasser, 2 Eßl. Salz, 1 Zwiebel, 1 Sträußchen Suppengrüns, 1 Gelbrübe

Wollen wir eine Fleischbrühe kochen, müssen wir uns erst überlegen, worauf wir den größten Wert legen, ob auf eine recht kräftige Brühe oder ob wir nebst einer guten Brühe noch ein saftiges Stückchen Kochfleisch auftragen wollen. Darnach richtet sich auch der Einkauf des zu verwendenden Stückes. Gute Brühen liefern das Wadenstück, sowie das Schwanzstück, dieselben enthalten wenig Fett, schmecken aber darum auch nach dem Kochen trocken. (Siehe Kochen des Ochsenfleisches.) Das gewaschene Fleisch wird nebst den gewaschenen, möglichst klein gehackten Knochen und der vorgeschriebenen Menge kalten Wassers beigestellt, und damit alle Säfte gut herausziehen können, möglichst langsam zum Kochen gebracht. Sobald das Wasser anfängt, Bläschen zu ziehen, würzt man mit Salz, auf der Herdplatte gebräunten Zwiebelscheiben, vorgerichteten Suppengewürzen, wie Petersilie, Lauch, Sellerie, Gelbrübenscheiben, welche mit einem Bindfaden zusammengebunden oder in einer Kapsel zusammengehalten werden. Ferner kommen in die Brühe, die man nach Belieben abschäumen kann, Gemüsestrunke von Blumenkohl, Weißkraut oder Wirsing, Tomaten oder getrocknete Tomatenschalen, Erbsenschoten, Spargelschalen und auch frische oder getrocknete Pilze, wie Champignons, Steinpilze, Eierschwämmchen u. dgl. Der Topf wird nach dem Würzen gut zugedeckt, damit mit dem Dampf keine Extraktstoffe entweichen können (dieselben verleihen nämlich der Brühe gerade ihren Wert), und zum Kochen 3 Stunden auf die Seite des Herdes gezogen. Die Brühe soll stets langsam fortkochen, darf nicht strudeln, sonst bleibt sie nicht klar. Während des Kochens darf auch kein heißes oder kaltes Wasser nachgefüllt werden. Beim Gebrauch wird die Fleischbrühe durch ein Sieb, oder wenn sie recht klar sein soll, durch eine gebrühte Serviette gegossen. Soll die Brühe eine dunklere Farbe erhalten, bräunt man 1/2 Teelöffel Zucker in einem Pfännchen, löscht mit kaltem Wasser ab und gibt nach dem Aufkochen von der gebrannten Zuckerlösung etwas zur Fleischbrühe. Soll die Brühe 1-2 Tage aufbewahrt werden, und hat man keinen Eisschrank zur Verfügung, dann füllt man sie, nachdem sie in kaltem Wasser möglichst schnell zum Erkalten gebracht wurde, in reine Flaschen, verschließt diese mit einem Wattekork und verwahrt sie in einem möglichst kühlen Raum. Die so aufbewahrte Brühe kann bei Gebrauch nebst einem Eiweiß in einen gut glasierten Topf gegeben und auf dem Feuer bis zum Kochen geschlagen werden. Alsdann wird der Topf auf die Seite des Herdes gezogen und nach etwa 10 Minuten durch ein reines Tuch filtriert. Das ausgekochte, trocken schmeckende Fleisch wird zu Frikadellen oder einer derartigen Resteverwertung verarbeitet.

Project Management

What's that?

"Science" of leading projects to success:
deadline, budget, quality

What belongs to project management?

1. External mechanisms: project environment, organizational framework

Budget, time management, decision-making power,
training possibilities, infrastructure

Project team: freedom and independence within certain rules
internal responsibility

2. Internal mechanisms

- Tools for dealing with tasks, which are often diffuse
- Team work
- Team organization
- Leadership skills
- Fine time management
- Process structuring
- Standardization of information flows

Problem of learning project management

Every project is singular.
- The same things are not always right.
- Experiences can only partly be generalized.
- Repeatability / reproducibility are missing.
- Different experience horizons

How do you learn project management?

More practically than theoretically

1. Learning by doing as a project leader
2. Apprenticeship (assistant of a project leader)
3. Playing projects, management games
4. Check lists, books
5. Project reviews of good project leaders (question of honesty)
6. Case studies
7. Everyday planning (current state vs. planned state)

You have to deal with people (soft skills)

1. The first one are you yourself (self-monitoring)

How do you get on with yourself?
How well do you know yourself?

Time management: day, week, holiday, work, studies, job
Budget planning

Initiative to learn and to take responsibility, flexibility
Ability of self-organization

Do you know which learning type you are?
Why do you study at a university (of applied sciences)? Why IS?

2. Then there are other people (monitoring by others)

How do you get on with other people? Others with you?

Ability:
- Assertion, assertiveness
- Oral and written explanations
- Presentation
- Questions
- Understanding
- Check-back
- Self-discipline
- Interest in power
- Difference factual (stick to the facts) – personal

→ Lab groups, working groups

Learning project management has a lot to do with character building

- Long process, evolutionary
- High degree of reflection and self-monitoring
- Conscious treatment of people and knowledge
- Learning from mistakes

Examples for everyday project management

Planning a working day or a working week

- Buffers for unexpected events
- Alternatives; flexible planning
- Time for relaxing
- Time for reflection
  - => Overview of your achievements (feeling of success)
  - => planning the next period
- estimating the daily productive time
- Choose your objectives in a way so that you have a feeling of success every day

=> Learn to estimate time requirements.
=> Choose your objectives precisely.

Disciplines related to project management

- Group dynamics
- Human resources psychology
- Personnel management

Projektmanagement

Was ist das?

"Lehre" von der erfolgreichen Durchführung von Projekten:
Termin, Budget, Qualität

Was gehört dazu?

1. Externe Mechanismen: Projektumfeld, organisatorische Rahmenbedingungen

Budgetierung, Zeitvorgaben, Entscheidungskompetenzen,
Schulungsmöglichkeiten, Infrastruktur

Projektteam: Freiheit und Selbständigkeit im Rahmen gewisser Spielregeln
Eigenverantwortlichkeit

2. Interne Mechanismen

- Handwerkszeug für den Umgang mit Aufgabenstellungen (unscharf)
- Teamarbeit
- Teamorganisation
- Führungskompetenzen
- feines Zeitmanagement
- Ablaufstrukturierung
- Institutionalisierung von Informationsflüssen

Problem der PM- Erlernung

Jedes Projekt ist einmalig.
- Es ist nicht immer das gleiche richtig
- Verallgemeinerbarkeit ist nur teilweise gegeben
- Wiederholbarkeit / Reproduzierbarkeit fehlen
- Verschiedene Erfahrungshorizonte

Wie lernt man Projektmanagement?

mehr praktisch als theoretisch

1. Learning by doing als Projektleiter
2. Lehrberuf (Assistent eines Projektleiters)
3. Spielprojekte, Planspiele
4. Checklisten, Bücher
5. Erfahrungsberichte guter Projektleiter (Frage der Ehrlichkeit)
6. Fallstudien
7. Alltagsplanung (Ist-Soll-Vergleiche)

Sie haben es mit Menschen zu tun (Soft- Skills)

1. Der erste sind Sie selbst (Selbstbeobachtung)

Wie kommen Sie mit sich selbst zurecht?
Wie gut kennen Sie sich selbst?

Zeitplanung: Tag, Woche, Urlaub, Arbeit, Studium, Beruf
Finanzplanung

Lernbereitschaft, Flexibilität, Verantwortungsbereitschaft
Fähigkeit zur Selbstorganisation

Wissen Sie, welcher Lerntypus Sie sind?
Warum studieren Sie? Warum FH? Warum Wirtschaftsinformatik?

2. Dann gibt es auch noch andere (Fremdbeobachtung)

Wie gut kommen Sie mit anderen zurecht bzw. andere mit Ihnen?

Fähigkeit: Durchsetzung
mündliche und schriftliche Erklärungen
Präsentation
Fragen
Verstehen
Rückfragen
Selbstdisziplin
Machtbewusstsein
Unterscheidung sachlich – persönlich

→ Übungsgruppen, Arbeitsgruppen

Projektmanagement lernen hat viel mit Persönlichkeitsbildung zu tun

- langer Prozess, evolutiv
- hoher (Selbst-)Reflexionsgrad
- bewusster Umgang mit Personen und Inhalten
- Lernen aus Fehlern

Beispiele für Alltags-Projektmanagement

Planung eines Arbeitstages oder einer Arbeitswoche

- Pufferzeiten für Unvorhersehbares
- Alternativen, flexible Planung
- Erholungszeiten
- Zeiten für Reflexion
  - => Überblick über das Geleistete (Erfolgserlebnisse)
  - => Planung des nächsten Zeitabschnittes
- Einschätzung der täglichen produktiven Zeit
- Zieldefinition so wählen, dass Sie täglich Erfolgserlebnisse haben

=> Zeitbedarfe einschätzen lernen
=> Zieldefinitionen präzise wählen

Mit Projektmanagement zusammenhängende Gebiete

- Gruppendynamik
- (Arbeits-)Psychologie
- Mitarbeiterführung

Alfred Holl

Principles of IS modeling

1 Motivation 1

1.1 Relation between IT and organizations

Information systems (IS)
– as a science and
– as technical systems
have the task
to systematically optimize information processing
in organizations (non-profit and profit, that is, enterprises),
to support business tasks in organizations.

Organizations, however, are open, dynamic, complex,
socio(-technical) information(-processing) systems.

Therefore, business tasks in organizations are not so formal
that a complete support by IT would be possible:
The organization level is always broader than the IT level.

Technical IS do not possess any technological end in itself.
The purpose is not
to just introduce technology into business environments.
The organization level has a clear priority:
Business IT alignment, IT Governance

There is no need that technology is part of a technical IS:
There are technical IS without IT, e.g. card indexes.

1 Motivation 2

Therefore, we need methods (1.2) to systematically
– produce technical information systems
– design models (systems analysis, requirements engineering, business process modeling, reference modeling, OO analysis)

1.2 Methods in IS

Methods to professionally, systematically and economically produce technical information systems
always include the
– support / management aspect (project management) and
– engineering / scientific / technical aspect (software engineering).

Project management has the task
to manage
– personnel teams, human resources psychology
– time and money
– documentations.

Software engineering has the task
to professionally, systematically and economically produce software systems, such as technical information systems.

In this sense, software engineering means
a lot more than software technology and
a lot more than applying graphic symbol systems, representation syntaxes, such as UML.

1 Motivation 3

1.2 Methods in IS

Methods to professionally, systematically and economically design models are closely related to epistemology.
They are not sufficiently considered
of today's software engineering.

Level	Partly methodic, partly structured	Epistemology-based	Epistemological foundation
Eliciting the current state	Systems analysis, Reverse engineering	(Missing)	Systems theory
Designing the planned state	Business concept modeling	Requirements engineering	Linguistics, psychology, …

(Holl / Maydt, Epistemological foundations of RE, 2007, 48)

2 How to systematically design a technical IS 1

Basically, an information system is some (open) information-processing system.
It can be human / social, technical or socio-technical.

2.1 Principle of key and lock: compatibility of IT and application field 1

We cannot design isolated technical information systems.
They have to be embedded in some organizational environment.

Opposition:
Techn. IS are formal tools and fit only formal application areas.
Organizations are open, dynamic, complex, soci(o-technic)al information-processing systems
which comprise many informal parts (e.g. humans).

It is obvious
that tools have to be adapted to their application areas.
Formal tools, however, cannot be applied
in completely informal application areas.
Therefore, up to some extent,
the organization has to be adapted to the technical IS,
some formalization of the applying organization is inevitable.

The deployment of a technical IS
requires formalization of the application area.
Formalize (straighten) lock before modeling a formal key:
Business process reengineering

A mutual adaptation is necessary.

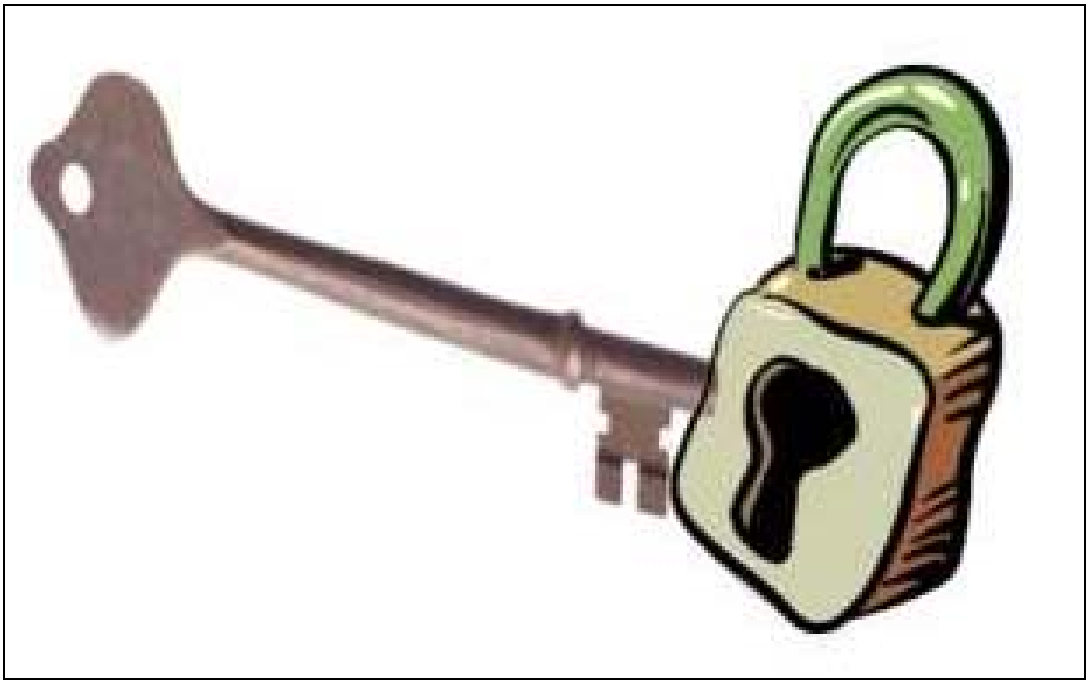
2 How to systematically design a technical IS 2

2.1 Principle of key and lock: compatibility of IT and application field 2

A good technical IS and its application area fit like key and lock in order to produce an efficient socio-technical system:

lock: organization level (social IS)
key: IT level (technical IS)

IT cannot cure the disastrous management of an organization.
A straight key cannot be put into a crooked lock.



Aim in 2.2-2.5:
an optimal technical information system
for a certain organizational environment (social IS).

2 How to systematically design a technical IS 3

2.2 Prescriptive model 1

Prescriptive models design a planned state to be achieved.

**Before we can program a technical information system,
we have to design a technical information system.**

**Before we can design a technical information system,
we have to design its application area (organization).**

**A mere model of a technical information system is not sufficient.
As a technical IS is closely interrelated with its application area,
the application area itself has to be modeled as well.**

The (conceptual) model of the planned state consists of:
**– formal model of the lock (application area, organization:
at least social system, also socio-technical system)**
– formal model of the key (technical information system)

Develop a clear idea of the technical IS's purpose and objectives.

**The design of prescriptive models has to be done
in close cooperation with the domain experts involved
(participative strategies).**

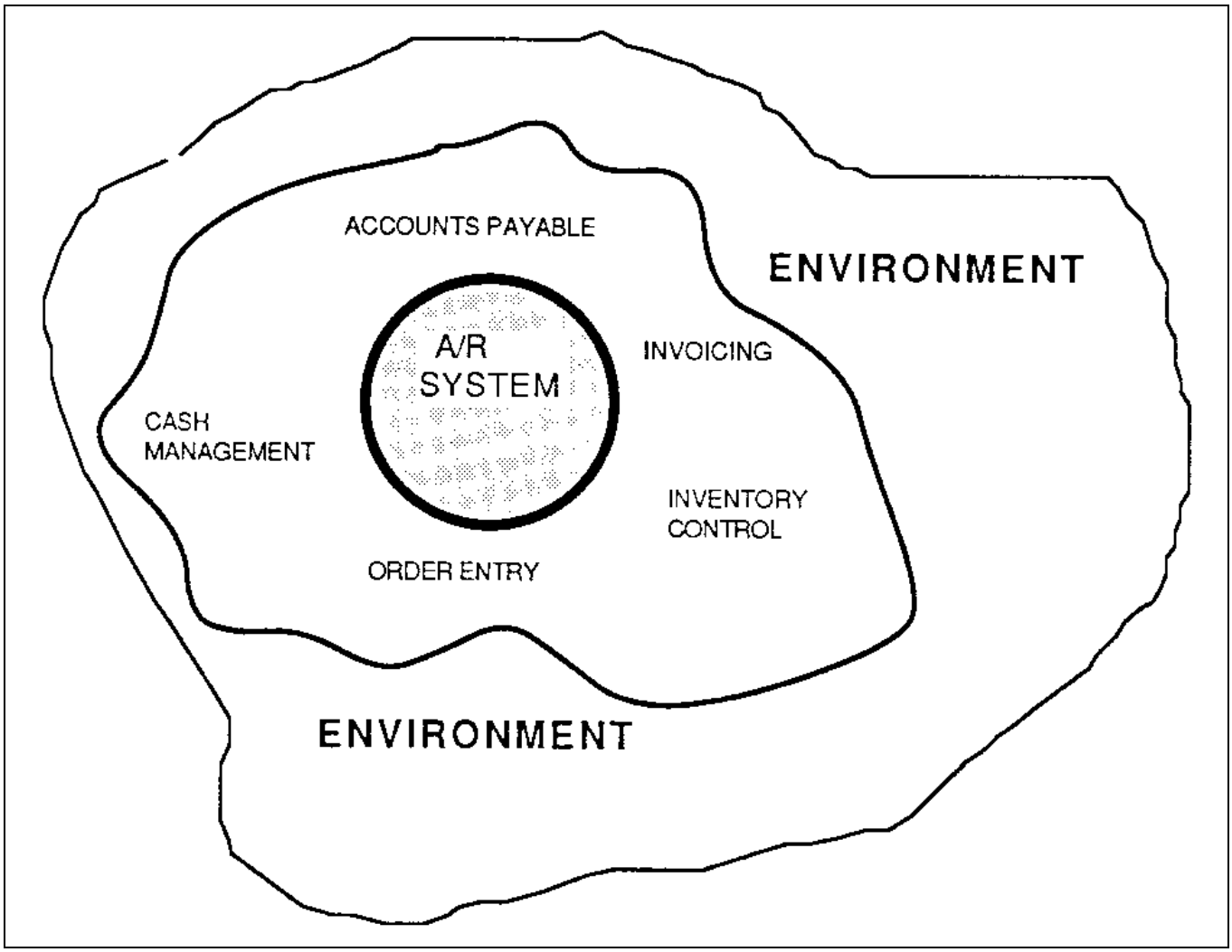
**A good IS expert understands the application areas
of the technical information systems developed by himself so well
that he is able to use them in their application areas.**

**Methods for the design of the concept of the planned state:
requirements engineering on the basis of epistemology**

2 How to systematically design a technical IS 4

2.2 Prescriptive model 2

As organizations / departments are open systems,
always use a magnifying glass model (problem of isolation):
soft, blending system boundary
with precision / magnification decreasing towards the rim.



Accounts receivable system embedded in an enterprise
(Yourdon, Modern structured analysis, 1989, 336)

2 How to systematically design a technical IS 5

2.3 Descriptive model

Descriptive models describe an existing current state.

Before we can design
a formal model of the planned state of the application area,
we must first understand and model its current state and
analyze it with regard to its accessibility to formalization.

Elicitation of the survey of the current state:
describe and model the existing socio-technical system:
– organization level (social IS, lock) and
– IT level (if there already is any technical IS)

Analysis of the current state:
– Is the lock pre-formalized (straight) or not (crooked)?
– How, to what extent can the lock be formalized (straightened)?

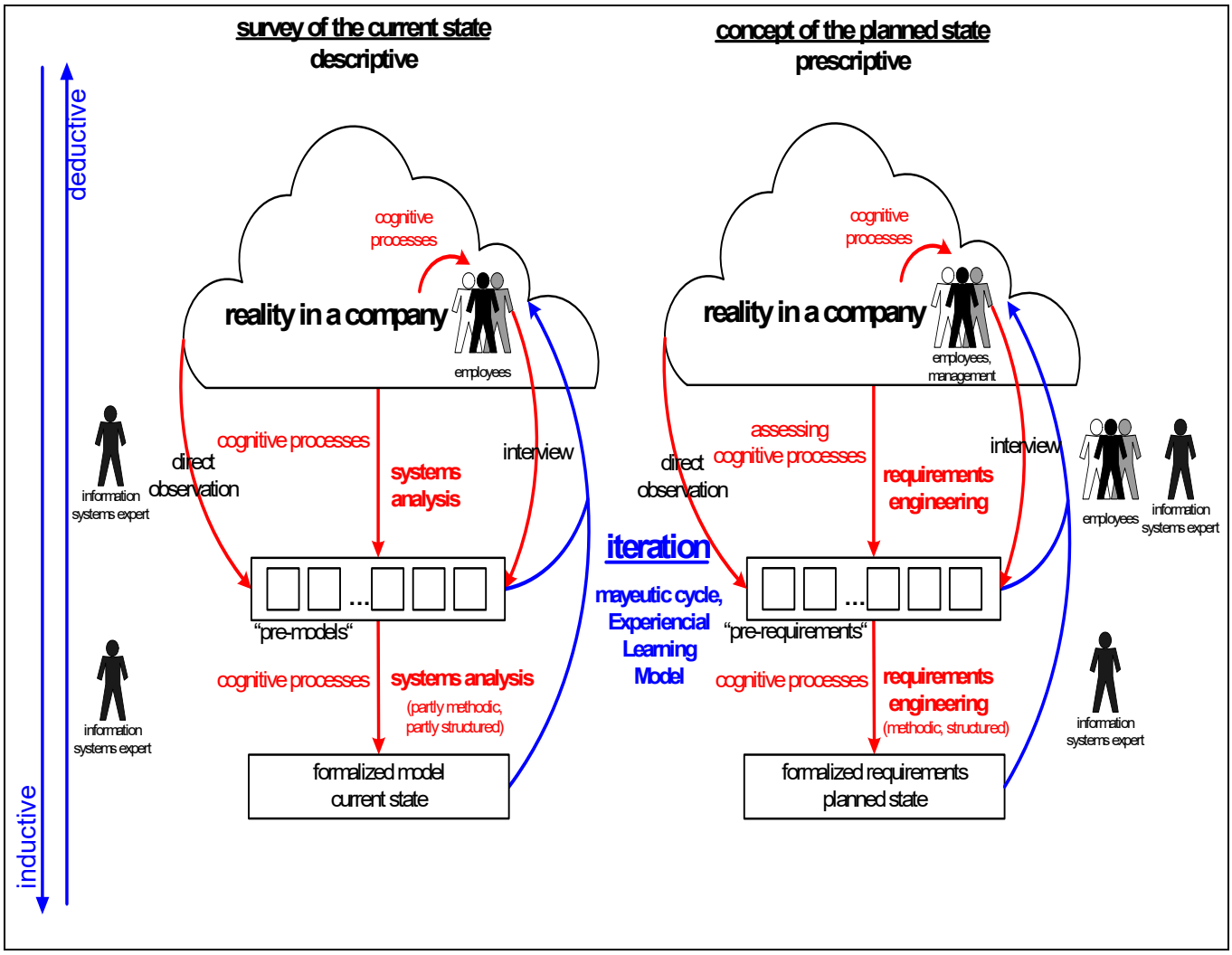
Regarding formalization, application areas of techn. IS differ in:
– pre-formalization
– potential for, accessibility to, suitability for formalization
(cf. deterministic vs. (non-) deterministic, chaotic domains)
– effort of formalization
→ not pre-formalized, scarcely formalizable object domains
→ partly pre-formalized object domains: implicit formal models
→ well pre-formalized object domains: explicit formal models

The design of descriptive models requires participative strategies.

Methods for the elicitation of the survey of the current state:
systems analysis on the basis of epistemology, systems theory

2 How to systematically design a technical IS 6

2.2-2.3 Details of modeling technical IS

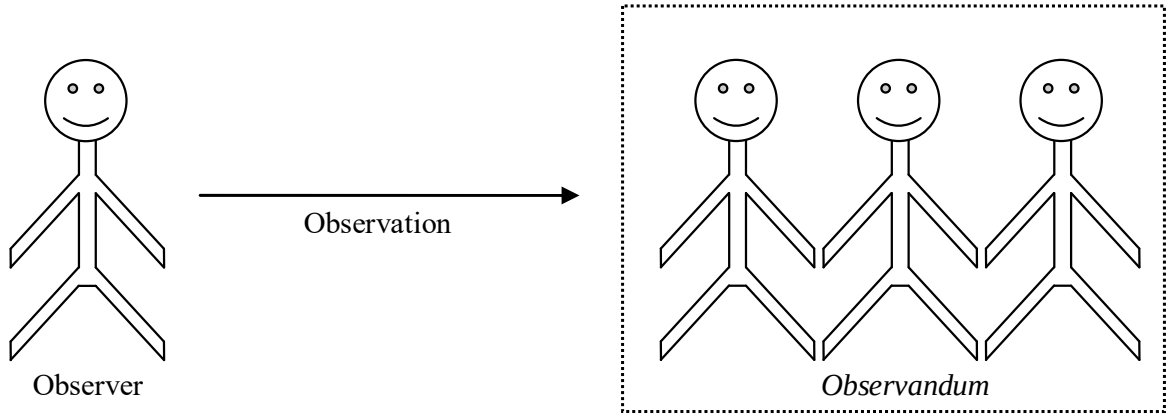


Cognitive processes in modeling technical IS
(Holl / Maydt, Epistemological foundations of RE, 2007, 53)

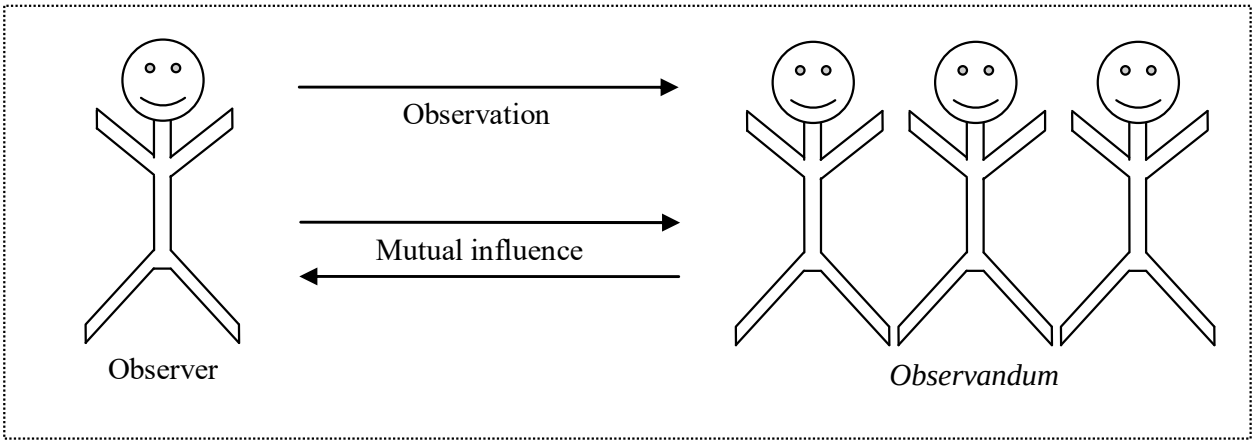
→ more in slides on empirism (mayeutic cycle)
The process of designing models of technical IS
is a lot more complex than demonstrated up until now.
See advanced courses on IS.
E.g., there are different design techniques, such as
top-down, inside out, view integration, umbrella models.
See course on software engineering.

2 How to systematically design a technical IS 7

2.2-2.3 Details of modeling technical IS
Mutual influence observer – observandum



1. Naïve-realistic view: separation of observer and *observandum*



2. Critical-realistic view: mutual influence of observer and *observandum* described as a new system consisting of observer and *observandum*

(Holl / Paetzold / Breun, Cooperative cyclic-iterative knowledge gain in IS anti-aging, 2011, fig. 3)

2 How to systematically design a technical IS 8

2.4 The two parts of the technical IS level

**We now have the necessary prerequisites
to produce a technical IS.**

**The design of the technical IS level (key) consists of two levels:
– information-relevant level ~ business concept
– implementation-relevant level ~ IT concept
cf. phase concept / software process model**

**The business concept has to be modeled
in close cooperation between IS experts and business experts.**

**The IT concept has to be derived from the business concept
by IS experts only.**

2 How to systematically design a technical IS 9

2.5 Phase concepts; software process models; summary of 2.2-2.4

main phase	subphase	model level	model purpose
analytical phase: problem analysis	elicitation of the current state of the soc-tech IS	information-relevant models	descriptive models (systems analysis, reverse engineering)
	analysis of the current state of the soc-tech IS		
	design of the planned state of the social IS (LOCK)		prescriptive models (require-ments engineering)
	design of the planned state (business concept) of the technical IS (KEY)		
synthetical phase: IT system development	design of the IT concept of the technical IS	implementa-tion-relevant models	
	programming		
	test		
	use	information-relevant models	
maintenance			

2 How to systematically design a technical IS 10

2.5 Unprofessional work without software process models

About German public projects going wrong:
Mertens, Peter: Schwierigkeiten mit IT-Projekten in der öffentlichen Verwaltung.
Informatik-Spektrum 35(2012) 433-446

Toll Collect (motorways)	2 years late
Health insurance card	permanent changes and delays
Administration of unemployed people data	hopeless
ELENA payroll data transfer	cancelled (data privacy)
eBalance: XBRL-based company profit data transfer	3 years late
Assignment of study places	at least 3 years late
Berlin airport	at least 3 years late

2 How to systematically design a technical IS

2.5 Software process models and information systems architecture concepts

Diaphasic multi-perspectivity

**On its way through a systematic phase concept –
through a software (development) process model,
a model of a technical IS has to be transferred in several steps
via different models,
each of which in turn is split vertically and horizontally,
from an organization / enterprise model on the information level
to a technical model on the implementation level.**

Every software process phase represents a certain perspective.

2 How to systematically design a technical IS

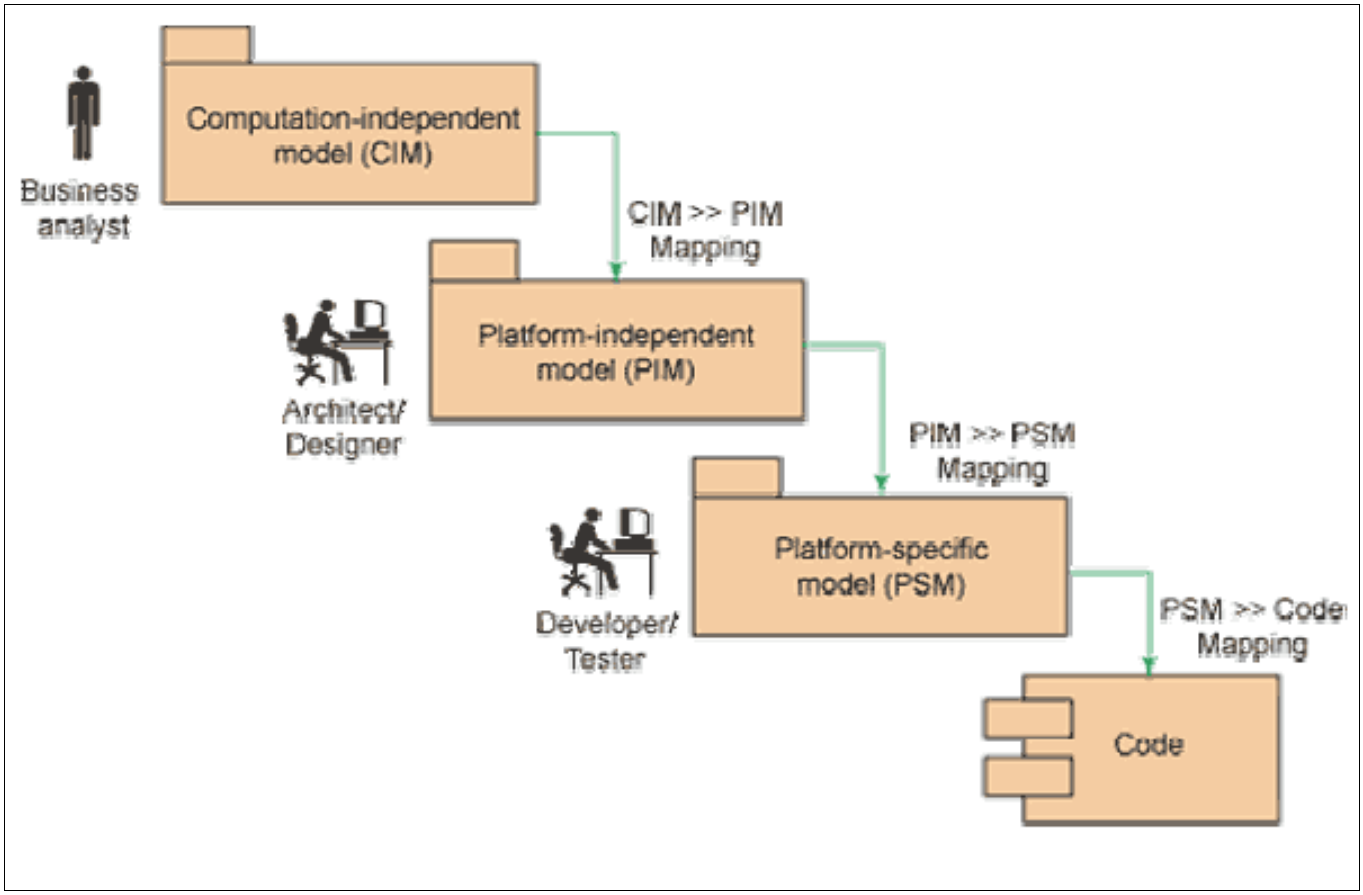
2.5 Software process models and information systems architecture concepts

Subphase	ARIS	MDA (Model Driven Architecture) by OMG (Object Modeling Group)
Design of the planned state / business concept	Business concept	Computation-independent model (CIM)
Design of the technical concept indep. of tools	IT concept	Platform-independent model (PIM)
Design of the technical concept dep. on tools	Implementation	Platform-specific model (PSM)
Examples	Siemens Systemhaus	Siemens Amberg (M^3 by MID Nürnberg) Application-specific model (ASM)
Tools	ARIS Toolset	Innovator (MID)

2 How to systematically design a technical IS

2.5 Software process models and information systems architecture concepts

Model-Driven Architecture (MDA)
by Object Management Group (OMG)

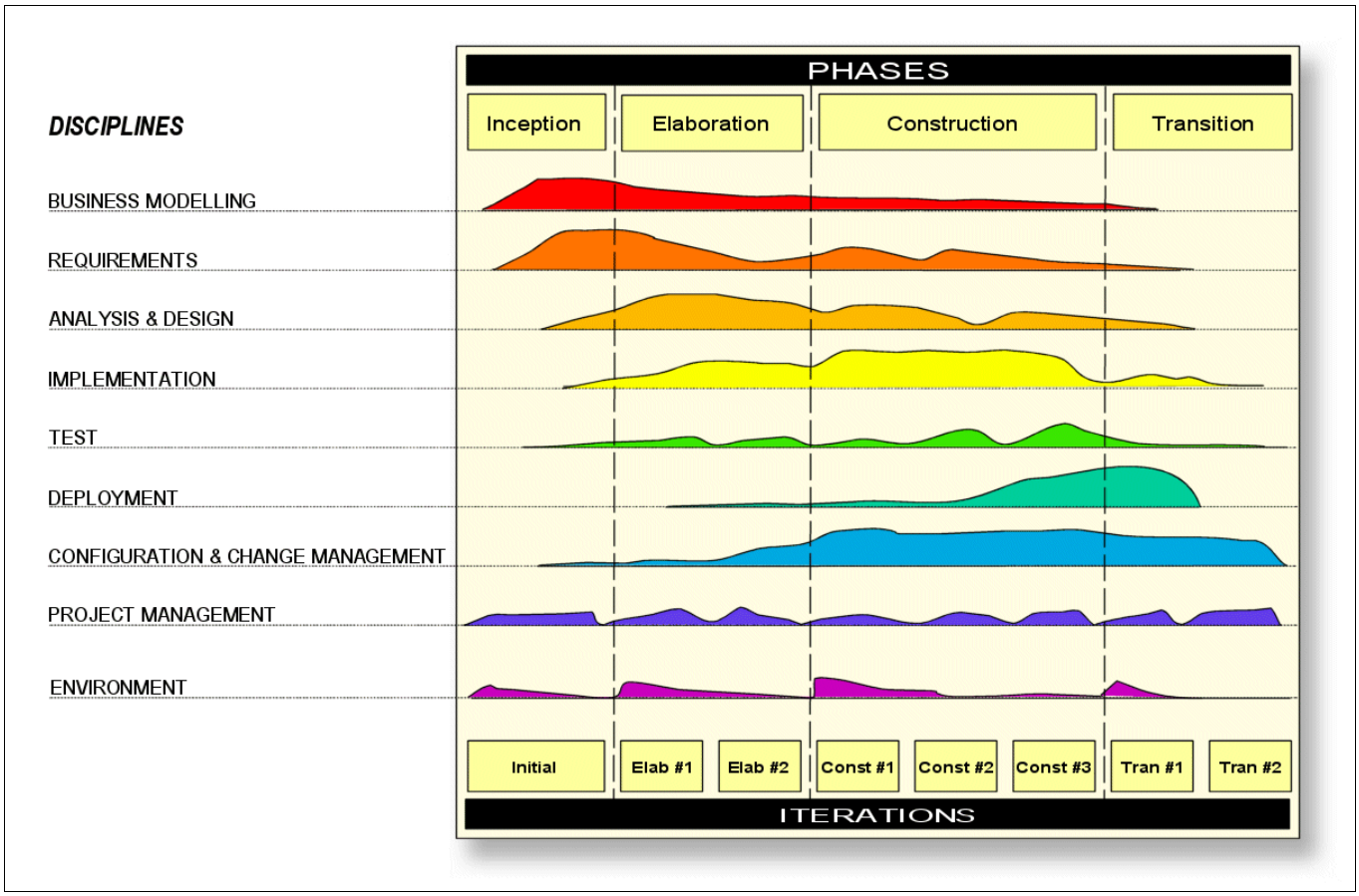


(Journal of Object Technology 2006, http://www.jot.fm/issues/issue_2006_03/column4/images/figure3.gif)

2 How to systematically design a technical IS

2.5 Software process models

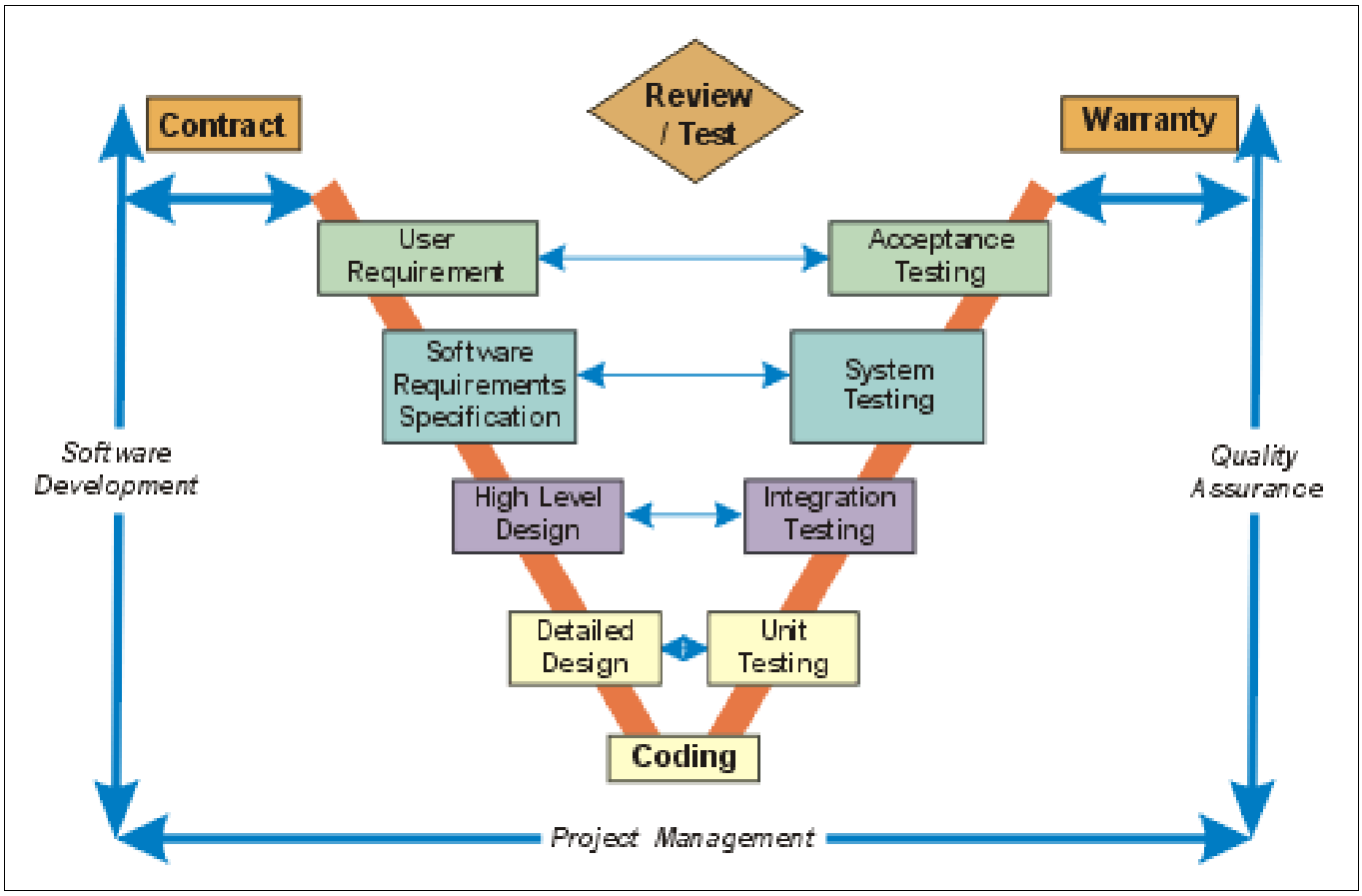
Rational Unified Process (RUP)
by Grady Booch, Ivar Jacobsson, Rumbaugh



2 How to systematically design a technical IS

2.5 Software process models

V-Model

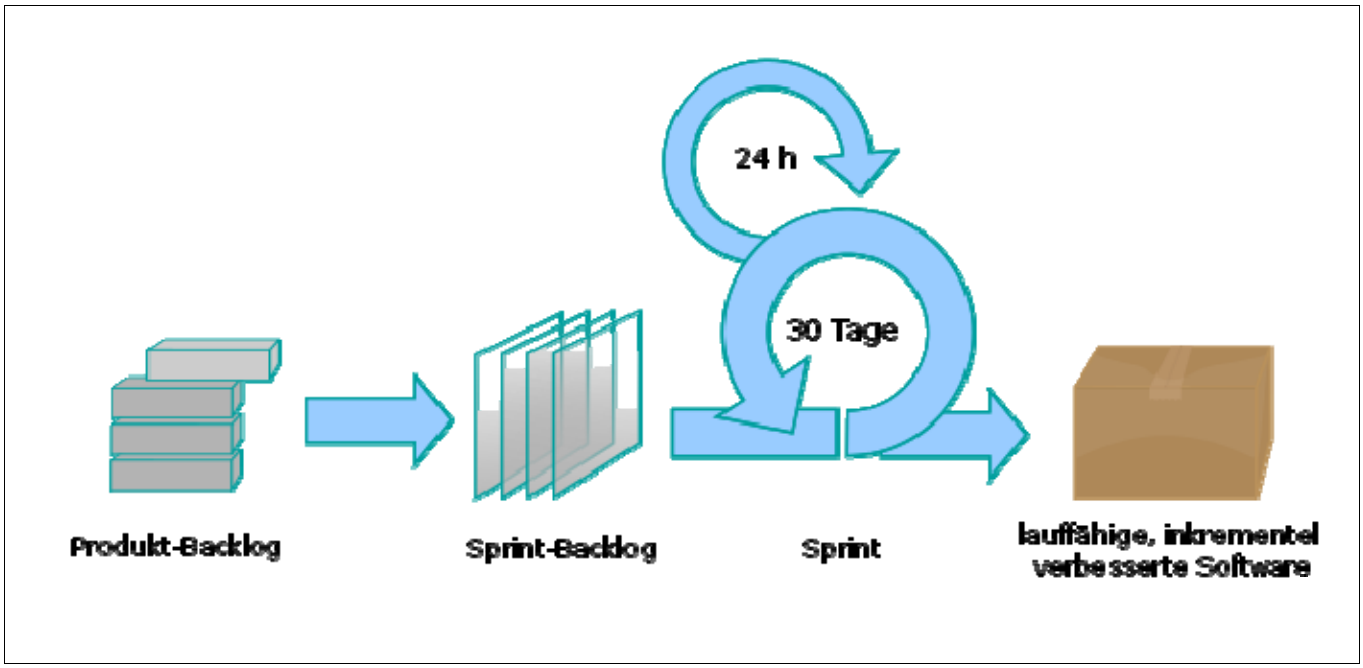


2 How to systematically design a technical IS

2.5 Software process models

Scrum process, agile software development

kleine Schritte: **sprints**
Zwischenergebnisse: **product increments**
Langfristplan: **product backlog**
Kurzfristplan: **sprint backlog**

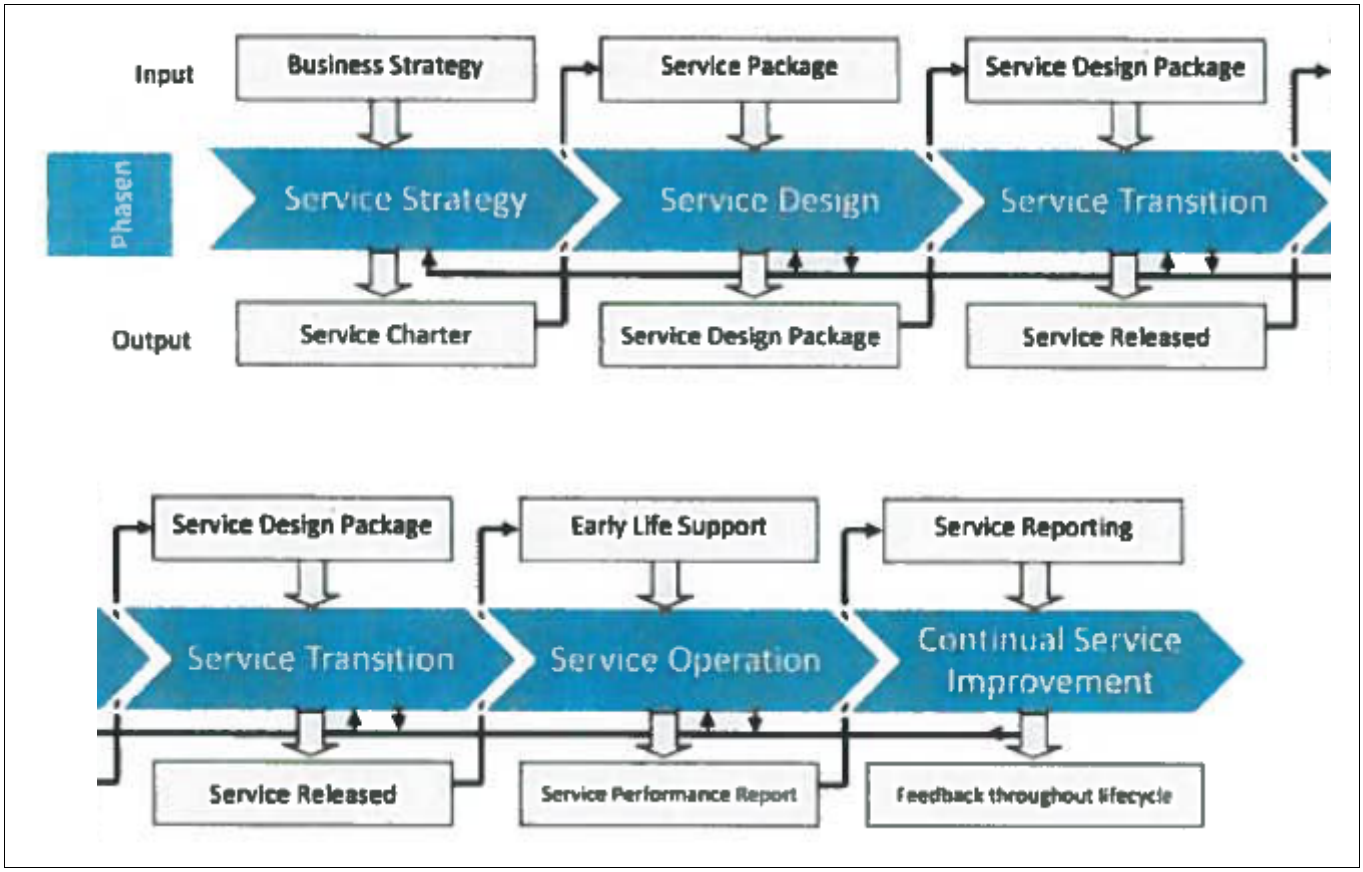


(wikimedia commons)

2 How to systematically design a technical IS

2.5 Software process models

ITIL (infrastructure library) Best Practices
für ITSM (service mgmt.)



(Günther 2012)

2 How to systematically design a technical IS

2.5 Phase concepts; model purposes and levels

Phase concepts or software life cycle models are process models for the professional, systematical development of software in well-defined steps (phases).

1 Model purposes:

– descriptive: models of ...
describe and analyze a current state: segment of reality
– prescriptive: models for ...
design a planned state:
application area in an organization and technical IS
present the desired information processing (test models)

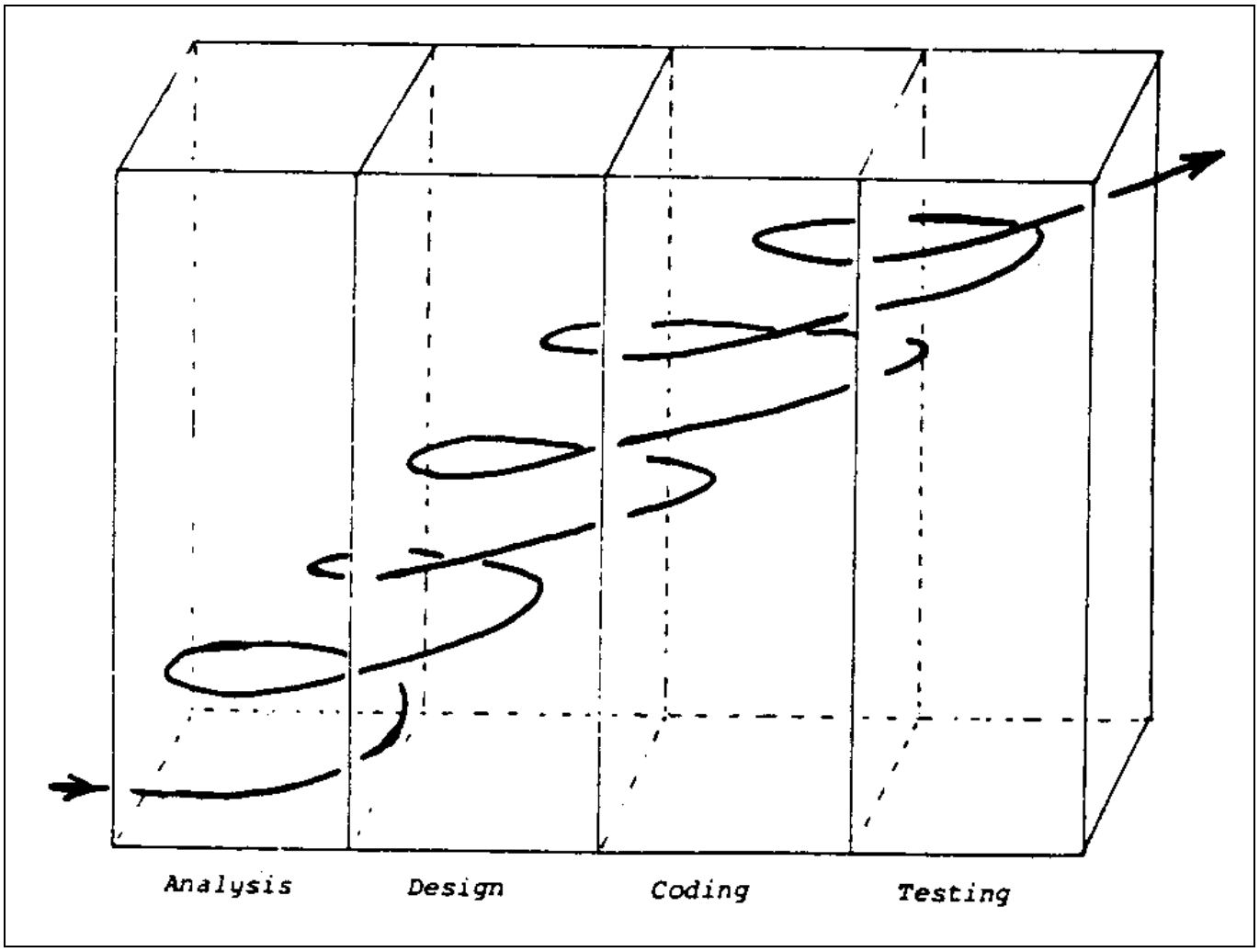
2 Model levels
corresponding to main phases in a software life cycle:

– models irrespective of IT details:
analytical phase,
information-relevant level:
aims at the detailed understanding of a problem and the construction of a conceptual model

– models with respect to IT details:
synthetical phase,
implementation-relevant level:
aims at the construction of an IT system

2 How to systematically design a technical IS

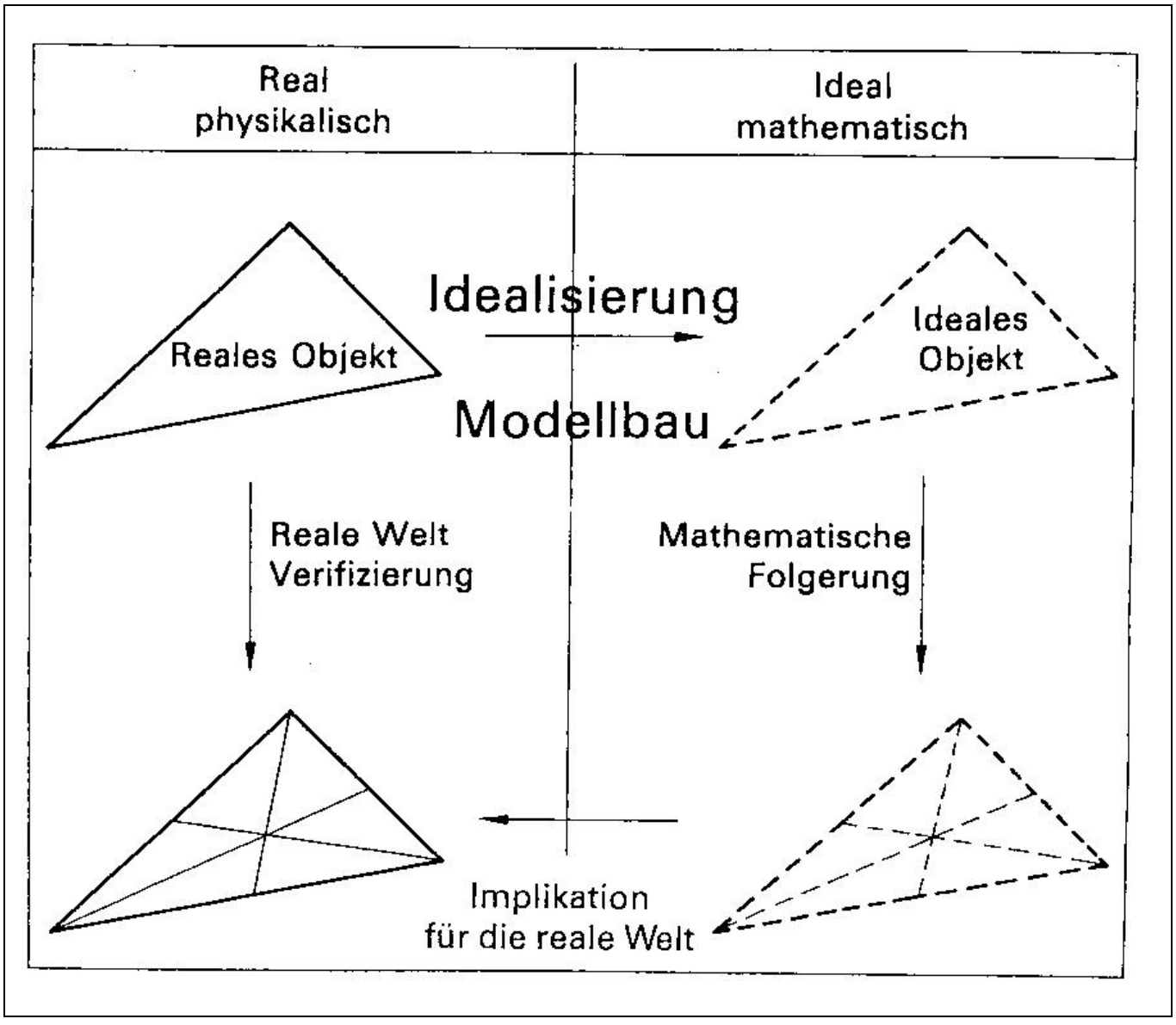
2.5 Phase concepts – simplification 1



Phase concept with loops, but without maintenance
(Hofstetter, SW-Entwicklung und human factor, *, 50)**

2 How to systematically design a technical IS

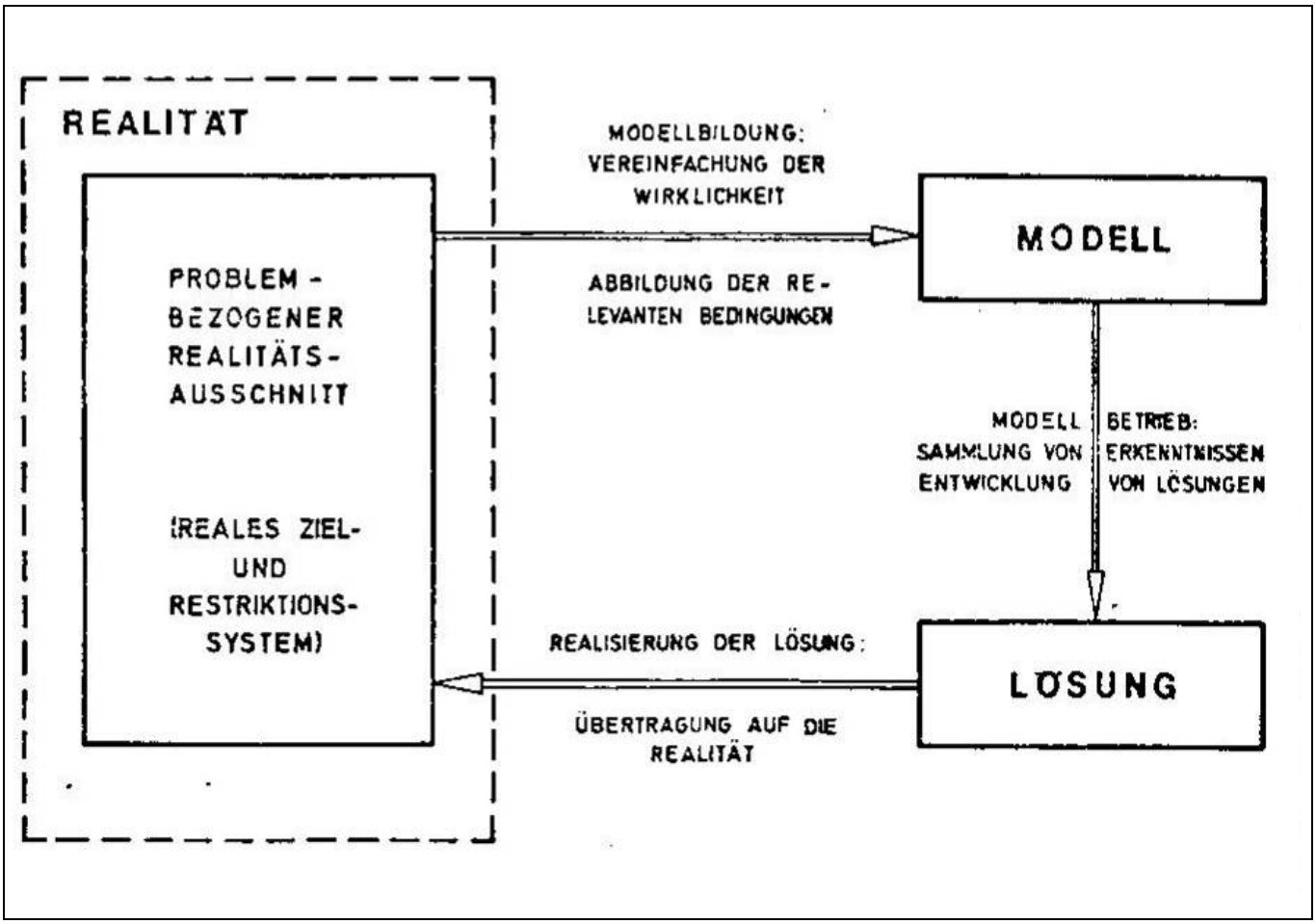
2.5 Phase concepts – simplification 2



Simple mayeutic cycle
(Davis / Hersh: Mathematical experience, 1994 [1981], 131)

2 How to systematically design a technical IS

2.5 Phase concepts – simplification 3



Simple mayeutic cycle
(reference ?)

2.6 How to change a technical IS – IS anti-aging, changed requirements management

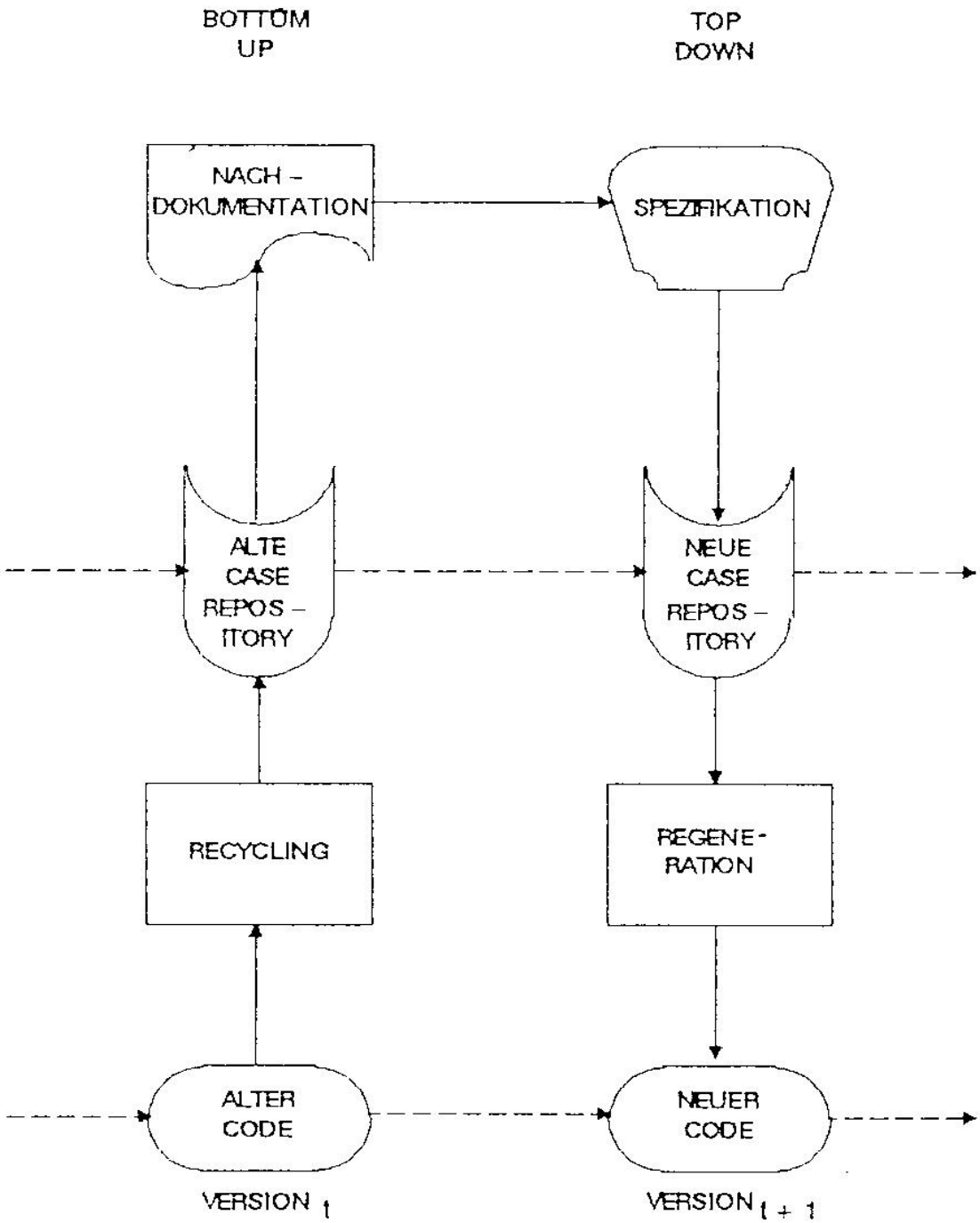


Abb. 2.10: Bottom-Up Software-Entwicklung

Bottom-up software development
(Sneed, Harry M.: Software maintenance, 1991, fig. 2.10)

3 Two sources for model construction 1

external world ↓ World 1 objects of cognition	phenomenon, individual experience ↓ World 2 knowledge of an individual subject of cognition			model, theory ↓ World 3 common knowledge
	perception, cognitive processes (empiristic) ↓ reconstruct. of World 1 →	memory	learning rationalistic ↓ activations of World 3 ←	
←	↓ creation, induction ↓ design, influence ←	← new ideas, knowledge →	→ publi-cation	→

Bi/trilateral linguistic sign		
form, materialized signifiant	code of interpretation	meaning, signifié, W2|W3
object of cog.		

Model as complex bi/trilateral linguistic sign		
materialized model repre-sentation	code of interpretation	model meaning W2|W3
object of cog.		

3 Two sources for model construction 2

Popper's World 1 (reality): empiristic method/approach
Organization, enterprise, department
observation and interviews (W3)
of domain experts by a model designer
(contrary to natural sciences: only observation)
preliminary description in pre-formal models: natural language
abstraction
check whether terminology is mathematically well-defined
final type construction
formalization (degree of pre-formalization is different)
reduction to axioms

often used for peripheral areas of models
often used for individual parts of an organization
(nominalist point of view: enumeration of individual objects)

Popper's World 3 (models, concepts, ideas): rationalistic method
reference models
activation in a model designer's brain
analogy-based transfer

often used for central areas of models
often used for standard parts of an organization, e.g. accounting
(universalist point of view: search for general principles)

Final step: integration of individual and reference models.

4 Empirism – Rationalism

Historically, there are
two different critieria for settling the truth of statements:
– naive empirism: experience and induction
– naive rationalism: reason and deduction

natural sciences	humanities
empirism **Aristotle (384-322)** **John Locke (1632-1704)** **David Hume (1711-1776)** **John Stuart Mill (1806-1873)**	**rationalism** **Socrates (470-399)** **Platon (427-347)** **René Descartes (1596-1650 Sth)** **Baruch Spinoza (1632-1677)** **G. W. Leibniz (1646-1716)**
Immanuel Kant (1724-1804): **– synthesis of empirism and rationalism,** **– transcendental epistemology** **Konrad Lorenz (1903-1989):** **– evolutionary epistemology**	

The mayeutic cycle contains empiristic and rationalistic parts,
that is, observations and theories mutually influence each other:
– Observations (experiences) change observation frameworks.
– Observation frameworks (intellect) exert an influence on
the selection of observation objects and
on observation interpretations.

5 References

pdf-files of my own publications: see my homepage.

Holl, Alfred:
Empirische Wirtschaftsinformatik und evolutionäre Erkenntnistheorie [Information systems as an empirical science and evolutionary epistemology].
In: Becker, Jörg et al. (ed.): *Wirtschaftsinformatik und Wissenschaftstheorie. Bestandsaufnahme und Perspektiven*.
Wiesbaden: Gabler 1999, 163-207, ISBN 3-409-12002-5.
English translation on my homepage.

Holl, Alfred; Krach, Thomas:
Ubiquitäre IT – ubiquitärer naiver Realismus [Ubiquitous IT – ubiquitous naive realism].
In: Britzelmaier, Bernd et al. (ed.): *Der Mensch im Netz. Ubiquitous Computing. - 4. Liechtensteinisches Wirtschaftsinformatik-Symposium an der FH Liechtenstein*.
Stuttgart: Teubner 2002, 53-69, ISBN 3-519-00375-9.

Holl, Alfred; Maydt, Dominique:
Epistemological foundations of requirements engineering.
In: Erkollar, Alptekin (ed.): *Enterprise and business management. A handbook for educators, consulters and practitioners*.
Marburg: Tectum 2007, 31-58;
short version = contribution to:
***Requirements Days 2006*, Nuremberg/Germany.**

„Kurzfristig mehr - langfristig weniger“

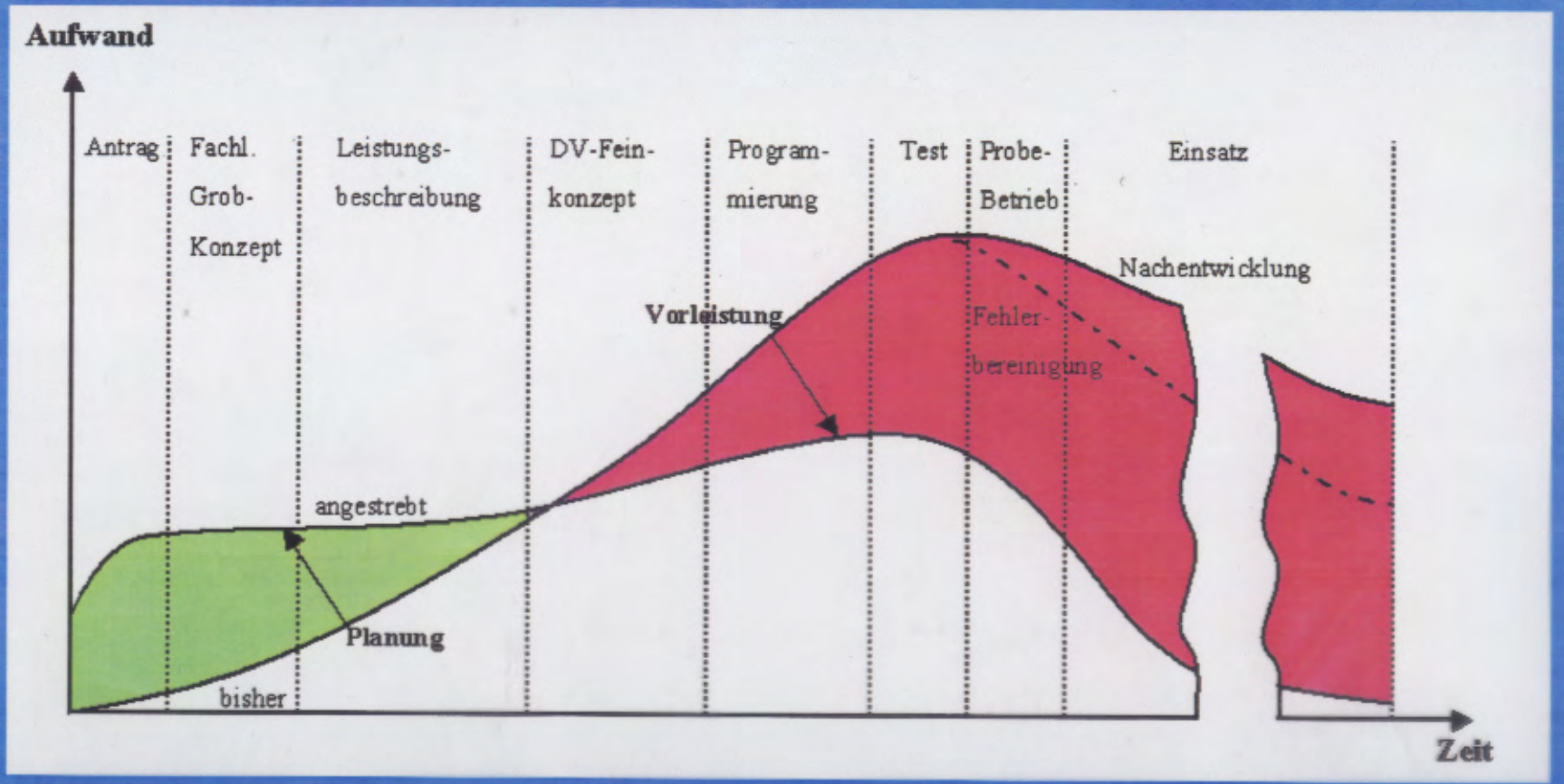
Die langfristige Rentabilität hoher Anfangsinvestitionen in die IT-Einführung oder Umstellung

Prof. Dr. Alfred Holl

„Kurzfristig mehr - langfristig weniger“

1. Kostenexplosion durch mangelnde Softwarequalität
   Die Kostensituation im Vergleich: Niedrige Anfangsinvestitionen führen zu hohen Folgekosten
2. Kostenreduktion durch erhöhte Softwarequalität
   Was ist Softwarequalität ?
3. Erhöhte Softwarequalität durch zeitliche Betonung der Problemanalyse
   a) Die Problemanalyse steht am Anfang der Entwicklung
   b) Die Analyse muß 55% der Zeit in Anspruch nehmen, die Implementierung dagegen nur 15%
4. Sinn und Zweck der Problemanalyse
   a) Steht das Nachdenken am Beginn, bleiben böse Überraschungen aus
   b) Um die Schwierigkeiten der Tätigkeiten während eine Systemanalyse zu meistern, benötigen Sie den Informatiker
   c) Am Ende steht weder ein Programm, noch Papierwust, sondern ein Konzept
5. Vorgehensweisen zur Problemanalyse
6. Sie benötigen die Problemanalyse
   a) Sie verschafft Ihnen einen klaren Überblick
   b) Sie bewahrt Sie vor der Kostenexplosion

Schematischer Aufwandsverlauf



Presse

Quelle: COMPUTERZEITUNG Nr. 35

Gesundheitswesen

IT-Projekte scheitern

Aufgrund unvollständiger Anforderungsprofile scheitern nach Analysen von First Consulting 84 Prozent der IT-Projekte im Gesundheitswesen.

„Projektziele müssen deutlicher definiert werden", fordert Frank Müller, Geschäftsführer der First Consulting Group. Sein Unternehmen hat die IT-Implementierung im Gesundheitswesen untersucht und ist zu erschreckenden Ergebnissen gekommen: So wurden nur 16% der Projekte erfolgreich abgeschlossen. Nahezu ein Drittel wurde völlig eingestellt, und mehr als die Hälfte kostete doppelt soviel wie veranschlagt. Neben einem unvollständigen Anforderungsprofil sind laut Müller die fehlende Unterstützung von seiten des Auftraggebers und mangelnde technische Kompetenz der Anbieter dafür verantwortlich.

Kostenbeispiel

Kosten richtig

Problemanalyse **35.000,-**

Implementierung **35.000,-**

Wartung **15.000,-**

Fehlerbehebung **15.000,-**

100.000,- DM

Kosten falsch

Problemanalyse **10.000,-**

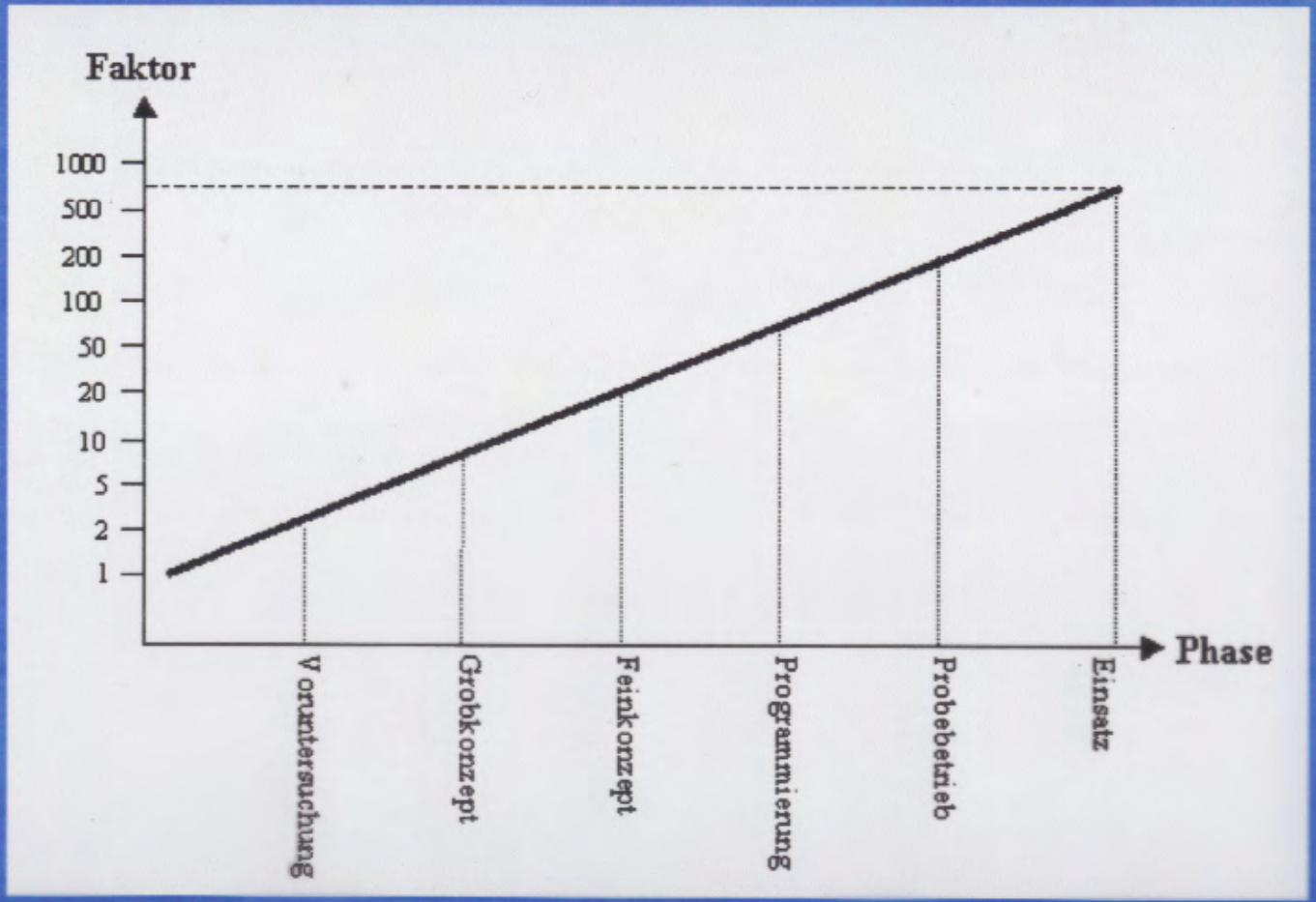
Implementierung **70.000,-**

Wartung **60.000,-**

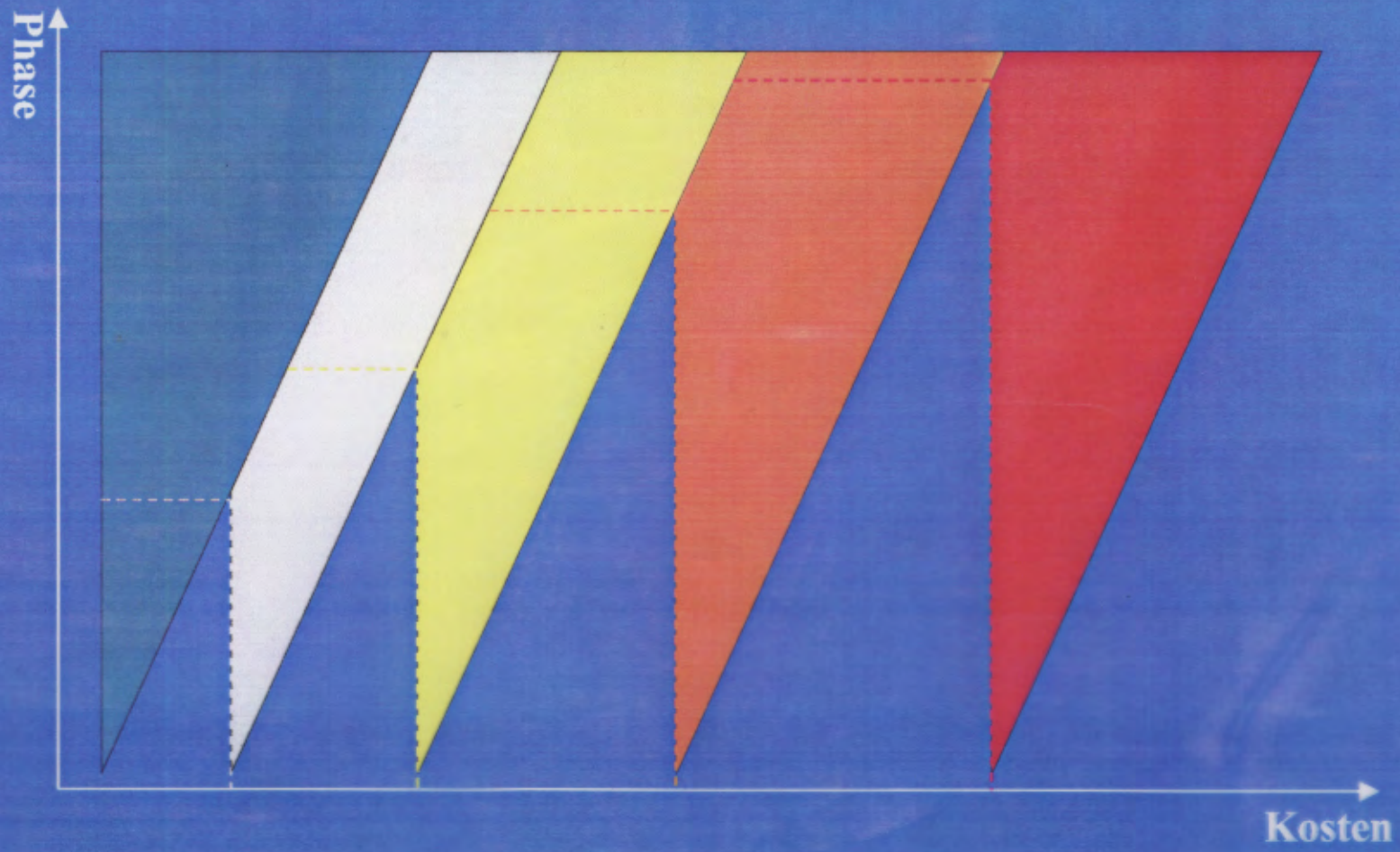
Fehlerbehebung **60.000,-**

200.000,- DM

Kostenentwicklung bei der Fehlerbehebung



Darum werden spät erkannte Kosten teuer



„Kurzfristig mehr - langfristig weniger“

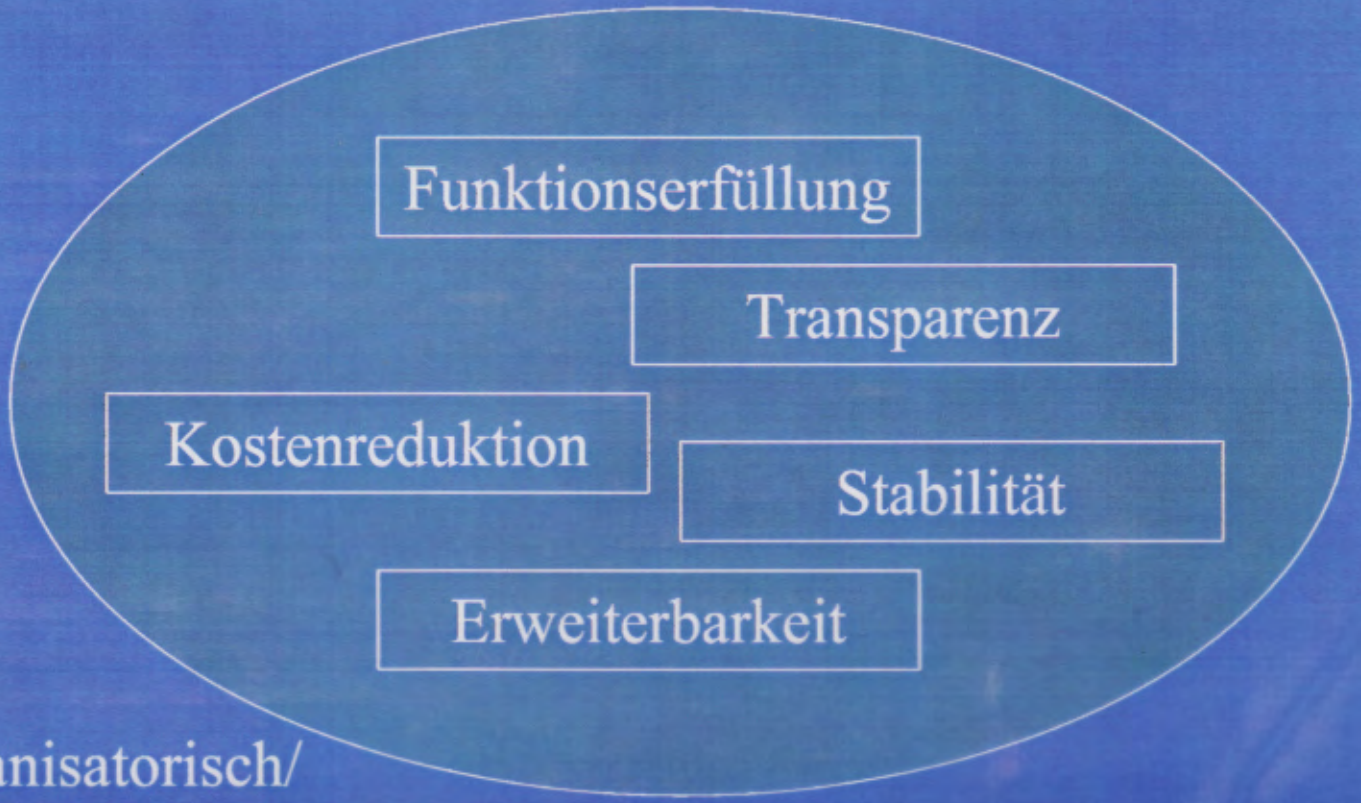
Relativ niedrige Kosten

Relativ hohe Software-Qualität

Qualitätsmerkmale aus Benutzersicht (Sachbearbeiter)

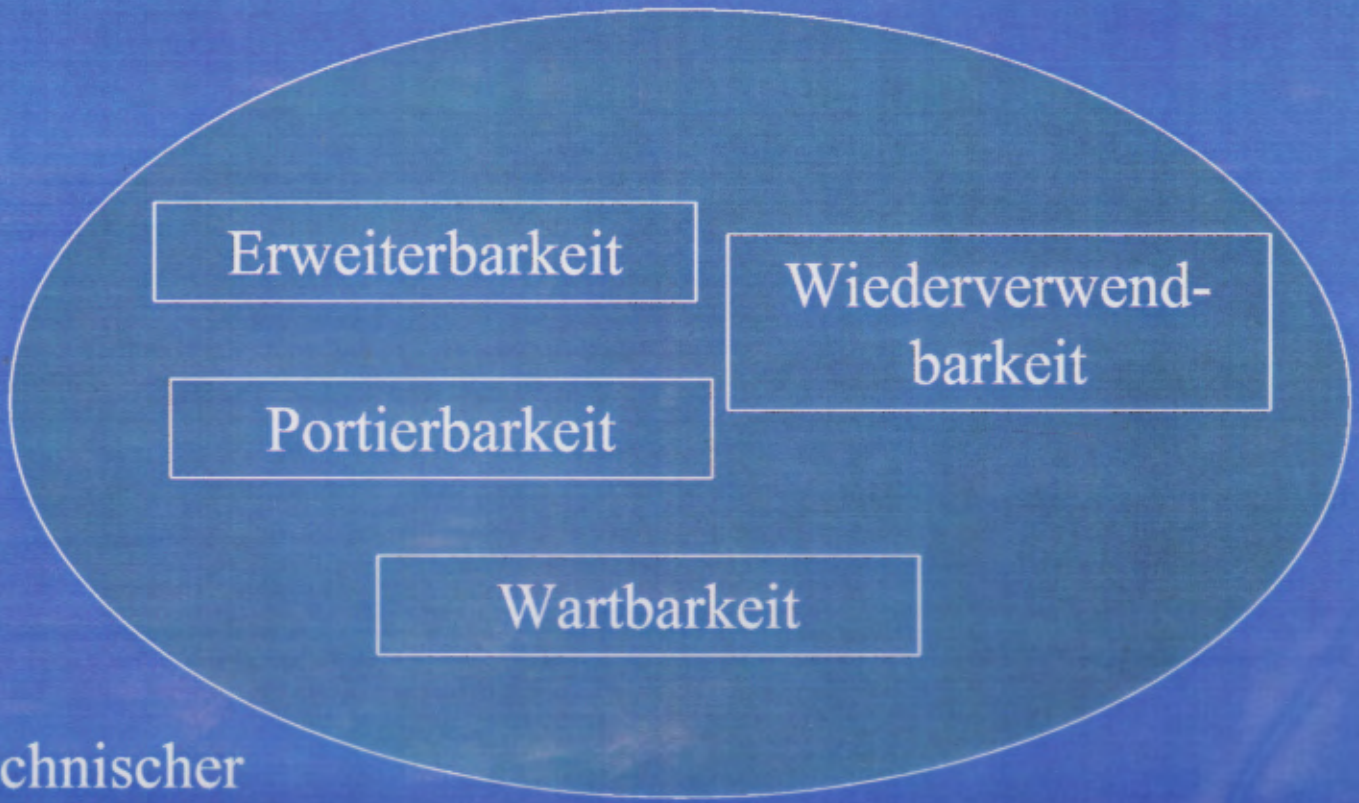
operational/
ergonomisch

Qualitätsmerkmale aus Anwendersicht (Institution)



organisatorisch/
finanziell

Qualitätsmerkmale aus Entwicklersicht



technischer
Aspekt

Qualitätsmerkmale

Die unterschiedlichen oberflächlichen Ansprüche der Beteiligten haben dasselbe gemeinsame Ziel: **Softwarequalität**

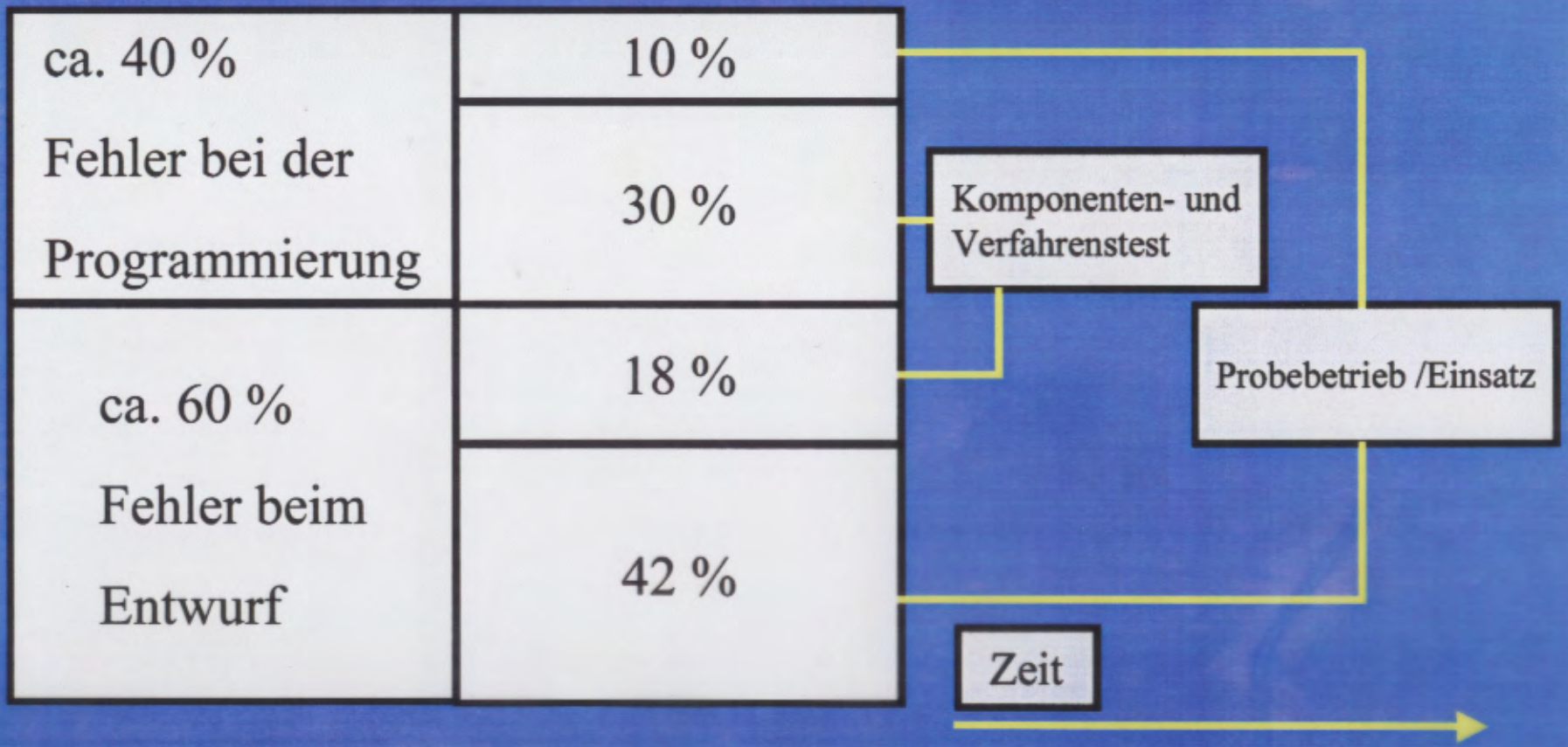
Phasen der Software-Entwicklung

- Analytische Phase (DV-unabhängig): Problemanalyse, Fachkonzept
- Synthetische Phase (DV-abhängig) Implementierung, Test
- Wartung, Pflege

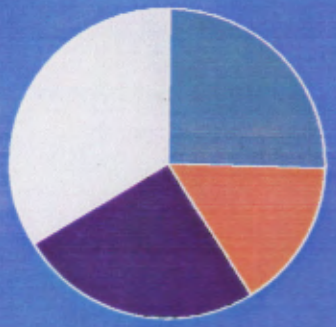
Vorgehensmodell zur Software-Entwicklung

Phase (Tätigkeit)	Ergebnis	Abstraktionsebene
Fachkonzept **55%**	Schlüssel-Schloß-Prinzip: Formale Modellierung des betrieblichen Einsatzbereiches und der zu entstehenden Software	Informationsmodell
DV-Konzept **15%**	Beschreibung der programmtechnischen Komponenten und Zusammenhänge (Daten-, Programmstrukturen usw.)	Implementations-modell
Implementierung **15%**	Realisierung der Beschreibungen des DV-Entwurfs in Form von Programmen, Datenbanken usw.	Anwendungssystem
Test **15%**		Getestetes Anwendungssystem

Bedeutung des Entwurfs

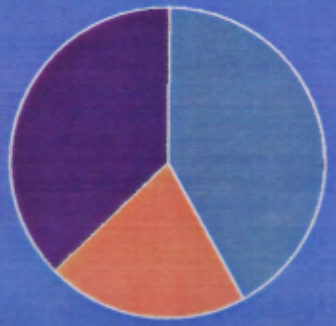


Ideen für Phasenaufteilung (1)



- Planung
- Codierung
- Komponententest
- Systemtest

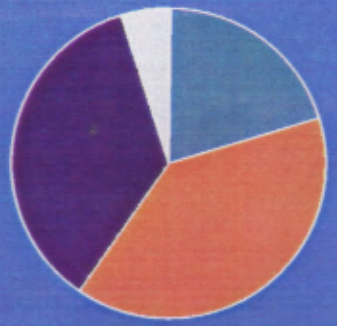
Lt. Brooks



- Analyse & Entwurf
- Codierung
- Test

Lt. Wolverton

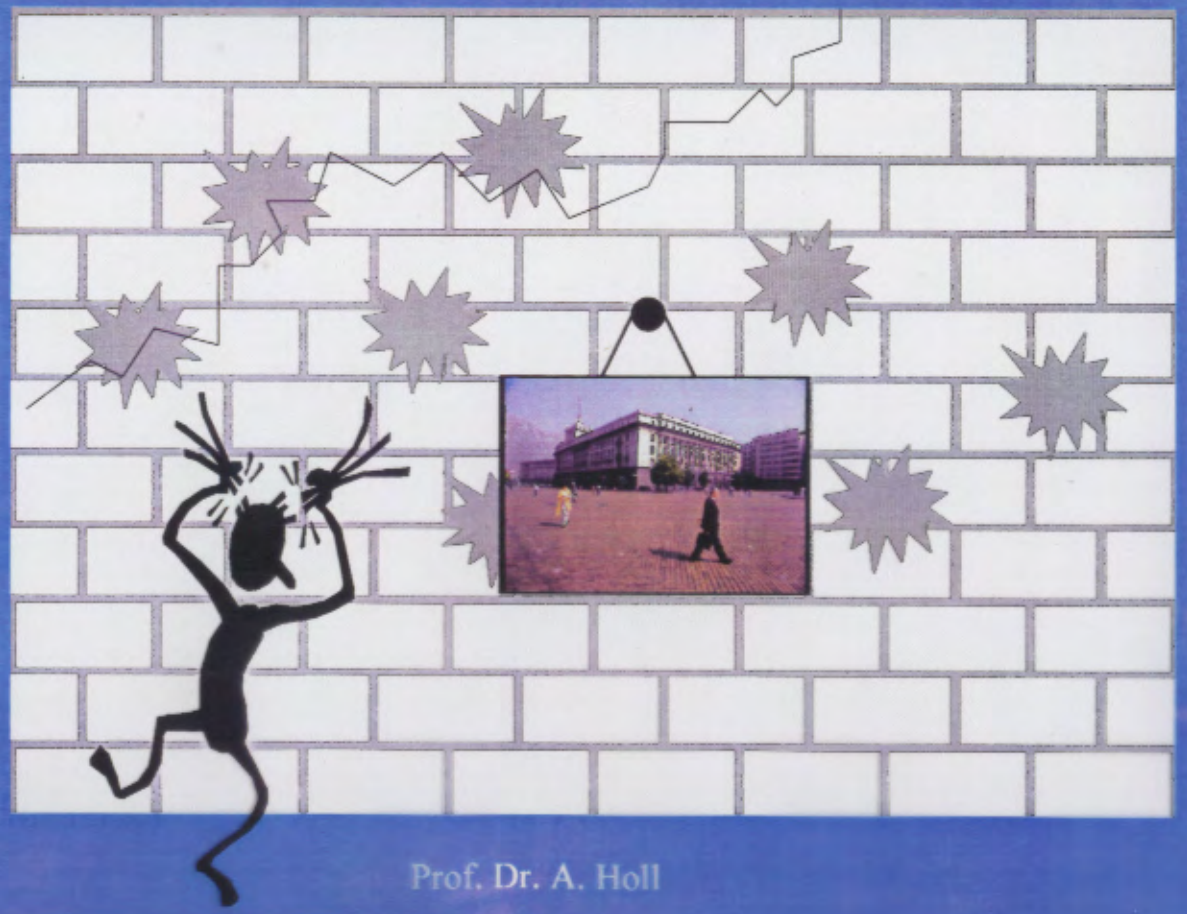
Ideen für Phasenaufteilung (2)



Lt. Mobil

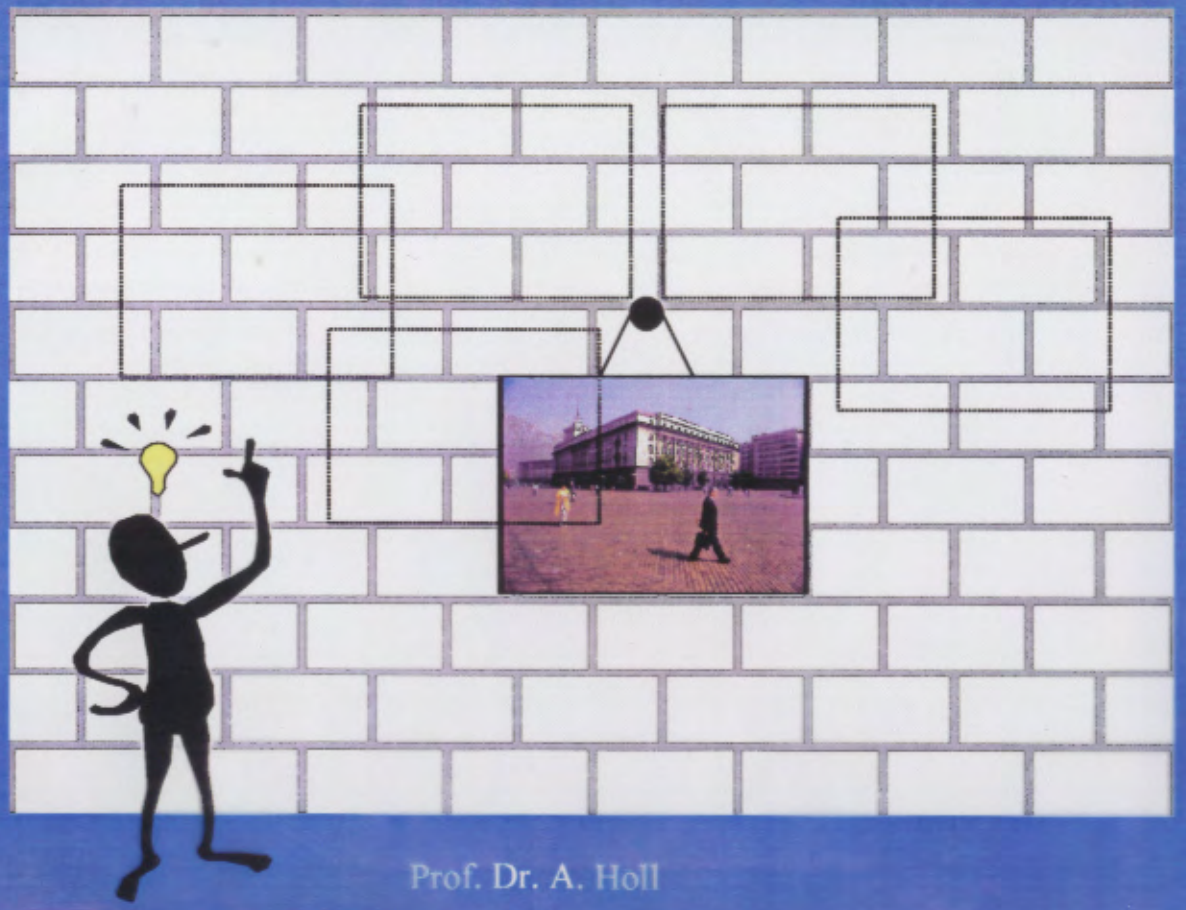
Sinn und Zweck der Analyse-Phase (1)

Ein kleines Beispiel: Sie wollen ein Bild aufhängen. **Methode 1:**



Sinn und Zweck der Analyse-Phase (2)

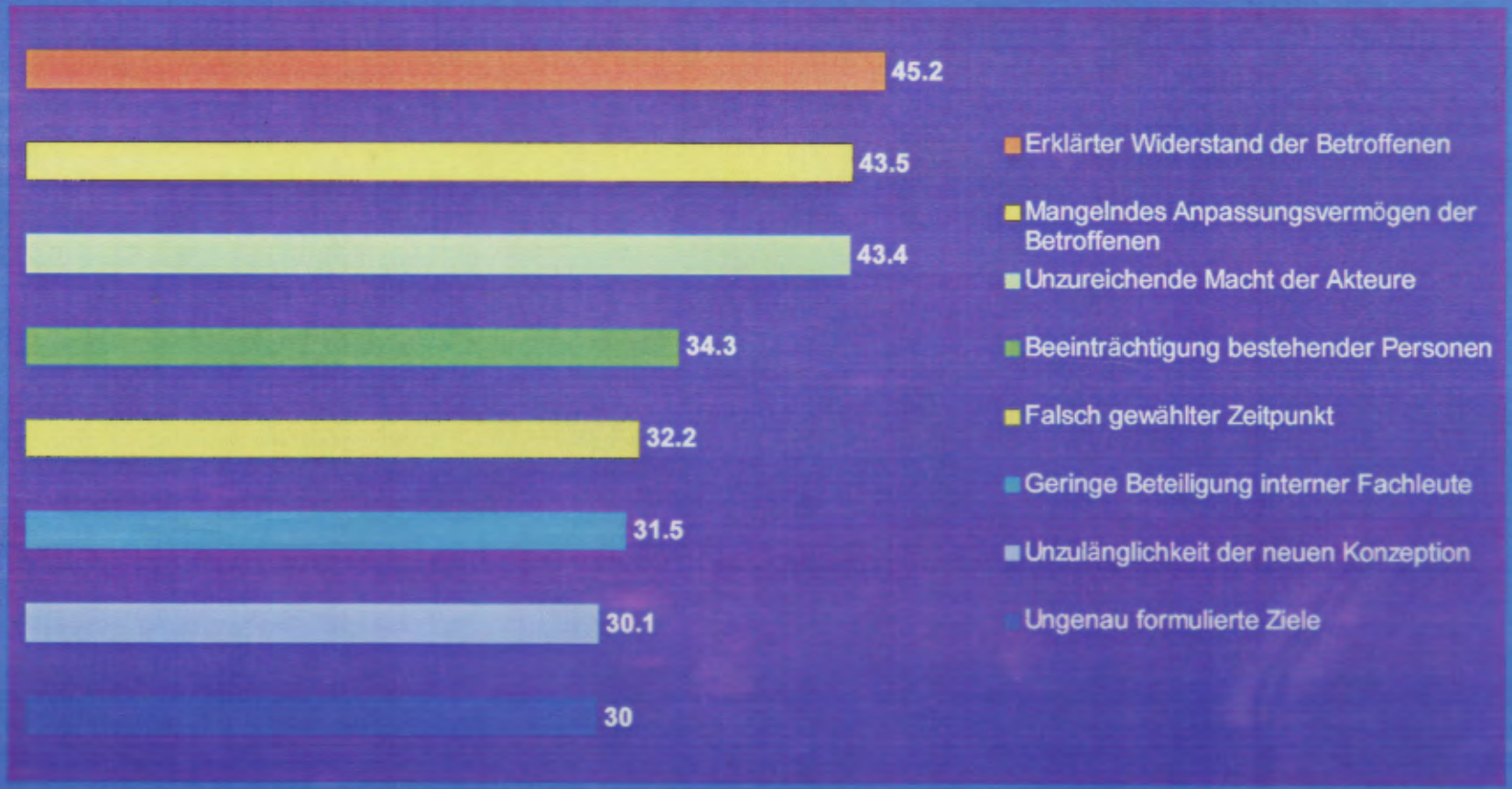
Ein kleines Beispiel: Sie wollen ein Bild aufhängen. **Methode 2:**



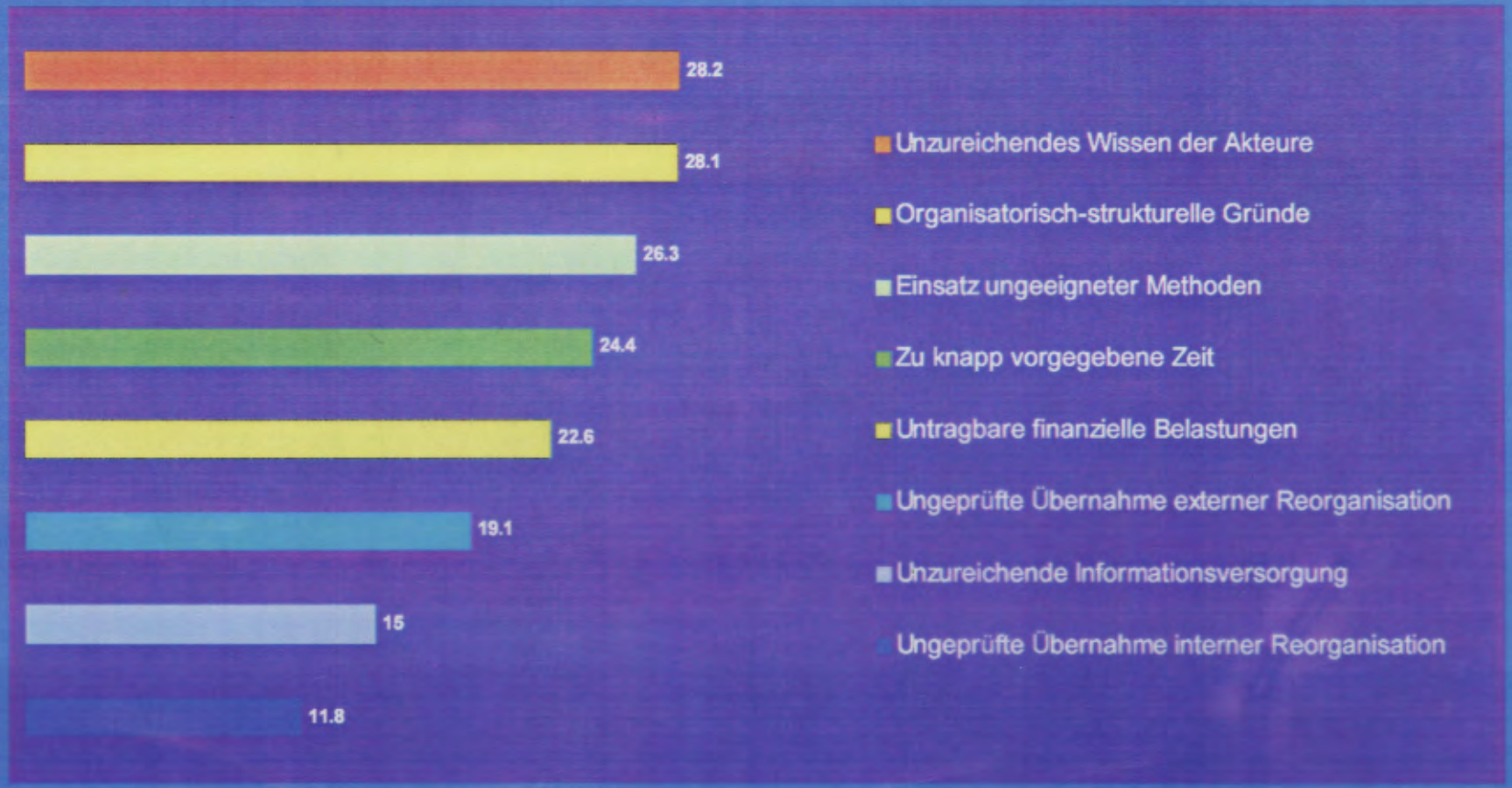
Einstellungs- und Verhaltensakzeptanz

		Verhaltensakzeptanz	
		ja	nein
Einstellungs-Akzeptanz	ja	Überzeugter Benutzer	Verhinderter Benutzer
	nein	Gezwungener Benutzer	Überzeugter Nicht-Benutzer

Gründe für das Scheitern von Organisationsprojekten (1)



Gründe für das Scheitern von Organisationsprojekten (2)



Tätigkeiten während der Problemanalyse

- Erhebung des Ist-Zustandes
- Analyse des Ist-Zustandes
- Diskussion von Lösungsalternativen
- Sollkonzept

Warum Informatiker für Problemanalyse ?

- **Erfahrung**
- **Distanz**
- **Fähigkeit zu**
  - Formalisierung und Mathematisierung
  - Abstraktem, formallogischem Denken
  - Requirements-Engineering
  - Analyse des IT-Bedarfs
  - Systemabgrenzung und Begriffsdefinition
  - Kommunikation mit Sachbearbeitern und Programmierung

Ziele der Problemanalyse

Klärung der Machbarkeit (was, wie):

- fachlich
- technisch
- organisatorisch
- finanziell
- personell
- zeitlich

Was Sie nicht erhalten

Fertige Programme mit undurchschaubarem Code-Wust

Was Sie nicht erhalten

Unüberblickbaren
Dokumentationswust

Nutzen der Ergebnisse

- Klares Konzept
- Bezug zum betrieblichen Umfeld
- Schnittstelle zur synthetischen Phase

Techniken zur Problemanalyse

Techniken

Planungstechniken

Zielformulierung/Aufgaben-
beschreibung, Netzplantechnik

Erhebungstechniken

Workshop, Fragebogen, Beobach-
tung, Dokumentenstudium,
Schätzungen

Analysetechniken

Geschäftsfeldanalyse, HEF-Analyse,
Geschäftsprozeßanalyse,
Aufgabenanalyse, Kommunikations-
analyse, Soziometrie

Erhebungstechnik - Beobachtung

- Offene und verdeckte Beobachtung
- Strukturierte und unstrukturierte Beobachtung
- Stichproben-Erhebungen aus punktuellen Beobachtungen
- Besichtigung des Betriebs

Erhebungstechnik - Fragebogen

Bedingungen:

- Erhebung quantitativer Sachverhalte
- Bekannte Erhebungsthematik
- Nicht erklärungsbedürftige Fragen
- Großer, aber homogener Kreis von Befragten

Erhebungstechnik - Workshop

- Persönliche Ansprache
- Steuern, Schwerpunkte setzen
- „Informelle Informationen“, Akzeptanz
- Feedback, Erklärungsmöglichkeit
- Kontrollierbare Situation (Psychologie)
- Anwendung von Beobachtungstechnik

➡ **Nur geschulte Informatiker !**

Analyse-Techniken

- **Geschäftsfeld-Analyse**

  Definition und Beurteilung der Geschäftsfelder hinsichtlich ihres strategischen und wirtschaftlichen Nutzens

- **Haupterfolgs-Faktoren (HEF)-Analyse**

  Ableitung der für den Erfolg eines Unternehmens typischen Faktoren unter Berücksichtigung der Unternehmensstrategie

- **Geschäftsprozeß-Analyse (Business Process Reengineering)**

  Branchentypische Geschäftsprozesse und unternehmensspezifische Besonderheiten werden zu einem Soll-Modell des Unternehmens

Prototyping

Definition:

„Ein Prototyp ist ein - mit wesentlich geringerem Aufwand als das geplante Produkt hergestelltes - vereinfachtes, einfach zu änderndes und zu erweiterndes Modell eines Systems, das nicht notwendigerweise alle Eigenschaften des Zielsystems aufweisen muß, jedoch so geartet ist, daß vor der eigentlichen Systementwicklung die wesentlichen Systemeigenschaften ersichtlich sind.

Die Methoden des Prototyping

explorativ ⇨ **Problemanalyse**	Ausgehend von ersten Vorstellungen wir ein Prototyp erstellt. Maßgeblich ist nicht die Qualität, sondern Funktionalität, leichte Veränderbarkeit und Kürze der Entwicklungszeit. Prüfung anhand von Beispielen.
experimentell	Zerlegung des Systems in einzelne Komponenten, Spezifikation der Teilsysteme durch Beobachtung der Wechselwirkungen mit den Schnittstellen; Prüfung der Flexibilität bei Erweiterung.
evolutionär	Prototyp wird nach klaren Benutzeranforderungen aufgebaut und inkrementell weiterentwickelt, indem neue Benutzeranforderungen integriert werden.

Sie benötigen die Problemanalyse!

Ziel-Situation nach Problemanalyse

Klarheit über:

- Eigene Situation
- Anfallende Kosten und Kostenersparnis
- Geschäftsabläufe

- Vorgehensweise

und vor allem

- Beziehungen im Unternehmen ➡ Gruppen-Dynamik

Alfred Holl

Problem analysis: the analytical phase in a phase concept of IS development (SW process)

1 Motivation: phase concept / software process (precise examination)

main phase	subphase	model level	model purpose
analytical phase: problem analysis	elicitation of the current state of the organization	information-relevant models	descriptive models (systems analysis)
	analysis of the current state of the organization		
	design of the planned state of the organization (LOCK)		prescriptive models (require-ments engineering)
	design of the business concept of the technical IS (KEY)		
synthetical phase: IT system development	design of the technical concept of the technical IS	implementa-tion-relevant models	
	programming		
	test		
	use	information-relevant models	
maintenance			

2 Preliminary examination (strategic phase) 1

2.1 Principles 1

Objectives:
- to figure out goal and purpose of the project
- to design a coarse business concept (Grobkonzept)
- to examine the feasibility (Durchführbarkeitsstudie)
- to finally decide about project initiation (Projektanstoß)

Coarse business concept: What is the problem / application area?
system definition: remember that
organizations are open socio-technical information systems!

1. System delimitation, system boundaries,
external coupling to the system surroundings
(Umweltkopplung zur Systemumgebung):
- **Remember: magnifying glass (geographic lens):**
  model boundaries wider than direct IT application area
- **Attention: cognitive processes during segmentation**
- **Attention: arbitrary object boundaries.**
  Where is the rim of an organization?

→ 2.2 Problem of isolation / separability

2. Components, interactions, internal linkage (Binnenkopplung),
partial systems on the next lower abstraction level

3. Goal and purpose of the system definition
Which purpose shall modeling during the precise examination
meet? Which aim shall it reach?
Which question shall it answer?
E.g. preparation of the deployment of individual SW,
selection of standard SW,
planning of business process reengineering

2 Preliminary examination (before SW process) 2

2.1 Principles 2

4. Feasibility

4.1 mathematical-formal accessibility to modeling
(important to estimate the duration of the project)
- pre-formalization (Vorformalisierung)
- accessibility to formalization (Formalisierbarkeit)
- effort of formalization (Formalisierungsaufwand)

→ 2.3 Problem of pre-formalization

4.2 technical (IT) feasibility
- coordination with existing IT systems
- individual SW, branch SW, standard SW

4.3 financial und temporal feasibility
- human and technical resources (Personal- und Sachmittel)
- possibility of financing, e.g. credits (Finanzierbarkeit)
- cost estimation (Aufwandsschätzung):
time, development costs, resulting costs (Folgekosten)

4.4 human resources psychology (Arbeitspsychologie)
- acceptance: hindered end-users (verhinderte Endbenutzer)
- resistance: forced end-users (gezwungene Endbenutzer)
- fear of rationalization (Rationalisierungsangst)

4.5 juridical and ethic feasibility
- technology assessment (Technikfolgenabschätzung)
- social acceptability/responsibility (Sozialverträglichkeit)
- data privacy (Datenschutz)

2.2 Problem of isolation / separability 1

Observations

trouble with not effective, isolated SW solutions
trouble with the limitation of systems

Critical realism

1. complexity reduction by creating segments and structures
2. there are natural system-like structures
with strong internal and weak external connections
but: there aren't any natural closed systems,
only open systems [socio-technical information systems in IS]
==> systems: descriptive categories, not reality-immanent
it is the observer who defines system boundaries

There is no empirical knowledge without isolation of systems!
(2nd epistemological dilemma)
Isolation is the pre-condition for
– the mere cognition of objects
– the transfer of feature sets, the perception of "gestalts"
– the comparison of objects
(Lorenz "The innate forms of possible experience" 1943: 319)

Evolutionary epistemology

1. cerebral cortex as carrier of cognitive processes
has its origin in optical neural centers;
consequence: cognitive strategies are transferred from
primary objects of cognition
visual-tangible (physical solids), simple,
few interactions, 'mesocosmic'
to secondary objects of cognition
socio-economical, sub-atomic particles, complex,
numerous interactions, macro/microcosmical
2. small segments are better suitable as basis of analogy

2.2 Problem of isolation / separability 2

Consequences for IS

1. at least: SA context diagram or UML use case diagram
with system surroundings and external connections
2. better: magnifying glass model:
soft, blending system boundary
with precision/magnification decreasing towards the rim
3. clear idea of the system's purpose and objectives

Remark 1 (ambiguity of segmentation)
There are different ways of decomposing an object domain.

Example: magnet: optical field vs. magnetic field

Remark 2 (abstraction levels)
Humans cannot understand complex systems at first glance
==> complexity reduction by decomposition
is necessary on different abstraction levels
==> problem of isolation occurs on every abstraction level

Remark 3 (process – system)
Processes can be interpreted as (linear) systems,
therefore, there are equivalences
– open process ~ open system
– process boundary ~ system boundary (defined by observer)
– process decomposition ~ system decomposition (→ IV.3.1.2)

2.2 Problem of isolation / separability 3

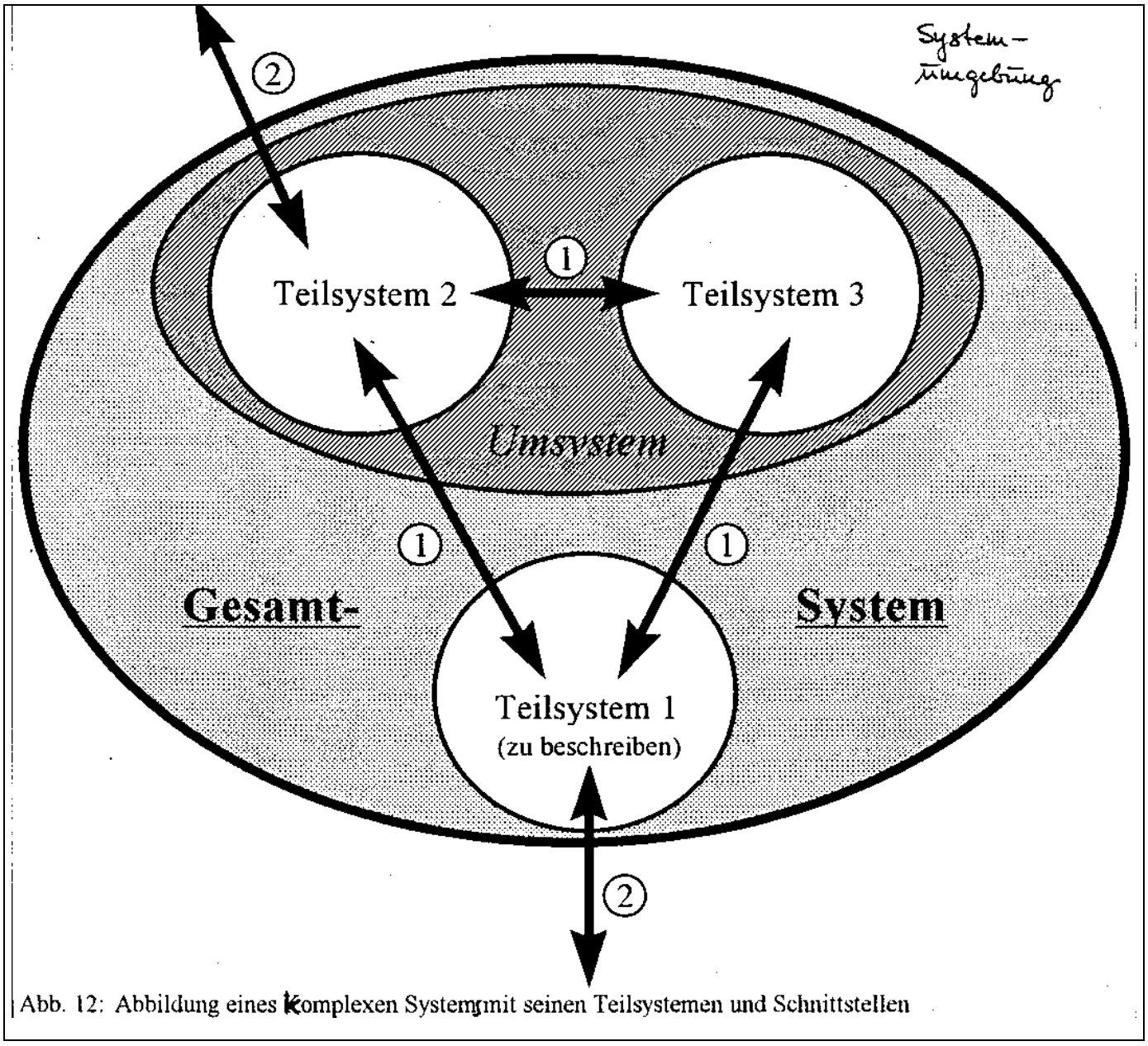
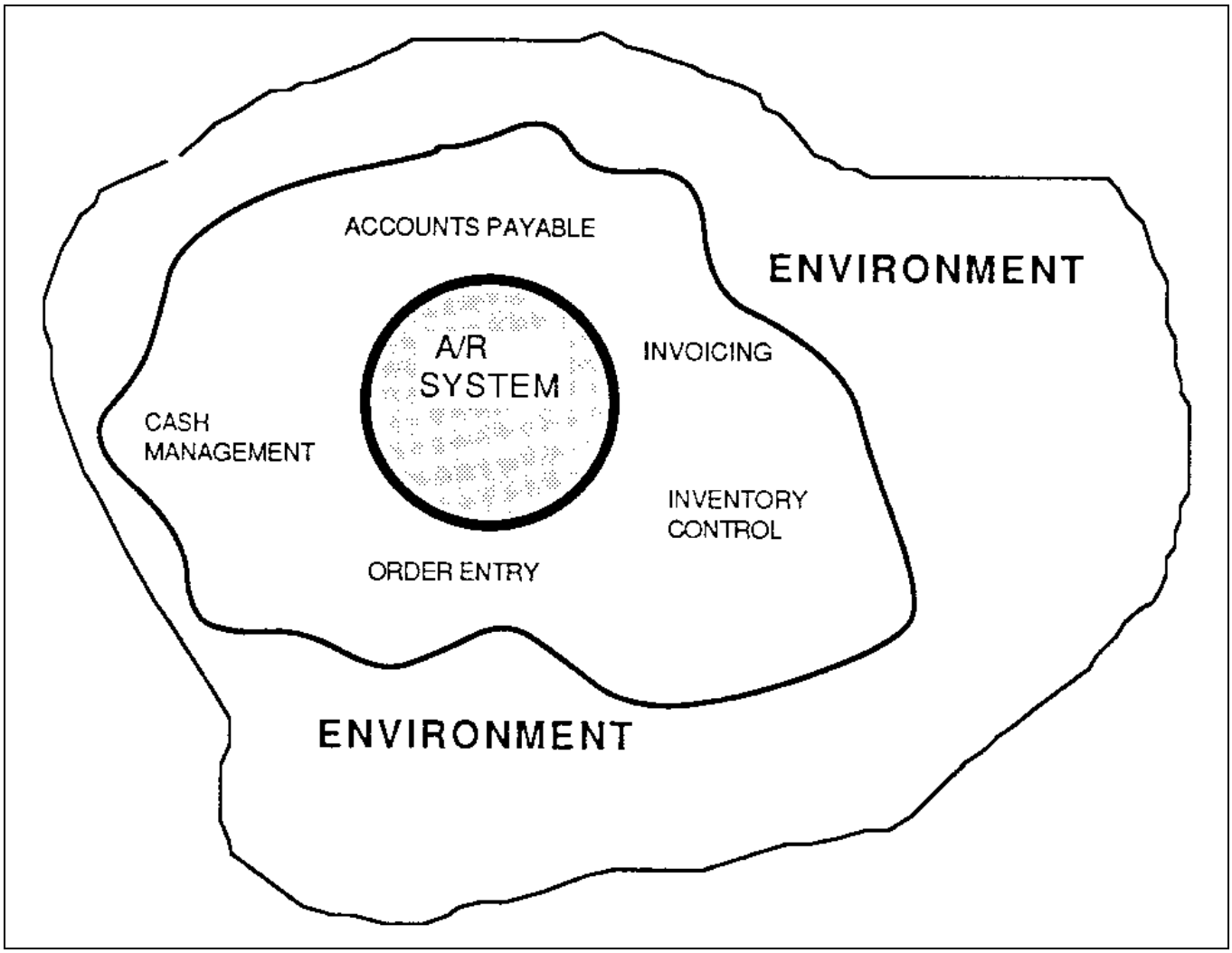


Abb. 12: Abbildung eines komplexen Systems mit seinen Teilsystemen und Schnittstellen

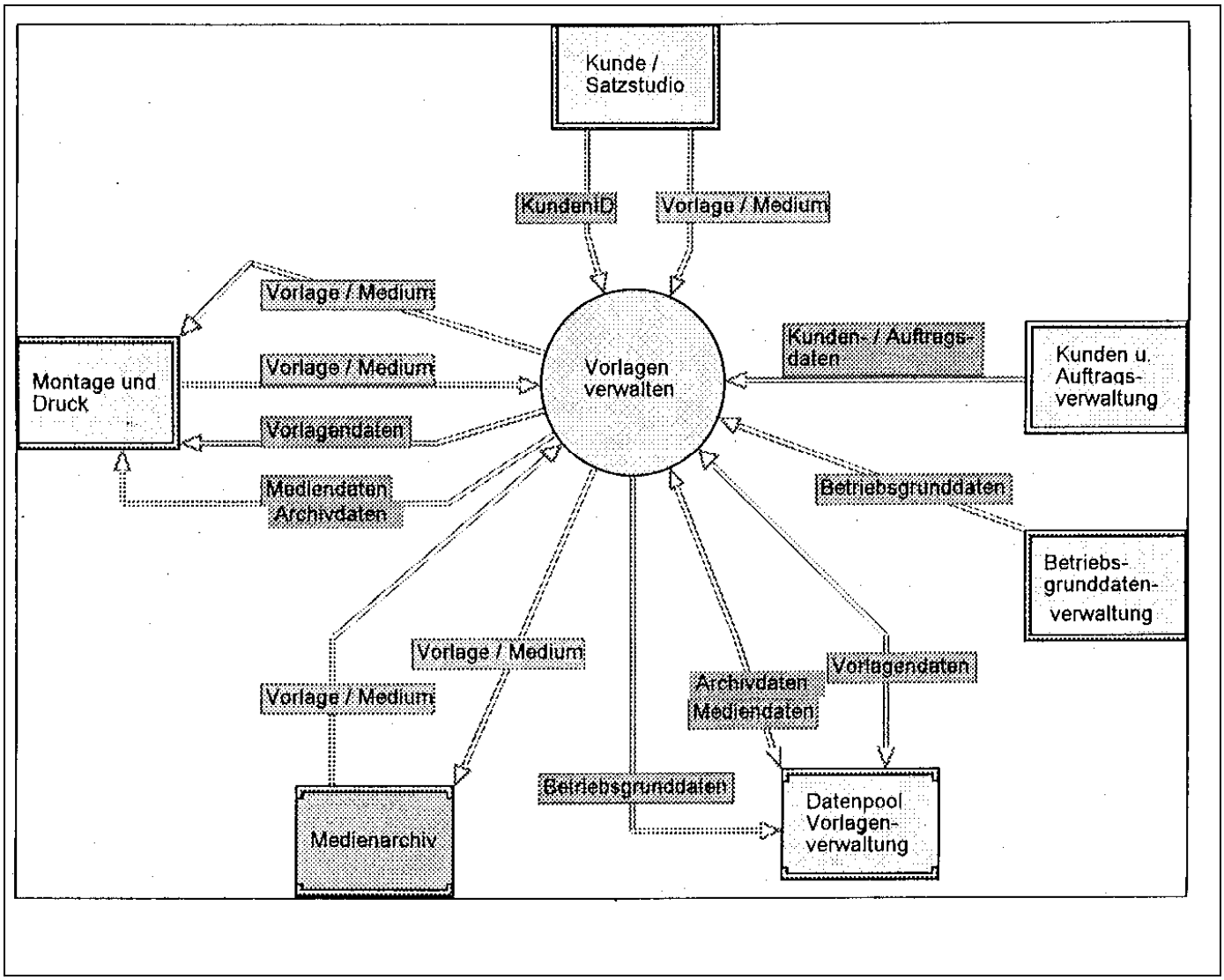
Surrounding system and system surroundings (from a diploma thesis)

2.2 Problem of isolation / separability 4



Accounts receivable system embedded in the enterprise (Yourdon, Modern structured analysis, 1989, 336)

2.2 Problem of isolation / separability 5



System surroundings in a structured analysis context diagram (from a diploma thesis)

2.2 Problem of isolation / separability 6

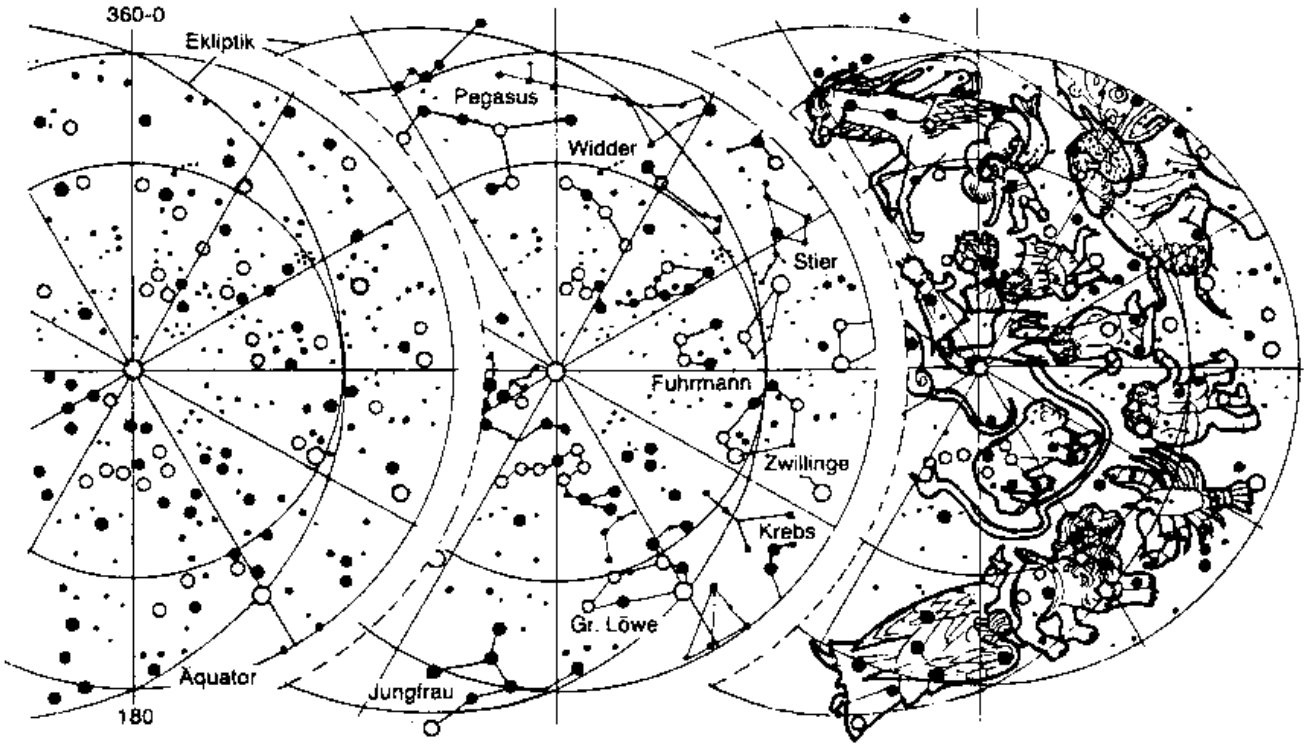


Abb. 6. Gestalt, wo keine ist, am Beispiel eines Ausschnittes des nördlichen Sternenhimmels. Man beachte links die reale Zufallsverteilung der Sterne und ihrer Größen, in der Mitte deren Verbindung zu Sternbildern, rechts die barocke Darstellung nach dem Himmelsatlas von Andreas Cellarius, der 1708 in Amsterdam erschien. Sie ist um 90° C gedreht und um die Achse 90–270° C gewendet, um der heute üblichen Orientierung zu entsprechen (vgl. Störig 1972).

**Constellations: form, where none exists,
exemplified by a section of the northern celestial sky;
to the right, the baroque representations by Cellarius 1708
(Riedl, Biology of knowledge, 1984, 157)**

2.2 Problem of isolation / separability 7

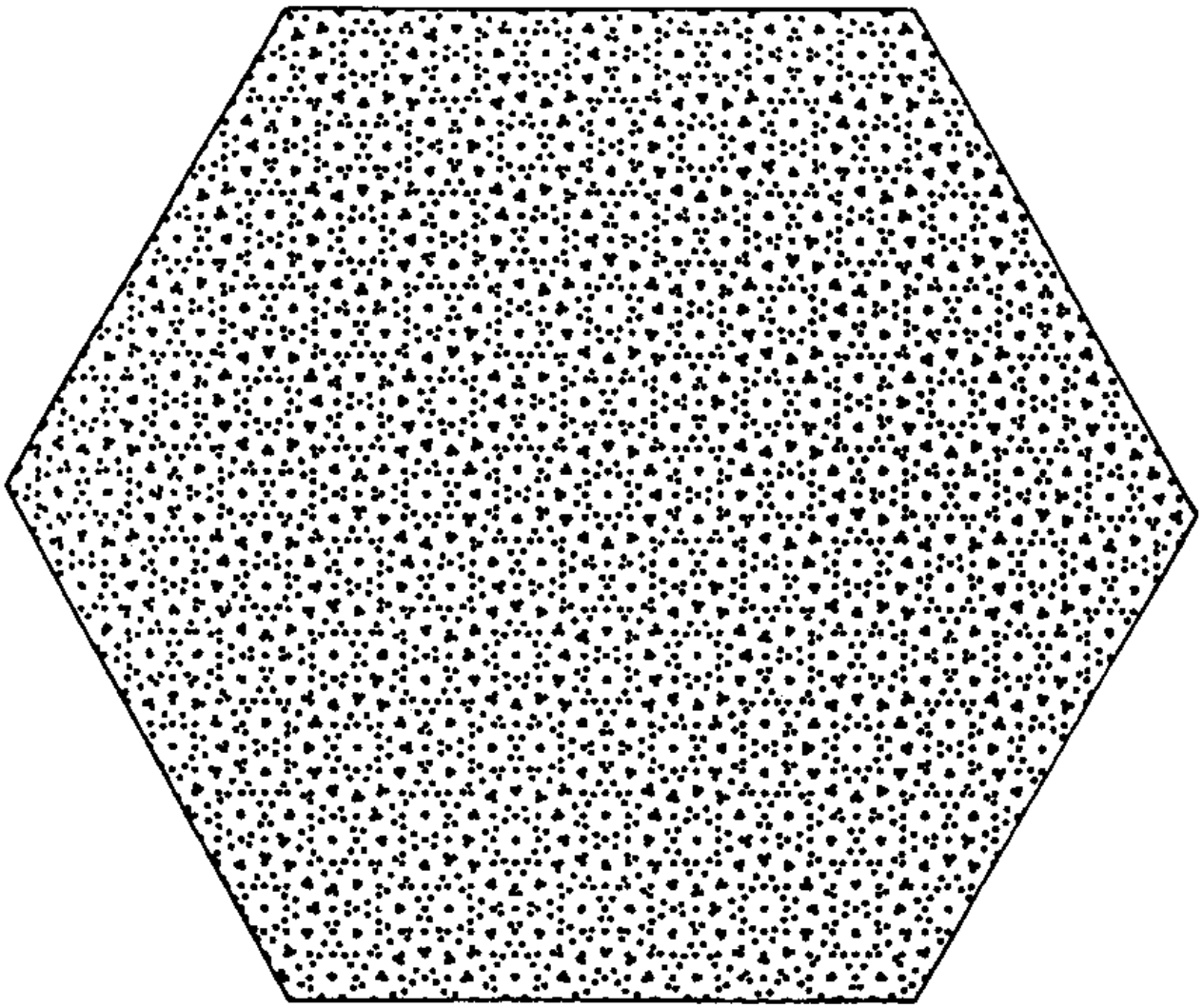


Abb. 2.30: Illustration des autonomen Strukturierungsprozesses unserer visuellen Wahrnehmung. Wir nehmen einen raschen Wechsel verschiedener Muster wahr. Unsere Wahrnehmung sucht nach Ordnung. Sie strukturiert das Wahrgenommene und interpretiert es auf verschiedene Weise. Aus D. MARR (1982).

Construction of structures by visual perception
(Eibl-Eibesfeldt, Biologie des menschlichen Verhaltens, 1995, 77)

2.3 Problem of pre-formalization, suitability for and effort of formalization

Observations

It is more difficult to model small organizations than large ones.
SW development for accounting is easier than for production.
Suitable descriptive categories
are often unknown in organizations
and model designers have to start from scratch to define them.
It is difficult to estimate the time necessary for formal modeling.

Critical realism

With regard to formalization, IS object domains differ in:
– pre-formalization
– suitability for formalization
(cf. deterministic vs. (non-) deterministic, chaotic domains)
– effort of formalization
→ not pre-formalized, scarcely formalizable object domains
→ partly pre-formalized object domains: implicit formal models
→ well pre-formalized object domains: explicit formal models

Consequences for IS

Examine object domains with respect to 3 views of formalization.
Respect the results in time and project management.
Pre-formalized domains are starting points for IT (accounting).
Don't force formalization, allow chaotic oscillations (production).
Check terminology used in organizations
with regard to suitability for formalization.

Remark (structuring)

These considerations apply
for pre-structures, suitability and effort for structuring as well.

3 Precise examination of the application area 1 (phase concept / software process)

3.1 Principles 1

1. Elicitation of the current state (Ist-Aufnahme):
description of the lock

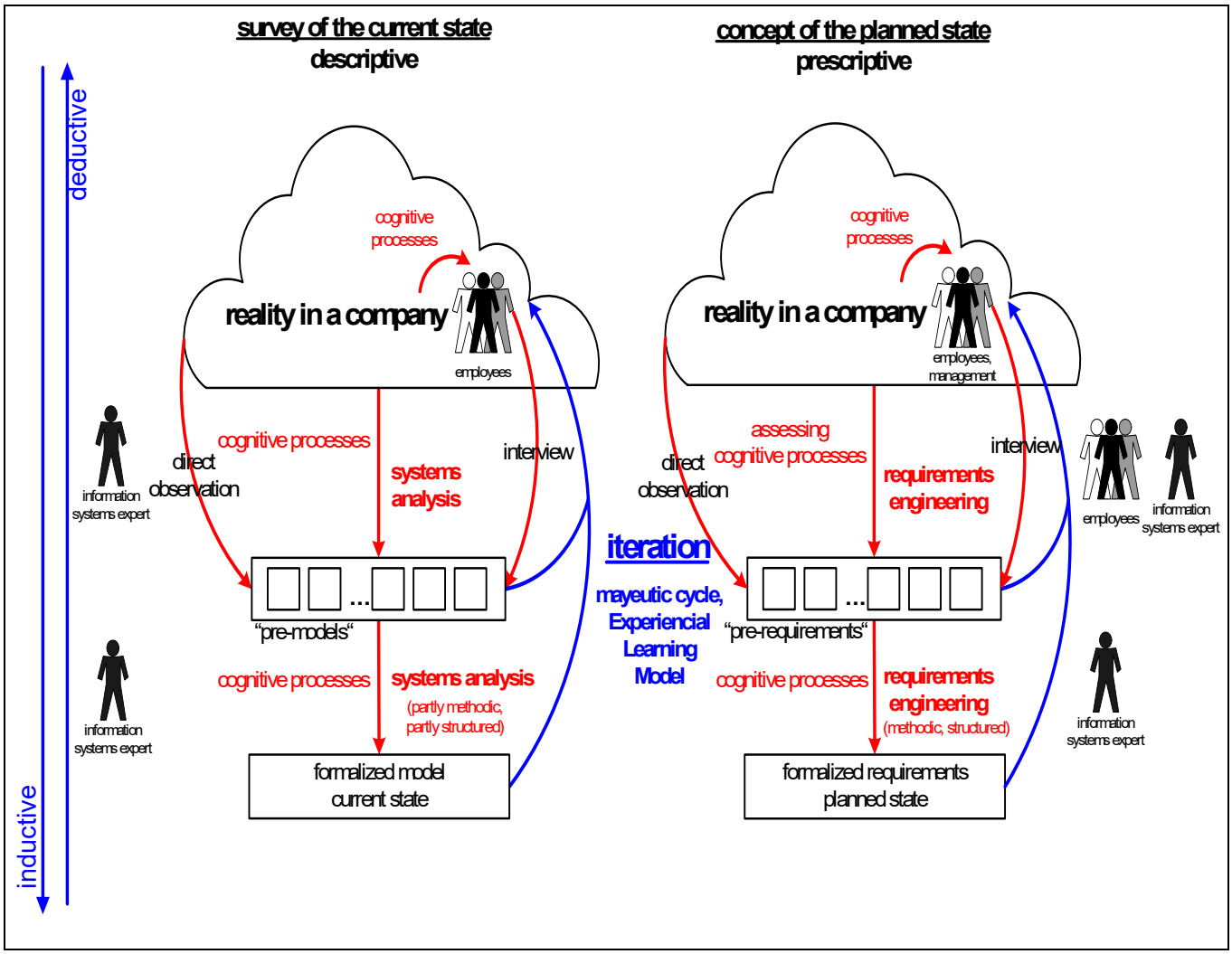
2. Analysis of the current state (Ist-Analyse):
Is the lock straight (pre-formalized), crooked,
can it be straightened?
Analysis of defects (Schwachstellenanalyse):
where are changes necessary?
Requirements Engineering (continued in step 3)
Proposals for solutions (Lösungsalternativen):
how should changes be made?

3. Concept of the planned state (Soll-Konzept):
formal model of the lock (organization)
formal model of the key (technical information system)
fine business concept (fachliches Feinkonzept, Pflichtenheft)

→ part 9: meta-models and notations of modeling approaches

3 Precise examination of the application area 2

3.1 Principles 2



Cognitive processes in IS modeling
(Holl / Maydt, Epistemological foundations of RE, 2007)

3 Precise examination of the application area 3

3.2 Requirements engineering: requirements in natural language

Rupp, Chris: Requirements Engineering. München 2001
based on R. Bandler/J. Grinder: Neuro-linguistic programming

This psychiatric approach tries to find the underlying meaning of utterances produced by means of transformations (cf. Freud's projection to others and exaggerated contrary)

Deletion
implicit assumptions
incomplete properties:
easily changeable: **how easily? what's easy? by whom?**
incomplete verbal nuclei/kernels
(missing objects and adverbials; cf. dependence grammar):
SW shall inform about errors: **whom? where? how? when?**
the development of a SW tool: **who develops? when? why?**

Generalization
universal quantors:
every error: **really every? any exceptions?**
incomplete conditions:
If the error X occurs in the last phase of program Y, then ...
What should be done if it occurs in another phase?
definite article without text reference:
the error: **which?**

Distortion
nominalization: resulting event instead of process
loss of data: **which data are lost? when? how? why?**

3.3 Requirements changes during problem analysis: temporal dynamics of the application area 1

Observations

SW does not meet requirements after long programming periods.

Critical realism

Every segment of the reality contains internal temporal dynamics, which can partly be deterministic and partly chaotic.
Prognoses of its future behavior are only partly reliable, especially if a segment is disturbed.

Consequences for IS

in general:

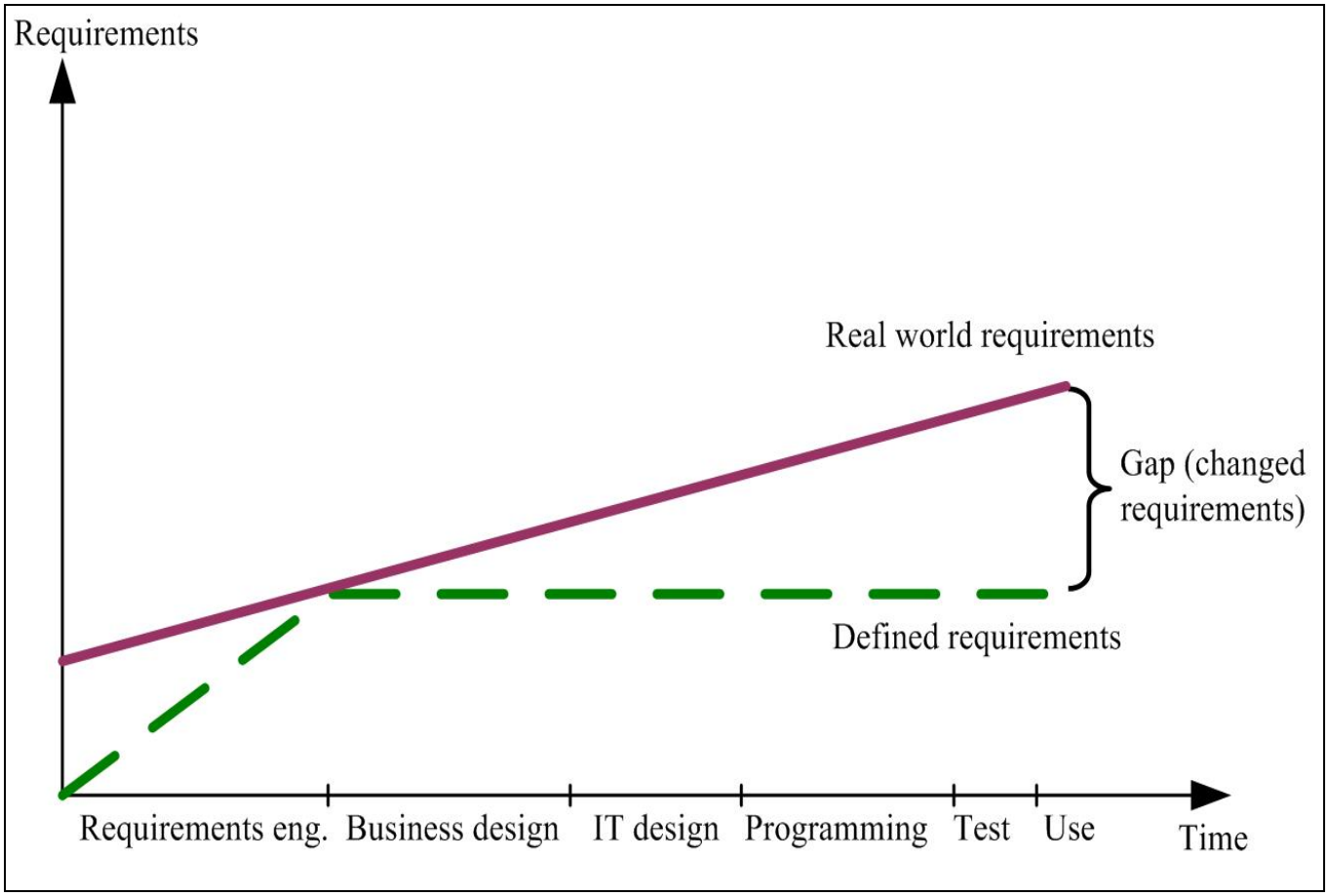
1. keep your model of the planned state valid
by quickly including changes in the application field in order to avoid using models of a past reality for programming
2. keep your models and your SW easily changeable in order to be able to easily change models and SW
in the case of changes in the application field

Consequences for IS

in detail:

an aspect of changed / creeping requirements management
overlapping phases and iterations in phase concepts
dynamic design concepts
permanent check of changes in the application field
permanent contact to the future users
participative strategies
evolutionary SW development
well-documented and easily adaptable SW
some aspects of 'extreme programming'
(user participation, quick development, small projects)

3.3 Requirements changes during problem analysis: temporal dynamics of the application area 2



Changed / creeping requirements gap
(Holl / Paetzold / Breun: IS anti-aging, 2011, fig. 17, p. 39)

4 Software evolution – IS anti-aging 1

Changes in reality require changes in software:

changed / creeping requirements management (cf. 3.3)

1st: introduction of changes in an IT independent (business) model

2nd: design of a new IT (dependent) model based on two sources:

1 the new IT independent model (information-relevant level)
2 the old IT (dependent) model (implementation-relevant level)

If the IT independent (business) model level is ignored and the new IT model is based only on the old IT model, SW complexity will soon increase a lot faster than the complexity of the underlying reality.

Reason: SW complexity has two sources:
– complexity of the underlying reality
– IT-immanent (IT dependent) complexity
due to performance optimization, features of IT products etc.

Long-term SW evolution restricted to the IT level is impossible!

Include possible future changes in reality you can already foresee always in the IT independent (business) model level!

cf. Meir Lehman: E (embedded) type programs

requirements engineering on the basis of epistemology

4 Software evolution – IS anti-aging 2

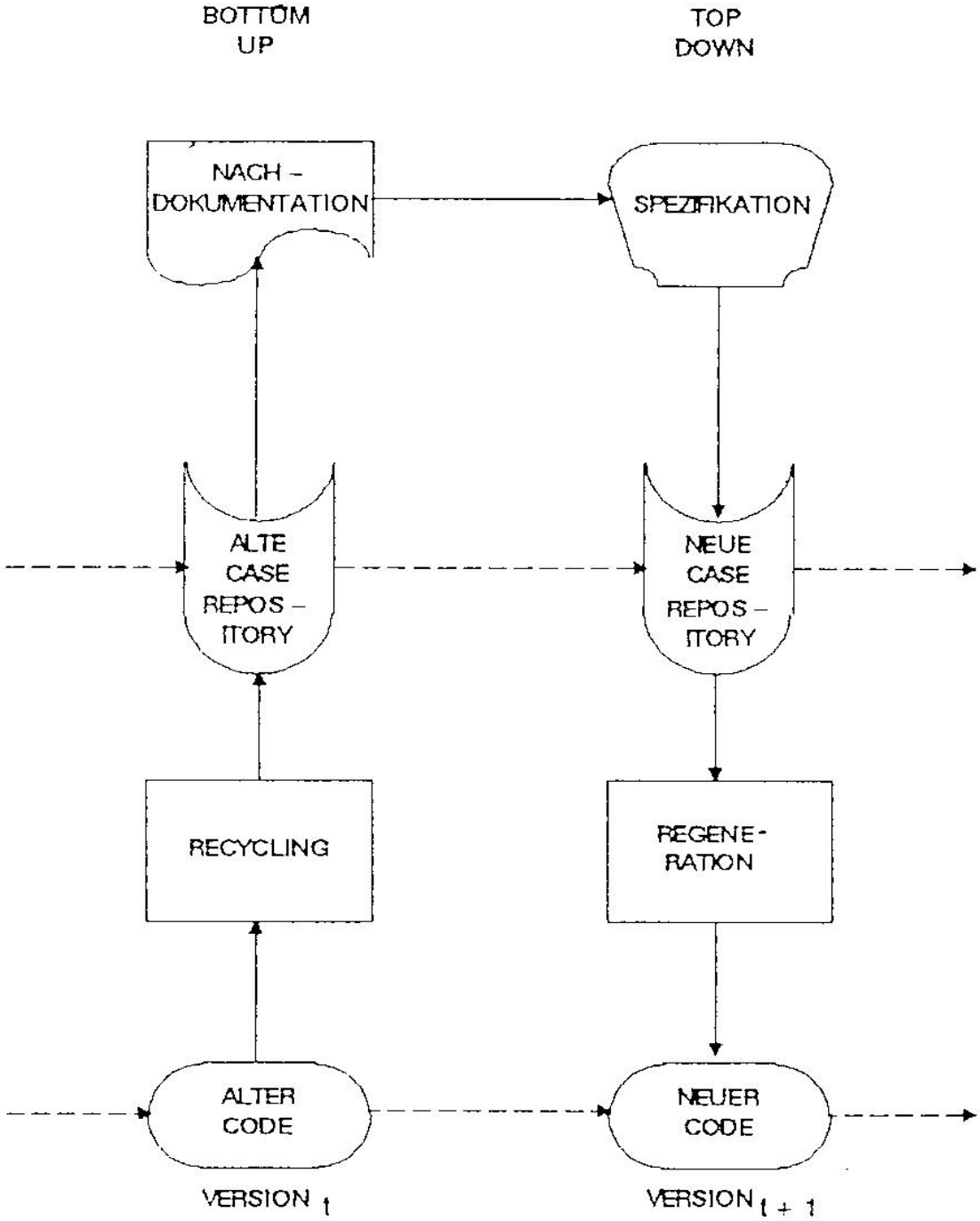
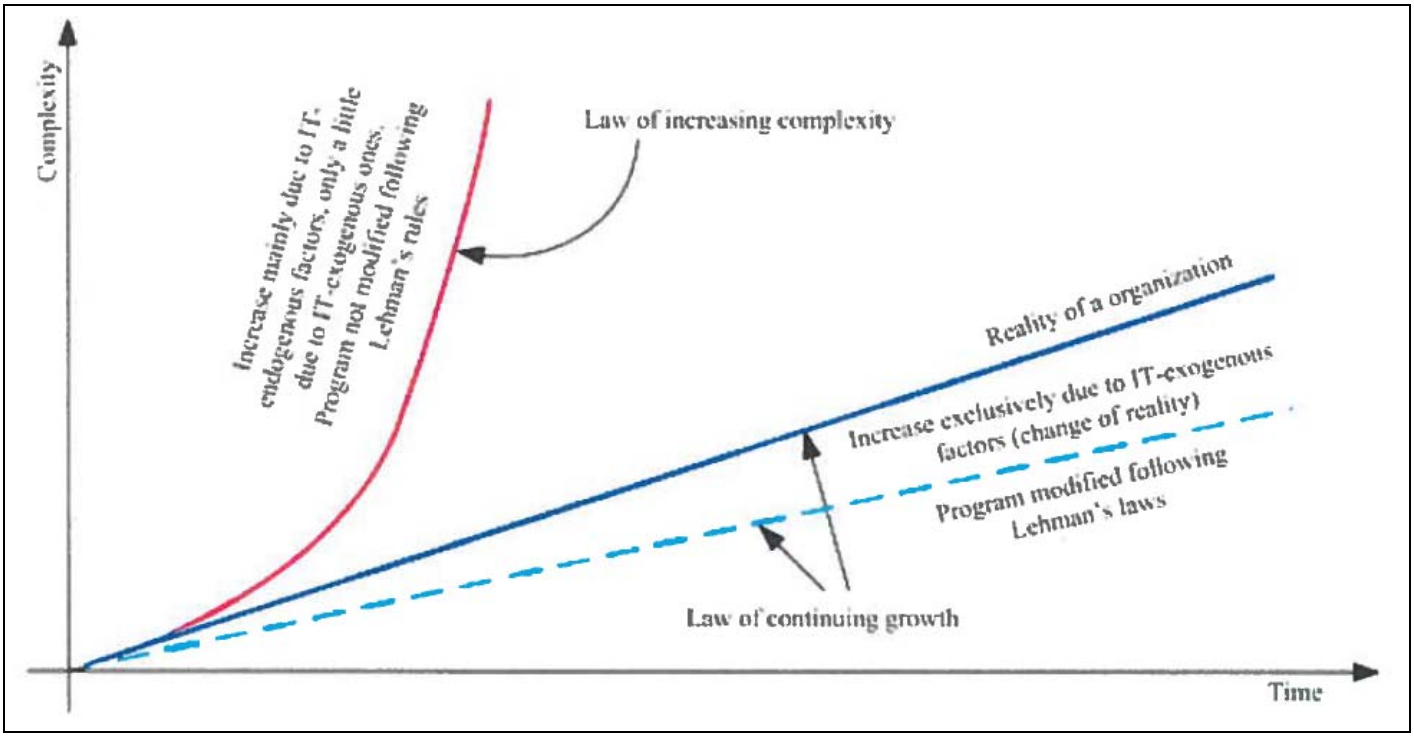


Abb. 2.10: Bottom-Up Software-Entwicklung

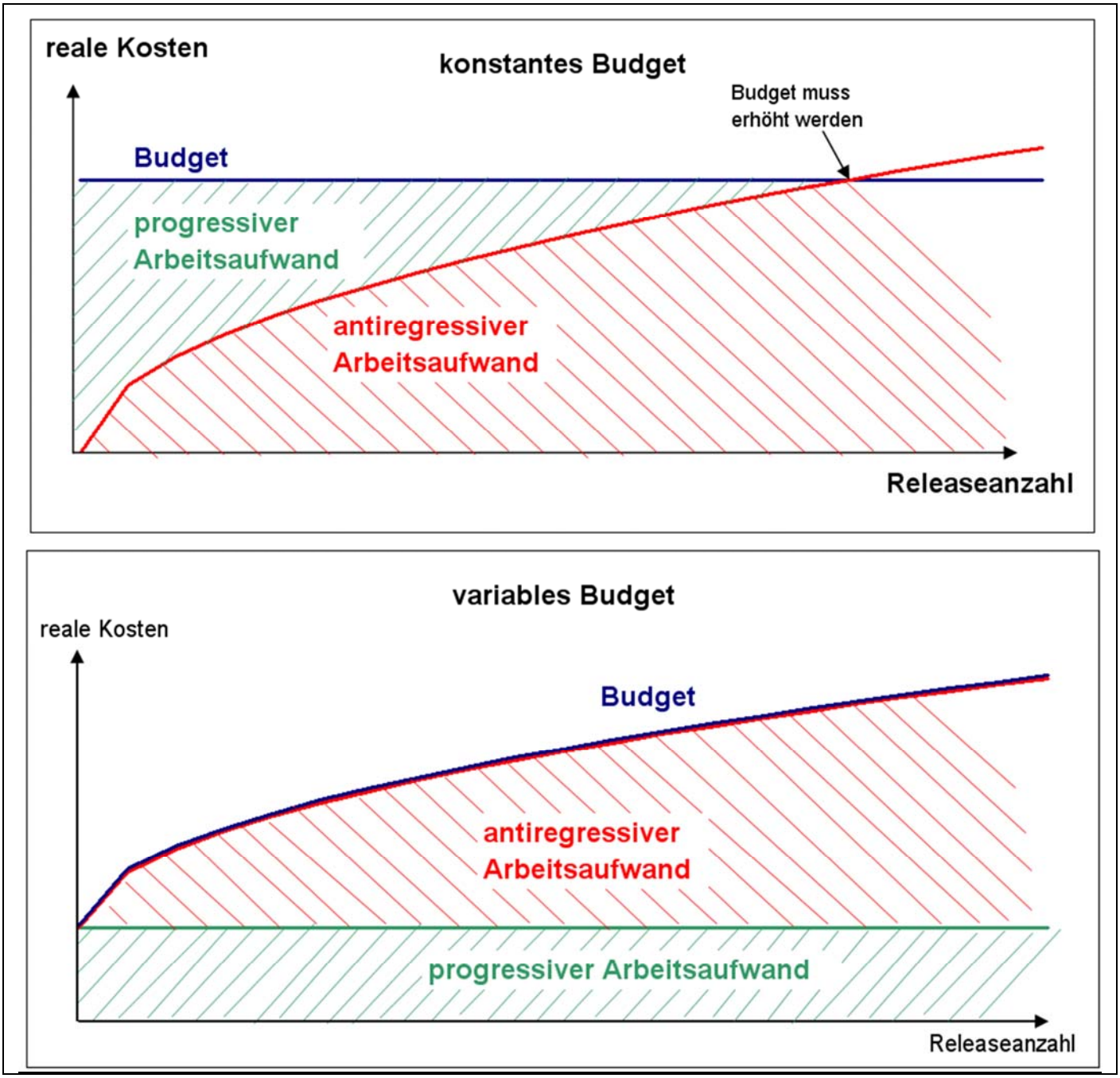
Bottom-up software development
(Sneed, Harry M.: Software maintenance, 1991, fig. 2.10)

4 Software evolution – IS anti-aging 3



Increasing complexity of E-type systems
(Holl / Paetzold / Breun: IS anti-aging, 2011, fig. 20, p. 44)

4 Software evolution – IS anti-aging 4



Progressive and anti-regressive costs over time

5 References

pdf-files of my own publications: see my homepage.

Holl, Alfred:
Empirische Wirtschaftsinformatik und evolutionäre Erkenntnistheorie.
In: Becker, Jörg et al. (ed.): *Wirtschaftsinformatik und Wissenschaftstheorie. Bestandsaufnahme und Perspektiven*.
Wiesbaden: Gabler 1999, 163-207, ISBN 3-409-12002-5.
English translation on my homepage.

Holl, Alfred; Maydt, Dominique:
Epistemological foundations of requirements engineering.
In: Erkollar, Alptekin (ed.): *Enterprise and business management. A handbook for educators, consulters and practitioners*.
Marburg: Tectum 2007, 31-58;
short version = contribution to:
***Requirements Days 2006*, Nuremberg/Germany.**

Holl, Alfred; Paetzold, Felix; Breun, Robert:
Cooperative cyclic-iterative knowledge gain in IS anti-aging.
Nuremberg: University of Applied Sciences 2011.

Alfred Holl

Rationalistic approaches to IS modeling: analogy and reference models

1 Motivation 1

1.1 Analogical thinking, a cognitive strategy 1

Cognitive dilemma 1

(Neolithic) Humans need(ed) information (knowledge)
to quickly master new situations,
but humans cannot know every object of cognition.
They have too compare them with well-known situations.
=> the cognitive necessity of comparisons:
analogical thinking / reasoning

Purpose of analogical thinking:
Quick extension of the knowledge about some new situation
based upon a comparison, upon analogy,
no logical conclusion, but a heuristic strategy.

Procedure:
1. Situation
A new and a well-known object of cognition (requires memory!)
coincide in some features.
2. Assumed consequence (analogical knowledge transfer)
Assumption of analogy:
They coincide in all their "important" features,
at least one more feature.

Assumption of a strong analogy starting from a weak one,
assumption of an extensibility of an existing analogy.
(In German: Analogieschluss = Schluss auf stärkere Analogie)

The correctness of assumptions of analogy cannot be proved.

1.1 Analogical thinking, a cognitive strategy 2

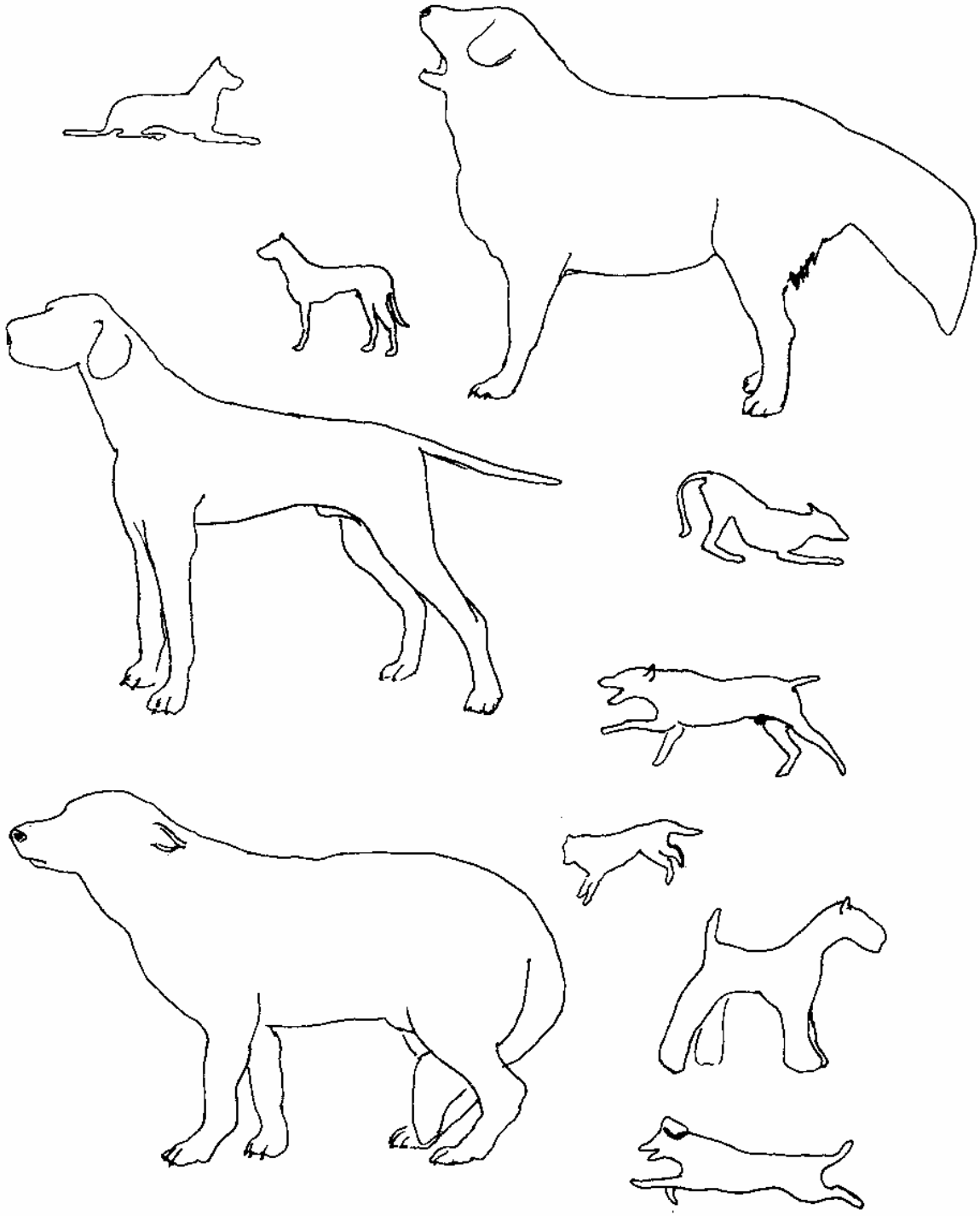


Abb. 13: Beispiel für Gestaltwahrnehmung des Menschen. Jede dieser Figuren erkennen wir als „Hund", obwohl es sich nur um grobe Umrißzeichnungen handelt. Wir abstrahieren die für Hunde essentiellen Merkmale, die „Hundegestalt", und erkennen diese auch in vereinfachenden Abbildungen, sofern nur die betreffenden Merkmale herausragen.

Assignment of individuals to a type using key features (Wuketits, Entdeckung des Verhaltens, 1995, 71)

1 Motivation 3

1.2 Examples for analogical thinking in IS 1

- **Data Mining** techniques (statistical and non-statistical), knowledge discovery in databases: similarities of data objects are used for inductive type construction.
- Transfer of **reference models** to "analogical" application fields;
  cf. purchase and sales depts. in a company
  (vs. the second source of modeling: observation / interview)
- Taxonomy in OO class models (**generalization**)
- Almost all IS models are **type models**.
  We don't model an individual customer, but a customer type.
  We don't model individual sales processes, but a sales process type.
  (Exception: model of an individual machine)
- The concept of analogy can be used for static and dynamic situations, e.g. data structures and process structures.
- Pattern recognition
- Design patterns
- etc.

1 Motivation 4

1.2 Examples 2 – Two sources for model design

Popper's World 1 (reality): empiristic method/approach
Organization, company, department
observation and interviews (W3)
of employees by a model designer
(contrary to natural sciences: only observation)
preliminary description in pre-formal models: natural language
abstraction
check whether terminology is mathematically well-defined
final type construction
formalization (degree of pre-formalization is different)
reduction to axioms

often used for peripheral areas of models
often used for individual parts of an organization
(nominalist point of view: enumeration of individual objects)

Popper's World 3 (models, concepts, ideas): rationalistic method
reference models
activation in a model designer's brain
analogy-based transfer

often used for central areas of models
often used for standard parts of an organization, e.g. accounting
(universalist point of view: search for general principles)

Final step: integration of individual and reference models.

All steps have to be taken in World 2.

1 Motivation 5

1.2 Examples 3 – Two sources for model design

external world ↓ World 1 objects of cognition	phenomenon, individual experience ↓ World 2 knowledge of an individual subject of cognition			model, theory ↓ World 3 common knowledge
	perception, cognitive processes (empiristic) ↓ reconstruct. of World 1 →	memory	learning rationalistic ↓ activations of World 3 ←	
←	↓ creation, induction ↓ design, influence ←	← new ideas, knowledge	→ publi-cation	→

Bi/trilateral semiotic sign		
materialized signifiant, vox	code of interpretation	signifié, conceptus W2|W3
object of cog.		

Model as complex bi/trilateral semiotic sign		
materialized model repre sentation	code of interpretation	model meaning W2|W3
object of cog.		

1 Motivation 6

1.2 Examples – Variables, type models 3

**IS experts do not design models of single real objects, such as of individual customers, of the processing of individual orders, (that is up to the organization's employees)
but general models, such as
the common properties of all of the customers,
of the processing of all of the orders.
This fact is the basis for the rationalization potential of IS.**

Models with variables: type / class models:
(intensional set definition, that is, no enumeration)

**– data model of a set of analogous / equivalent real objects:
tuple of attributes (variables); entity type; OO-class
e.g. customers in general**

**– function model (algorithm) for a set of equivalent problems:
e.g. algorithm for the calculation of the greatest common divisor
of two natural numbers (variables) in general
e.g. algorithm for the processing of orders in general**

Models without variables: individual / instance models:

**– data model of a single real object:
tuple of attribute values (constants); entity; OO-instance
e.g. one individual customer**

**– function model for a single problem:
e.g. for the calculation of the greatest common divisor
of the two natural numbers 12 and 30 (constants)
e.g. for the processing of order no. 4711**

2 Analogy – coincidence of feature values 1

Relation between two objects of cognition
(segments of reality, models; objects, data, processes):
Similarity, comparability, compatibility, associability,
equivalence (in terms of mathematics; → equivalence relation)
– some equal / common features
(tertium comparationis: base of comparison)
– some different features

Example: debtor and creditor management in a company
common: flow of data, goods, money between business partners
different: flow direction (inward, outward),
incoming / outgoing orders,
status of goods (raw material, final product)

Distinction:
– functional analogy: two processes deliver the same result
irrespective of the way of constructing the result
(→ functional model)
Example: copying a text with a copying machine vs. by hand
– structural analogy: two objects of cognition coincide
in selected structural components

We restrict ourselves to the latter kind of analogy.

In biology, analogy has a special meaning (vs. homology):
two recent similar morphological forms
without phylogenetic relationship, without a common ancestor
Examples:
– fins of whales and fish
– wings of bats, birds and flying reptiles

2 Analogy – coincidence of feature values 2

Formalization of the principle of analogy
in order to make models more transparent and better comparable

Feature *F* (based on theory of gestalt):
– dimension *D*
– value *V*
(cf. attributes and attribute values in data modeling)

Example: Feature *F* (*D* color, *V* red)

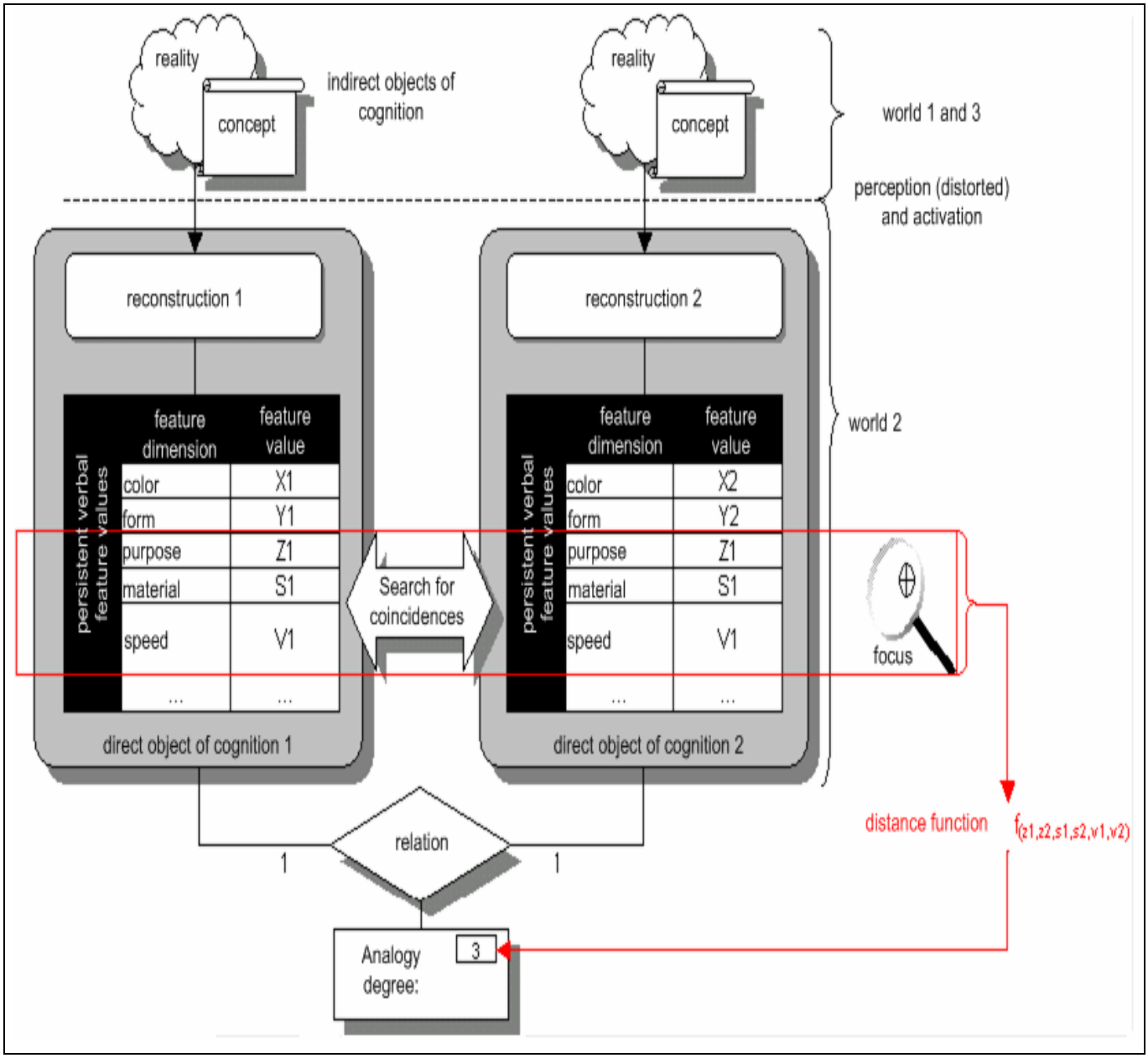
Degree of analogy
between two objects of cognition based on *n* features
calculated by using a weighted measure / function of proximity / similarity:

$$f(V_{11},V_{12},V_{21},V_{22},..,V_{n1},V_{n2}) = \sum_{\substack{i=1..n \\ V_{i1}=V_{i2}}} W_i$$

Pick out the common features in the above set of *n* features.

Two (or *m*) objects of cognition are defined as analogous iff they have
– equal essential (distinctive) features
(are considered as relevant for the comparison)
– different / equal accidental (non-distinctive) features
(don't play any role for the comparison)

2 Analogy – coincidence of feature values 3



Analogies are based upon coincidences of feature values (Holl / Auerochs, Analogisches Denken, 2004, Fig. 2)

3 Analogical thinking 1

3.1 Type construction – induction 1

Type (some sort of a model):
– constituted by equal / common essential features
– found via induction from similar objects of cognition
– a verbal description (umbrella term) can be constructed comprising just the analogous objects of cognition belonging to this type
– different or equal accidental features (e.g. size, number of employees of an organization etc.)

Example:
Customer and supplier (business partners) with
– essential features: name, address, contact person, turnover etc. (short for formal Boolean features (name-yes-no, yes) etc.)
– accidental features: receiver or sender of orders

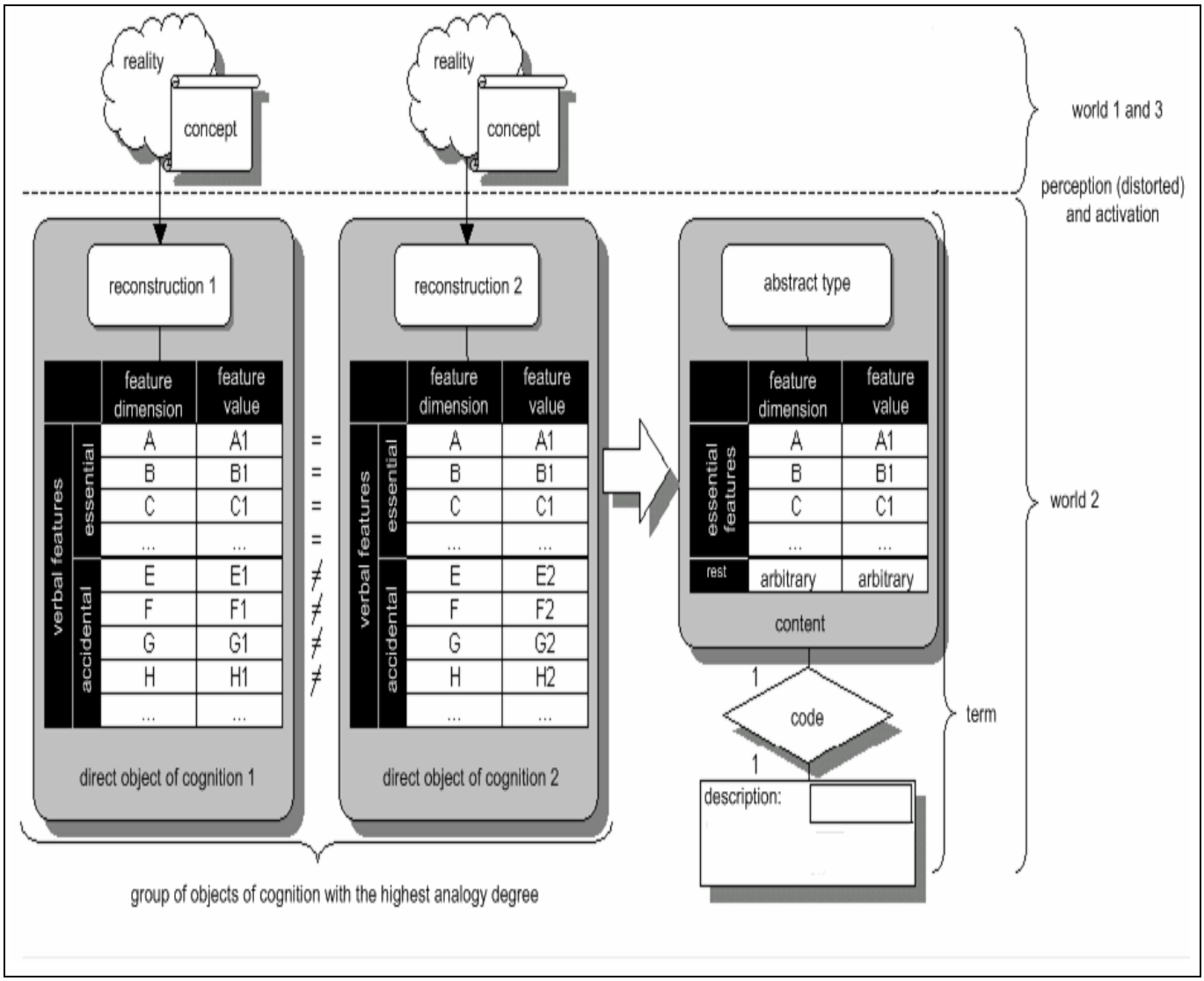
Type construction is done in every natural language where the essential features often remain implicit.

It can be formalized to serve scientific purposes.

Up until now, we distinguish between two kinds of features:
– essential features: common / equal (within a type), distinctive (towards other types)
– accidental features: common or not common non-distinctive

3 Analogical thinking 2

3.1 Type construction – induction 2



Induction step due to postulated analogy
(Holl / Auerochs, Analogisches Denken, 2004, Fig. 3)

3 Analogical thinking 3

3.1 Type construction – induction 3

**An object of cognition can be assigned
to different essential features, that is, to different types,
depending on the compared object of cognition.**

Analogy is always relative to a given set of essential features.

Example:
**Customer 1 – customer 2: customers with
more than 10,000 $ turnover a year**
**Customer 1 – customer 3: customers with
A-rating**
Customer 1 – customer 4: regular customers

Weak analogy: “few” essential features
Strong analogy: “many” essential features

**“An analogy can be more or less detailed and
hence more or less informative.”
(Konrad Lorenz, Analogy as a source of knowledge, 1974, 186)**

3 Analogical thinking 4

3.2 Levels of analogy

Analogy can be defined between objects of cognition on various levels of cognition/existence, between
– 1 objects of cognition of World 1
– 2 types, (parts of) models (World 3 objects of cognition)
– 3 objects of cognition of World 1 and types (World 3)

A type is also an object of cognition!

Examples (case 1):
Socrates, Aristotle;
this swan, that swan
customer 1, customer 2

Example (case 2):
philosopher, human;
ostrich, swan, bird
customers, suppliers

Examples (case 3):
Socrates, humans;
this swan, swans
customer 1, customers

3 Analogical thinking 5

3.3 Reasoning – deduction 1

1 Classification using essential features
2 Transfer using a pars-pro-toto strategy

Example (case 3) with true conclusion:
modus ponens (a sort of a syllogism = logical conclusion)
Humans are mortal.
common accidental (non-distinctive) feature of a type
Classification:
Socrates is a human.
coincidence object of cognition - type
in essential features (or key features, see 4)
Transfer:
Socrates is mortal.
common accidental feature of an object of cognition
(or essential feature if one starts with key features)

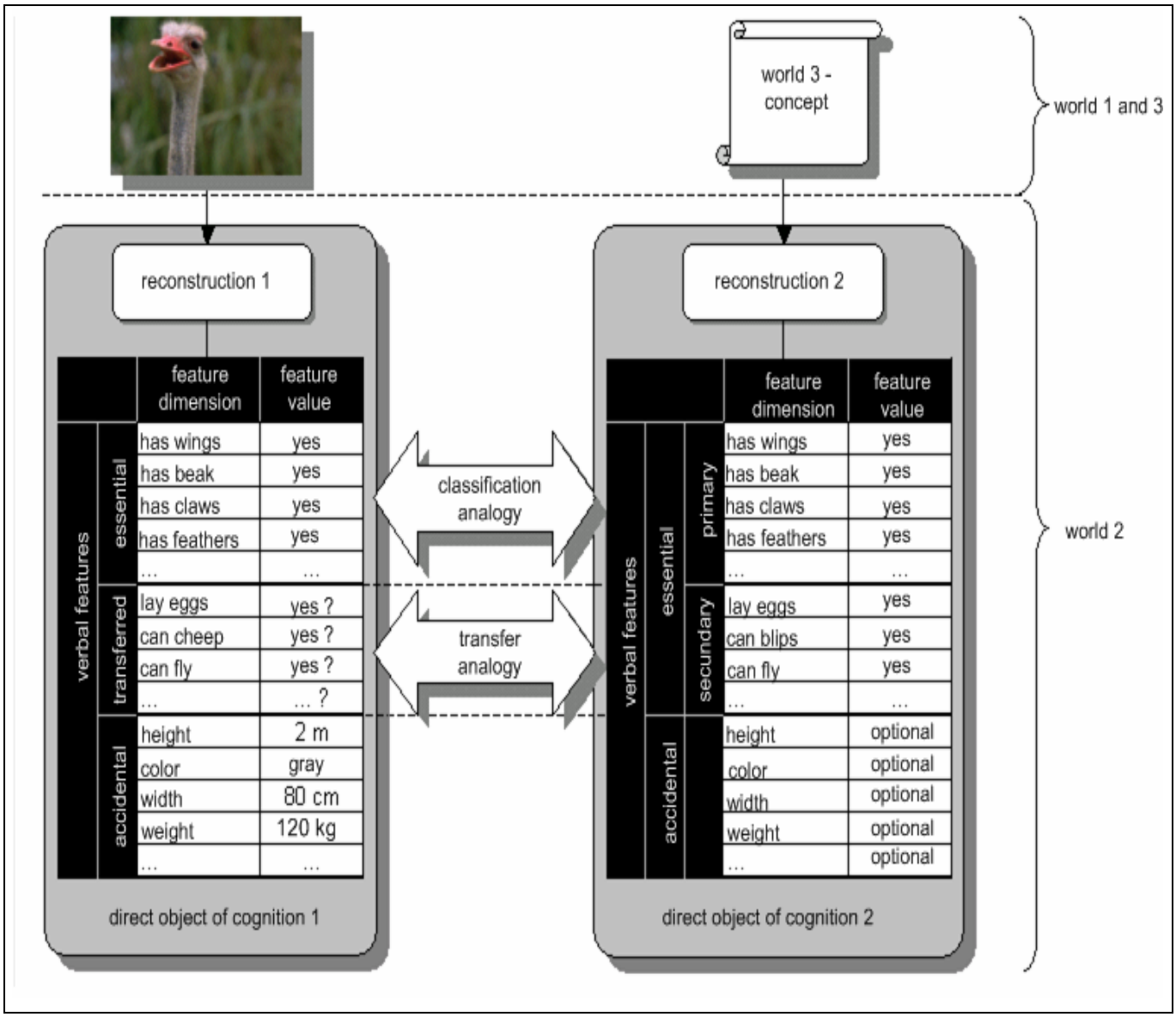
Example (case 3) with false conclusion:
Every swan is white.
This bird is a swan.
This bird is white.

Example (case 2) with false conclusion:
A swan can fly.
Ostrich and swan are analogous (are birds).
An ostrich can fly.

Correctness of assumptions of analogy:
– ⇔ adequacy of selected essential features (or key features)
– cannot be proved.
Risk: This kind of thinking can be a cognitive trap!

3 Analogical thinking 6

3.3 Reasoning – deduction 2



Deductive conclusion with the help of analogy
(Holl / Auerochs, Analogisches Denken, 2004, Fig. 4)

Up until now, we distinguish between <u>three kinds of features</u>:
– essential features (classification)
– common accidental features (transfer)
– different accidental features

3 Analogical thinking 7

3.4 Relation between analogy and induction / deduction

Induction
(due to cognitive dilemma 1)
Starting from some similar / analogous objects of cognition of the same type,
that is objects of cognition with the same essential features,
a theory / model of a common accidental feature is derived.
This is a creative, heuristic (not logical) procedure!

Deduction
Situation:
There is a theory about a common accidental feature of a type.
Classification: The type and some other object of cognition coincide in their essential features.
Transfer – analogical assumption – (logical) conclusion:
Type and object of cognition are analogous,
that is, they coincide in all their essential features,
therefore, the theory applies for the object of cognition.
(analogical transfer of common accidental features)

Or even in a weaker form (see 4):
Classification: The type and some other object of cognition coincide in key features.
Transfer – analogical assumption – (logical) conclusion:
Type and object of cognition are analogous,
that is, they coincide in all their key features,
therefore, the theory applies for the object of cognition.
(analogical transfer of common accidental features and secondary essential features)

3 Analogical thinking 8

3.5 Popper's fallibilism 1

Verification / falsification (Karl Popper)

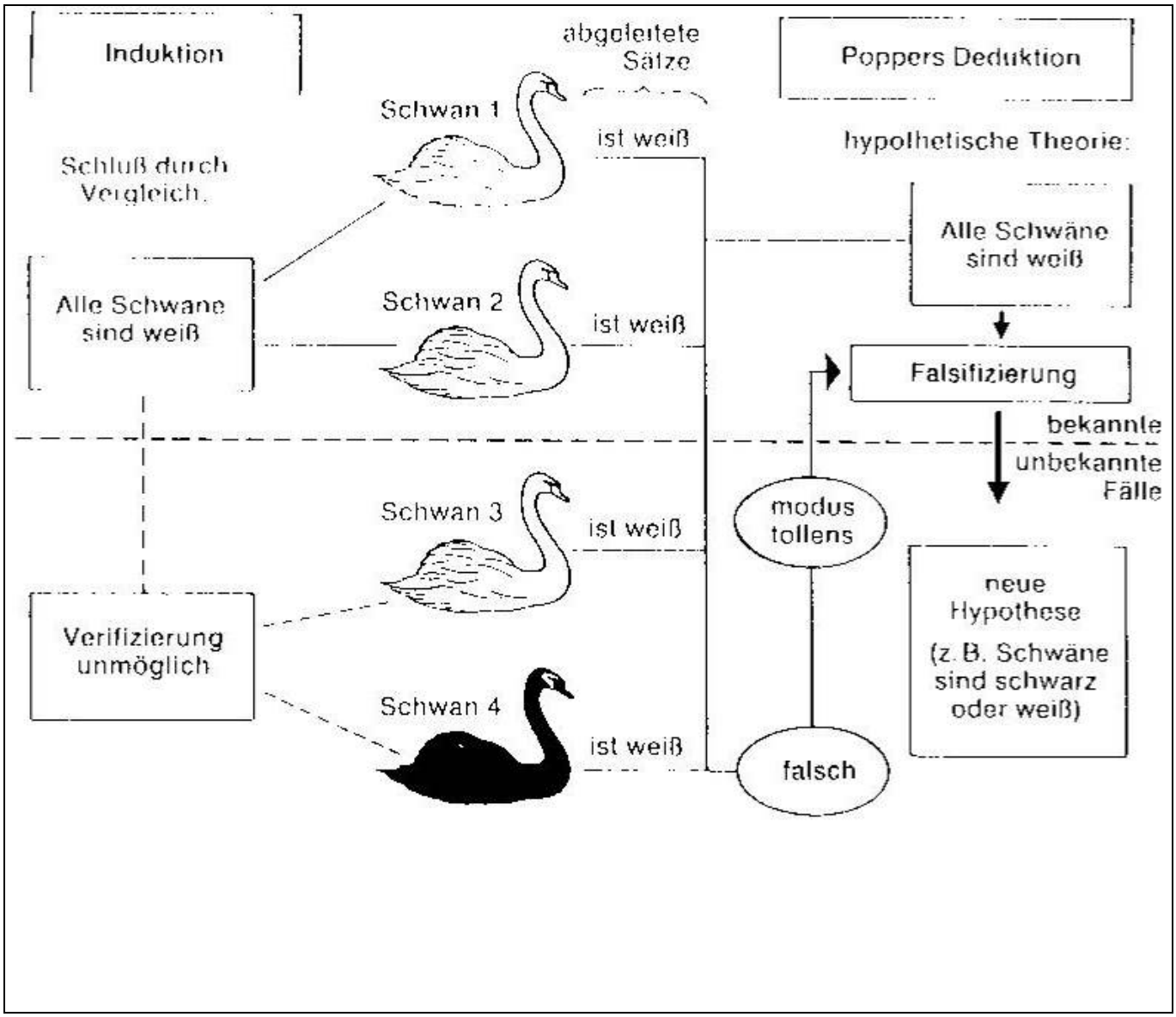
As we do not know all the objects of cognition of a given type, inductively derived theories cannot be proved;
cf. *every swan is white*, *every bird can fly*

That is – as we already know –
the correctness of assumptions of analogy cannot be proved and
the correctness of logical deductions
starting from an inductively derived (only falsifiable) theory cannot be proved.
The results cannot be more true than the pre-conditions.

Deduction works correctly only with well-defined mathematical objects.

3 Analogical thinking 9

3.5 Popper's fallibilism 2



Can swans be black?
(dtv-Atlas Philosophie, *, 228)**

3 Analogical thinking 10

3.5 Popper's fallibilism 3

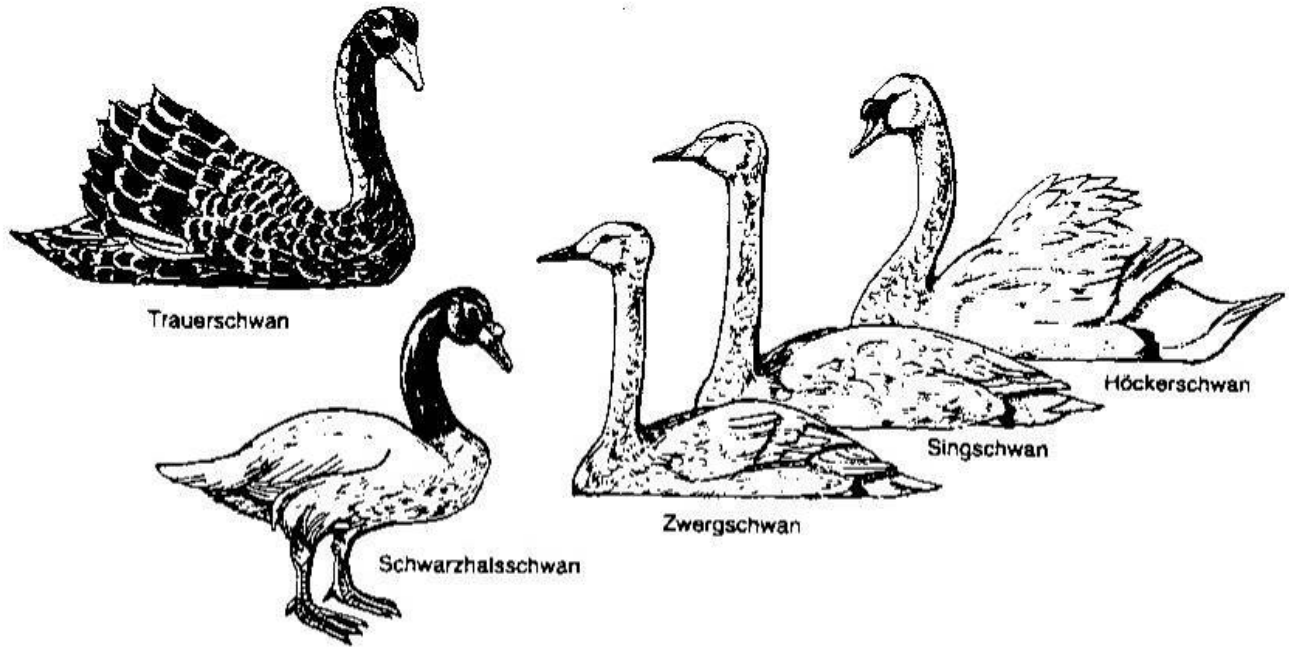


Abb. 14. Die Arten der Schwäne; die europäischen Arten sind jenen der Südhemisphäre (links) gegenübergestellt; dem australischen Trauerschwan und dem Schwarzhalsschwan aus Südchile und dem Feuerland (nach Peterson, Montfort und Hollom 1954, Klös und Klös 1968).

The genus "swan"
(Riedl, Biology of knowledge, 1984, 83)

4 Key feature based analogical thinking 1

Cognitive dilemma 2

(Neolithic) Humans need information to master these situations in the most adequate possible way, but
every object of cognition has numerous features,
among them not easily observable ones and even hidden ones.
The complete observation of all the essential features of an object of cognition is impossible,
it would take too much time or even destroy the object,
but quick reactions are necessary for survival.
cf. lion in the bush, roars, but is not visible

=> the cognitive necessity of partial comparisons
based upon only few features ("key features")

The cognitive strategy of analogical thinking is originally a heuristic cognitive pars-pro-toto (part instead of total) strategy based upon so-called key features
(Konrad Lorenz, Die angeborenen Formen möglicher Erfahrung, 1943, 240: key stimuli, pars-pro-toto reactions)

Key features (directly perceptible, e.g. optical):
– considered as important in the sense of the theory of gestalt
Konrad Lorenz 1959:
"Gestalt perception as source of scientific knowledge."
– (un)consciously, heuristically defined by observer/scientist
– not type-immanent, depending on object and observation

Example:
Customers and suppliers are
companies connected with our own company
by business transactions (data, goods, money)

4 Key feature based analogical thinking 2

Highly significant essential features can serve as key features.

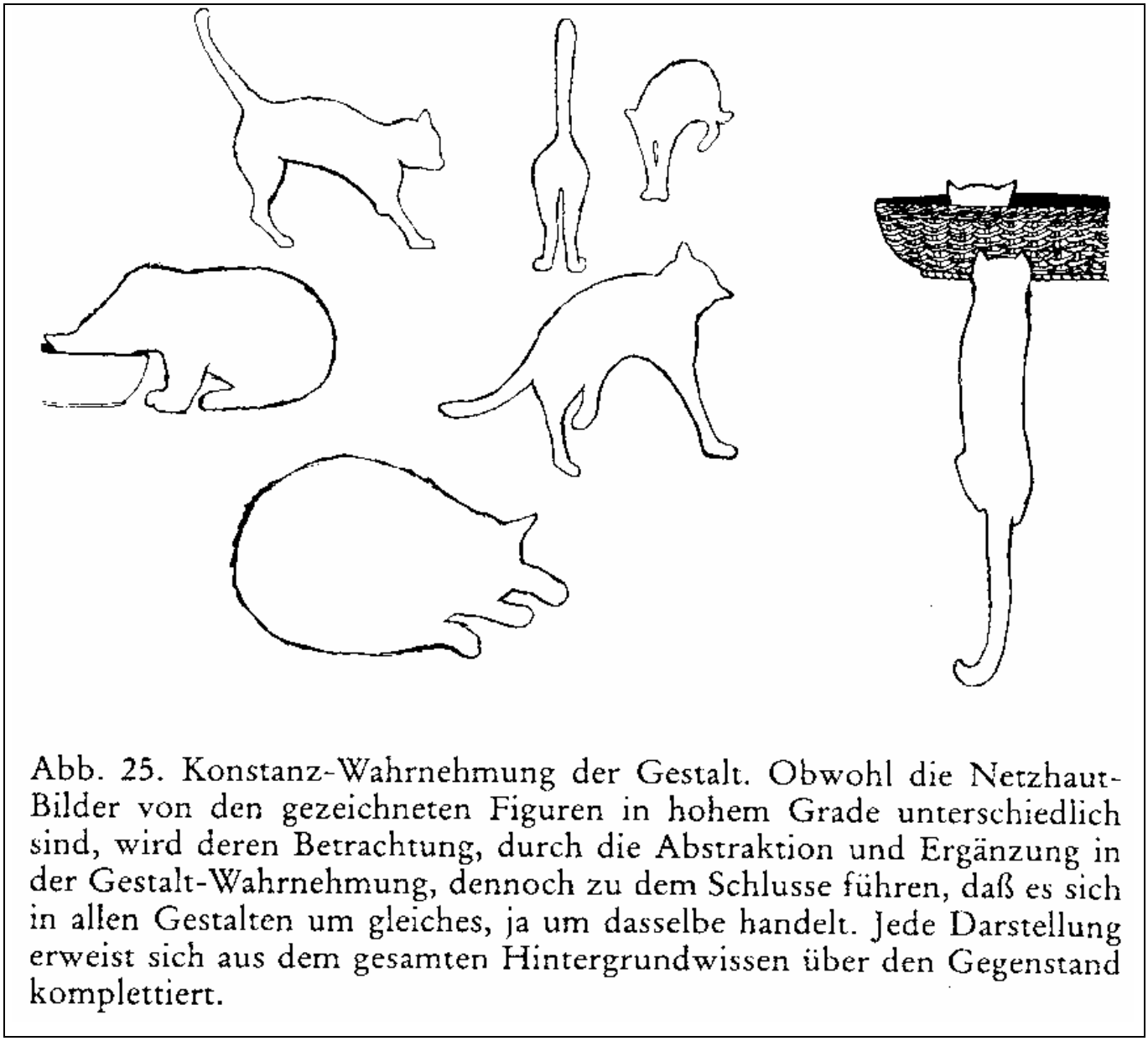


Abb. 25. Konstanz-Wahrnehmung der Gestalt. Obwohl die Netzhaut-Bilder von den gezeichneten Figuren in hohem Grade unterschiedlich sind, wird deren Betrachtung, durch die Abstraktion und Ergänzung in der Gestalt-Wahrnehmung, dennoch zu dem Schlusse führen, daß es sich in allen Gestalten um gleiches, ja um dasselbe handelt. Jede Darstellung erweist sich aus dem gesamten Hintergrundwissen über den Gegenstand komplettiert.

One animal or different animals?
(Riedl, Biology of knowledge, 1984, 167)

4 Key feature based analogical thinking 3

At last, we distinguish between four kinds of features:
– primary essential features (suitable as key features)
– secondary essential features (not suitable as key features)
– common accidental features (transfer)
– different accidental features

Example: human

Primary essential features (suitable as key features)
– shape of the body
– shape of the face
– movement on two legs
– ability to speak

Secondary essential features (not suitable as key features)
– cortex of the brain

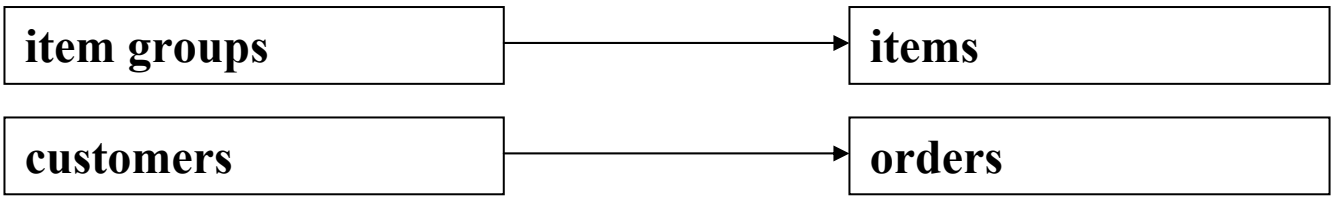
Common accidental features
– mortality

Different accidental features
– color of hair
– color of skin
– height
– sex

Essential features are common /equal and distinctive.
Accidental features are common or not and non-distinctive.
Secondary essential features and common accidental features can be used for analogical transfer.

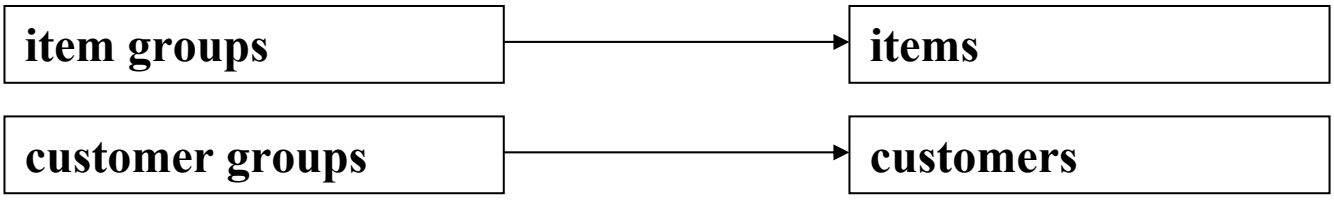
5.1 Data models: What degrees of analogy occur?

1 mere syntactic

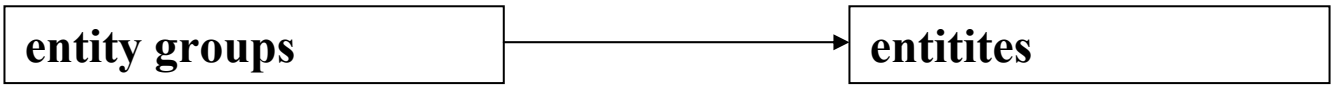


analogy: one-to-many relationship

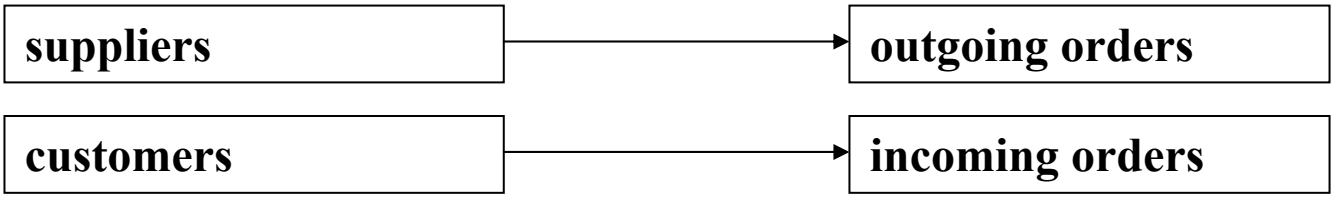
2 low degree, weak semantic



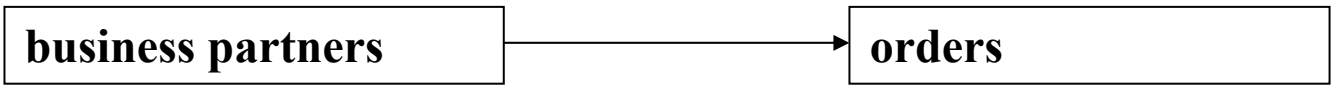
umbrella terms:



3 high degree, strong semantic



umbrella terms:



5.1 Data models: generic models; reference models

Two data models are analogous iff

(1) they are syntactically equal, that is
the structures of entity types and relationships are equal,
the mere diagrams without text coincide

(2) they are semantically analogous in the same degree, that is
syntactically corresponding entity types
are analogous in the same degree, that is
an umbrella term can be constructed
for each pair of corresponding entity types

Example

creditor	debtor	umbrella terms generic model
supplier groups ↓ suppliers ↓ outgoing orders ↓ order lines ↑ raw materials ↑ material groups	customer groups ↓ customers ↓ incoming orders ↓ order lines ↑ products ↑ product groups	business partner gr. ↓ business partners ↓ orders/contracts ↓ order lines ↑ items ↑ item groups

→ one-to-many relationship

5.1 Data models: What about partial analogies?

Complete model analogies are rare, that is syntactic equality is often not complete.

Example 1:
Number of order lines
orders with only one or with more order lines

customers **library users**	→	**orders** **-**	→	**order lines** **borrow transactions**	←	**products** **books**

Example 2:
Individual identifiability of items
individually identifiable items (library books, cars) or not individually identifiable items

borrow transactions **order lines**	←	**books (copies)** **-**	←	**books (titles)** **products**

5.2 Main functional areas of a company

Company management
Information management
Financial management, investments
Personnel management = human resources management
Accounting (ledger, accounts receivable, accounts payable)
Marketing, distribution, sales, order management
Materials management, inventory, purchasing, procurement
Production
Quality assurance/management
Product development, research and development
Customer support/service

Decomposition into smaller functional areas
which can be assigned to
employees (employee groups) in a matrix

6 References

pdf-files of my own publications: see my homepage.

Holl, Alfred:
Empirische Wirtschaftsinformatik und evolutionäre Erkenntnistheorie.
In: Becker, Jörg et al. (ed.): *Wirtschaftsinformatik und Wissenschaftstheorie. Bestandsaufnahme und Perspektiven*.
Wiesbaden: Gabler 1999, 163-207, ISBN 3-409-12002-5.
English translation on my homepage.

Holl, Alfred; Auerochs, Robert:
Analogisches Denken als Erkenntnisstrategie zur Modellbildung in der Wirtschaftsinformatik.
In: Frank, Ulrich (ed.): *Wissenschaftstheorie in Ökonomie und Wirtschaftsinformatik. Theoriebildung und –bewertung, Ontologien, Wissenmanagement.*
Wiesbaden: DUV 2004, 367-389, ISBN 3-8244-0738-8.

Alfred Holl

Meta-Models of IS Modeling Approaches

The primary focus are principles of modeling and basic elements of models independent of their graphical representation.

Only the secondary focus are individual notations of model representations (semantic networks with nodes and arcs / edges), such as Jackson, SA, UML etc.

0 Aspects of IS models and their notations (multi-perspectivity)

1 Rules for graphical model representations

2 Static function(-oriented) models: function structure models

3 Dynamic data(-oriented) models: information flow models

4 Static data(-oriented) models: data (structure) models

5 Dynamic function(-oriented) models: control flow models

6 Conclusion

0 Aspects of IS models and their notations

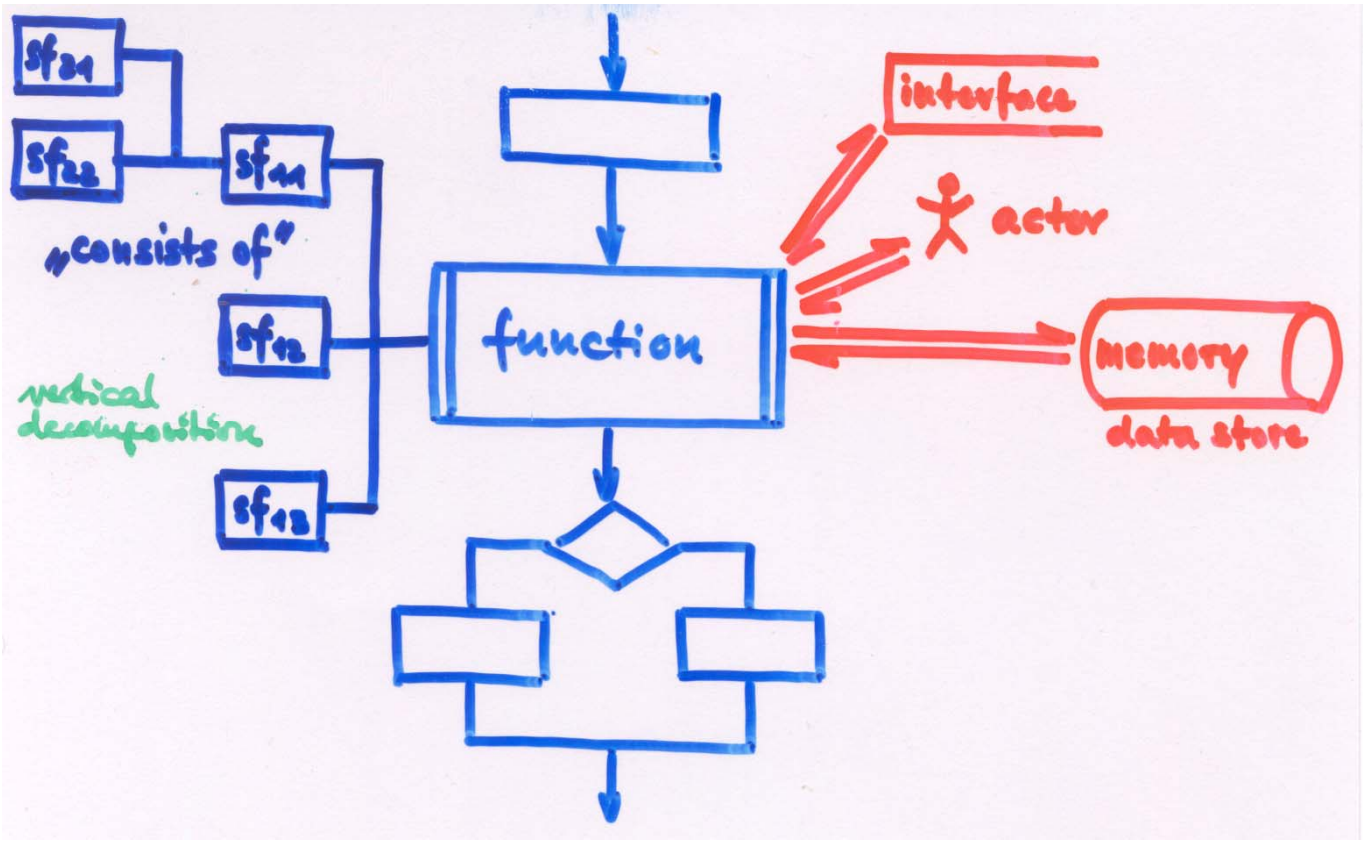
Horizontal multi-perspectivity / decomposition: static and dynamic data and function models 1

	static/structure models	dynamic/behavior models
data models	data (structure) models: data structure diagrams; entity-relationship models (ERM) UML class diagrams	information flow models: information / data flow charts / diagrams; Structured Analysis (SA); UML use case diagrams
function models	function structure models: compositional function trees; Jackson trees	control flow models: algorithms (functions); Nassi-Shneiderman diagrams, block diagrams (flow charts); business process models; UML activity diagrams; (UML sequence diagrams)

Each of the four aspects represents a certain perspective.

0 Aspects of IS models and their notations

Horizontal multi-perspectivity / decomposition: static and dynamic data and function models 2



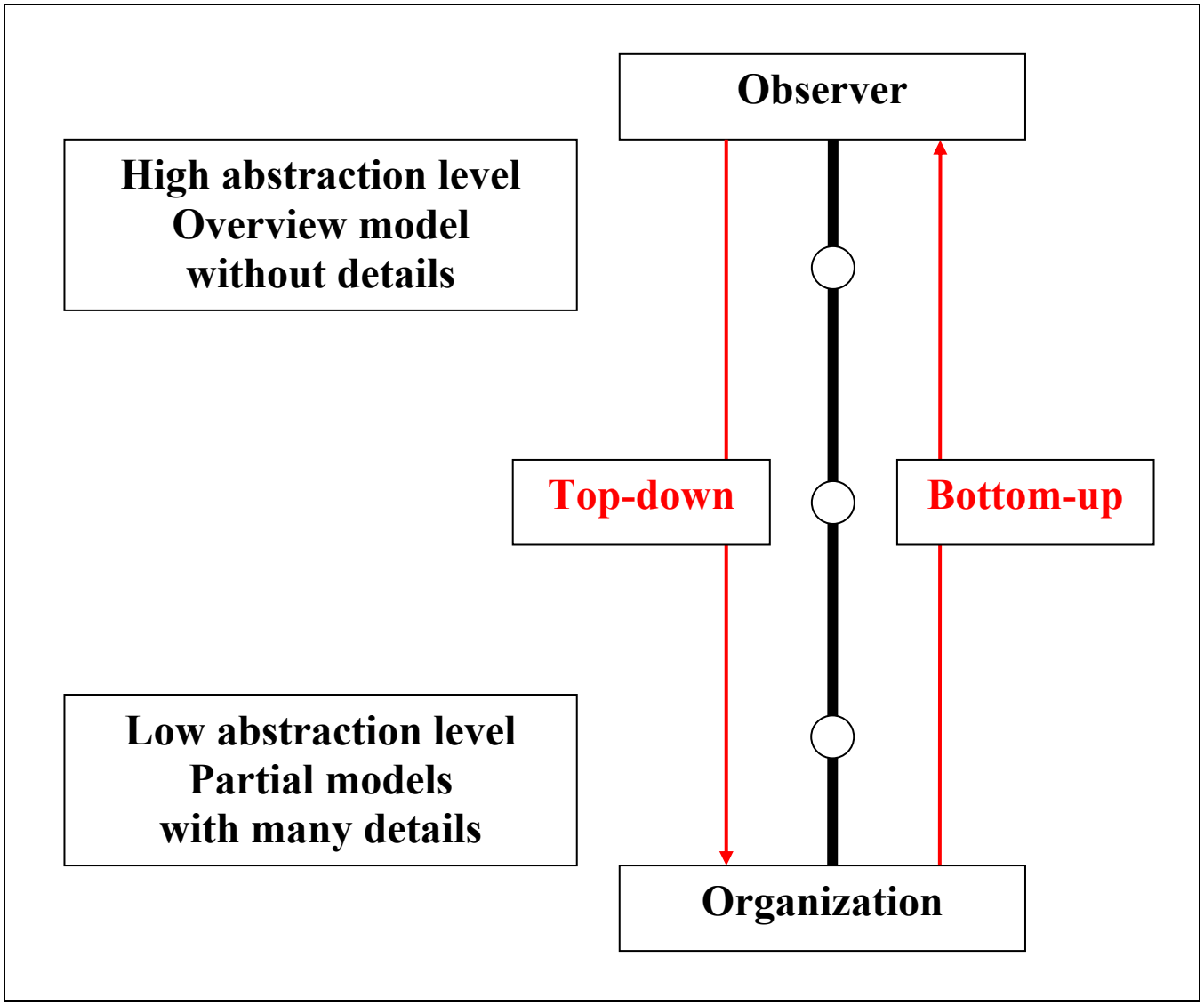
Static function model: function structure model irrespective of tests, iterations, sequences	Dynamic function model: control flow model	Dynamic data model: information flow model	Static data model: data structure model

0 Aspects of IS models and their notations

Vertical multi-perspectivity / decomposition: levels of abstraction

Using design methods (top-down, bottom-up, inside-out), models have to be decomposed into small and transparent partial models on different levels of abstraction (hierarchical levels with different degrees of abstraction).

Every level of abstraction represents a certain perspective.



1 Rules for graphical model representations

All of the nodes have unequivocal (distinct) names.

Vertical multi-perspectivity

Hierarchic decomposition (top-down refinement)
of all structure components on different abstraction levels
→ compatibility of adjacent abstraction levels

Horizontal multi-perspectivity

→ compatibility of adjacent model aspects

---

7 to max. 9 nodes per diagram ('chunk', 'Superzeichen')
cf. Miller, George A.: *The magical number 7, plus or minus 2*, The Psychological Review 63(1956), 81-97

→ transparent arrangement of the symbols
→ sequence of comments according to this arrangement

Diagrams and comments mutually complete each other.

2 Function structure models

function trees, ‘consists of’-hierarchies

irrespective of control flow (conditions, iterations)

3.1 Information flow models: Elements 1

Nodes:

1. functions

– template for function names: verb + noun (object)
→ quantifier if singular and plural cannot be distinguished

2. stores, memories

Types of information media ('Datenträger'; to be marked in diagrams):
– digital stores / media
– paper (printout, document ('Beleg', 'Schriftstück'), card index ('Kartei'), binder ('Ordner'))
– mental memories, skills, knowledge (head)
→ type purity (only one information type in every store)
→ type compatibility of the attached information flows
→ type conversion only with functions

3. actors (intra-system and extra-system (SA -interfaces); extra-system ones can be intra- or extra-organizational): function owners, responsible persons ('Aufgabenträger', 'Funktionsträger'), persons involved ('Beteiligte'); institutions; IT systems

3.1 Information flow models: Elements 2

Arcs:

data / information flows (type purity)
– between functions and stores / actors
→ not between different functions
– distinction analogous to store types
(digital, paper, oral communication)
– flow direction (cf. store access modes read / write)
if necessary material and money flows

Not: processes, procedural structures
Auxiliary approaches:
– graphic arrangement of the function symbols
– [event flows (a second arrow type; cf. CASE4.0)]

3.2 Information flow models: Notations

Traditional data flow chart

Process matrix (data flow chart in the form of a table):
– functions, stores, access mode

Structured Analysis (SA) according to Tom de Marco 1978:
– level 0: context diagram (system delimitation)
– only extra-system (external) interfaces / actors

UML use case diagram (‘Anwendungsfalldiagramm’)
– intra-system and extra-system actors

4.1 Data structure models: Elements
1 Entity / object types 1

Nodes:

Entity types, object types, (object) classes

1. attribute structure (attribute ≈ property dimension)
– attributes of the objects (instances) of a class (Instanzattribute)
– attributes of the corresponding class (Klassenattribute)

2. attached procedures, elementary functions, services, 'methods' (called by messages): data encapsulation
– services of the objects (instances) of a class (Instanzmethoden)
– services of the corresponding class (Klassenmethoden)

3. object types
– transient: exist during run-time only
– persistent: stored in permanent memories

→ individual elements: entities, objects, object instances

Relation between SA and OO
– refine SA data stores to object types / classes
– refine SA functions to OO services

4.1 Data structure models: Elements
1 Entity types 2

Entity, object
not: a real object, but: a model, a tuple of attribute values

Entity structure: attribute structure
2 entities have the same structure iff they have the same attributes

Set of entities
Set of arbitrary entities with the same structure

The following problem leads to a definition of an entity type:
A customer table shall only contain one address for each customer, the current address.

Entity type: 2 properties
structure property: invariable attribute structure
compatibility property: definition of a primary key
2 entities with the same structure are compatible iff they can be elements of the same entity set iff their primary key values are different (entity integrity).

Entity set
Set of compatible entities with the same structure (table).

There are always several entity sets of an entity type.
Not every set of entities with the same structure is an entity set.

	World 1	World 3
single object	one real object	entity
set - type	set of homogeneous real objects	entity type

Data store (digital, paper, head): can contain several entity types

4.1 Data structure models: Elements
2 Relationships, associations 1

Arcs:

Relationships, associations (OO) regard
the interrelations between entity types from 3 points of view:

1. numeric classification: cardinality, multiplicity (OO)
– n:m many-to-many relationship
– 1:n one-to-many relationship
– 1:1 one-to-one relationship
– 1:c conditional relationship (c = 0, 1)
– special two-dimensional relationships such as c:c, c:n
– multidimensional relationships: n:m:p, n:m:p:q (e.g. time-table)

CAUTION: The cardinality is also a type
that is, it can have different values for individual entities.

Example:
A one-to-many relationship "customers → orders" can contain
– individual customers with many (two or more) orders
– individual customers with one order
– individual customers with no orders

2. syntactic description: primary key → reference key
– on entity type level: key attributes
– on entity level: key attribute values

→ Referential integrity required!

4.1 Data structure models: Elements
2 Relationships, associations 2

3. semantic interpretation

3.1 simple association without any semantic interpretation

Examples:
1:n customers – orders
1:c employees – (head of) – department

3.2 compositional relationship:
‘consists of’, ‘whole-part’, aggregation

Composition (special case of aggregation): existential dependence

Examples:
1:n orders – order lines
1:c car – air condition

4.1 Data structure models: Elements 2 Relationships, associations 3

3. semantic interpretation

3.3 taxonomic relationship:

'is a', generalization (umbrella term) / specialization
inheritance of attributes and services
polymorphism of services

Example:
1:c business partners – customers in combination with
1:c business partners – suppliers

Interpretation different in OO class diagrams:
no instances of abstract classes

general concept	basic class abstract class	↑ generalization
special concept	derived class	↓ specialization (with inheritance)

4.2 Data (structure) models: Special models and their notations

1. Pure data models in general

all features mentioned without services (attached procedures)

ERM: entity-relationship model (Chen) and its extensions

DSD: data structure diagram (Bachman) and
Oracle diagram:
no semantic interpretation of relationships

2. 3NF models: special pure data models reduced to axioms (controlled redundancy)

– no services (attached procedures)
– no semantic interpretation of relationships
– one-to-many relationships only

favorite diagram: DSD

3. Static object models

Thesis: should be based on 3NF data models in order to have a stable basis quite independent of subjective influence

all features mentioned
logical primary and reference keys are not always used

UML class diagram

5.1 Control flow models: Elements 1

In the following, behavior meta-models will be examined from the point of view of information systems.

That is, there will be a focus on the activity-on-node variant.

The activity-on-arc variant (state transition networks, Petri nets), which is important for theoretical computer science approaches, will be excluded.

5.1 Control flow models: Elements 2

Nodes:

1. function, action (computer-aided or not)
   function unit, function module

– name from the view of the organization
– decomposition-marker: reference to sub-processes
– algorithm, internal logic in a note
– duration, start time, end time
– features, feature values (→ theory of gestalt)
– IT support: computer-aided or manual

2. initiating and resulting events

3. actor: person/role/department responsible for the action
   partly connected with data flow

4. external (business/communication) partners
   connected with data flow

5. data stores accessed: input data and output data
   connected with data flow

6. resources used (machines etc.)

	World 1 (reality)	World 3 (model)
single object, "instance"	one individual course of events in an organization	business process instance
set - type of similar objects	set of homogeneous courses of events	business process type

5.1 Control flow models: Elements 3

Arcs:

1. control flow: temporal interrelation of functions
(cf. structured programming)

– temporal succession: sequence (predecessors and successors)
mandatory or arbitrary (pseudo-parallelism)

– condition: alternative, selection (IF, XOR)
case discrimination (CASE)
or complex rule (decision table)
disjoint and complete (if incomplete a standard path)

– repetition: iteration, loop (WHILE or REPEAT)
test-first loop and test-last loop

– recursion

– simultaneousness: parallel processing, parallelism (AND)
mandatory or arbitrary (pseudo-parallelism)

CAUTION:
all control flow elements without the mere sequence must have
a divergent delimitor (begin) and
a convergent delimitor (end, synchronization);
the delimitors have to be arranged symmetrically in a diagram:
IF – ENDIF, CASE – ENDCASE, LOOP – ENDLOOP etc.

2. data flow (only partly)

3. mere connectors to actors and resources used

5.2 Control flow models: Special models and their notations 1

1. Classical notations

1.1 Traditional notations for structured programming

flow chart, block diagram ('Programm-Ablauf-Plan')
structure diagram, structogram (Nassi-Shneiderman diagram)
Jackson tree
- Jackson structured design (JSD)
- Jackson structured programming (JSP)

functions and control flow

1.2 Decision table

complex conditions and functions: rules

1.3 Network model(ing technique)

functions, sequence, parallel processing,
duration, start time, end time
→ critical path

1.4 Control flow plus data flow

HIPO: hierarchy plus input-process-output (Mills 1972, IBM)
functions, control flow, data stores, data flow

5.2 Control flow models: Special models and their notations 2

1.5 Swim lane diagrams

functions, control flow, responsible departments
predecessor of UML activity diagram
Arbeitsablaufdiagramm: Arbeitsschritte – Abteilungen
Organisationsprozessdarstellung (H. F. Binner)

2. Business process models

Event-driven process chain (EPC)
OMG Standard: Business Process Diagrams (BPD) using Business Process Model and Notation (BPMN)

functions, control flow
events
actors, partners, data stores, resources, data flow
swim lanes (responsible departments)

3. Dynamic object models

UML activity diagram

functions, control flow
events (non standard)
actors, partners, data stores, resources, data flow
swim lanes (responsible departments)

UML sequence diagram

classes, elementary functions called by messages, control flow

6 Conclusion: Model notations as semantic networks

All the models mentioned can be represented by semantic networks with nodes and arcs.

Analogies:
data model: entity types and relationships
process model: functions and control flow
different types of control flow correspond to different types of relationships between entity types
in addition:
data flow models contain different types of nodes:
functions, actors and data stores (entity types)

---

7 References

Böhm, Corrado; Jacopini, Giuseppe:
Flow diagrams, Turing machines and languages with only two formation rules.
***Communications of the ACM* 9(1966) 5, 366-371.**

Dijkstra, Edsger:
GOTO statement considered harmful.
***Communications of the ACM* 11(1968) 3, 147-148.**

Alfred Holl

Structured design of process models, structured business process modeling

1.1 Motivation 1

BPM is a type of process (dynamic function) modeling,
a subtype of behavior modeling,
represented by
– event-driven process chain [A. W. Scheer, ARIS]
– UML activity diagram
– BPMN business process modeling notation

What other modeling approaches belong to this type?
Control flow modeling in program design and programming
represented by
– block diagram (flow chart)
– Nassi-Shneiderman diagram
– UML activity diagram

Comparison of current diagrams:
– BPM unstructured: spaghetti [Scheer 1994]
– control flow diagram structured

Control flow modeling styles		BPM styles	
1950s 1960s	Spaghetti code programming and spaghetti design	late 1980s	Spaghetti BPM
early 1970s	Structured programming and structured design	2010 ?	Desire: Structured BPM (not only in WFM)

Historic comparison (Holl / Valentin 2004)

Why did BPM not realize the similarity and learn from structured program design?
– BPM ← business, information systems
– structured program design ← computer science

1.1 Motivation 2

The problem of structuring is independent of the notation used.

"There is nothing to prevent the systems analyst from creating an arbitrarily complex, unstructured flowchart." [Yourdon 1989,222]

Not only
– mapping of spaghetti reality
but even
– higher complexity than the complexity of the reality

"Unless great care is taken, the flowchart can become incredibly complicated and difficult to read." [Yourdon 1989, 290]

Only Nassi-Shneiderman is restrictive
with regard to structuring,
but it is not applied to BPM

"The Nassi-Shneiderman diagrams are generally more organized, more structured and more comprehensible than the typical flowchart." [Yourdon 1989, 224]

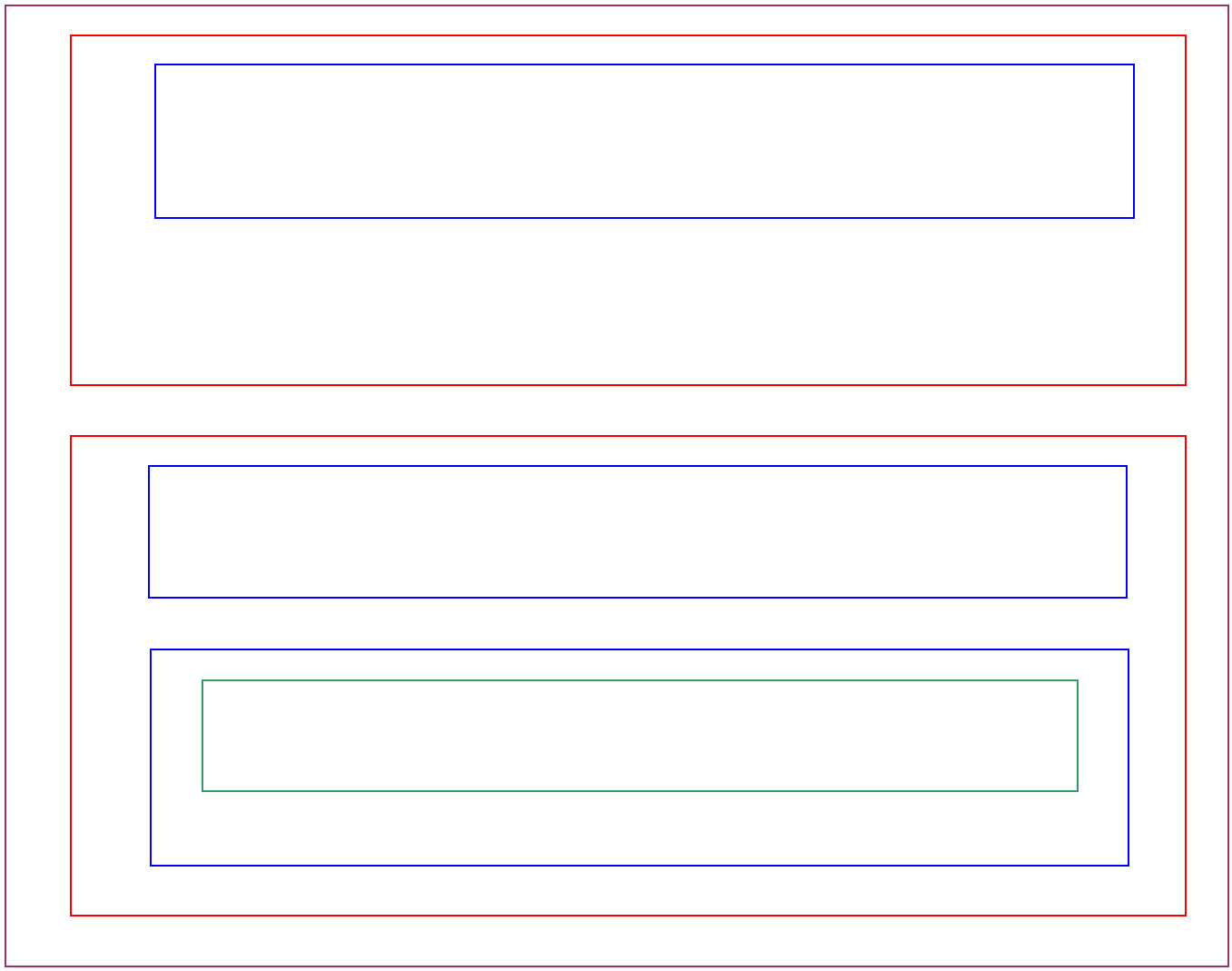
Improvement

"To create a structured flowchart, the systems analyst must organize his or her logic with nested combinations of the flowchart symbols (by Böhm-Jacopini)." [Yourdon 1989, 222]

Böhm-Jacopini proof 1966 shows the sufficiency of sequence, selection (alternative / test) and repetition (iteration) for every mathematically describable process.

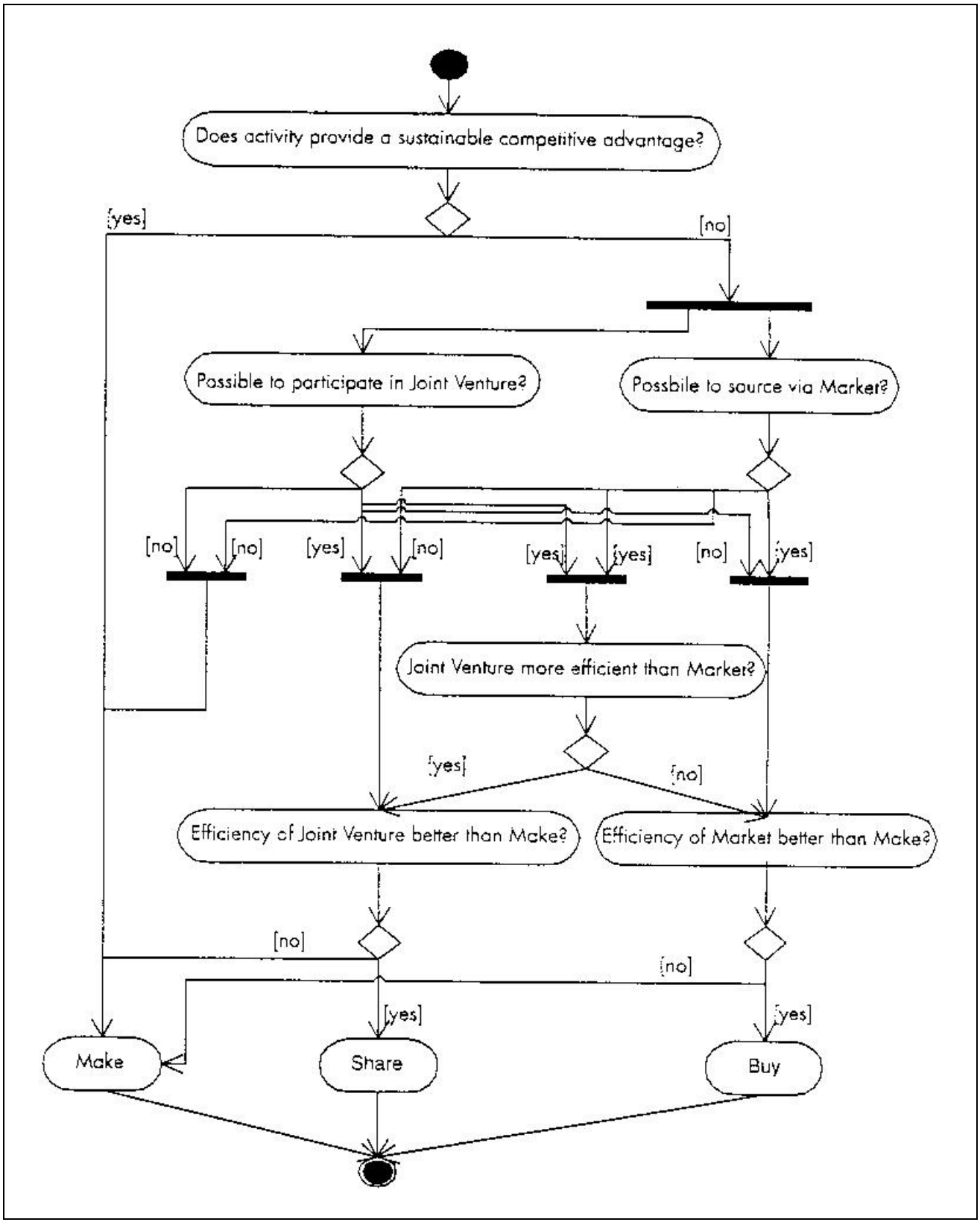
1.1 Motivation 3

Nested structure components



cf. latest version of UML sequence diagrams

1.2.1 Unstructured examples: current literature 1



(Wirtschaftsinformatik 46(2004) 207)

1.2.1 Unstructured examples: current literature 2

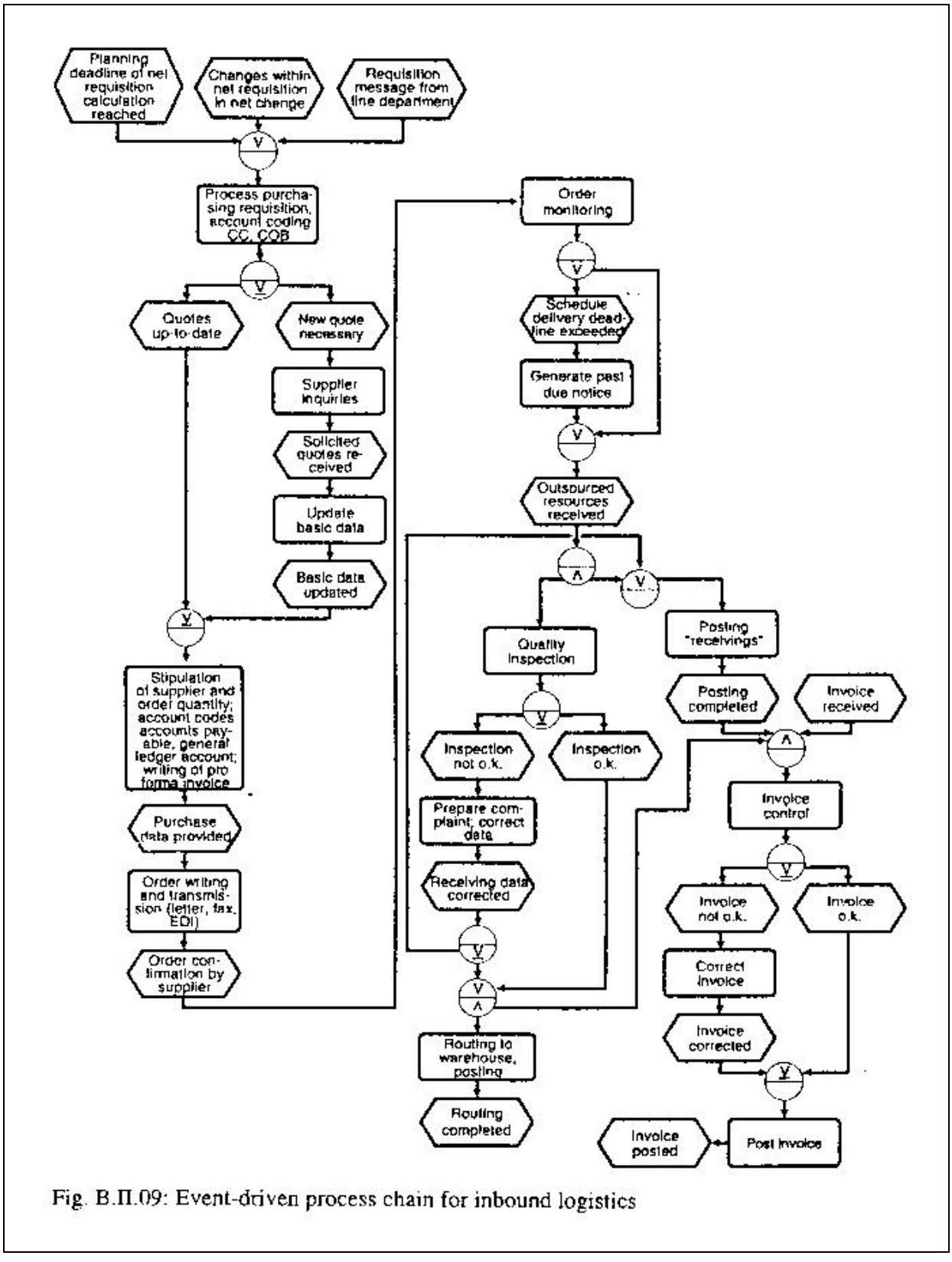


Fig. B.II.09: Event-driven process chain for inbound logistics

(Scheer, Business Process Engineering, 1994: 404)

1.2.1 Unstructured examples: current literature 3

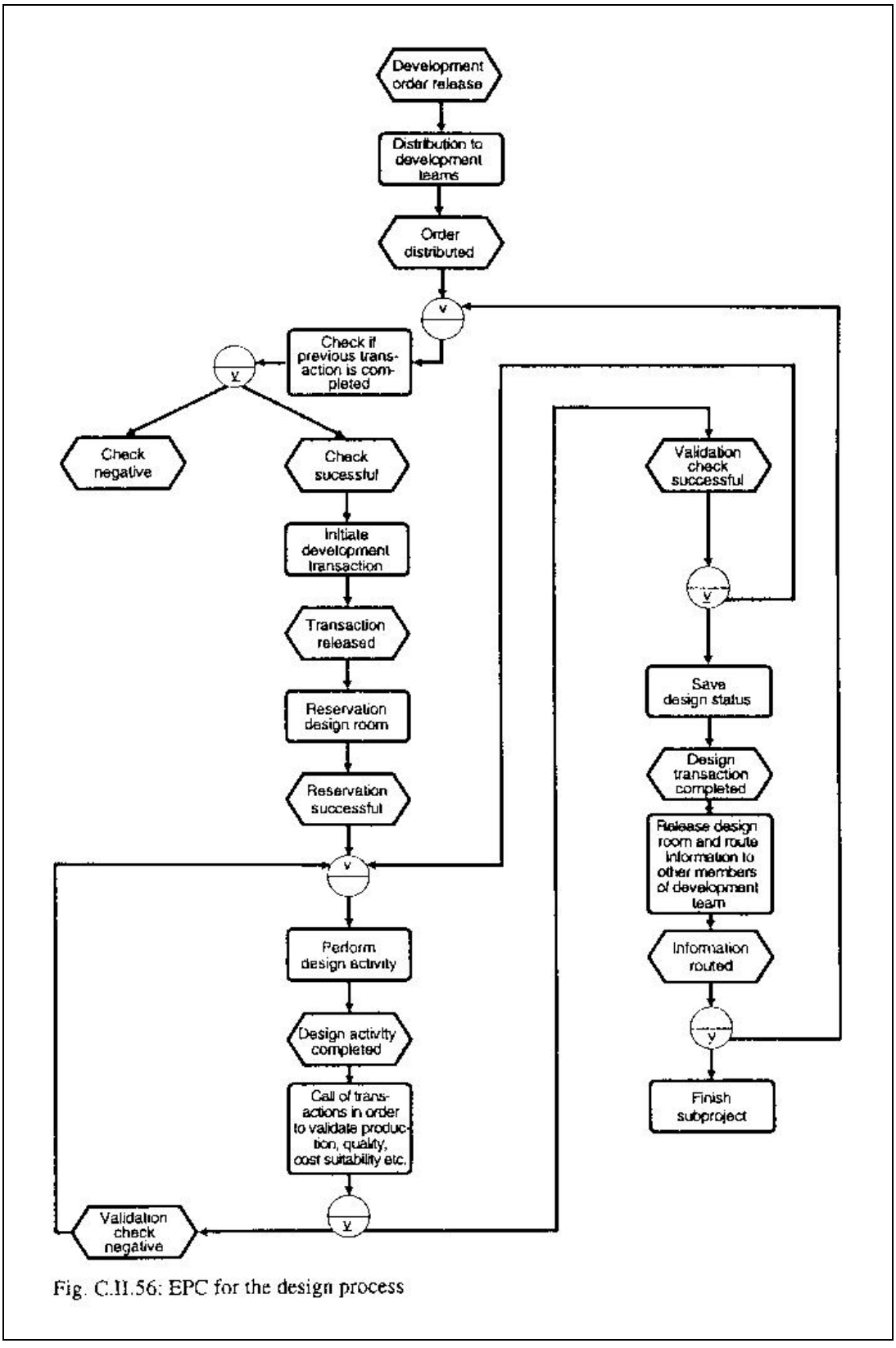


Fig. C.II.56: EPC for the design process

(Scheer, Business Process Engineering, 1994: 589)

1.2.1 Unstructured examples: current literature 4

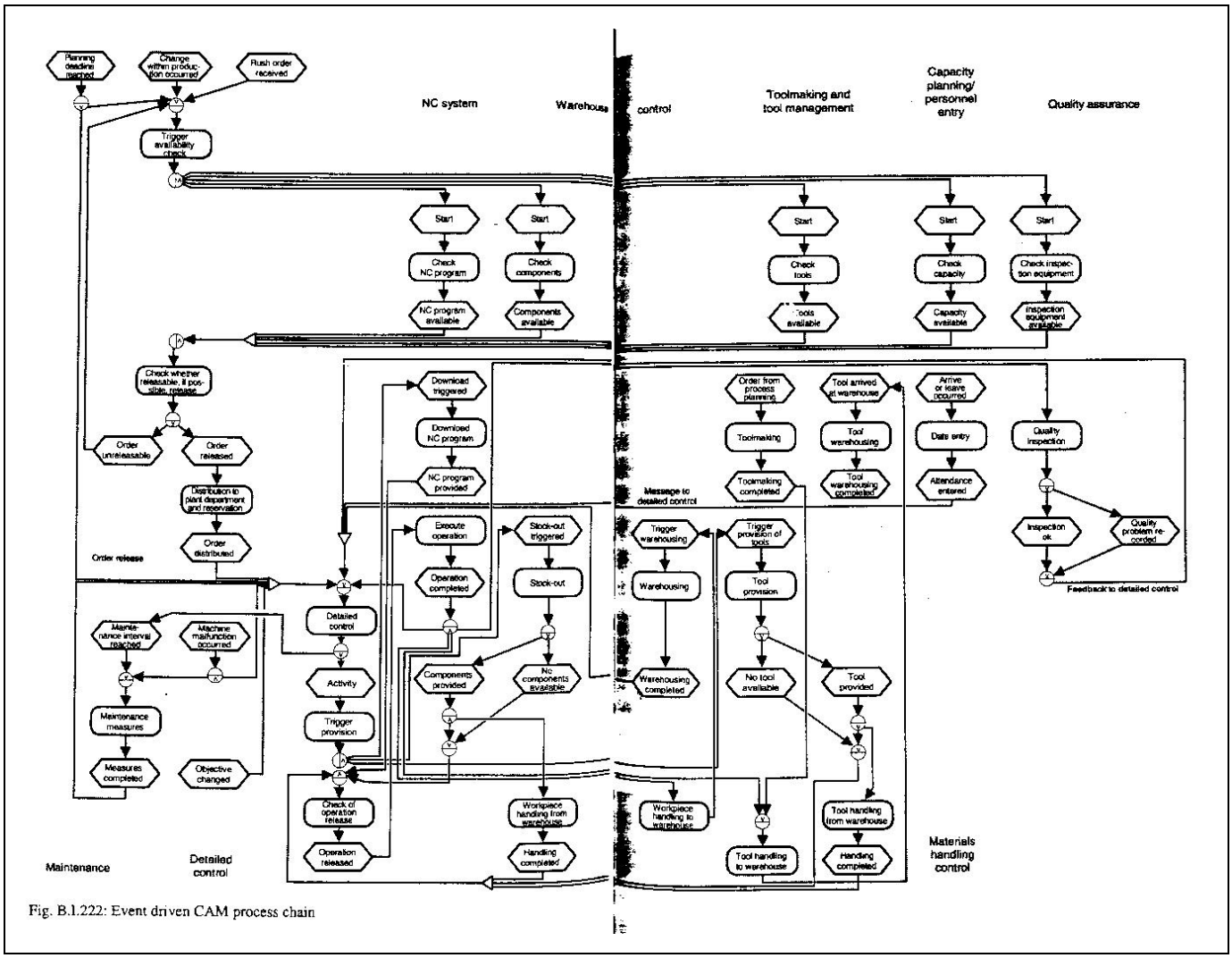
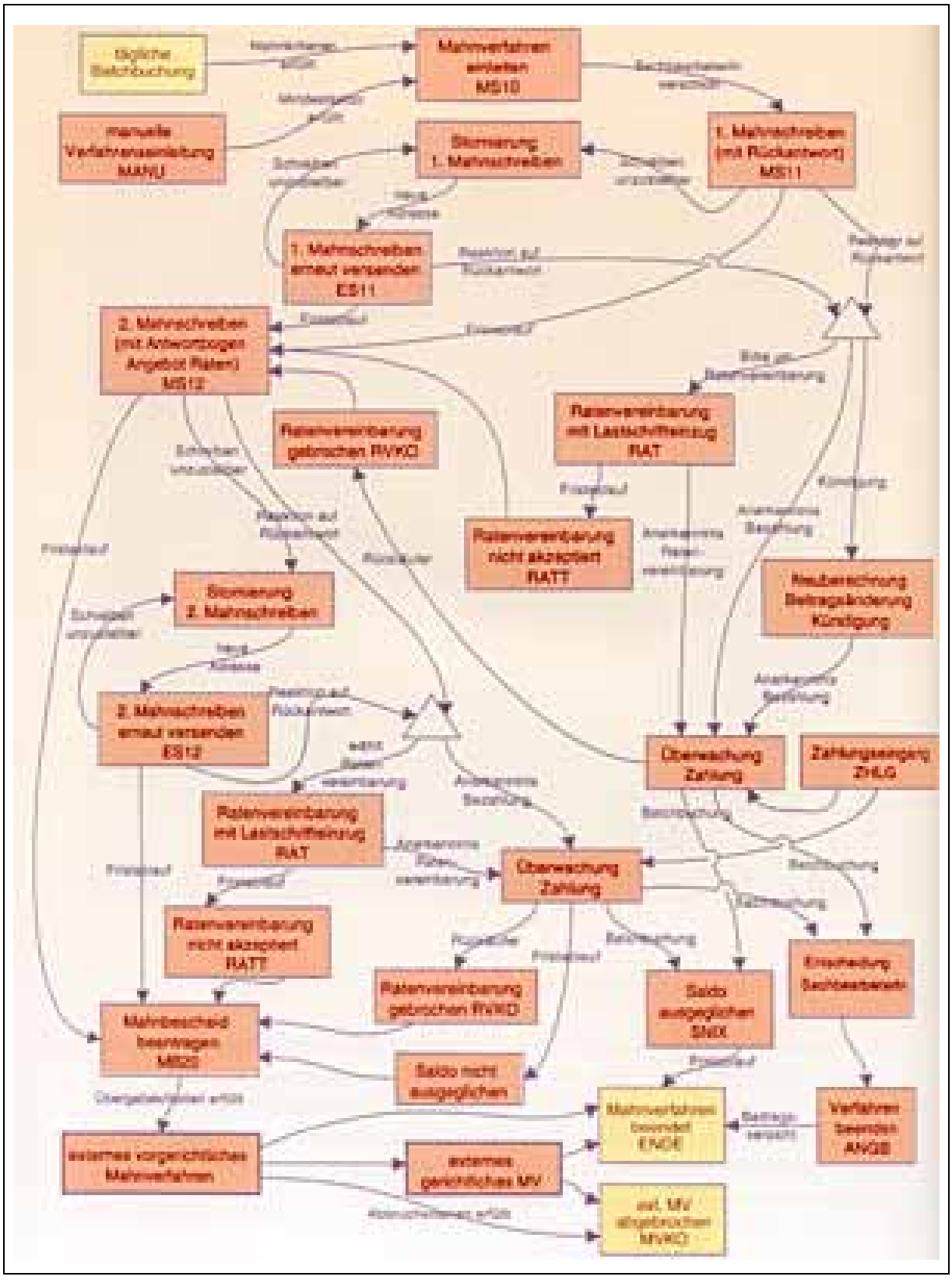


Fig. B.1.222: Event driven CAM process chain

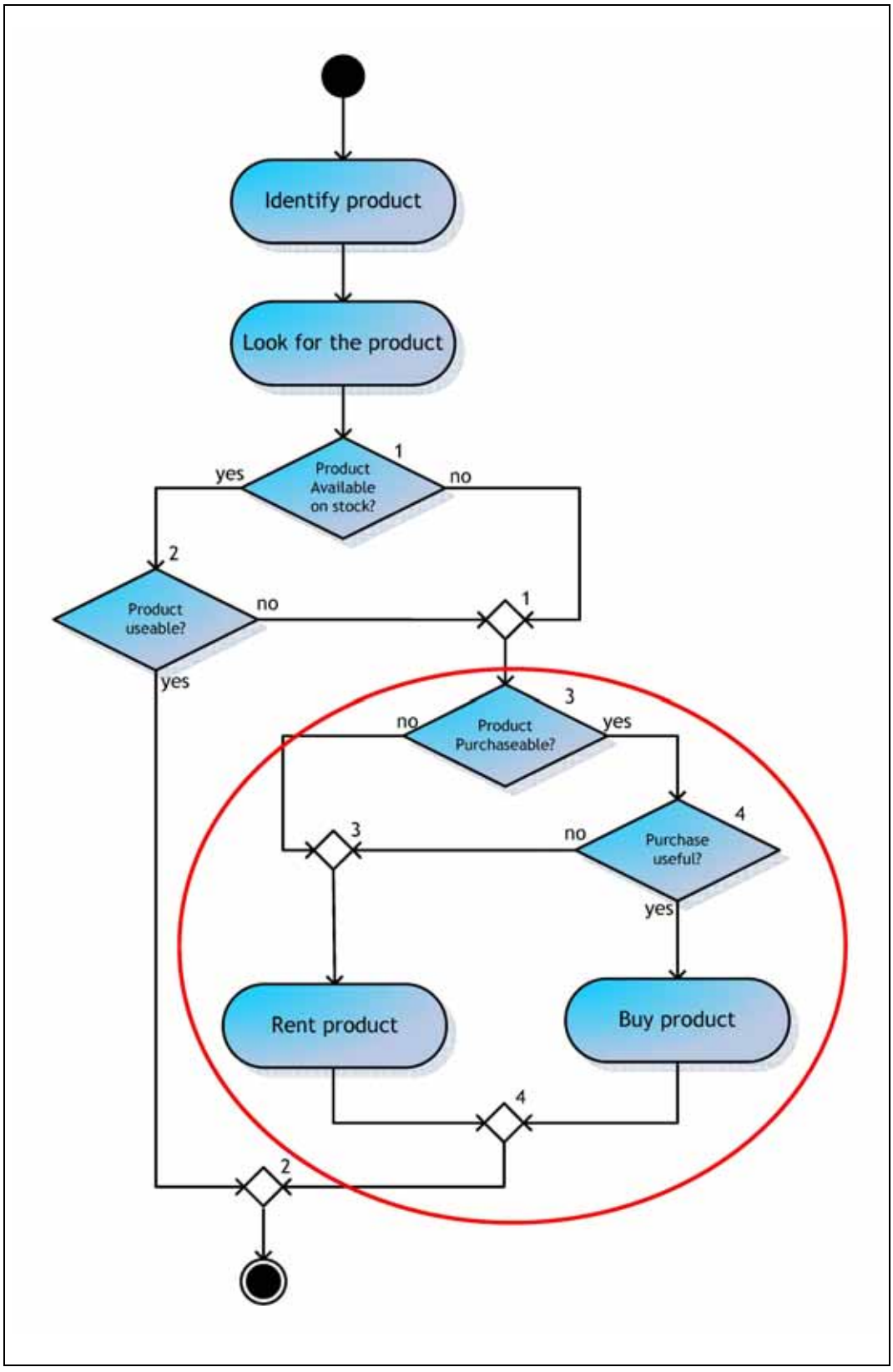
(Scheer, Business Process Engineering, 1994: 350-351)

1.2.1 Unstructured examples: current literature 5



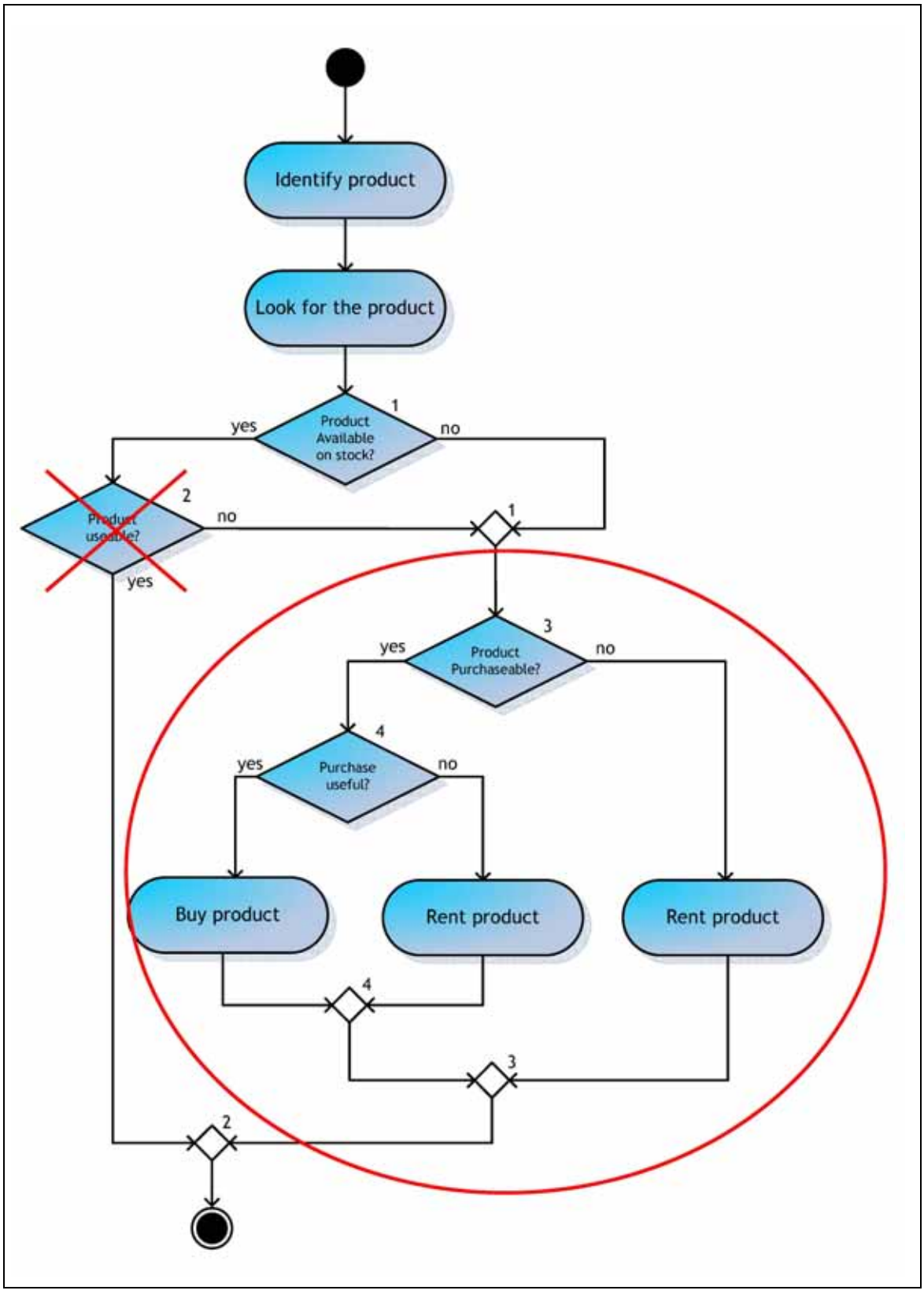
(Süddeutsche Zeitung 14.04.2008)

1.2.2 Unstructured examples: structuring 1



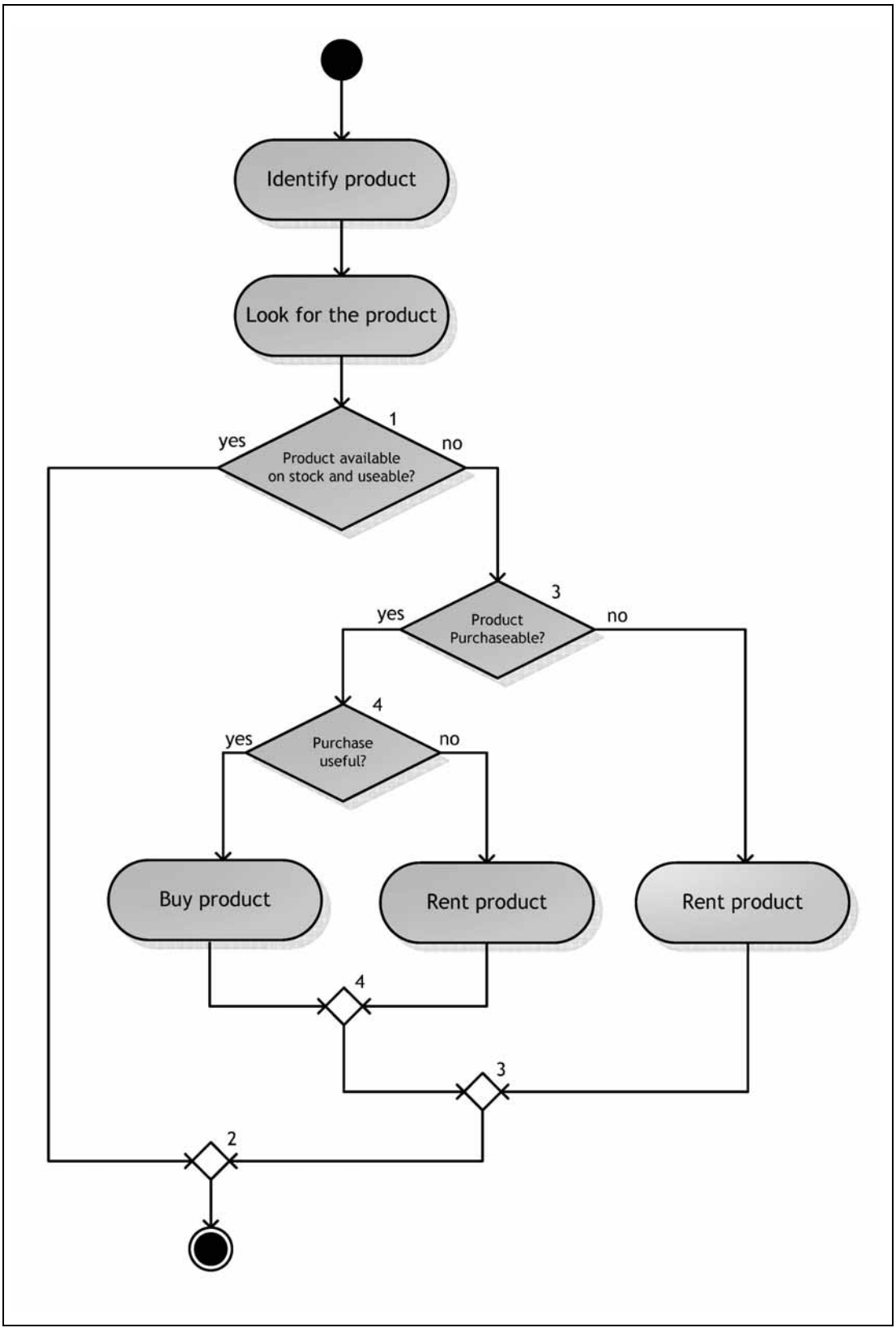
Typical example of the current BPM style in the form of a UML activity diagram; example only covers unstructured tests (Holl / Valentin 2004)

1.2.2 Unstructured examples: structuring 2



Improved business process model (Holl / Valentin 2004)

1.2.2 Unstructured examples: structuring 3



Well-structured business process model (Holl / Valentin 2004)

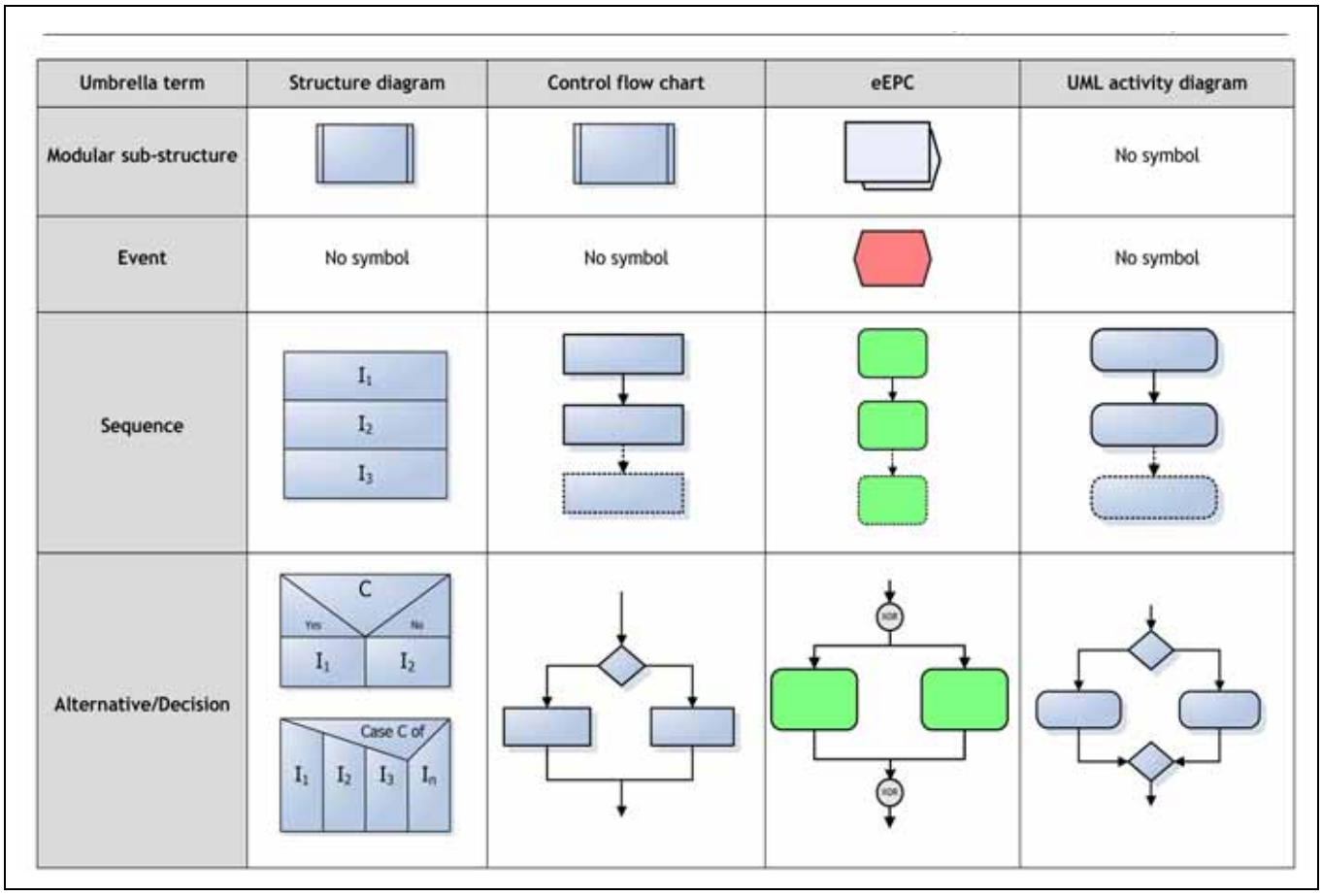
1.3 Basic components of process models 1

→ **Aim: to convince the BPM community with the presentation of a detailed analogy**

Umbrella term	BPM	Control flow modeling
Modular substructure	partial process	subprogram, subroutine
Event	business event	operating system event, interrupt
Sequence	sequence	sequence
Test, alternative, decision	XOR	IF
Iteration	cycle	loop
Simultaneity, parallelism	AND	parallel functions
Process unit	business activity	instruction or block of instructions

Analogy (umbrella terms) of the basic components of BPM and control flow modeling (Holl / Valentin 2004)

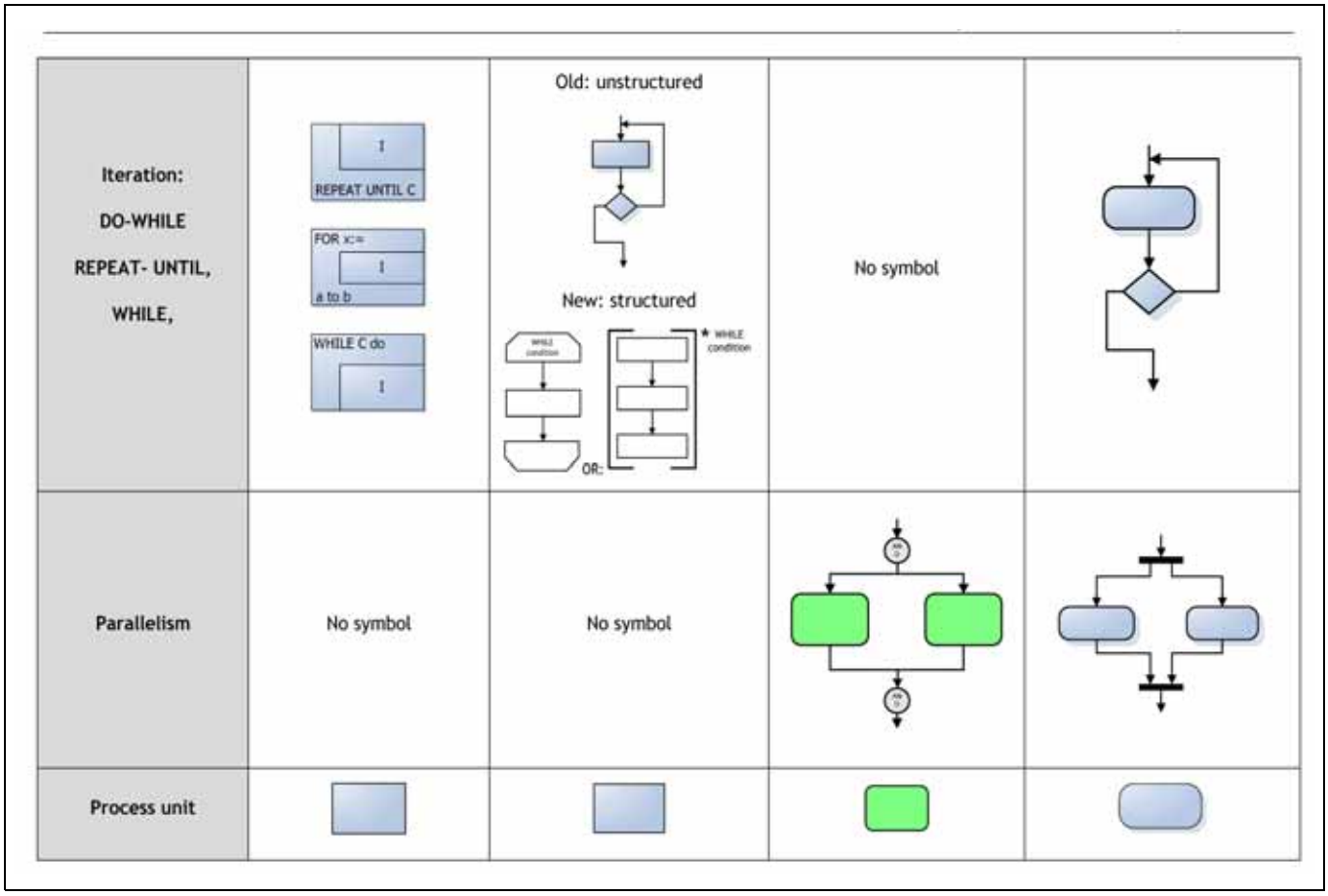
1.3 Basic components of process models 2



Analogy of the notations of BPM and control flow modeling (Grünauer 2008: 102 according to Holl / Valentin 2004)

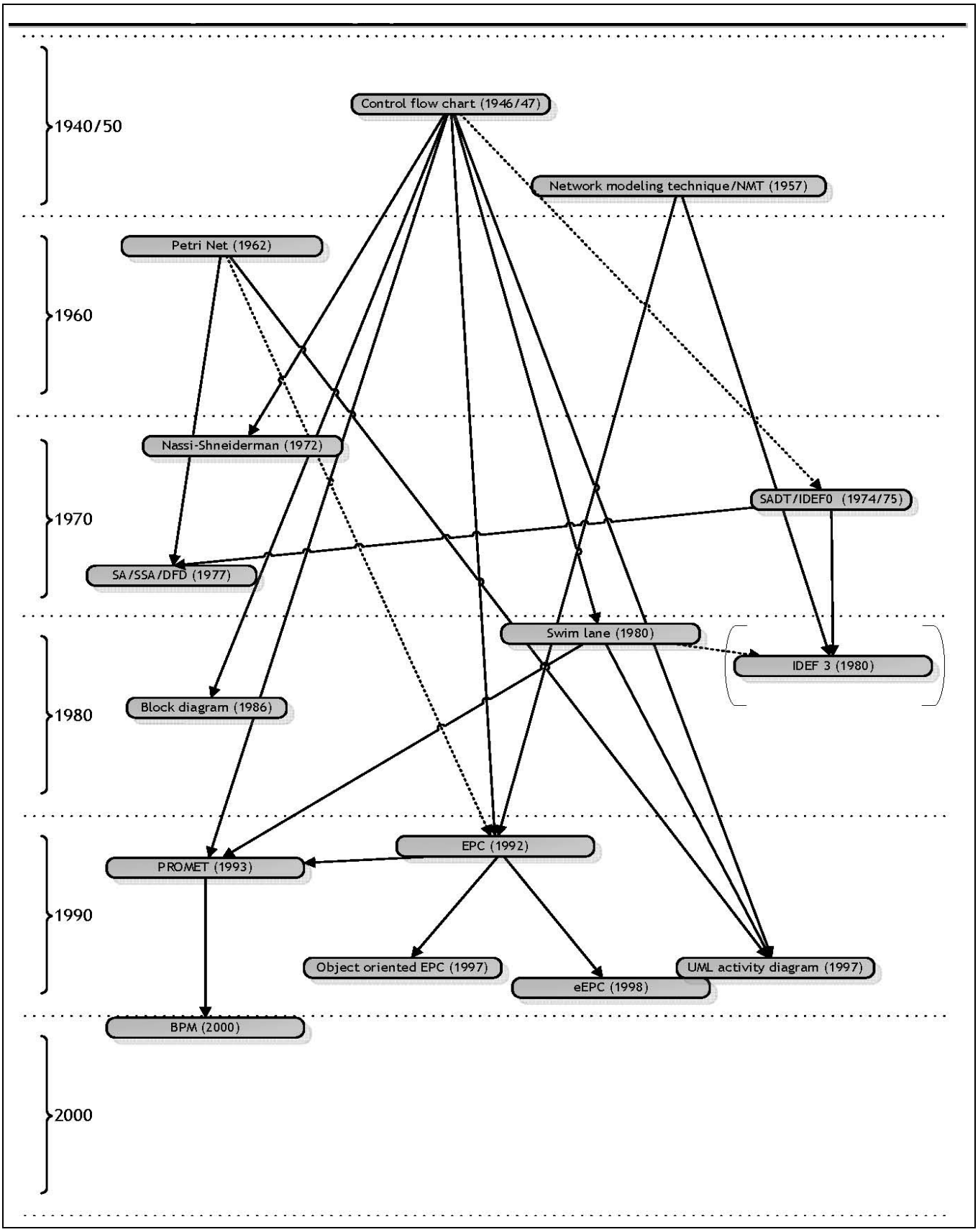
Structure diagram: DIN 66 261, according to Nassi-Shneiderman
Control flow chart: DIN 66001

1.3 Basic components of process models 3



Analogy of the notations of BPM and control flow modeling (Grünauer 2008: 102 according to Holl / Valentin 2004)

1.3 Genealogical tree of process notations



(Grünauer 2008: 30)

1.4 Process meta-model: elements 1

In the following, process meta-models will be examined from the point of view of information systems.

That is, there will be a focus on the activity-on-node variant.

The activity-on-arc variant (state transition networks, Petri nets), which is important for theoretical computer science approaches, will be excluded.

1.4 Process meta-model: elements 2

Nodes of a semantic network:

1. function, action (computer-aided or not)
function unit, function module

– name from the view of the organization
– decomposition-marker: reference to subprocesses
– algorithm, internal logic in a note
– duration, start time, end time
– features, feature values (→ theory of gestalt)
– IT support: computer-aided or manual

2. initiating and resulting events

3. actor: person/role/department responsible for the action
partly connected with data flow

4. external (business/communication) partners
connected with data flow

5. data stores accessed: input data and output data
connected with data flow

6. resources used (machines etc.)

	World 1 (reality)	World 3 (model)
single object, "instance"	one individual course of events in an organization	business process instance
set - type of similar objects	set of homogeneous courses of events	business process type

1.4 Process meta-model: elements 3

Arcs of a semantic network:

1. control flow: temporal interrelation of functions (cf. structured programming)

– temporal succession: sequence (predecessors and successors)

– condition: alternative, selection (IF, XOR)
case discrimination (CASE)
or complex rule (decision table)
disjoint and complete

– repetition: iteration, loop (WHILE or REPEAT)
test-first loop and test-last loop

– recursion

– simultaneousness: parallel processing (AND)

– coroutine: mutual call

CAUTION:
all control flow elements without the mere sequence must have a divergent delimitor (begin) and a convergent delimitor (end, synchronization); the delimitors have to be arranged symmetrically in a diagram: IF – ENDIF, CASE – ENDCASE, LOOP – ENDLOOP etc.

2. data flow (only partly)

3. mere connectors to actors and resources used

1.4 Process meta-model: special notations 1

1. Classical notations

1.1 Traditional notations for structured programming

flow chart, block diagram ('Programm-Ablauf-Plan')
structure diagram, structogram (Nassi-Shneiderman diagram)
Jackson tree
- Jackson structured design (JSD)
- Jackson structured programming (JSP)

functions and control flow

1.2 Decision table

complex conditions and functions: rules

1.3 Network model(ing technique)

functions, sequence, parallel processing,
duration, start time, end time
→ critical path

1.4 Control flow plus data flow

HIPO: hierarchy plus input-process-output (Mills 1972, IBM)
functions, control flow, data stores, data flow

1.4 Process meta-model: special notations 2

1.5 Swim lane diagram

functions, control flow, responsible departments
predecessor of UML activity diagram
Arbeitsablaufdiagramm: Arbeitsschritte – Abteilungen
Organisationsprozessdarstellung (H. F. Binner)

2. Business process models

Event-driven process chain

functions, control flow (ridiculous: no iterations!)
events
actors, partners, data stores, resources, data flow

3. Dynamic object models

UML activity diagram

functions, control flow
events
actors, partners, data stores, resources, data flow
swim lanes (responsible departments)

UML sequence diagram

classes, elementary functions called by messages, control flow

1.5 Conclusion

Changes to be made in BPM

– **block structures**:
  BEGIN – END, LOOP – ENDLOOP,
  IF(XOR) – ENDIF, CASE – ENDCASE
  BEGIN OR – END OR, BEGIN AND – END AND

– **corresponding notations for block structures:**
  divergent and convergent delimiters
  symbol for iterations

– **hierarchically nested structures (LIFO principle)**

– **vertical decomposition with motivated cuts**
  hierarchic modular structure

– **transparent diagrams**

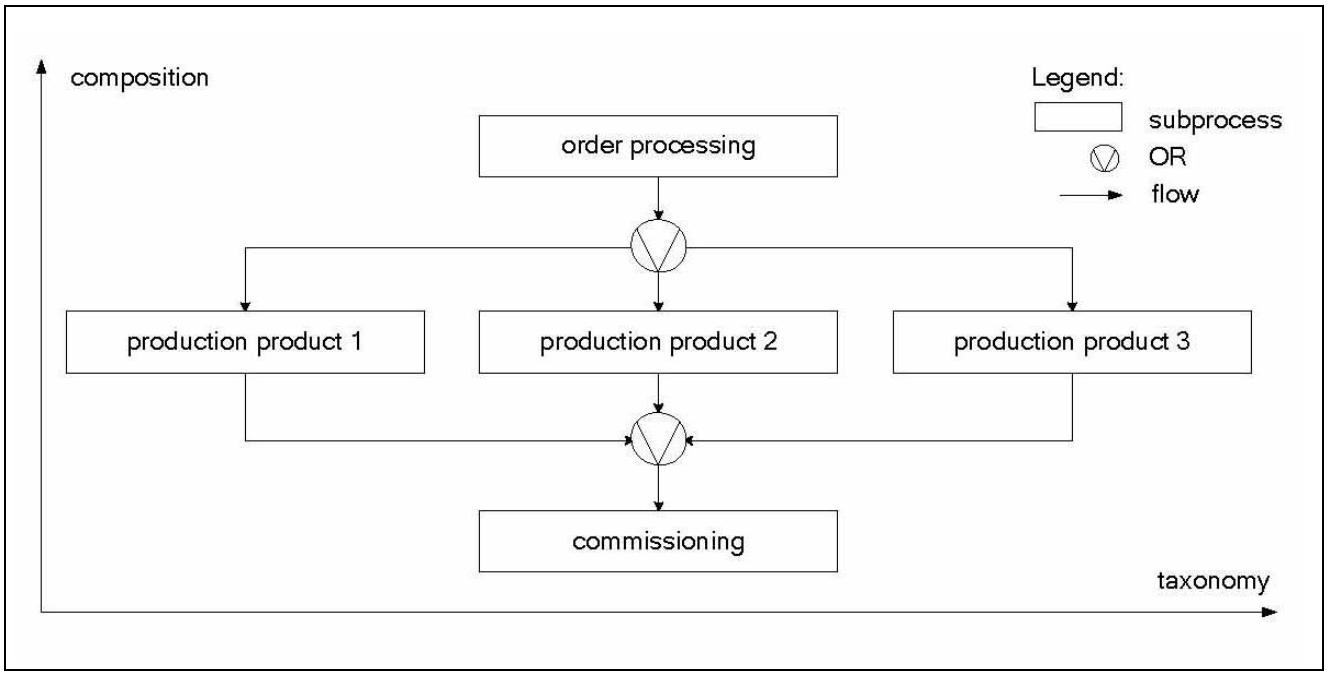
Advantages

– **more transparent description of the reality**
– **easier optimization of BP models (BP reengineering)**
– **easier modification and adaptation of BP models**
– **more effective mapping to workflow management tools**

– **better, transparent basis of communication**
– **more effective requirements engineering**
– **better usable reference models**

2 Structured business process decomposition

2.1 Motivation 1: teaser

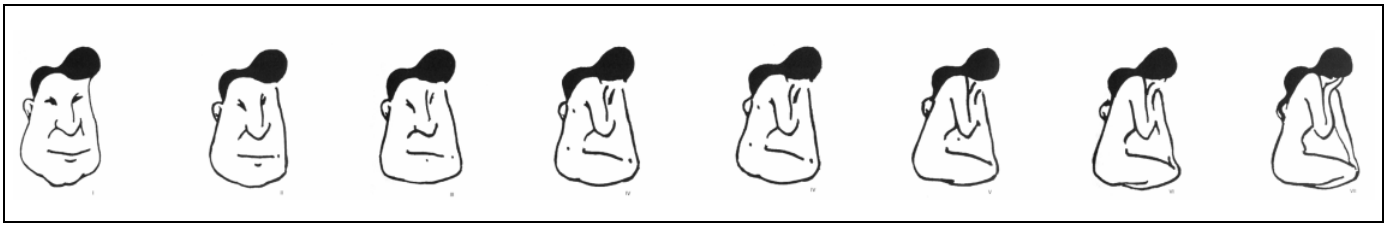


Decomposition of processes in sub-processes
(Holl / Krach / Mnich 2000, 198)

1 Decomposition in sequential sub-process (compositional)
2 Decomposition in parallel sub-processes (taxonomic)

The former is the subject of the following considerations.

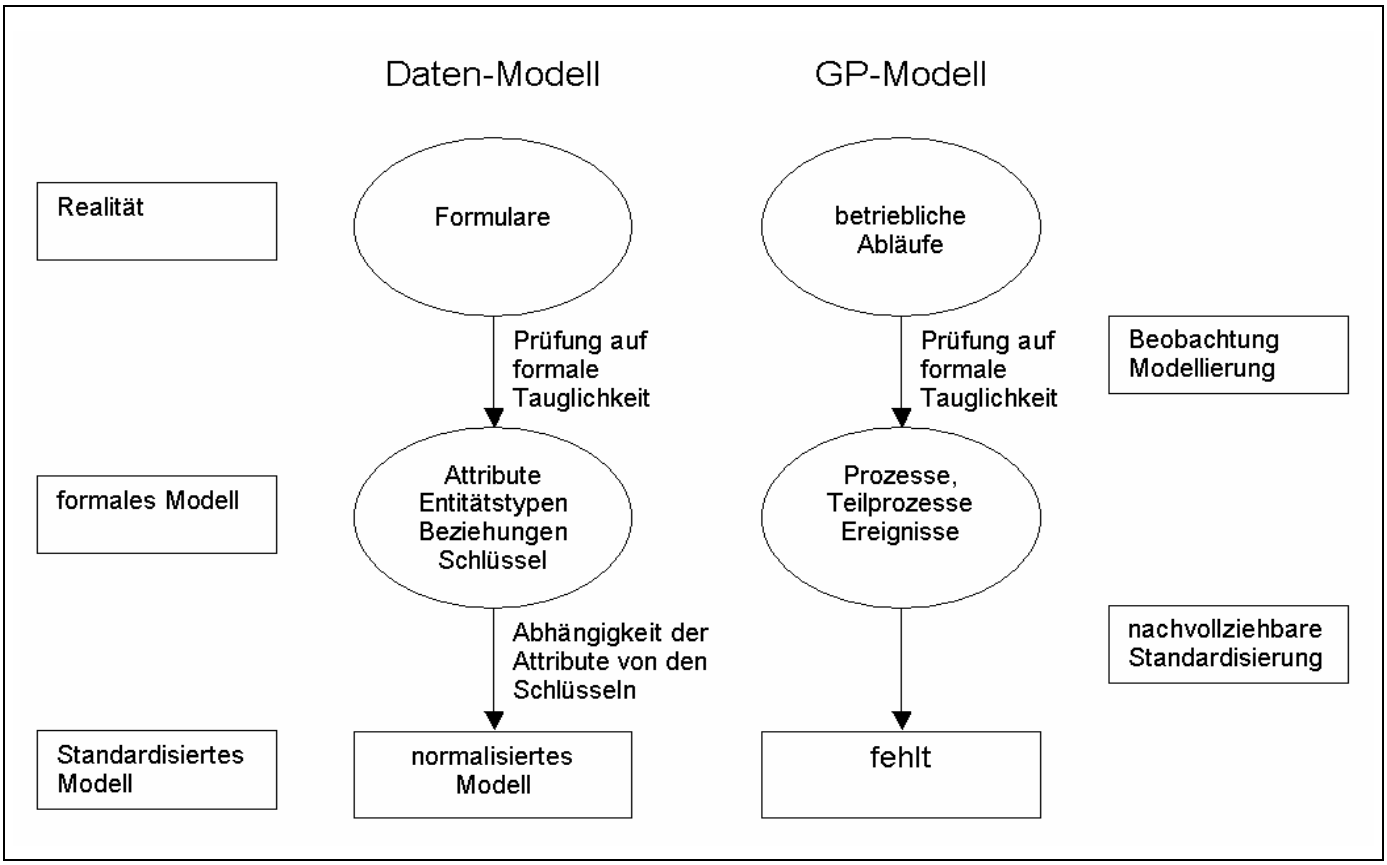
Where can the following process be divided into sub-processes?



A man's face to a woman's body (Riedl 1987: 74-77)

2.1 Motivation 2: two starting points and their synthesis

1 Different model designers construct different BP models vs. data and static OO models are more independent of designers



**Comparison between data and BP modeling:
a method analogous to normalization is missing
(Holl / Krach / Mnich 2000, 203)**

2 Examination of similarity and features as cognitive principles in evolutionary epistemology and theory of gestalt: becoming aware of decomposition features changes hypotheses of decomposition, of splitting points

**3 Aim / synthesis of the two starting points:
gestalt-theoretical business process decomposition:
processes are split up where a feature changes its value.**

2.2 Theory of gestalt 1

The theory of gestalt dates back to considerations of
- Johann Wolfgang von Goethe
- Christian von Ehrenfels
- Max Wertheimer

It is an interdisciplinary theory with applications in
- epistemology, psychology of perception
- biology
- pedagogic
- architecture, arts

The whole (semantics) is more than the sum of its parts (syntax).
'Forms' (German "Gestalten") can be
– static: physical objects
– dynamic: melody, ritual, process

What is this?

2.2 Theory of gestalt 2

Decomposition of static and dynamic ‘forms’ (“Gestalten”)

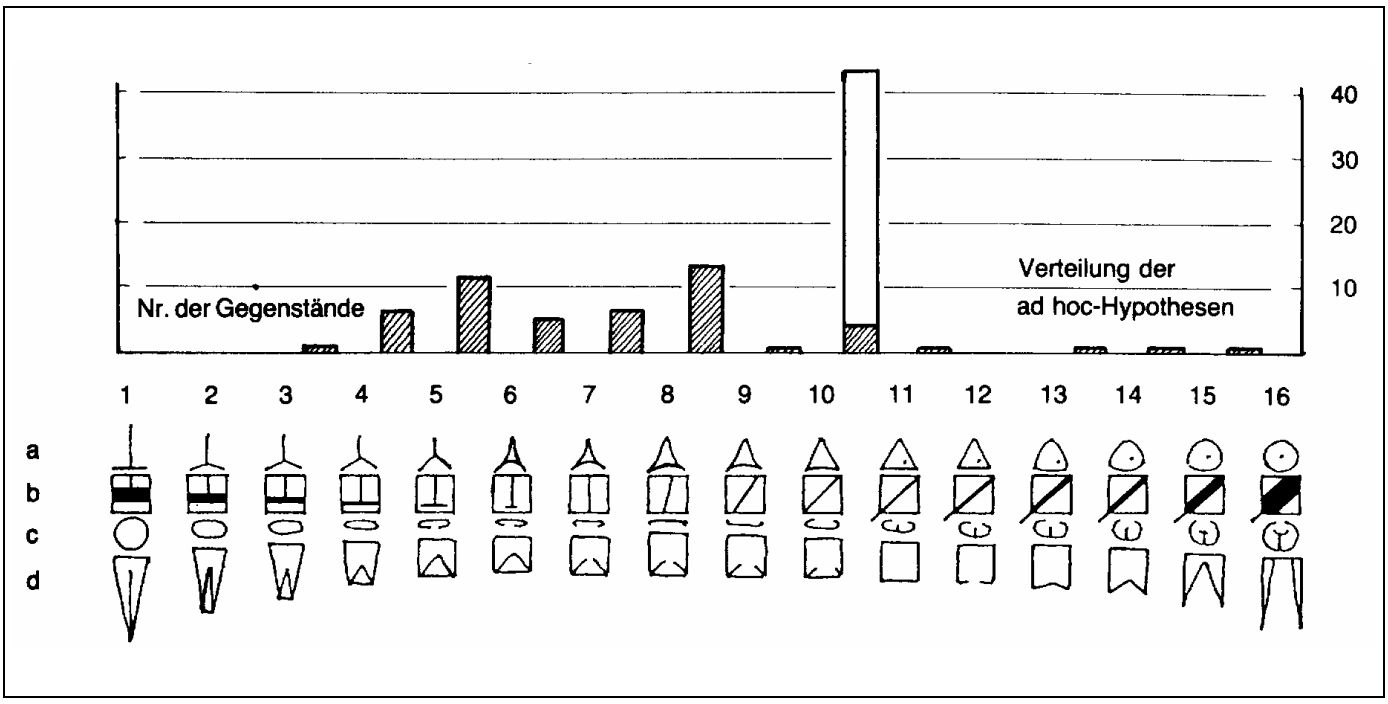
For humans, it is easy to decompose static ‘forms’ (pictures), difficult to decompose dynamic ‘forms’ (courses of events, business processes, morphing processes).

Features

A particularity or a property of a ‘form’ can be called a feature.

Rupert Riedl has systematically examined the idea of a feature in his book “Begriff und Welt” (‘Concept and reality’) 1987.

Riedl shows that features cannot only be used to find similarities between different static ‘forms’ but also to decompose / subdivide dynamic ‘forms’.



Splitting of a process according to changes of features (Riedl 1987: 195)

2.3 Business process decomposition and gestalt-theoretical features 1

BP decomposition is done using features.

The model designer has to be aware of these features, has to lift them from the unconscious to the conscious level and has to make them explicit.

Thus, we obtain BP models which can be followed and, therefore, be discussed and motivated.

Possible features in business processes:
- responsible person
- order status
- machine

Processes are split up where a feature changes its value.

Relation between features and events

When a feature changes its value, an event happens.

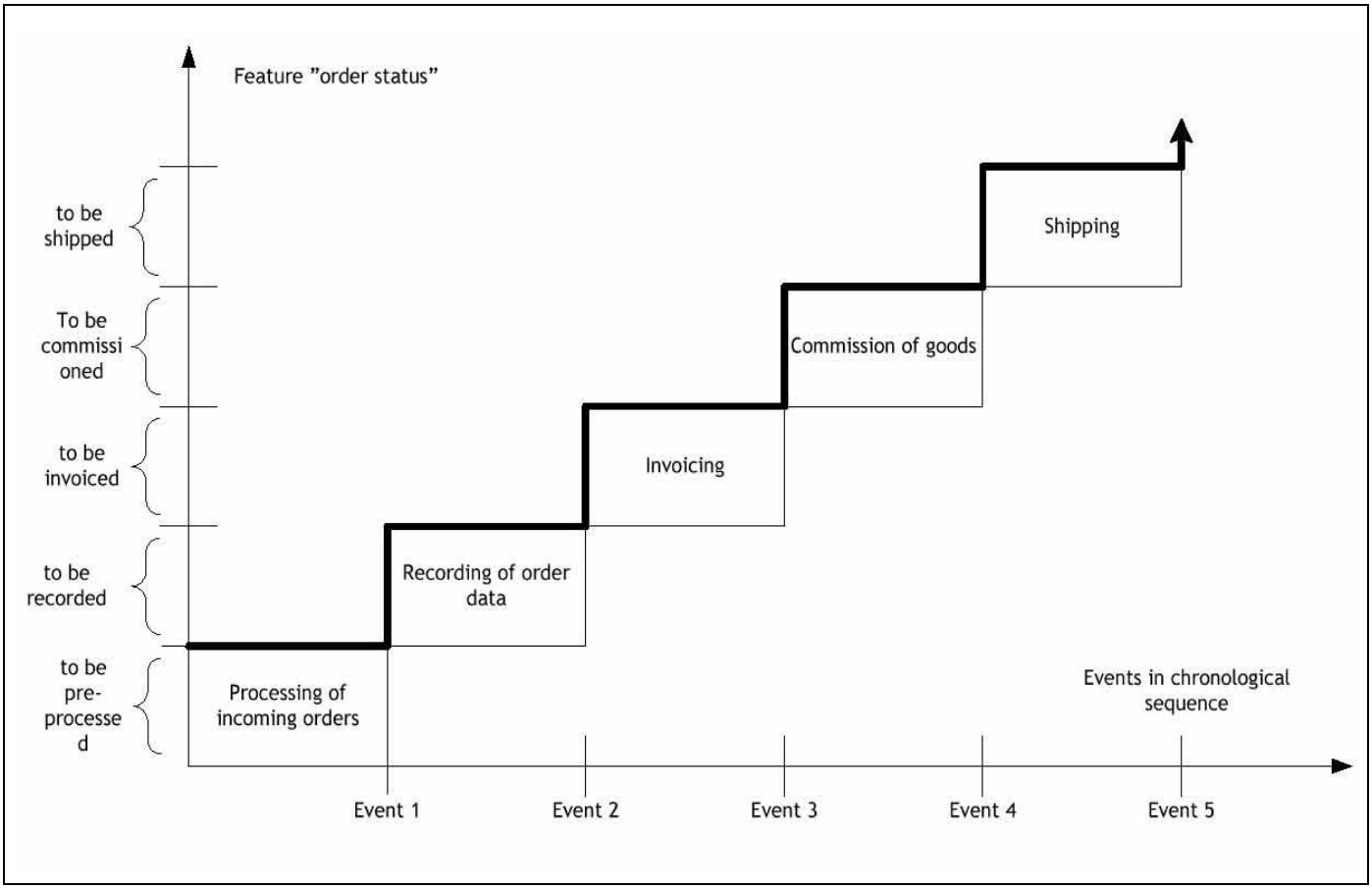
→ Feature-based event-driven process chains

Models do not become better automatically, but this approach makes it easier to discuss and, thus, to improve them.

2.3 Business process decomposition and gestalt-theoretical features 2

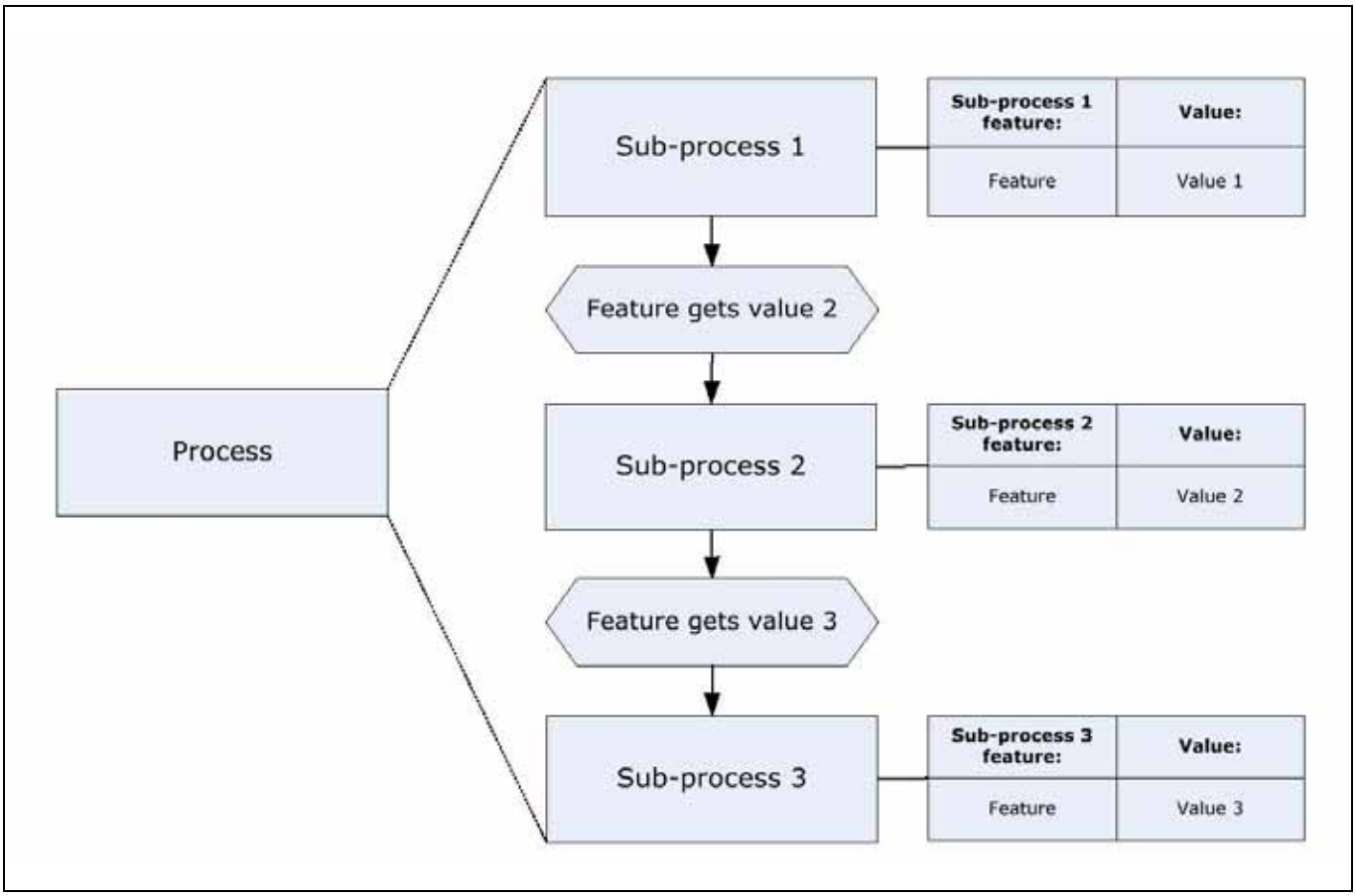
Sub-processes	Values of the feature "order status"
Order acceptance check	To be checked
Order data recording	To be recorded
Invoicing	To be invoiced
Commissioning	To be commissioned
Shipping	To be shipped

Sub-processes and their feature values



Changes of a feature visualized as mathematical step function (Holl / Krach / Mnich 2000, 207)

2.3 Business process decomposition and gestalt-theoretical features 3



Process representation with sub-processes, events and features (Holl / Krach / Mnich 2000, 208)

3 References

Böhm, Corrado; Jacopini, Giuseppe:
Flow diagrams, Turing machines and languages with only two formation rules.
***Communications of the ACM* 9(1966) 5, 366-371.**

Dijkstra, Edsger:
GOTO statement considered harmful.
***Communications of the ACM* 11(1968) 3, 147-148.**

Grünauer, Karin:
***Business process modeling*. Växjö (Master thesis) 2008**

Holl, Alfred; Valentin, Gregor:
Structured business process modeling.
Contribution to:
***Information Systems Research in Scandinavia (IRIS'27)*,**
Falkenberg/Sweden 2004, CD-ROM.

Holl, Alfred; Krach, Thomas; Mnich, Roman:
Geschäftsprozessmodellierung und Gestalttheorie.
In: Britzelmaier, Bernd et al. (ed.): *Information als Erfolgsfaktor. 2. Liechtensteinisches Wirtschaftsinformatik-Symposium an der FH Liechtenstein*.
Stuttgart: Teubner 2000, 197-209, ISBN 3-519-00317-1.

Lorenz, Konrad (1903-1989):
Gestalt perception as fundamental to scientific knowledge
[original 1959 in German: Gestaltwahrnehmung als Quelle wissenschaftlicher Erkenntnis. *Zeitschrift für experimentelle und angewandte Psychologie* 6(1959) 118-165].
***General systems* 7 (1962) 37-56 [= Bertalanffy, L. v.; Rapoport, A. (ed.): *Yearbook of the Society for General Systems Research*].**

Riedl, Rupert:
***Begriff und Welt – Biologische Grundlagen des Erkennens und Begreifens*. Berlin, Hamburg: Parey 1987.**

Yourdon, Edward:
***Modern structured analysis*. Englewood Cliffs NJ 1989.**

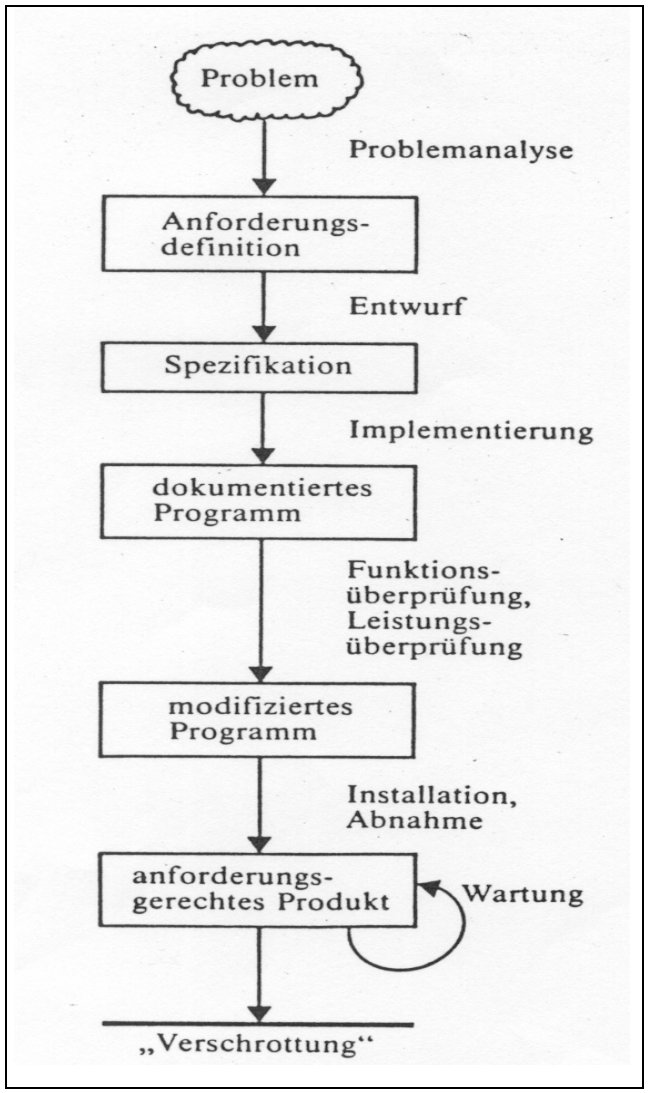
pdf-files of my own publications: see my homepage

Material for software engineering

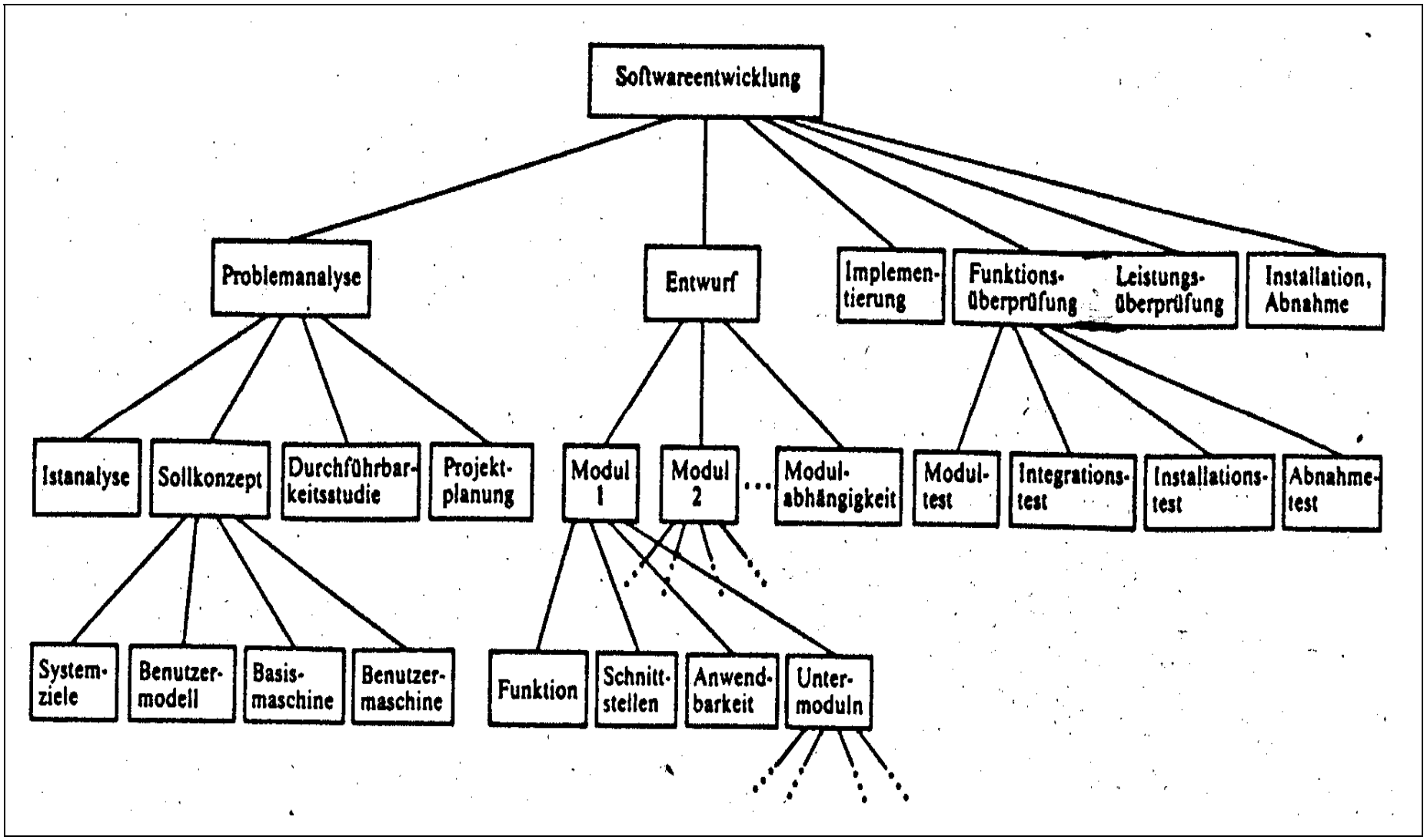
1 Examples for temporal structures in software life cycle models
1.1 Linear
1.2 Tree
1.3 Network model
1.4 Loop
1.5 Overlap
Note: do not use these models for the lab, but the model from the course!

2 Examples for aspects of information system models and their notations
2.1 Information flow models; function structure models
2.2 Data (structure) models
2.3 Behavior models or process models

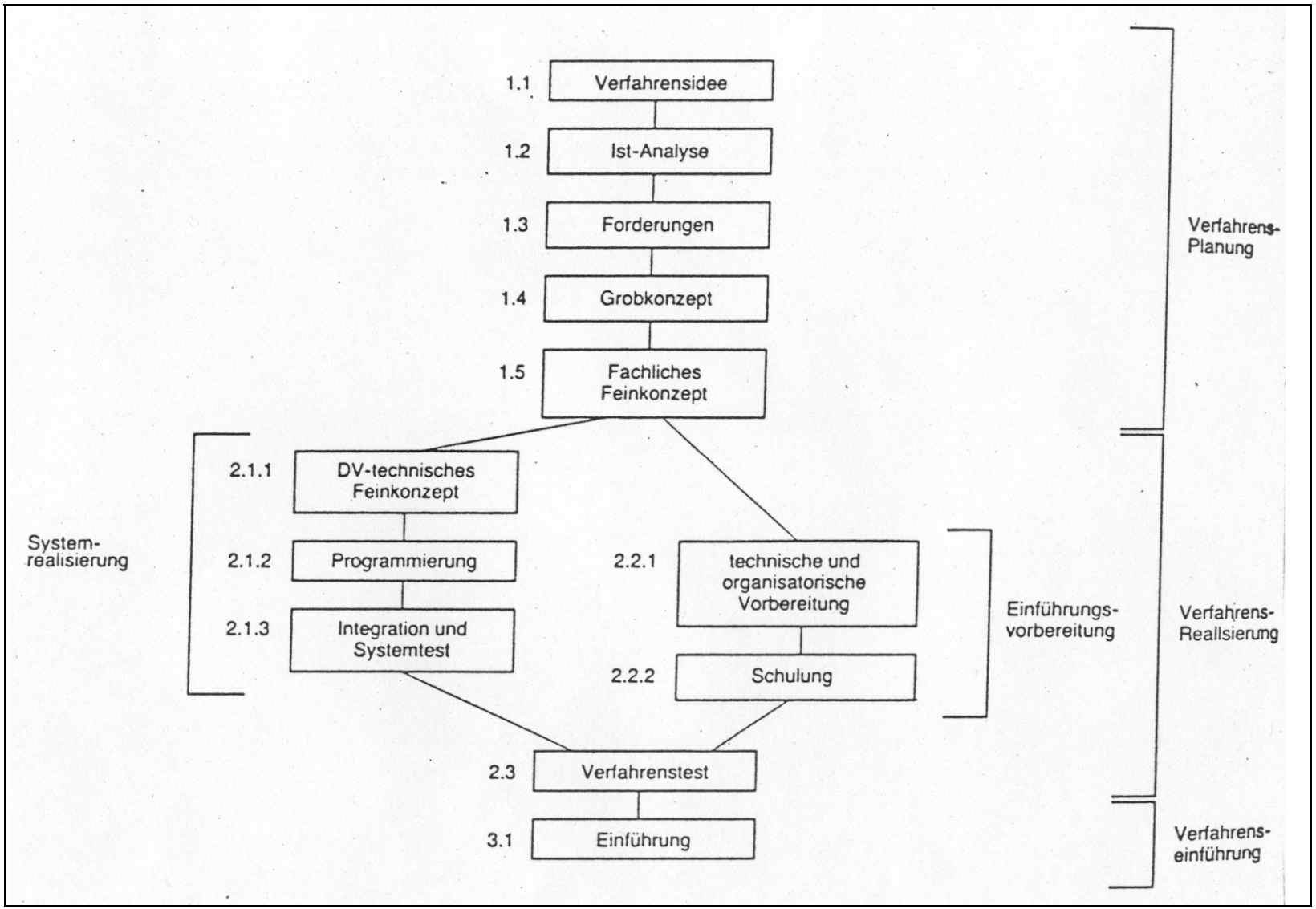
1.1 Linear SW life cycle model / software process model



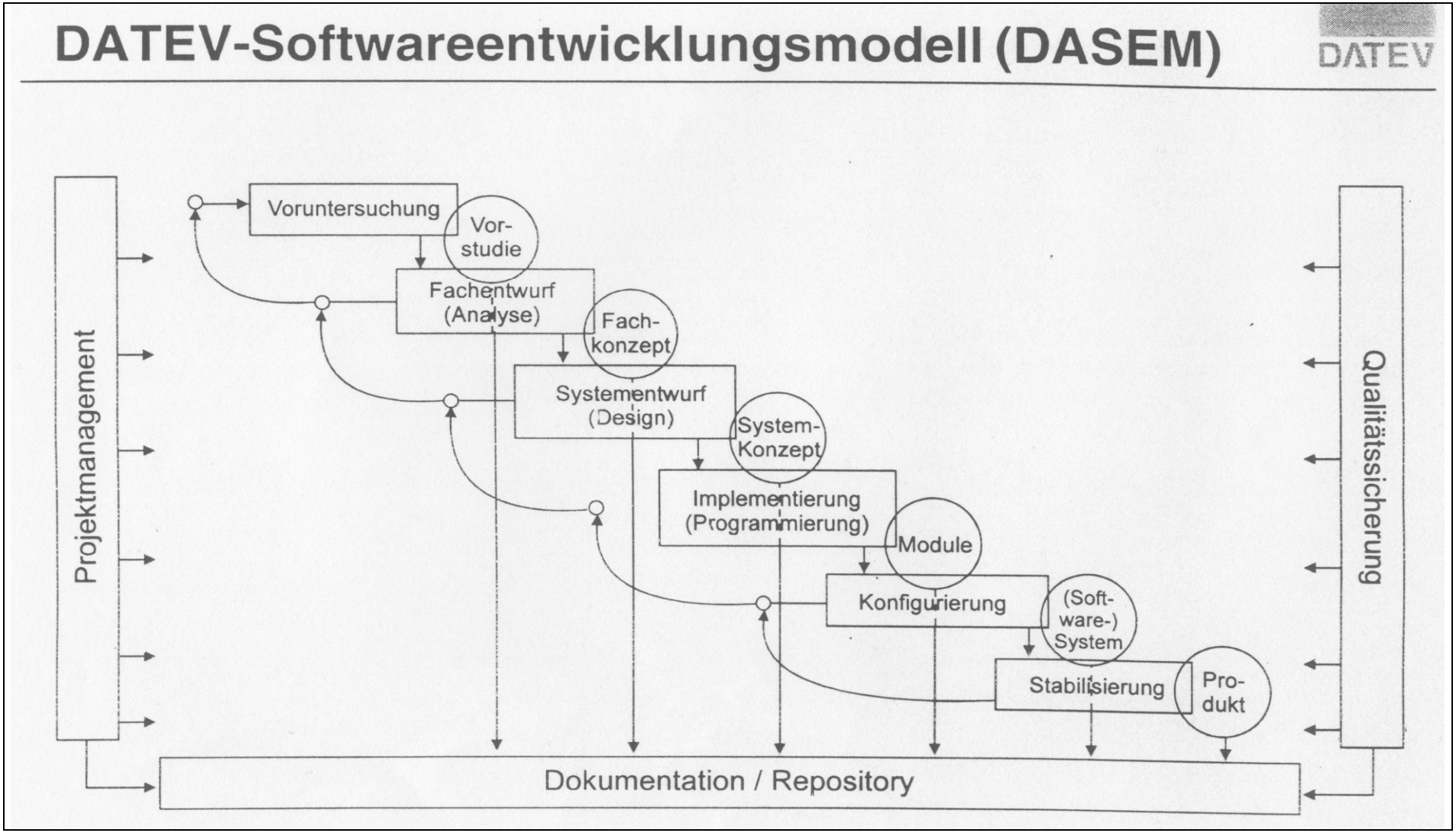
1.2 Tree model



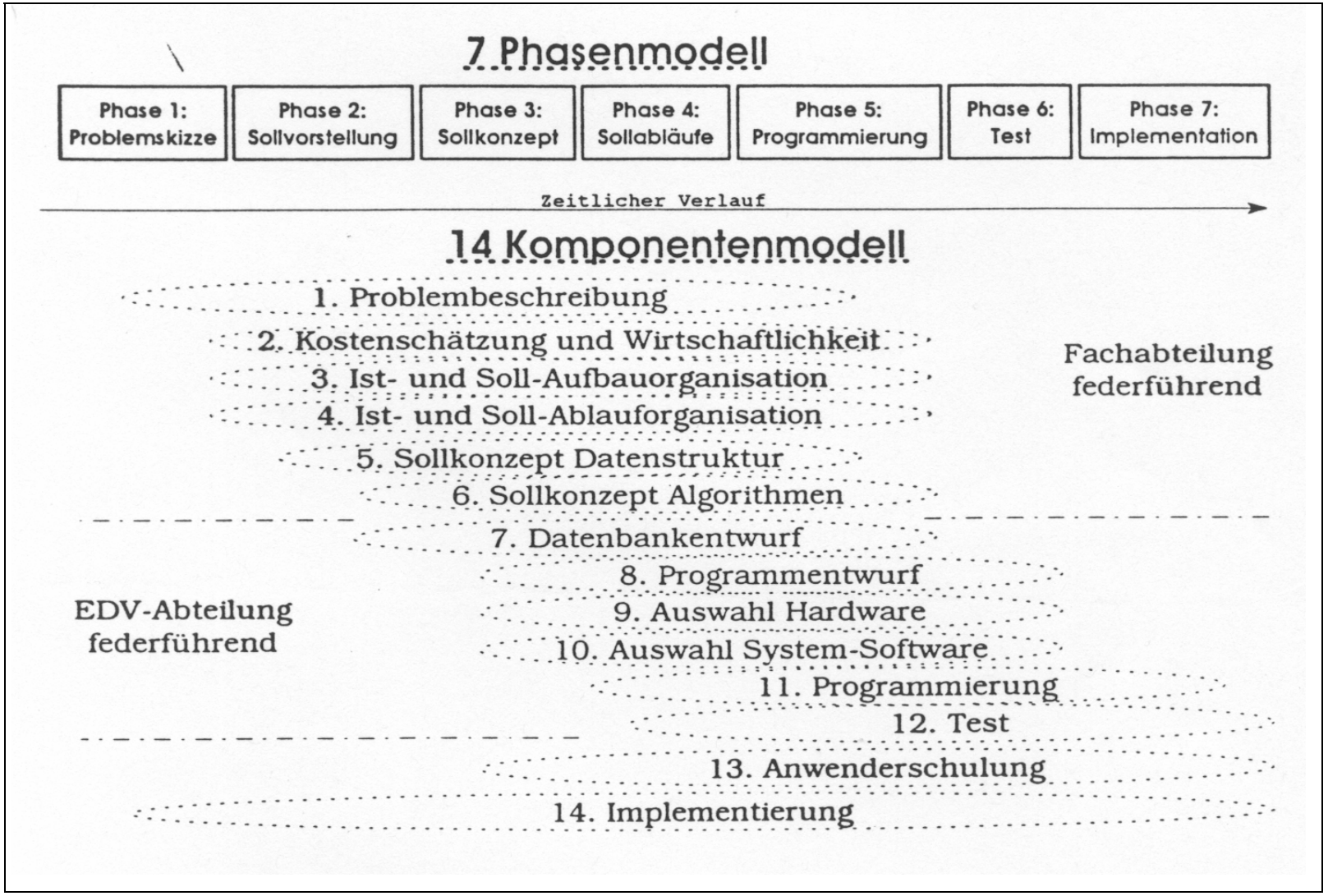
1.3 Network model



1.4 Loop model



1.5 Overlap model



2.1 Information flow models: structured analysis

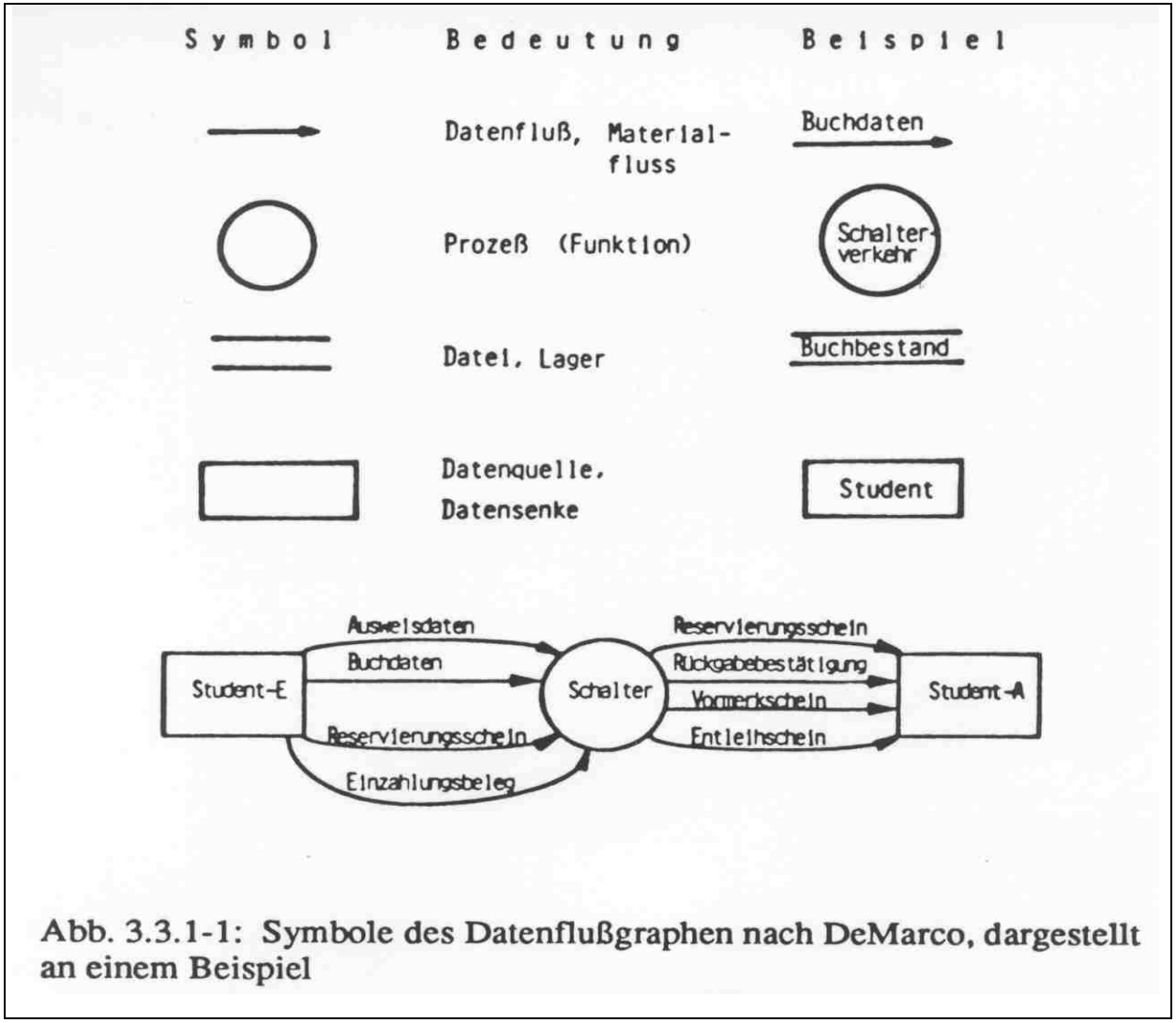
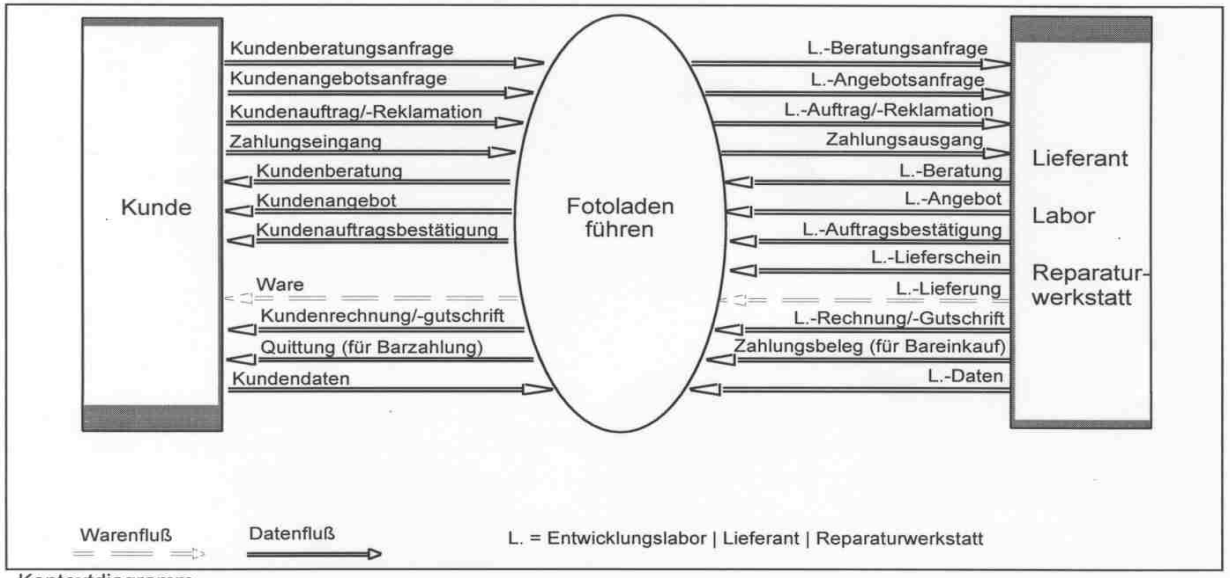


Abb. 3.3.1-1: Symbole des Datenflußgraphen nach DeMarco, dargestellt an einem Beispiel

2.1 Information flow models: SA level 0 (context diagram)

3.3.1. Schnittstelle des Fotoladens zur Außenwelt / Kontextdiagramm

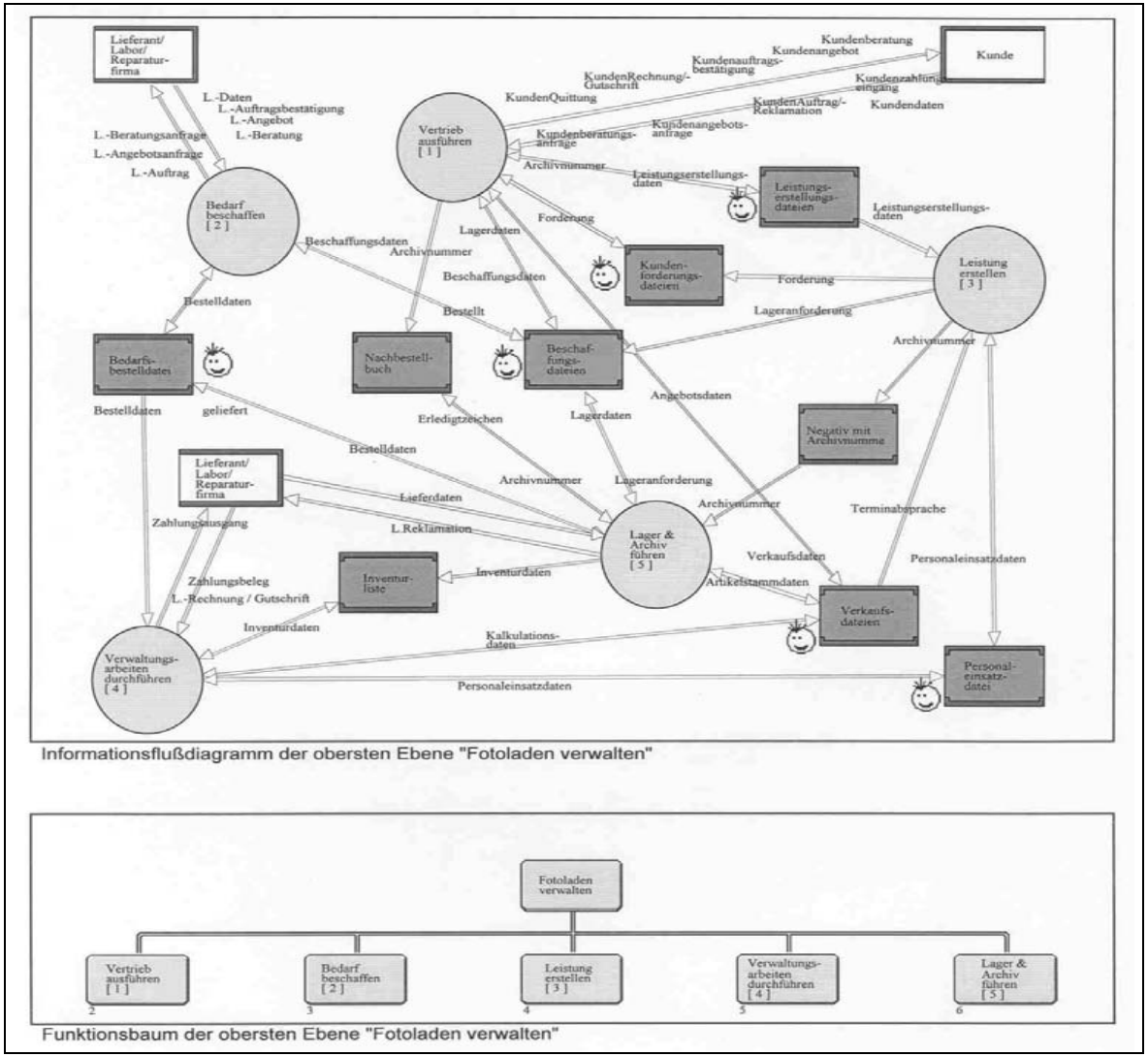
Nach Analyse der in Gesprächen mit dem Besitzer des Fotoladens gewonnenen umfangreichen Datensammlung kristallisierten sich zwischen dem Unternehmen und seiner Umwelt folgende Daten- und Warenflüsse heraus. Sie werden in einem Kontextdiagramm beschrieben. Aus Gründen der Übersichtlichkeit fassen wir dabei einzelne Datenflüsse unter Oberbegriffen zu Datenflußgruppen zusammen. Im Anschluß an das Kontextdiagramm (Punkte 3.3.1.1. und 3.3.1.2. - Zusammensetzung der Datenflüsse -) werden ihre Einzelbestandteile detailliert aufgeführt.



Kontextdiagramm

Die Hauptschnittstellen zur Außenwelt bilden der Kunde und der Lieferant (inklusive Entwicklungslabor und Reparaturwerkstätten). Die Schnittstelle Bank bleibt in unserem Modell unberücksichtigt, da es sich bei diesem Kontakt nach außen um standardisierte Vorgänge handelt, die in jedem Betrieb nach dem gleichen Schema ablaufen.

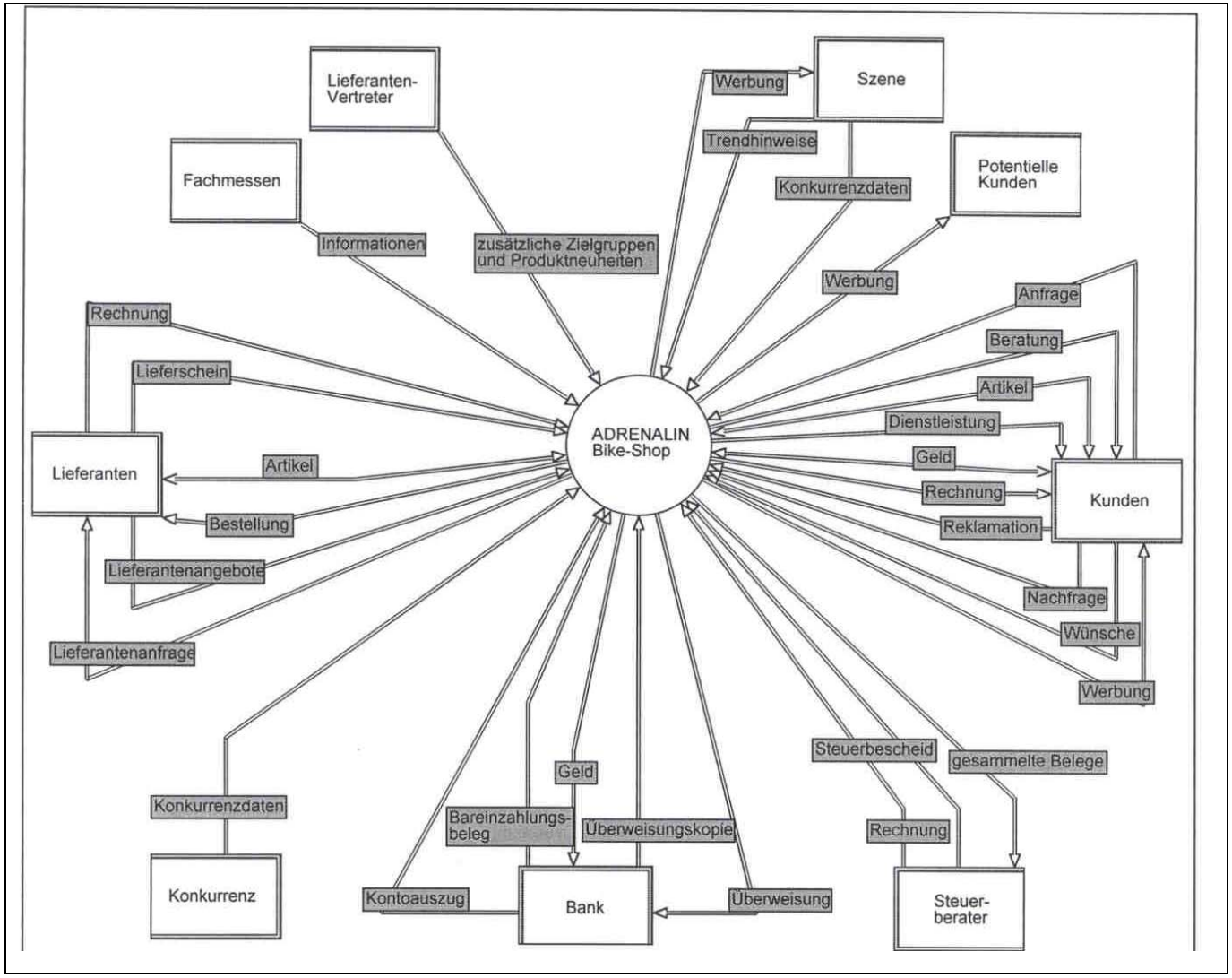
2.1 Information flow models: SA level 1; function structure model



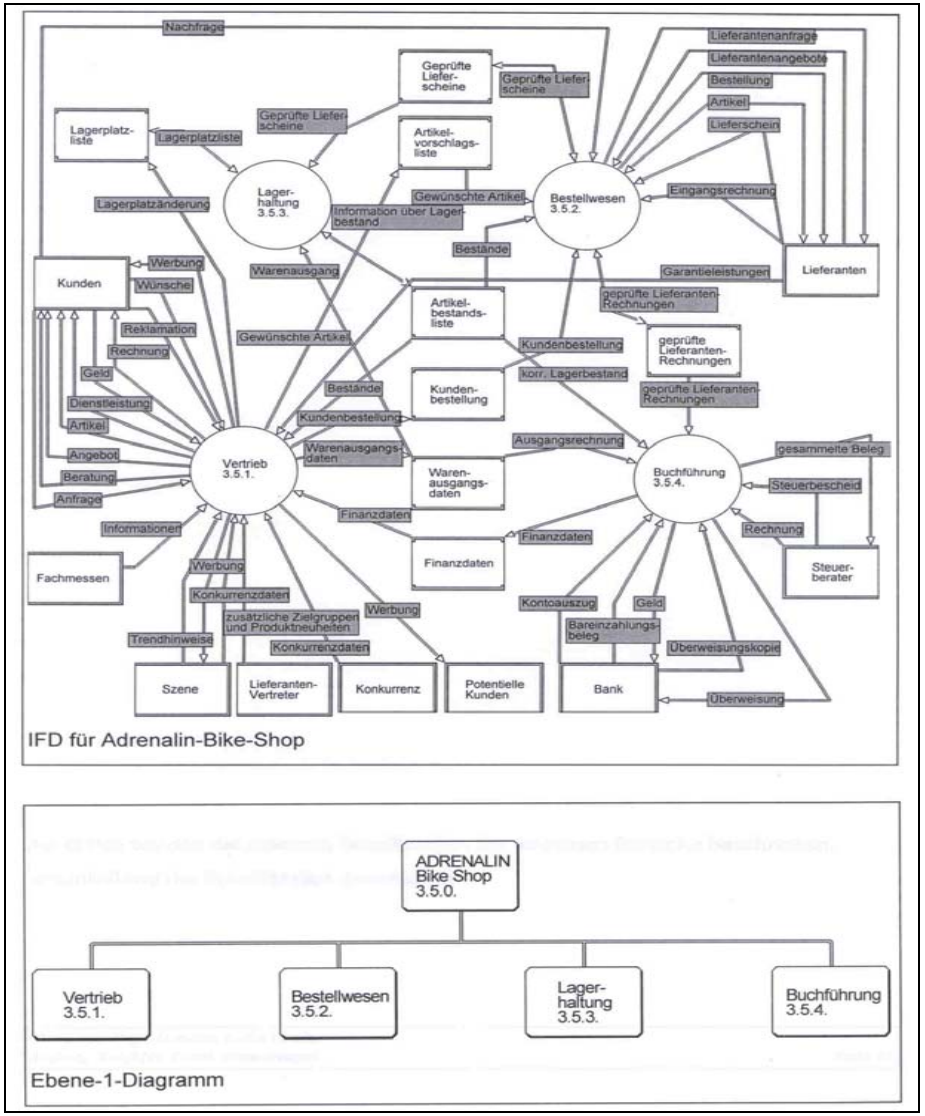
Informationsflußdiagramm der obersten Ebene "Fotoladen verwalten"

Funktionsbaum der obersten Ebene "Fotoladen verwalten"

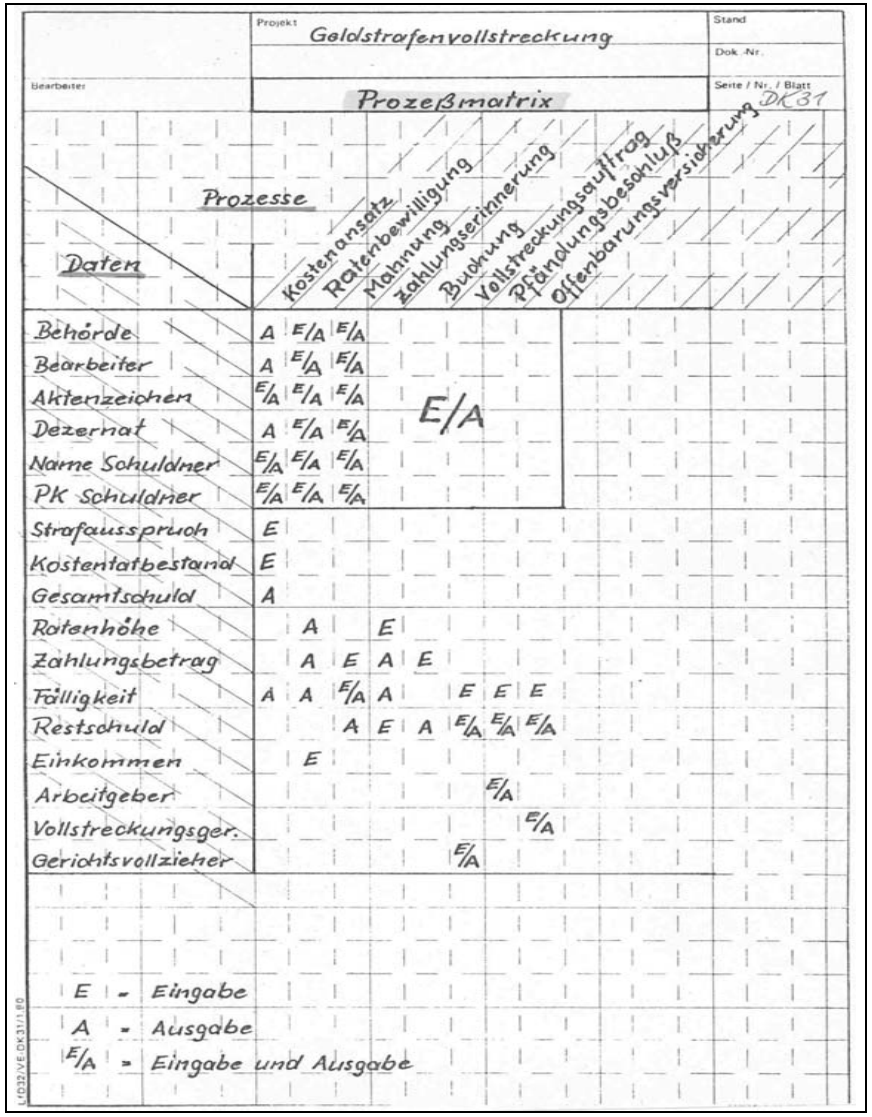
2.1 Information flow models: SA level 0



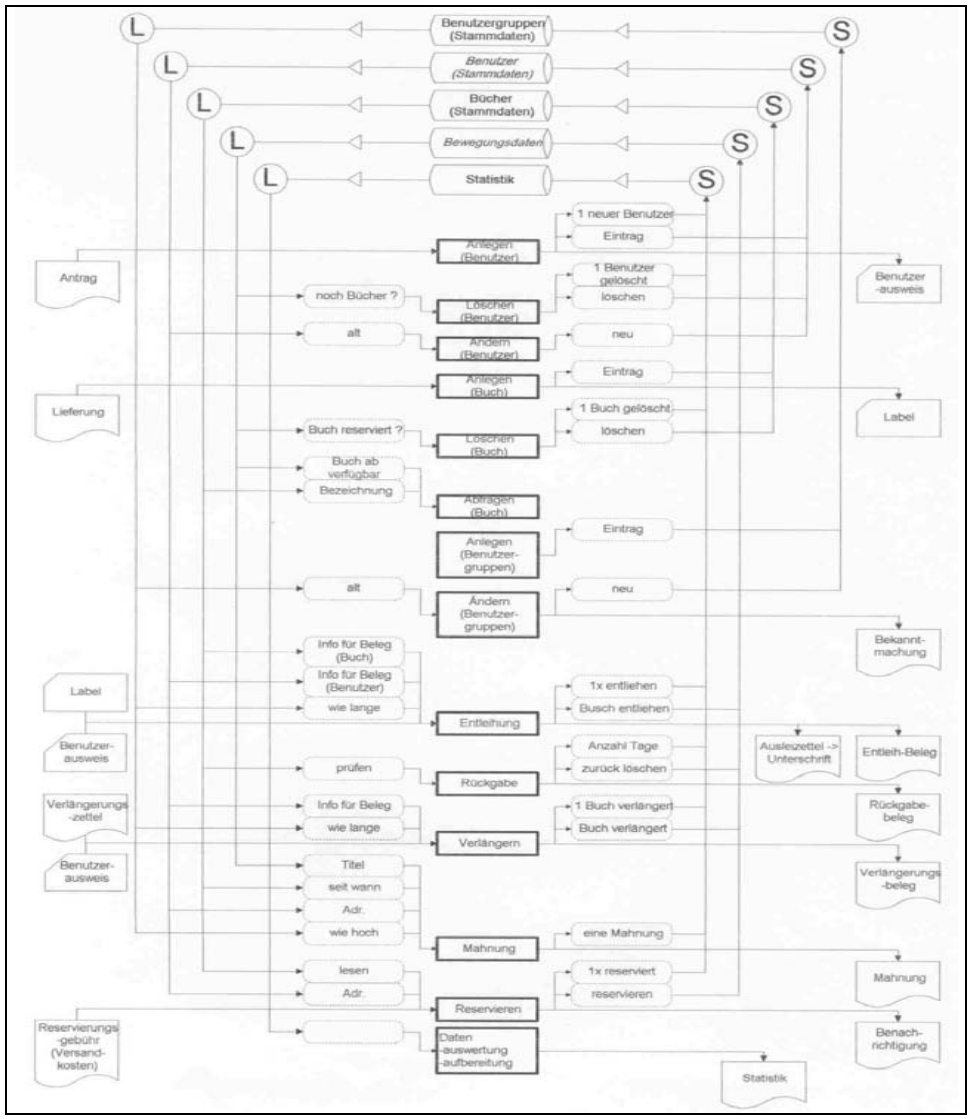
2.1 Information flow models: SA level 1; function structure model



2.1 Information flow models: process matrix



2.1 Information flow models: individual design



2.1 Information flow models: UML use case diagram

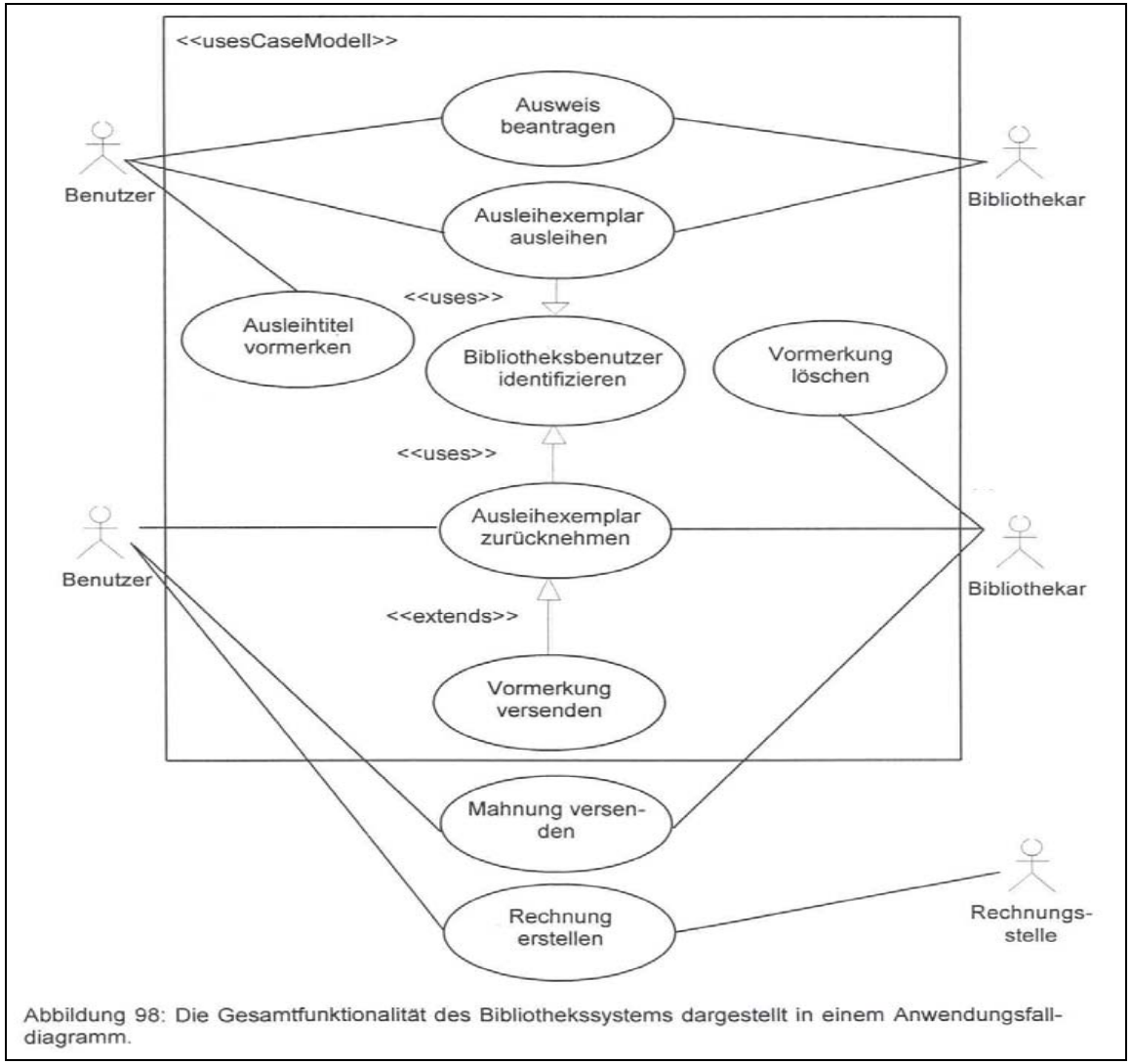


Abbildung 98: Die Gesamtfunktionalität des Bibliothekssystems dargestellt in einem Anwendungsfall-diagramm.

2.2 Data (structure) models: UML class diagram

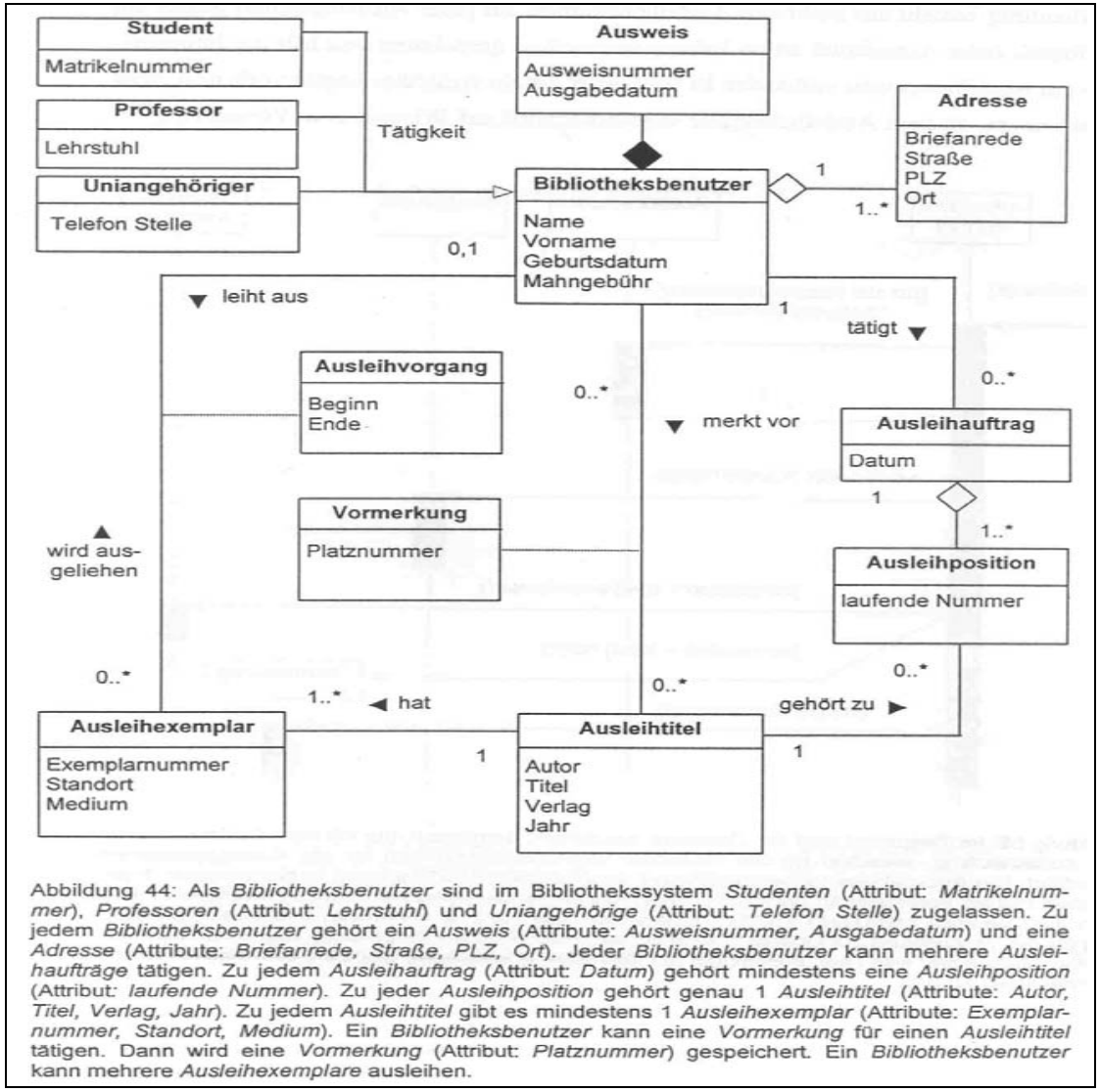


Abbildung 44: Als *Bibliotheksbenutzer* sind im Bibliothekssystem *Studenten* (Attribut: *Matrikelnummer*), *Professoren* (Attribut: *Lehrstuhl*) und *Uniangehörige* (Attribut: *Telefon Stelle*) zugelassen. Zu jedem *Bibliotheksbenutzer* gehört ein *Ausweis* (Attribute: *Ausweisnummer, Ausgabedatum*) und eine *Adresse* (Attribute: *Briefanrede, Straße, PLZ, Ort*). Jeder *Bibliotheksbenutzer* kann mehrere *Ausleihaufträge* tätigen. Zu jedem *Ausleihauftrag* (Attribut: *Datum*) gehört mindestens eine *Ausleihposition* (Attribut: *laufende Nummer*). Zu jeder *Ausleihposition* gehört genau 1 *Ausleihtitel* (Attribute: *Autor, Titel, Verlag, Jahr*). Zu jedem *Ausleihtitel* gibt es mindestens 1 *Ausleihexemplar* (Attribute: *Exemplarnummer, Standort, Medium*). Ein *Bibliotheksbenutzer* kann eine *Vormerkung* für einen *Ausleihtitel* tätigen. Dann wird eine *Vormerkung* (Attribut: *Platznummer*) gespeichert. Ein *Bibliotheksbenutzer* kann mehrere *Ausleihexemplare* ausleihen.

2.2 Data (structure) models: UML class diagram

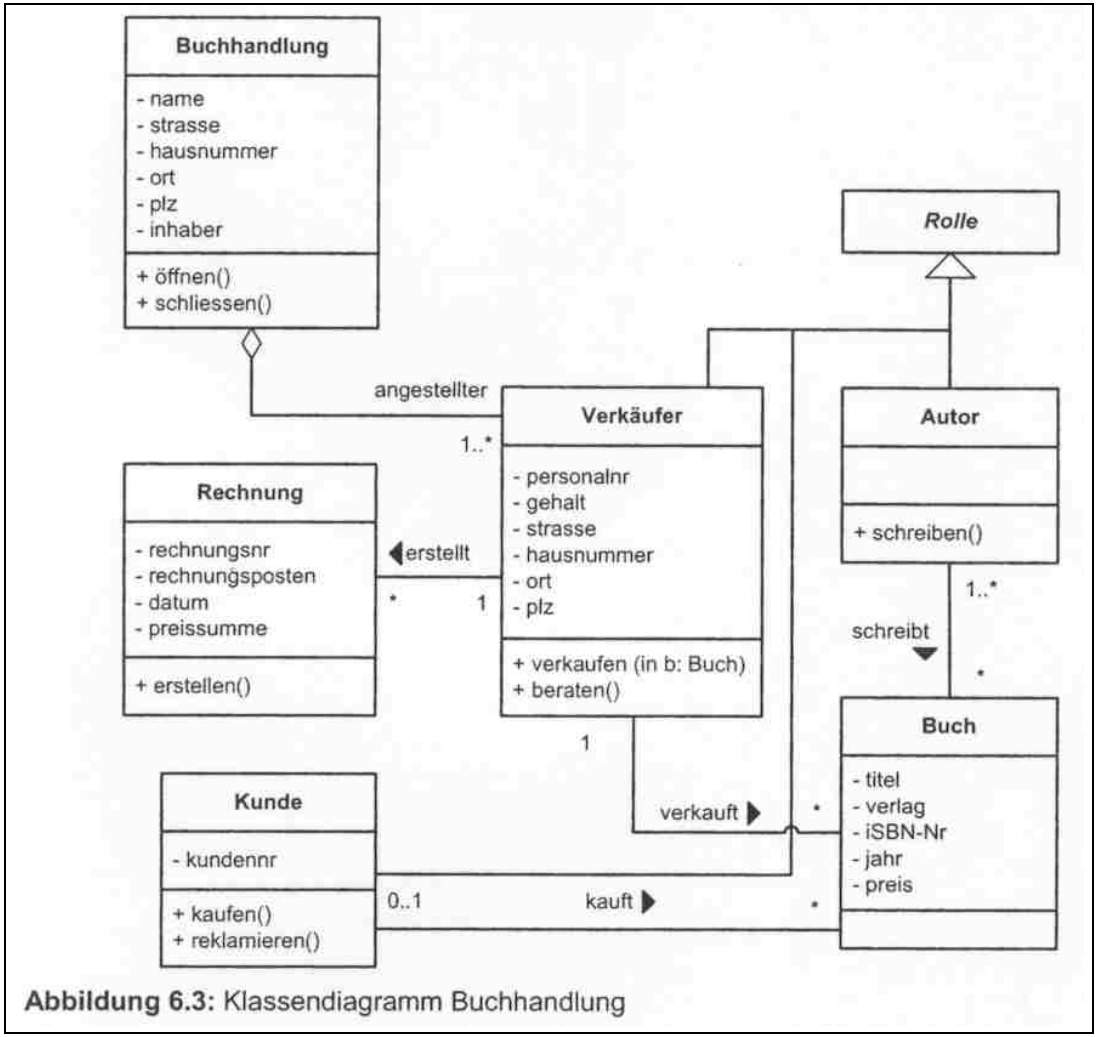
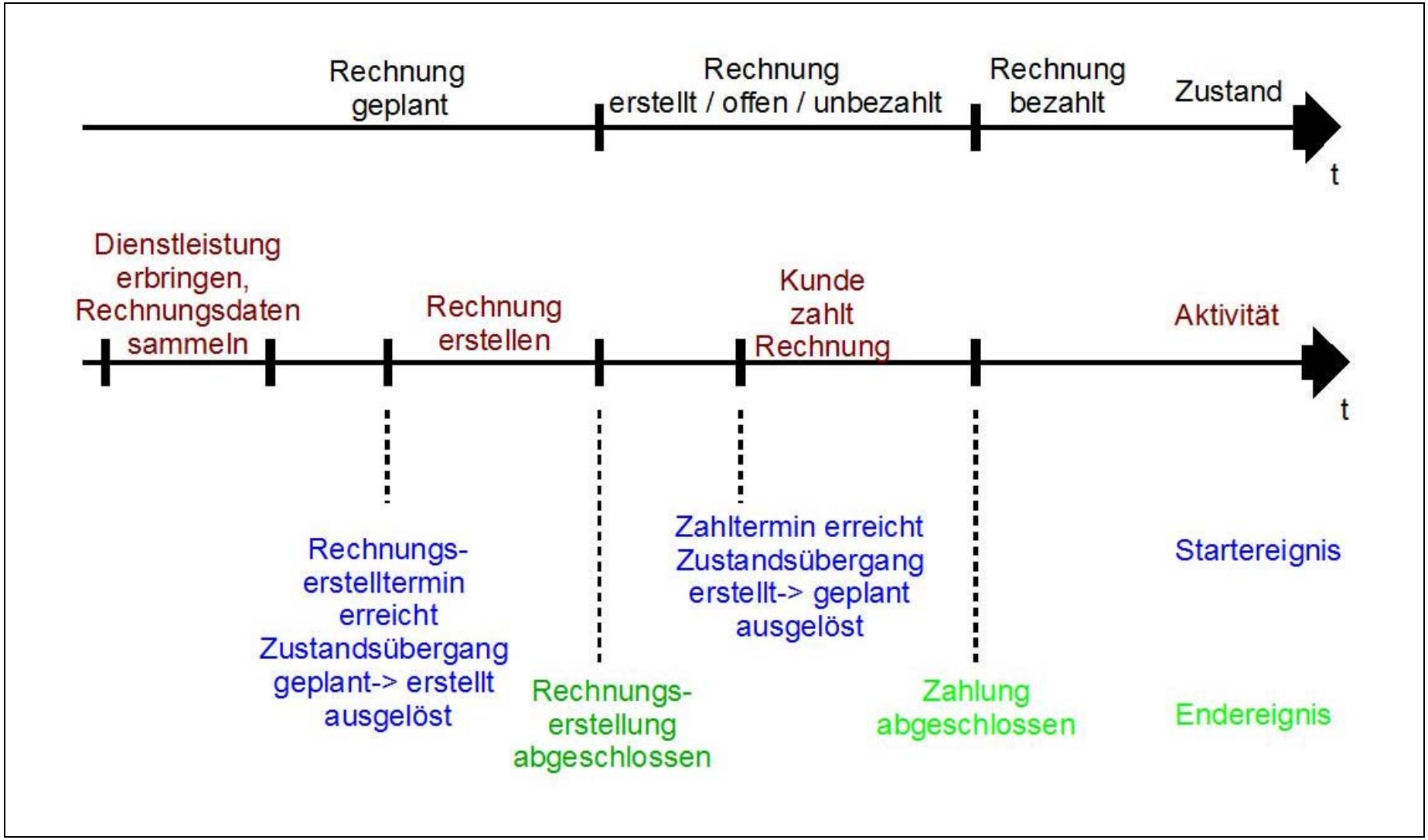


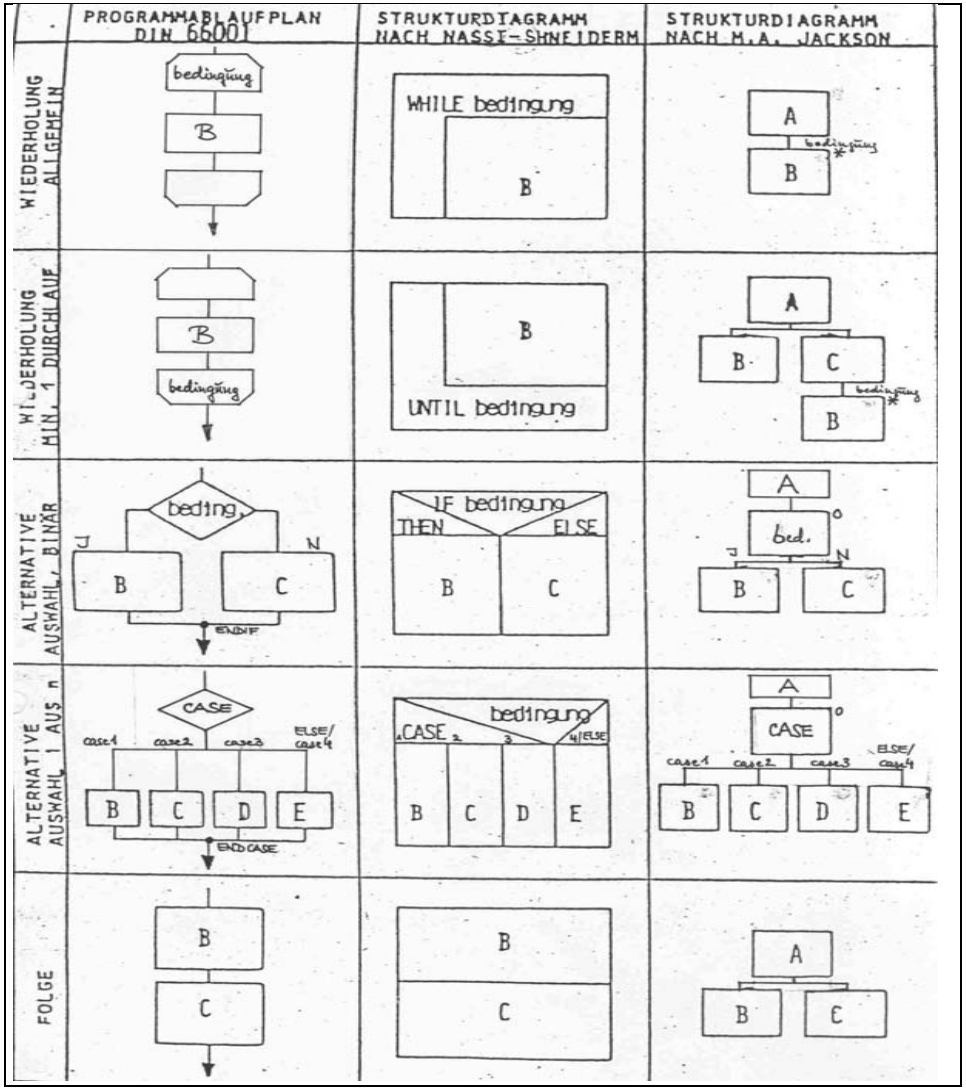
Abbildung 6.3: Klassendiagramm Buchhandlung

(Rupp, Hahn, Queins et al.: UML2 glasklar. München 2005, p. 100)

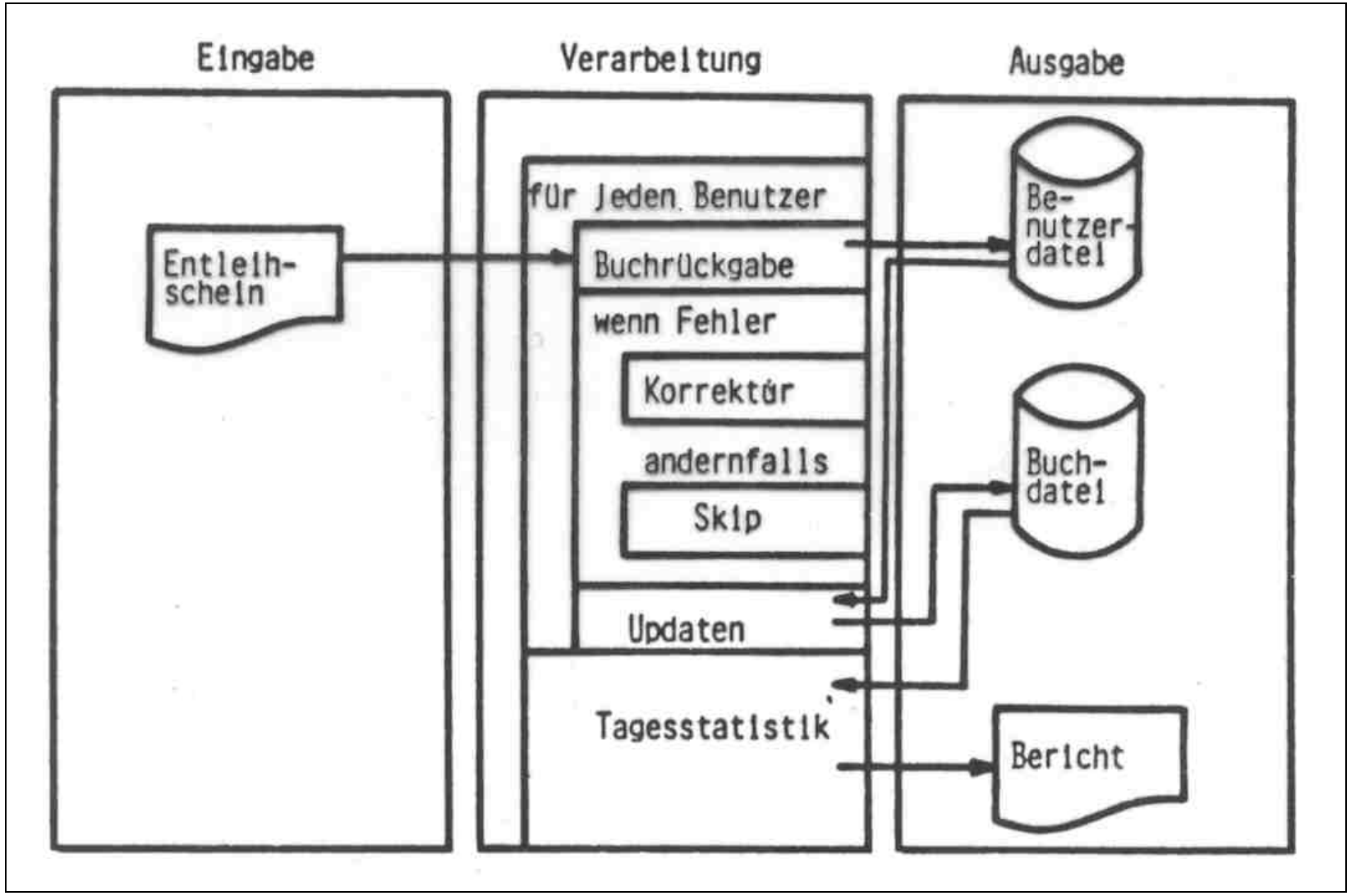
2.3 Behavior models: process / activity vs. state transition



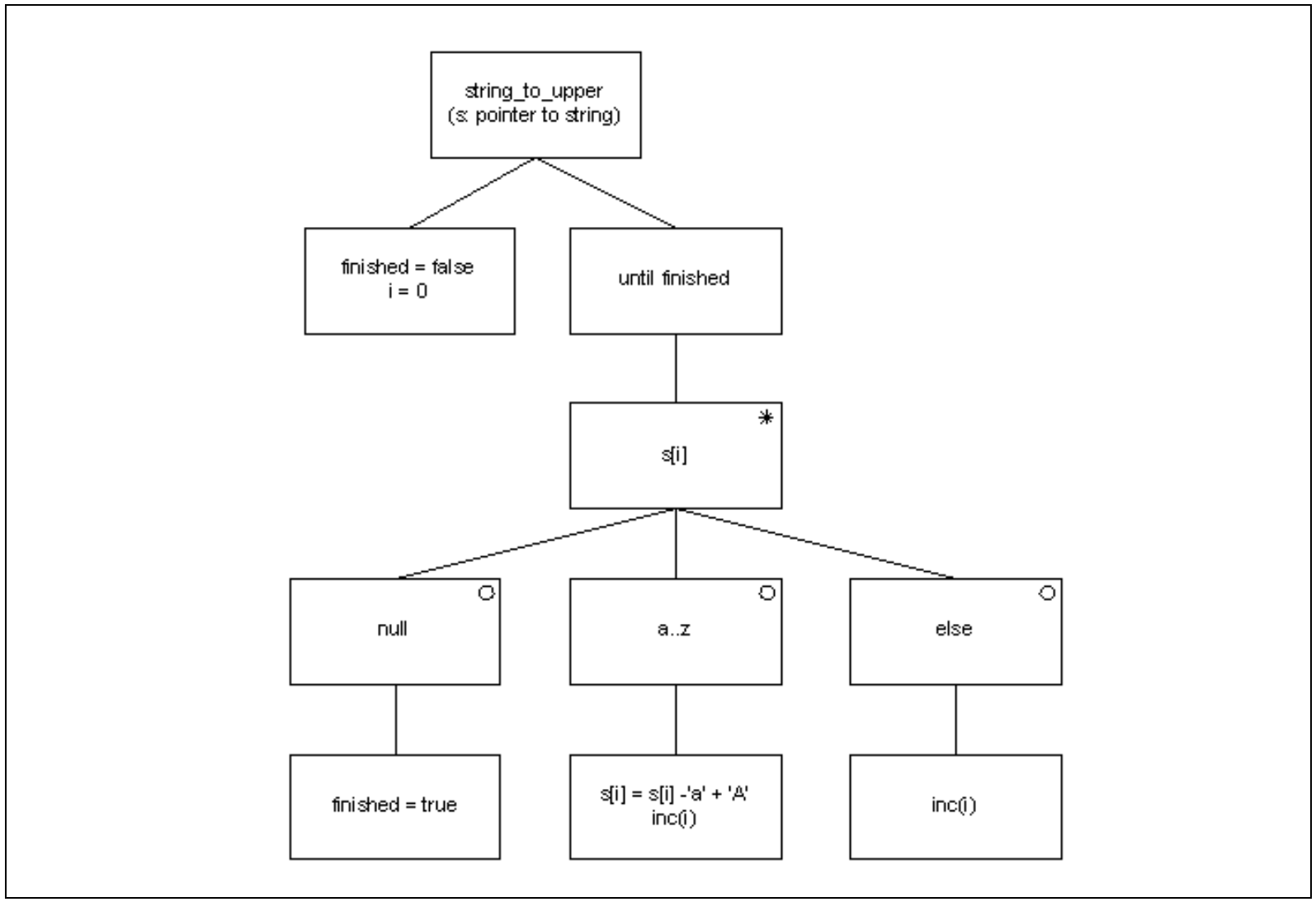
2.3 Behavior models: overview of traditional notations



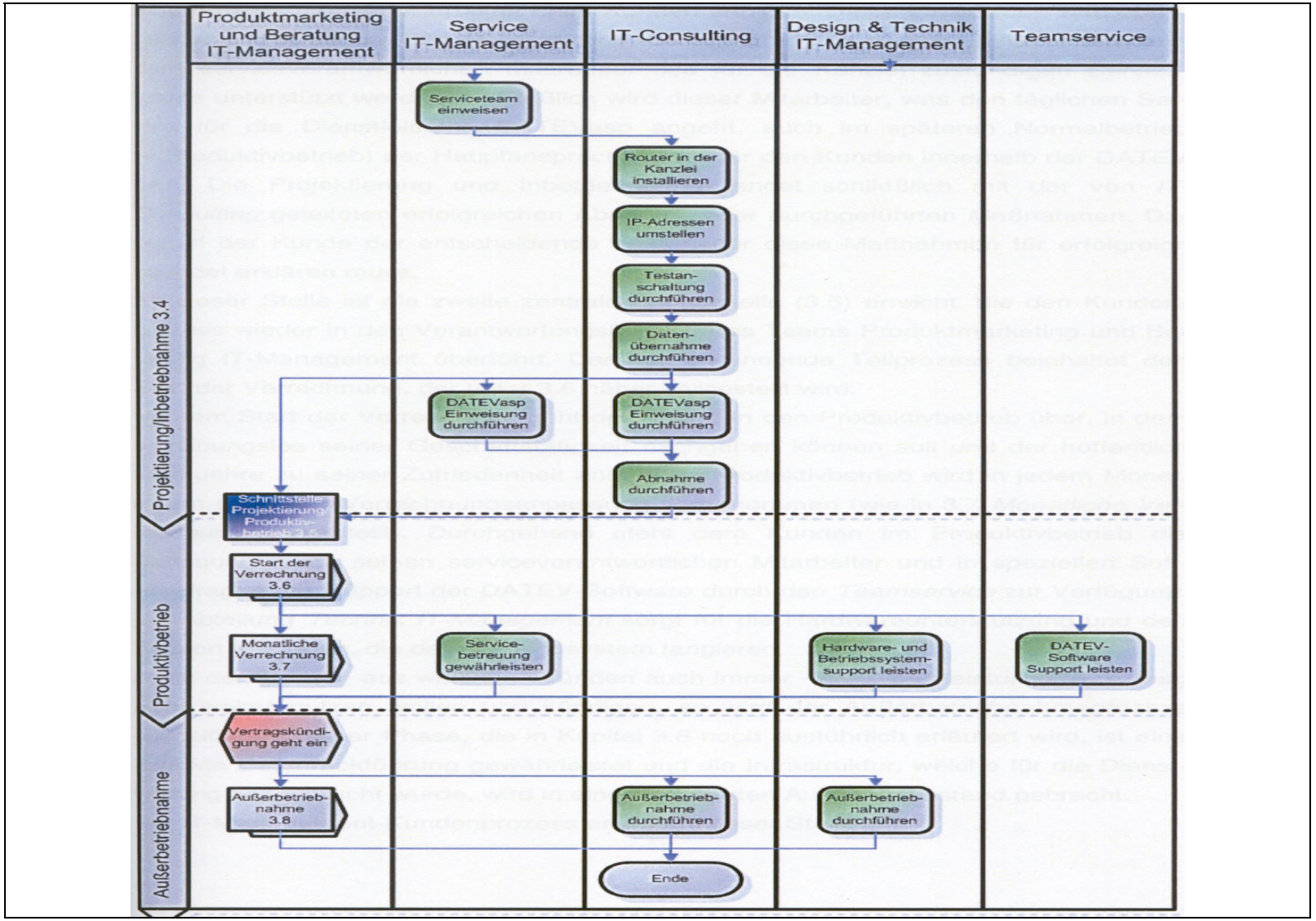
2.3 Behavior models: HIPO



2.3 Behavior models: Jackson Structured Design / Programming



2.3 Behavior models: EPC with swimlanes



2.3 Behavior models: BPMN, EPC, Petri Net (InfSp 37(2014) 196)

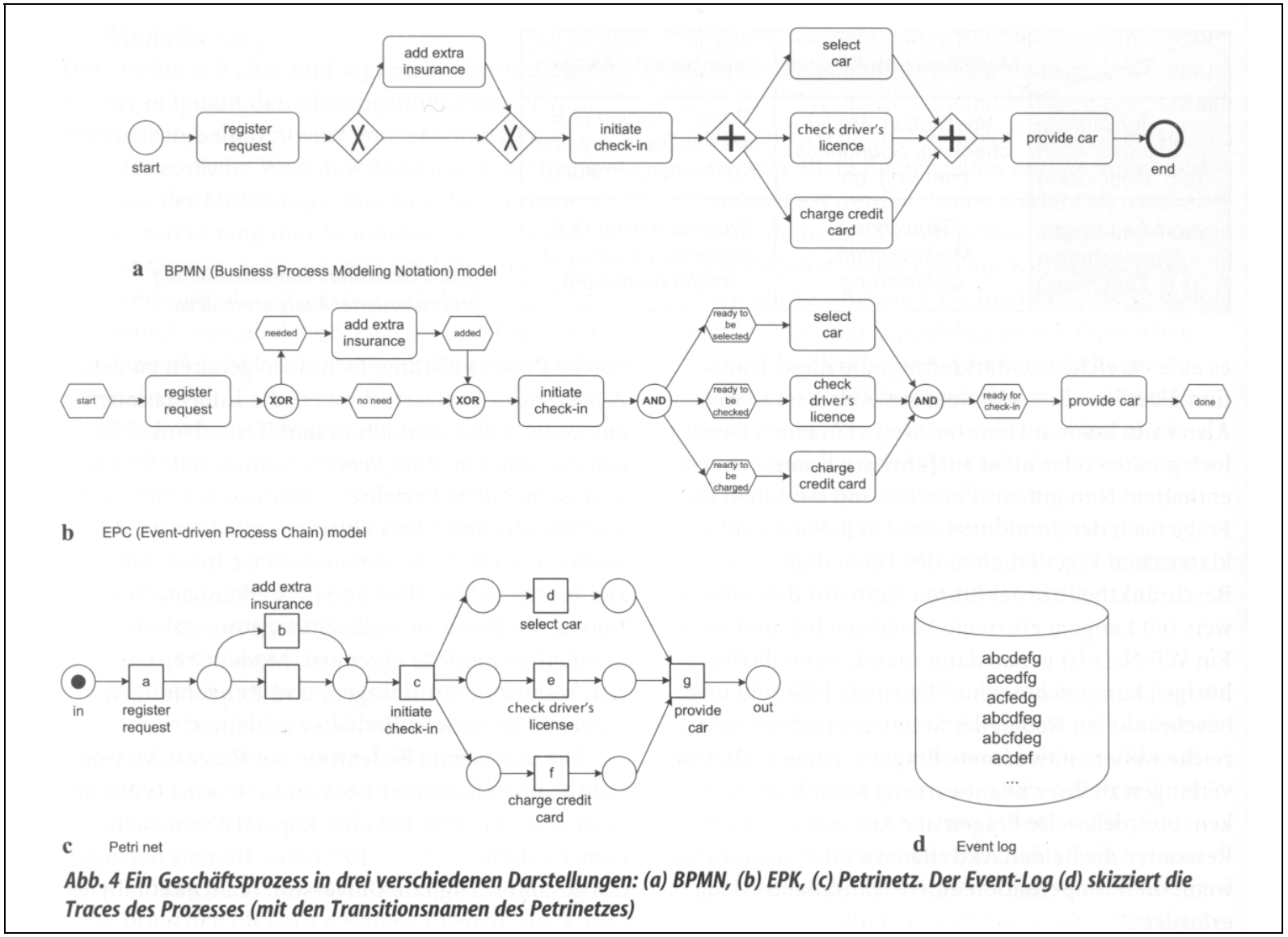
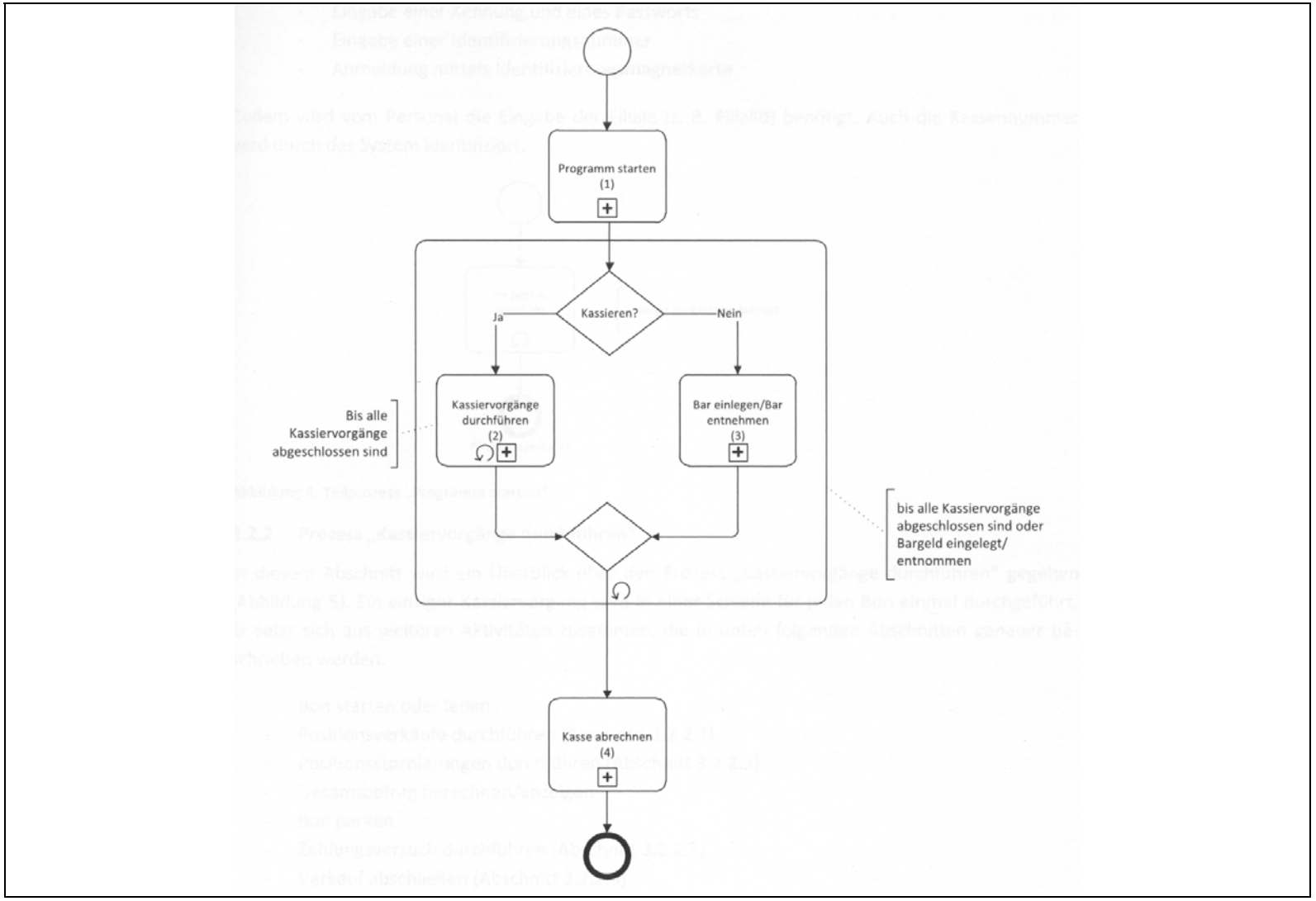
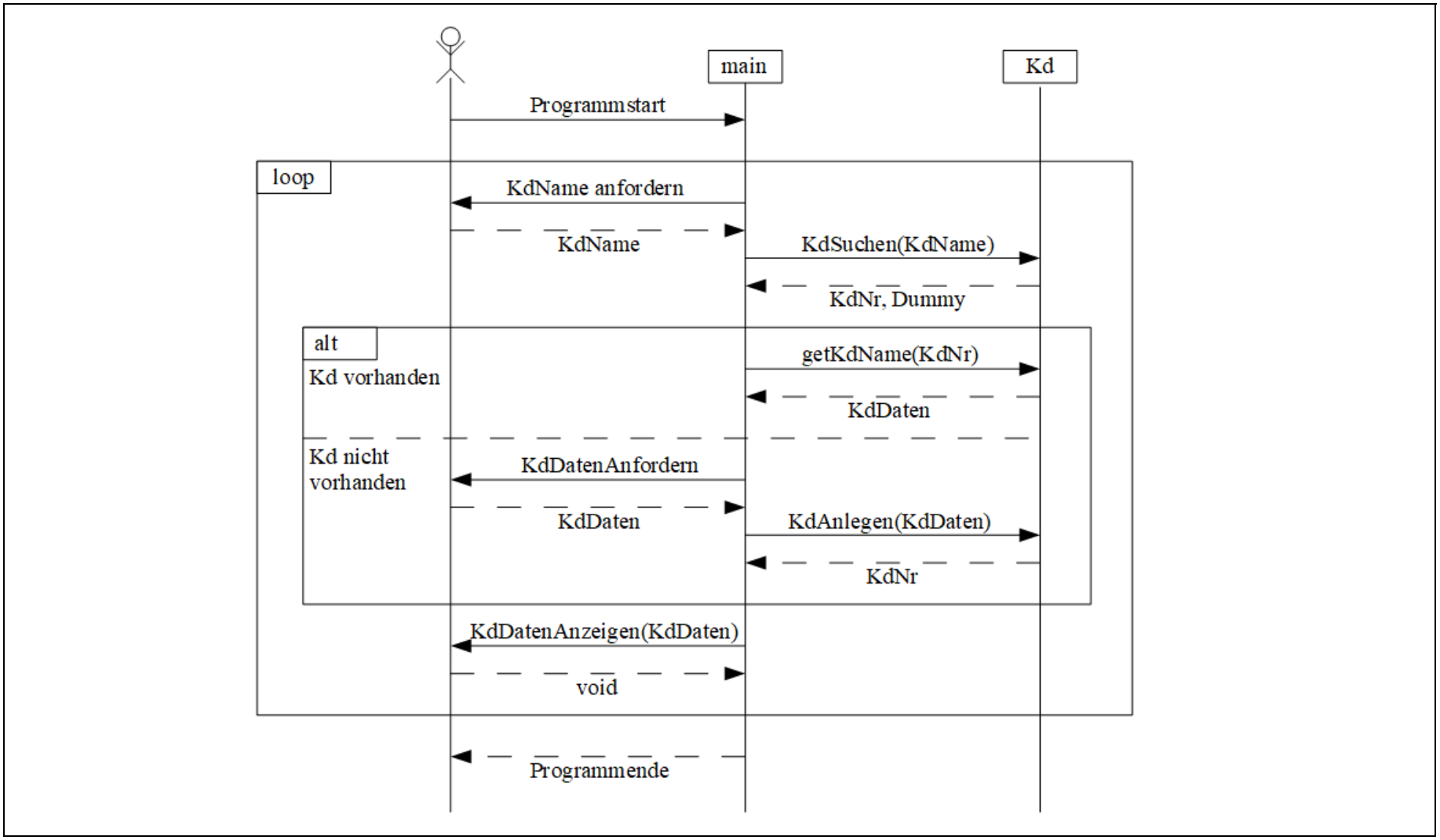


Abb. 4 Ein Geschäftsprozess in drei verschiedenen Darstellungen: (a) BPMN, (b) EPK, (c) Petrinetz. Der Event-Log (d) skizziert die Traces des Prozesses (mit den Transitionsnamen des Petrinetzes)

2.3 Behavior models: structured BPD using BPMN

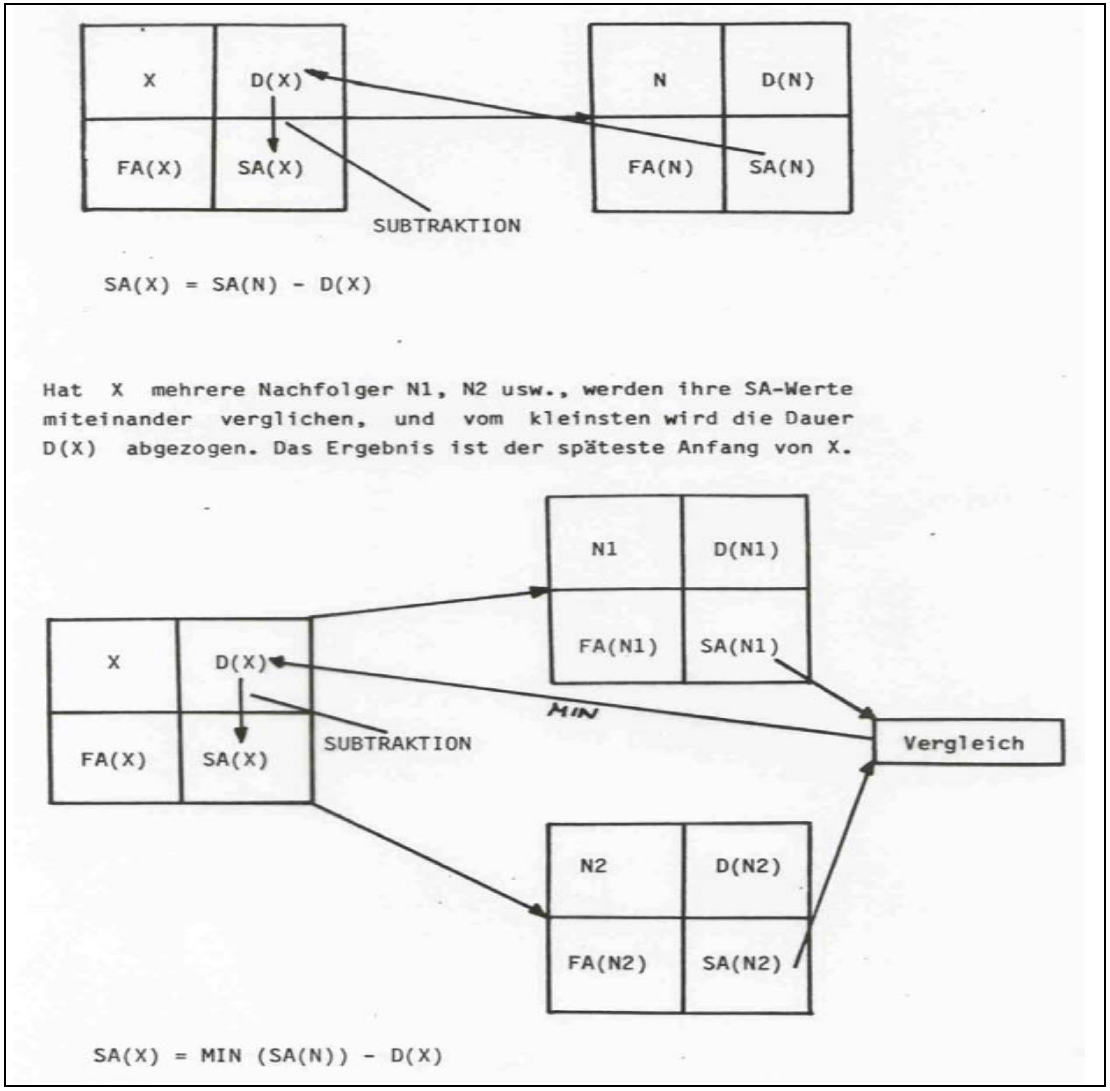


2.3 Behavior models: structured UML sequence diagram

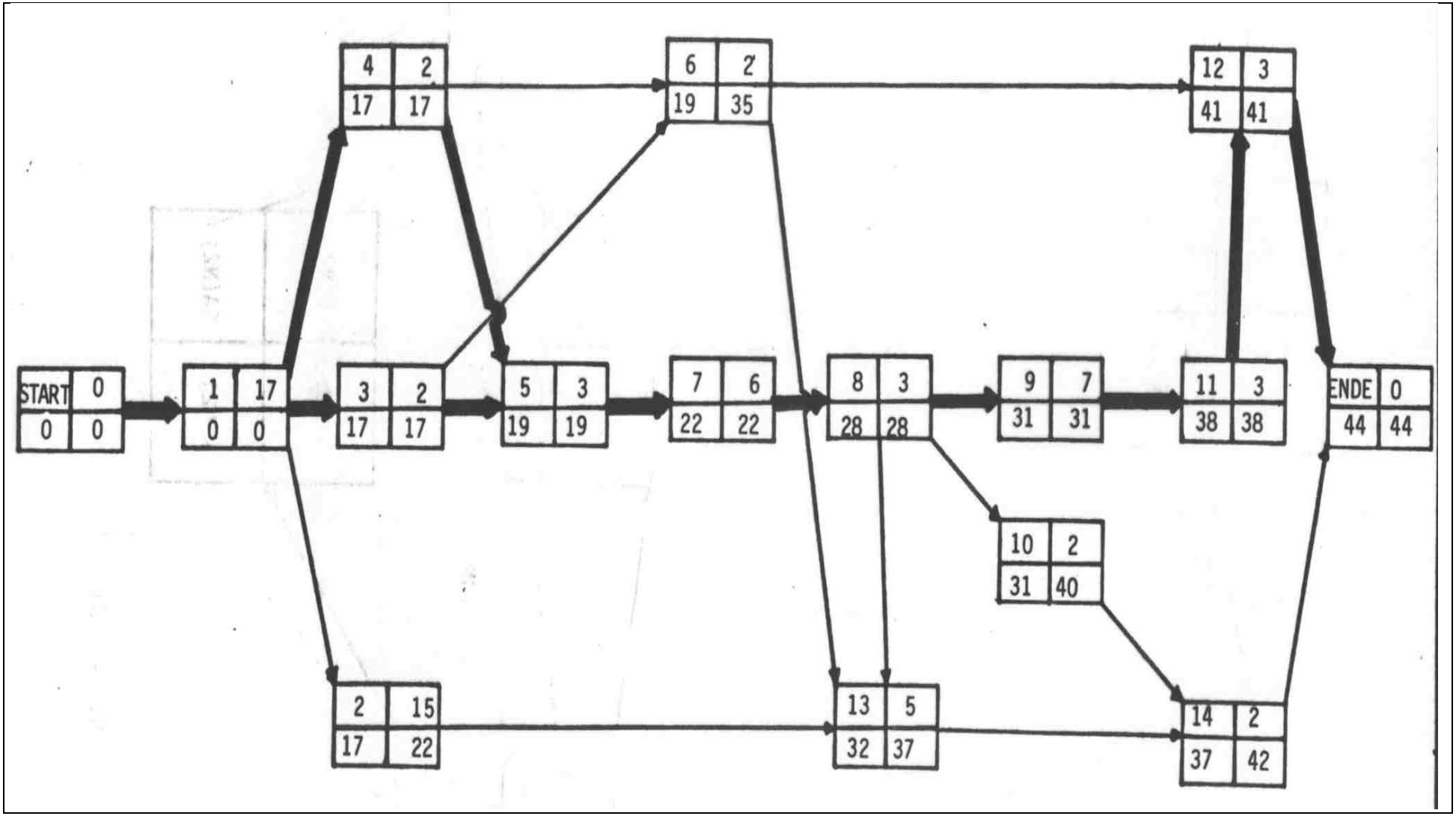


Return Klasse.Methode(Parameter)

2.3 Behavior models: network modeling technique (activity on node)



2.3 Behavior models: network modeling technique (critical path)



2.3 Behavior models: decision table (conditions and actions)

2.3 Behavior models: decision table

Selection of shipment types

	1	2	3	4	5	6
Inland	J	J	J	J	N	N
Dringlich	J	J	N	N	J	N
Gewicht < 10 Kg	J	N	J	N	-	-
Normal			X			
Eilsendung	X					
Lkw-Transport				X		
Luftfracht		X			X	
Bahn-Transport						X

2.3 Behavior models: decision table

CRM: selection of advertisement types depending on customer types

Interessensgrad	-	O	O	O	+	+	+
- keines O latent + gezielt							
Kenntnisstand	x	-	O	+	-	O	+
- ohne Vorkenntnisse O Überblickswissen + sehr genau							
KV legt Wert auf Beratung des Kunden durch Pre-Service?	x	J N	x	x	J N	x	x
Akquisemaßnahmen							
Kunden durch Pre-Service beraten		✓			✓		
Kunden durch KV beraten		✓			✓		
Infomaterial versenden (3.1.1)			✓			✓	
DATEVasp Angebot an Kunden versenden (3.1.2)				✓		✓	✓
Prozessabbruch	✓						

J= Ja, N= Nein, x= beliebig, ✓= zu wählende Aktion

2.3 Behavior models: decision table

Ancient Greek: morphological features depending on infinitive

→ : Verschiebung des Akzents um eine Silbe nach rechts
← : Verschiebung des Akzents um eine Silbe nach links
→ E : Verschiebung des Akzents nach rechts auf die Endung
AZ: Änderung des Akzents von Akut zu Zirkumflex

Silbenzahl		2	2	2	2	>= 3
Vokalquantität der vorletzten Silbe		kurz	lang	kurz	lang	kurz oder lang
Anlaut		vokal.	vokal.	kons.	kons.	vokal. oder kons.
Beispiele		ἄγω	ᾄδω	κτίζω	κλίνω	παιδεύω
Akzent: Änderung/ Verschiebung						
Impv.	2.Sg.	keine	AZ	keine	AZ	←
	3.Sg.	→	→	→ E	→ E	→ E
	3.Pl.	→	→	→ E	→ E	→ E
Impf.	1.Sg.	AZ	AZ	←	←	←
	2.Sg.	AZ	AZ	←	←	←
	3.Sg.	AZ	AZ	←	←	←
	3.Pl.	AZ	AZ	←	←	←

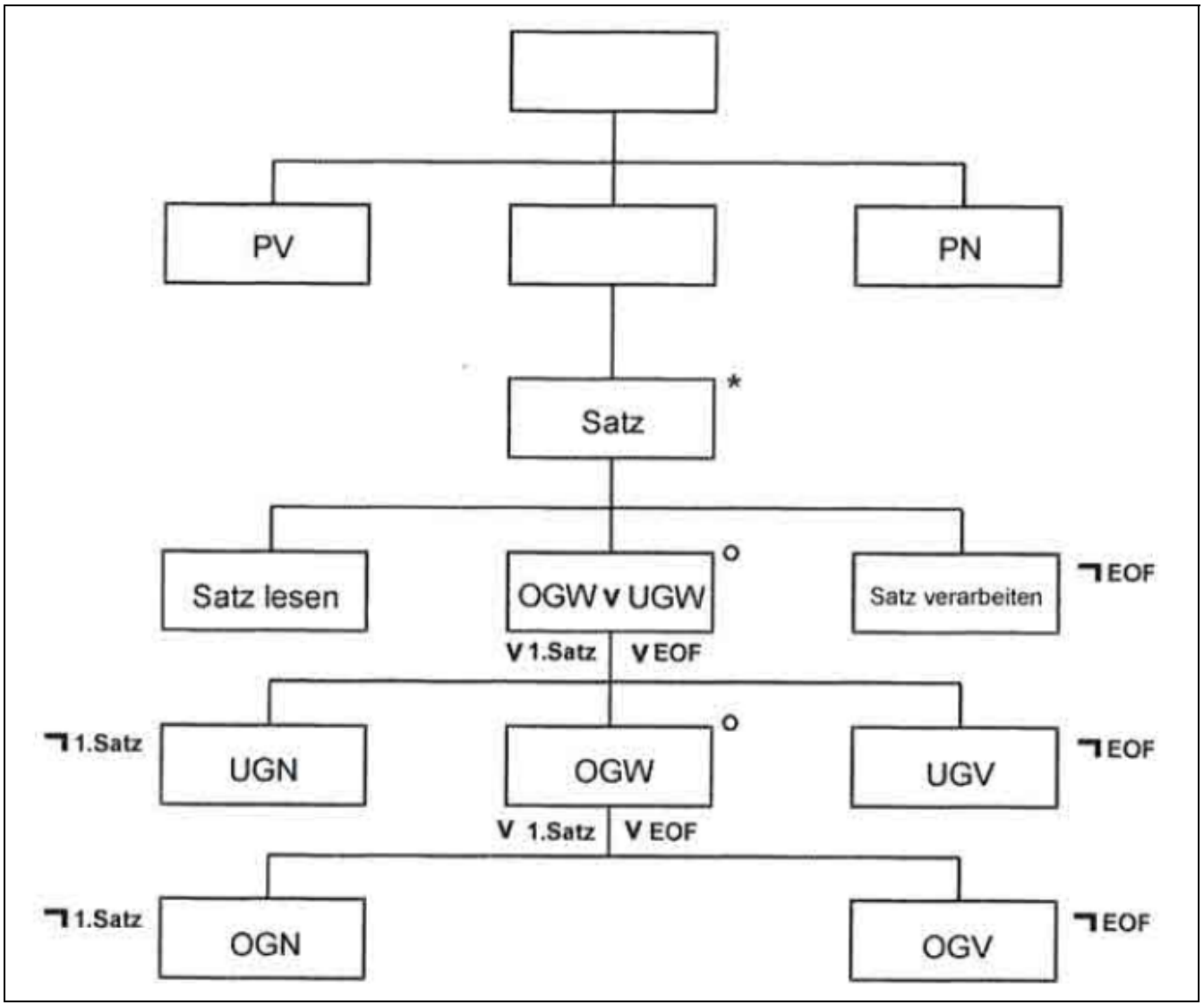
(Holl, Pavlidis, Urban: Rückl. Wörterb. gr. Verbalmorph. 2006: III.66)

2.3 Behavior models: breakpoint analysis – examples

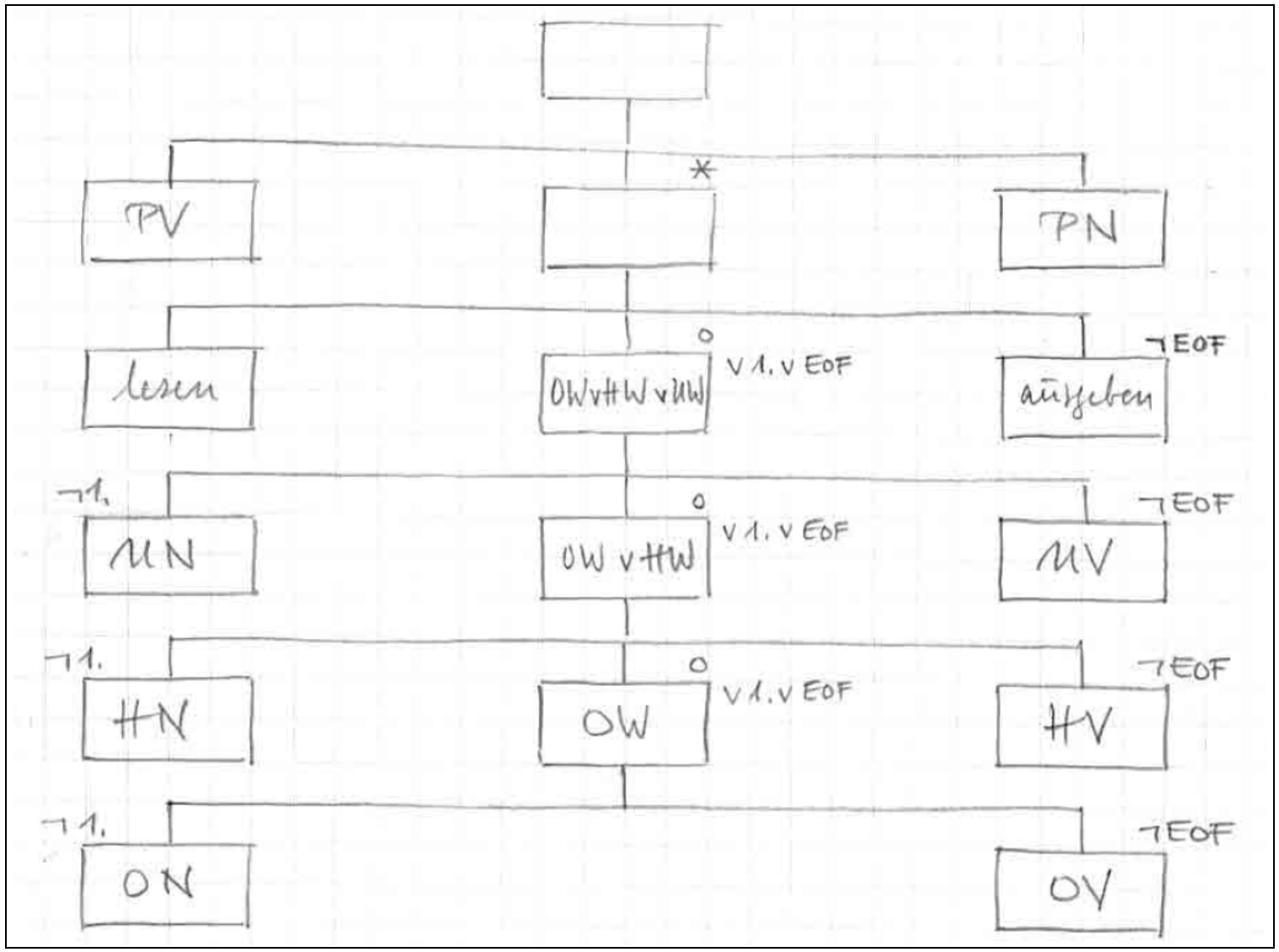
Einstufig:
KdID, LiefDat, LiefWert
→ Sammelrechnungen

Zweistufig:
KdID, LiefDat, ArtID, PosBetrag
→ Sammelrechnungen

2.3 Behavior models: breakpoint analysis – algorithm structure



2.3 Behavior models: breakpoint analysis – algorithm structure



2.3 Behavior models: breakpoint analysis – module description

Programmvorlauf:	Dateien öffnen Gesamtsumme = 0
Programmnachlauf:	Gesamtsumme ausgeben Dateien schließen
Gruppenvorlauf:	Gruppenüberschrift ausgeben Gruppensumme = 0
Gruppennachlauf:	Gruppensumme ausgeben Gesamtsumme = Gesamtsumme + Gruppensumme
Satzverarbeitung:	Datensatz ausgeben Gruppensumme = Gruppensumme + Einzelwert

Abstract/Motivation

Business processes represent a transformation: input is transformed and output with a higher value, higher usefulness and effectiveness for the recipient is created. Depending on the business strategy and the process goals, a business process creates profit for the company. This is the reason why business process modeling (BPM) is so popular nowadays. Every company wants to optimize their processes for creating higher effectiveness and of course larger profits. Furthermore, business processes are normally modeled with the aim that parts of their functionality are supported or implemented by a workflow management system.

As it seems is there a close relation between Computer science (CS) and Information systems (IS). But this is not the fact. For modeling business processes, a model type which was originally developed for software engineering is nowadays used for business process modeling (BPM). This development led to a big jumble of graphical BPM notations: Sometimes graphical notations are used which are not adequate for BPM.

By a genealogical comparison and a comparison using a standardized example the different graphical notations will be evaluated with a strengths and weaknesses profile. The found analogies of those notations and control flow modeling are presented in Table 19 in the end of Section 3.

A high-quality business process model is a good basis for workflow models which are derived from the refinement of business process models.

Therefore, is it very distressing to see that now, nearly 40 years after the introduction of structured software design, structured design for business process modeling is still being neglected. Figure 1 shows a business process model of a german organization, published by Sueddeutsche Zeitung in April 2008.

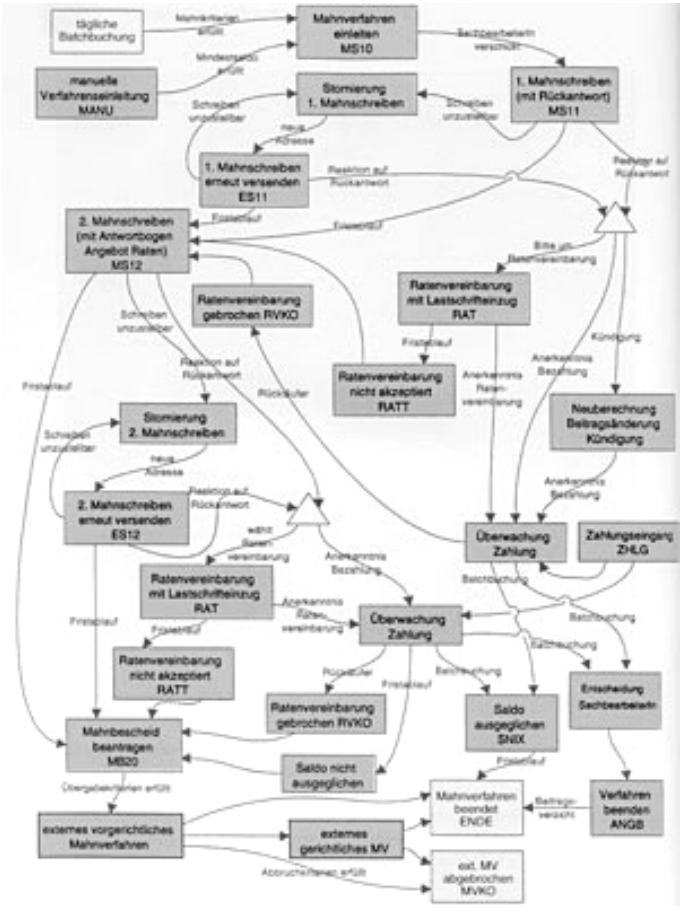


Figure 1: Example of the current modeling style (Sueddeutsche Zeitung, April 14, 2008 [online])

3.2. Distinction of the different graphical notations

Basically, we can distinguish between two types of graphical notations in BPM:
Data flow oriented,
Control flow oriented notations, and
Data flow and control flow oriented notations.

Table 2 shows which graphical notation supports the control-flow aspect, the data-flow aspect or both, also regarding the classical notations as well as the object oriented ones.

		Model type		
		Data flow oriented	**Data- and control flow oriented**	**Control flow oriented**
Notation	**Classical notations**	Data flow diagram (SA, SSA), Section 3.9	SADT/IDEF0, Section 3.8 PROMET (Process Method) diagram, Section 3.13 Business process diagram, Section 3.16	Control flow chart, Section 3.4 Network modeling technique, Section 3.5 Petri net, Section 3.6 Structure diagram (Nassi-Shneiderman), Section 3.7 Swim lane diagram, Section 3.10 Block diagram, Section 3.11 Event driven process chain (EPC), Section 3.12 Extended event driven process chain (eEPC), chaper 3.15
	Object oriented notations	e.g. UML-Use case diagram (only minor usable for BPM)	Not available	UML – Activity diagram, Section 3.14 Object oriented event driven process chain (oEPC), Section 3.17

Table 2: Different model types and notations

3.3.1. Notation of control flow charts

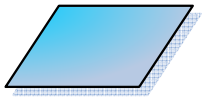
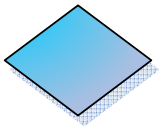
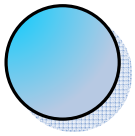
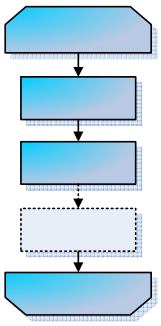
Symbol	Naming	Meaning
	Terminator, Start or End	Shows the materials, information, or action (inputs) that start the process, and the results (output) at the end of the process.
	Data	In- and output of data
	Process	Illustration of a process activity
	Alternative	Illustration of alternatives or decisions. Normally the control flow is split up in two directions (true/false)
	Control flow	Connection to the other elements
	Sub-course/Sub-program	Link to other flow charts.
	Connector	Connectors are used for combinations after alternatives or for reasons of clearness to prevent a crossing of lines.
	Iteration/ Loop	The Iteration defined in the upper square is executed until a certain condition becomes true.

Table 3: Control Flow Chart (DIN66001, 1966)

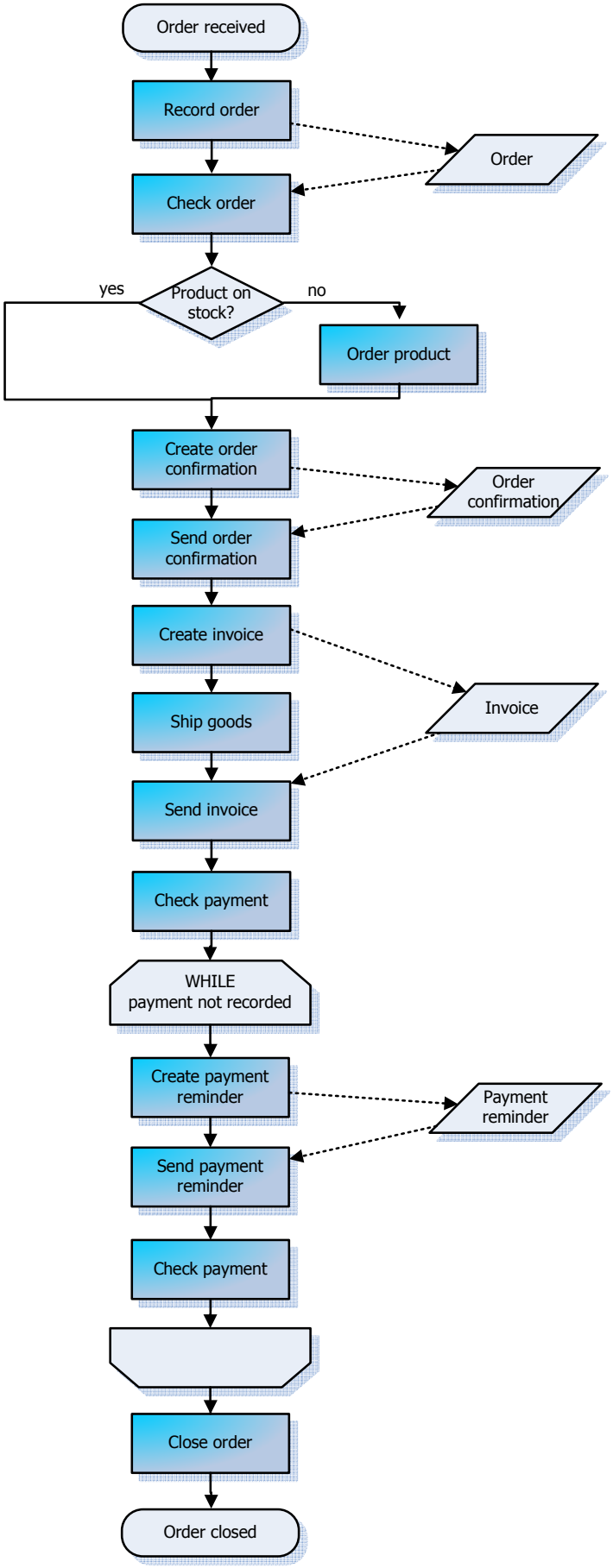


Figure 13: Demonstration of a control flow chart – Improved notation

Pseudo activities – except start and end – are not necessary. The clarity is not influenced by pseudo activities.
Changes in the structure of the model are easily applicable. Arrows can easily be added or removed without the need to re-calculate the model.

3.4.3. Activity on arc network

Every activity (represented by an arrow) belongs to one start- and one end-event (represented by circles). This requires that every activity has to be terminated before another activity can start. Except for the start- and end-event, every node represents a start- and an end-event for different activities. An event can be start- or end-event for more than one activity. However: two events can just be connected by one activity. That is why we need "pseudo activities" for illustrating the parallelism of activities. The use of pseudo activities is also necessary when two or more activities with different start- and end-events are connected with each other. A pseudo activity is represented by a dashed arrow. Please see Figure 15 for an illustration of a pseudo activity. (cf. Heinrich, 2003:218, translated by KG)

3.4.4. Activity Table

Before we can begin with the network model, is it necessary to list the different activities in an activity table. The following table shows the activities for the introduced example of an order process:

Activity	Activity label	Predecessor	Required time (days)
A	Record and check order	--	0,5
B	Order product if necessary	A	0,2
C	Create order confirmation	A, B	0,1
D	Send order confirmation	C	0,1
E	Create invoice	D	0,1
F	Send invoice	E	0,1
G	Ship order	D	1
H	Check payment	E	0,2
I	Create payment reminder	H	0,1
J	Send payment reminder	I	0,1
K	Close order	J	0,2

Table 4: Table of activities

The critical path, which runs without interruption from the start activity to the end activity, is represented by a thick black line. The critical path represents the critical activities where the time buffer is null.

3.4.6. Demonstration of an activity on arc network

The following demonstration shows the network model of the order example, based on the activity table with eleven activities (A to K); all important relationships are easily cognizable. The earliest possible end date of activity K, and so for the entire process, is in time unit 1,3. The latest end date is in time unit 29,3 (planned end date). The model contains four pseudo activities (illustrated by dashed arrows).

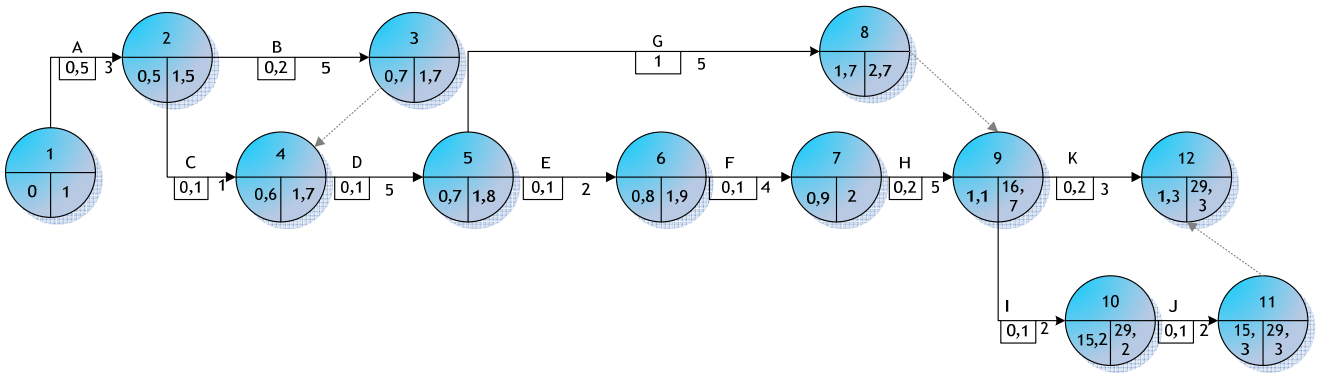


Figure 15: Demonstration of an activity on arc network

3.4.7. Evaluation of the NMT activity on arc networks notation

NMT activity on arc networks are not understandable without the corresponding activity table. The notation offers no possibility for a direct naming of the different activities in the diagram. Another problem talking about clarity and clearness are the pseudo activities. Pseudo activities are necessary for illustrating the process logic in our order example, but they are making a structured design impossible. The data flow aspect is not regarded, but NMT activity on arc networks are a good tool for time planning and analysis of business processes.

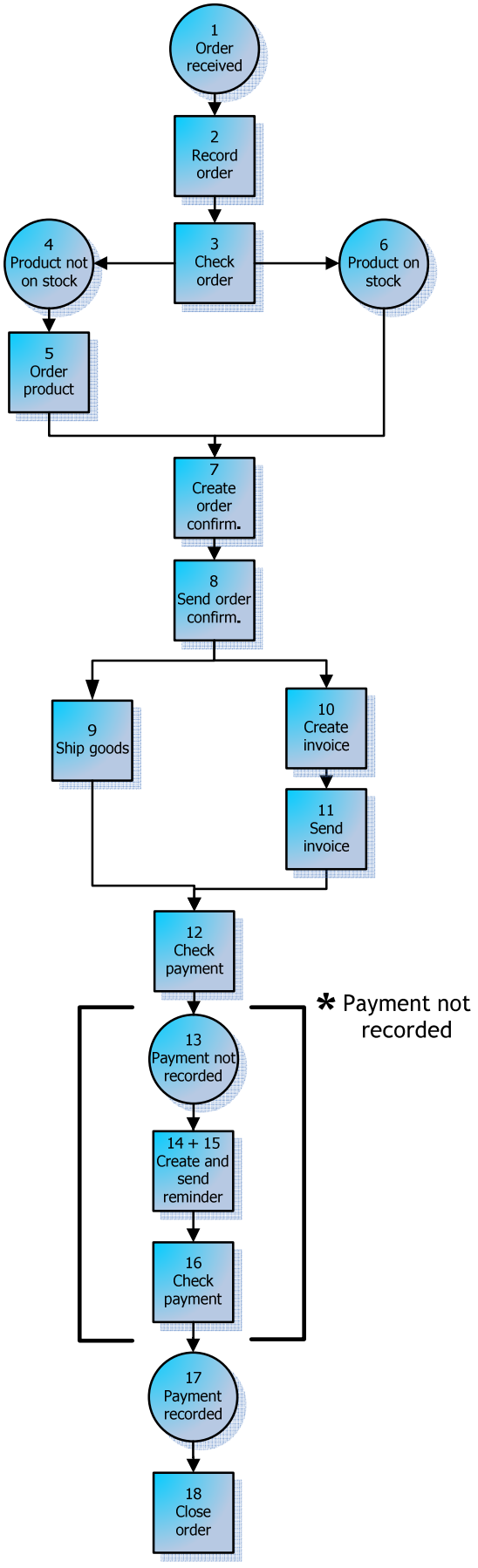


Figure 22: Demonstration of a Petri net (P/T) without marks – Improved notation

3.6. Structure diagram (Nassi Shneidermann)

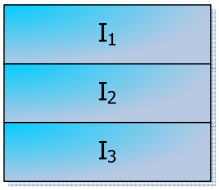
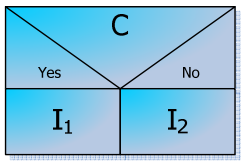
Nassi Shneiderman diagrams are also called “structure diagrams”. Those diagrams are normally used for an abstract description of algorithms, but they can also be used in business process modeling. The diagrams describe procedural program sequences. The regarding processes are described on a code independent level (without a direct connection to a programming language) and so they are representing “top-down” design. (Grotehusmann, [online], translated by KG)

When structured programming became popular in the 1970s, Nassi-Shneiderman diagrams were introduced as tools for the design of structured program courses.

‘Nassi Shneiderman diagrams are often better organized, structured and also more understandable then other notations. That is the reason why this kind of notation is preferred for representing process specifications. But a considerable amount of diagrams is though necessary.’ (Yourdon, 1992:263, translated by KG)

The Nassi Shneiderman notation is the only graphical notation which forces the model designer to a structured design. (cp. Section 2)

3.6.1. Notation of Nassi Shneiderman structure diagrams

Symbol	Naming	Meaning
	Sequence	The instructions of a program or business process are placed in rectangles. Within those rectangles a description of the instruction, or the instruction itself is listed. The instructions are serialized in the following order: Instruction I_1, Instruction I_2, Instruction I_3…
	Alternative (IF/THEN/ELSE)	Within a sequence is it sometimes necessary to execute one or more instructions under a certain condition. We can also call those IF/THEN/ELSE selections. Interpretation: If the condition C is true, instruction I_1 is going to be executed. Else instruction I_2 is going to be executed.

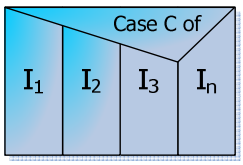
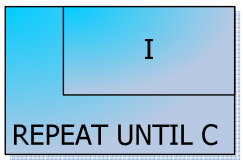
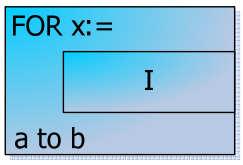
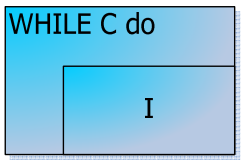
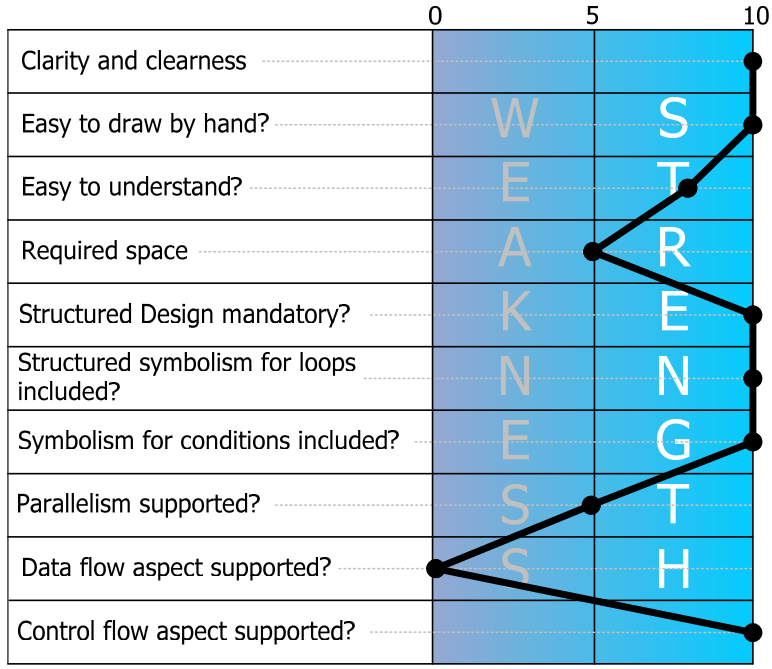
	Case differentiation (CASE)	With the help of case differentiations, an instruction can have multiple exits.
	REPEAT UNTIL Iteration	Iteration I is executed until condition C becomes true.
	FOR Iteration	FOR iterations are executed for a known, certain amount of runs. An internal control variable (x) runs with an interval (a, b). As long as the control variable lies within the interval (a, b), iteration I is executed and the control variable increases (x + increment parameter). If there is no parameter listed, the increment is automatically 1.
	WHILE Iteration	As long as condition C is true, iteration I is executed.

Table 7: Elements of Structure Diagrams (Kubicek and Kamin, 2005, [online] translated by KG)

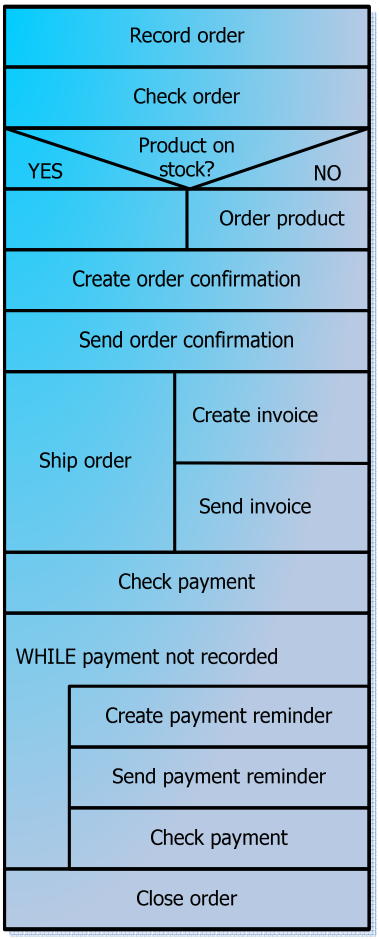
Clarity and clearness	10
Easy to draw by hand?	10
Easy to understand?	8
Required space	5
Structured Design mandatory?	10
Structured symbolism for loops included?	10
Symbolism for conditions included?	10
Parallelism supported?	5
Data flow aspect supported?	0
Control flow aspect supported?	10



Please see Section 3.3.4 for more information about the evaluation procedure.

Figure 24: Strengths and weaknesses profile of Nassi Shneiderman diagrams

3.6.4. Idea for improvement of Nassi Shneiderman diagrams



An idea for improvement of the Nassi Shneiderman notation is a possibility for illustrating parallel activities. As you can see in figure 25 I split up the “Ship goods” box to illustrate the parallelism to “Create invoice” and “Send invoice”.

Figure 25: Demonstration of a Nassi Shneiderman diagram - Improved notation

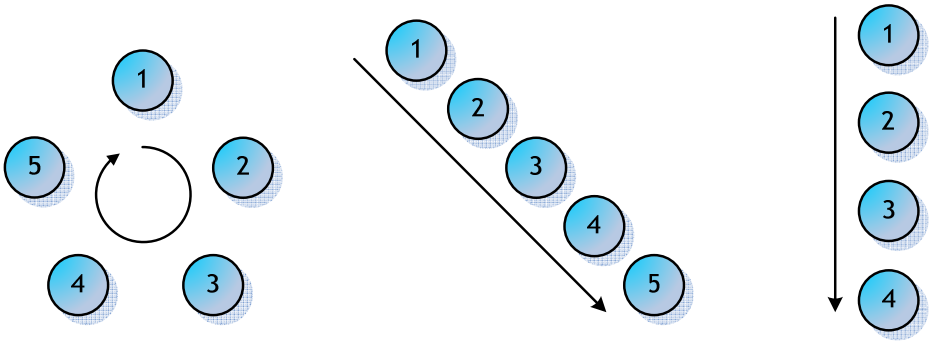


Figure 33: Improvement suggestions for the control flow representation in DFDs

I supplemented the demonstration example from Figure 31 with both of my improvement suggestions. With the help of numbers and letters is it also possible to represent parallelism of two actions, as you can see in Figure 34.

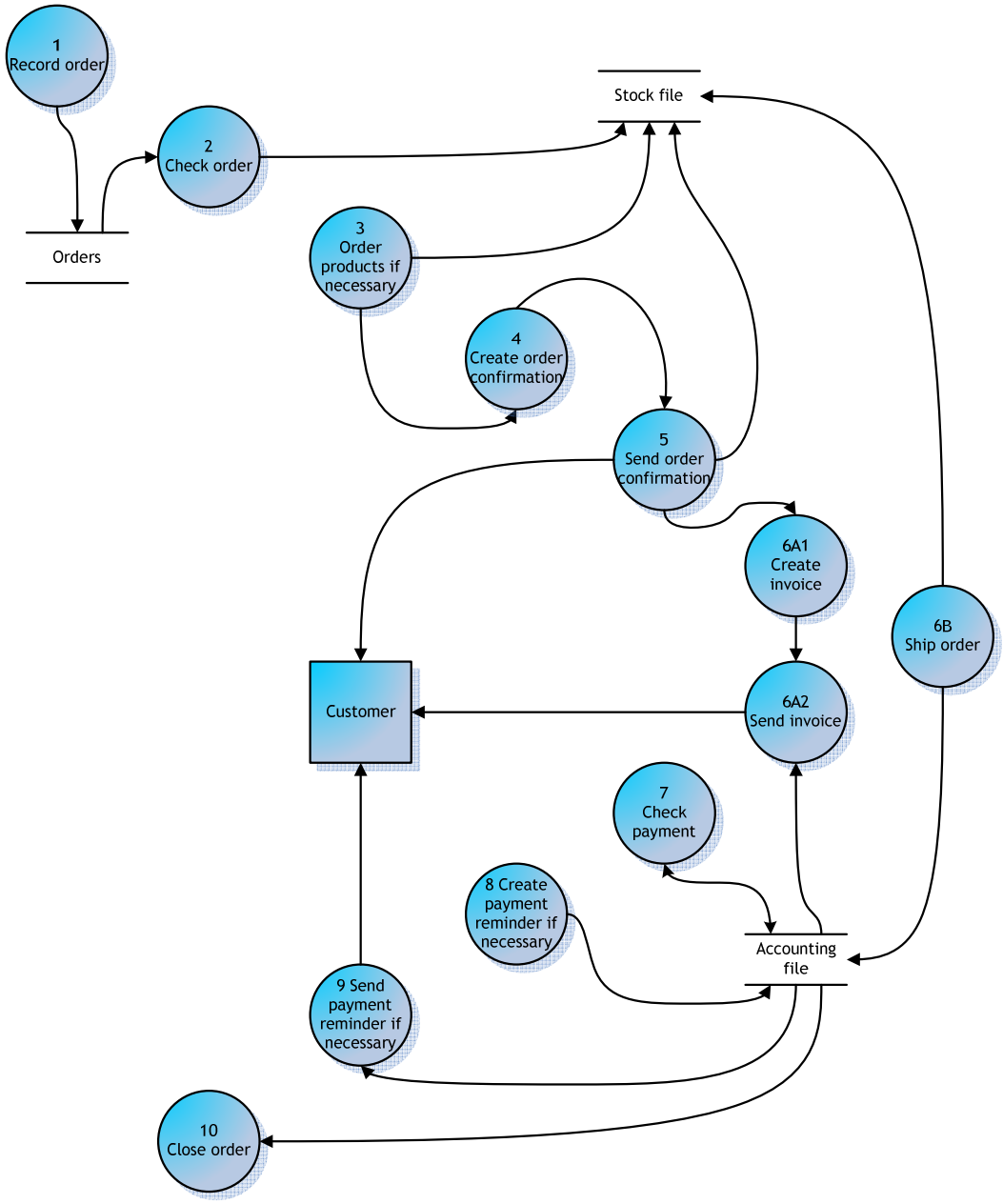


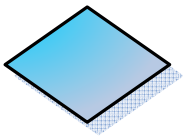
Figure 34: Demonstration of a data flow diagram – Improved notation

3.9. Swim lane diagram

Swim lane diagrams were originally called “Organisationsprozessdarstellung (OPD)” [Organizational process representation] and developed by H. F. Binner in the early 1980s. A swim lane is analogous to an area of responsibility for the process actors, where the dedicated responsibility is see-sawing in between until the procedure is completed. UML activity diagrams and the business process diagrams from Österle inherited swim lanes (cp. Gadatsch, 2005:62, translated by KG).

3.9.1. Notation of Swim lane diagrams

The notation of swim lane diagrams has been developed in different ways and can have different characteristics, depending if you are modeling a rough process model or a detailed workflow model.
In the simplest form, swim lanes are designed with just a few elements that make them very easy to draw. For workflow modeling, more precise elements are available e.g. control flow operators.

Symbol	Naming	Meaning
	Process step	Illustration of activities
	Control flow	Temporal logical course of process steps supplemented with alternatives (yes, no) and probabilities in %
	Alternative	Alternative in the course of activities
	Logical operator ‘exclusive OR’	For workflow models logical linking operators can be used additionally
	Logical operator ‘OR’	
	Logical operator ‘AND’	

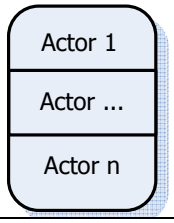
	Swim lane	Responsibilities of the actors (persons, departments, companies)
	Document	Document, Database or Information object relating to the course

Table 11: Notation swim lane diagrams (Gadatsch, 2005:63, translated by KG)

3.9.2. Demonstration of a swim lane diagram

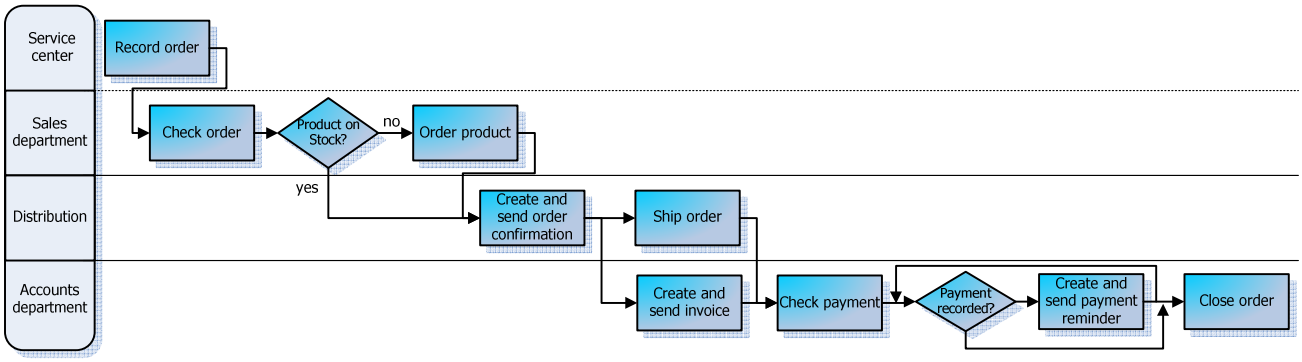
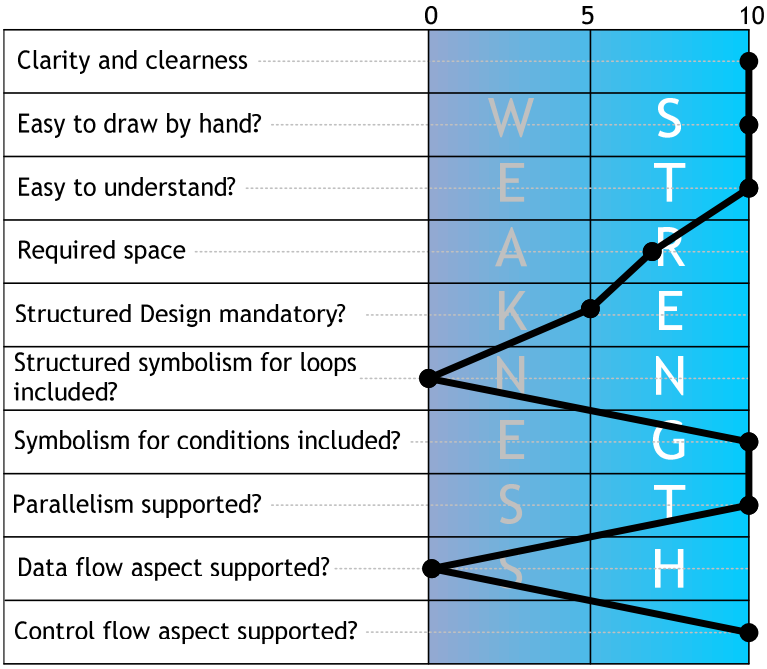


Figure 35: Demonstration of a swim lane diagram - Original notation

3.9.3. Evaluation of the swim lane diagram notation

Swim lane diagrams give a fast, clear and simple overview over the process responsibilities and the process course itself. The graphical notation is easy to draw by hand and supports all the important and required symbolism for BPM.

Clarity and clearness	10
Easy to draw by hand?	10
Easy to understand?	10
Required space	7
Structured Design mandatory?	5
Structured symbolism for loops included?	0
Symbolism for conditions included?	10
Parallelism supported?	10
Data flow aspect supported?	0
Control flow aspect supported?	10



Please see Section 3.3.4 for more information about the evaluation procedure.

Figure 36: Strengths and weaknesses profile of swim lane diagrams

3.9.4. Idea for improvement of swim lane diagrams

Structured symbolism for loops:

With the use of []* the unstructured diagram (Figure 35) can be easily transformed into a structured diagram (Figure 37). The original notation includes no structured symbolism for loops.

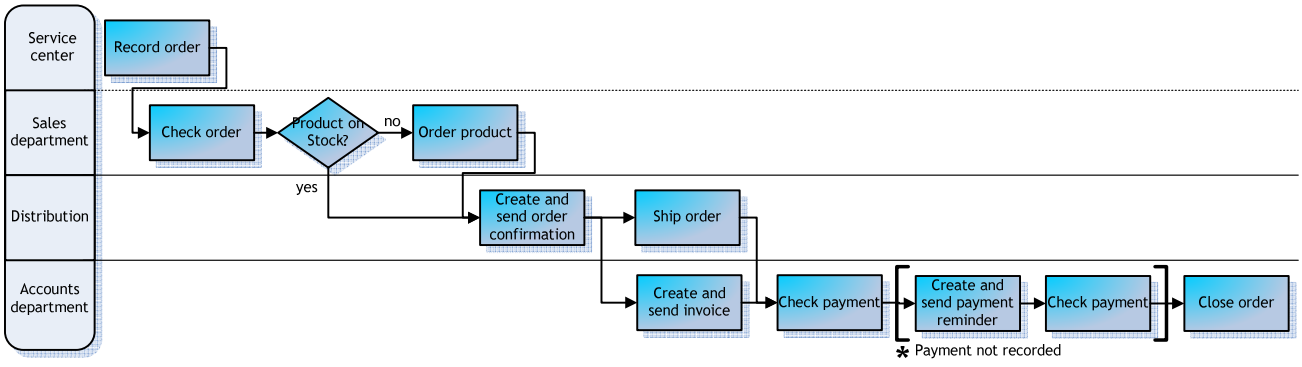


Figure 37: Demonstration of a swim lane diagram - Improved notation

3.10.2. Demonstration of a block diagram

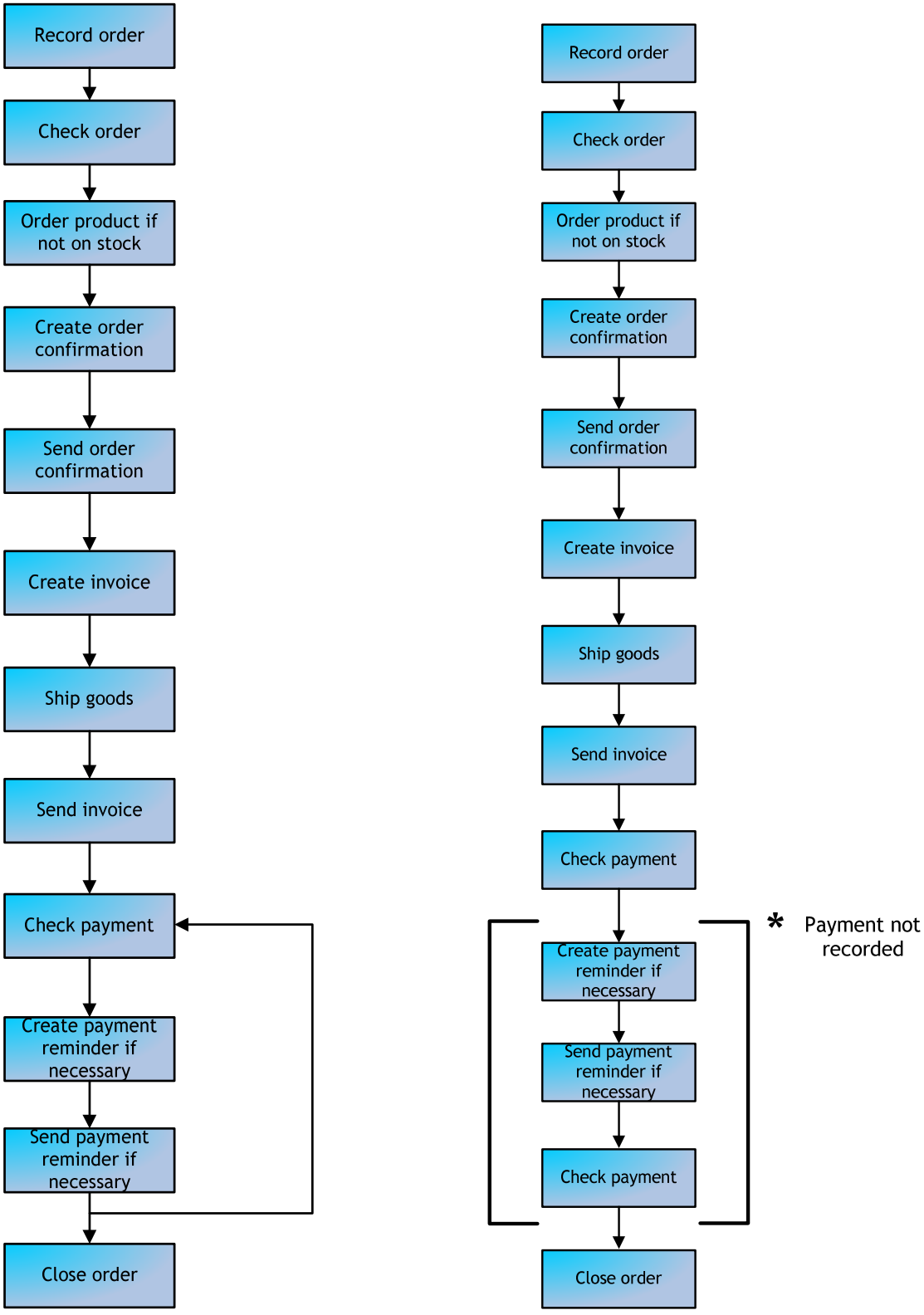


Figure 38: Demonstration of a block diagram - Original notation (left) and Improved notation (right)

3.11. Event-driven process chain

Event driven process chains (EPCs) were developed by Keller, Nüttgens, and Scheer and are representing a semi-formal notation for modeling business processes. Next to the description of business processes in the as-is state, EPCs are used for the design of process models in the target state. (cp. Heinrich 2003:403 and Staud, 2006:59, translated by KG)

EPCs are based on the concept of the directed graph (like Petri Nets, please see Section 3.6). Nodes are representing events; directed links are modeling the control flow.

3.11.1. Notation of EPCs

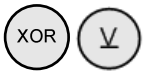
Symbol	Naming	Meaning
	Event	Description of a occurred state from which the rest of the process course is dependent. Events are initiating functions or are the results of functions. Events are passive elements in the process chain.
	Function	Functions are active elements in the process chain. They can be defined as tasks or performances. Functions are changing the system state, meaning that they are processing or executing material and informational objects by reading, changing, deleting, or producing them. So the aim of the function is the transformation of input to output. Therefore, functions can make decisions and influence the following course of the process. Functions are linked by events.
	Logical operator ‘exclusive OR’	Logical linking operators are describing the logical linking of events and functions
	Logical operator ‘OR’	
	Logical operator ‘AND’	

Table 12: Basic Elements of EPCs (Gadatsch, 2005:158, translated by KG

Events and functions are connected by directed links which are represented by dashed lines.

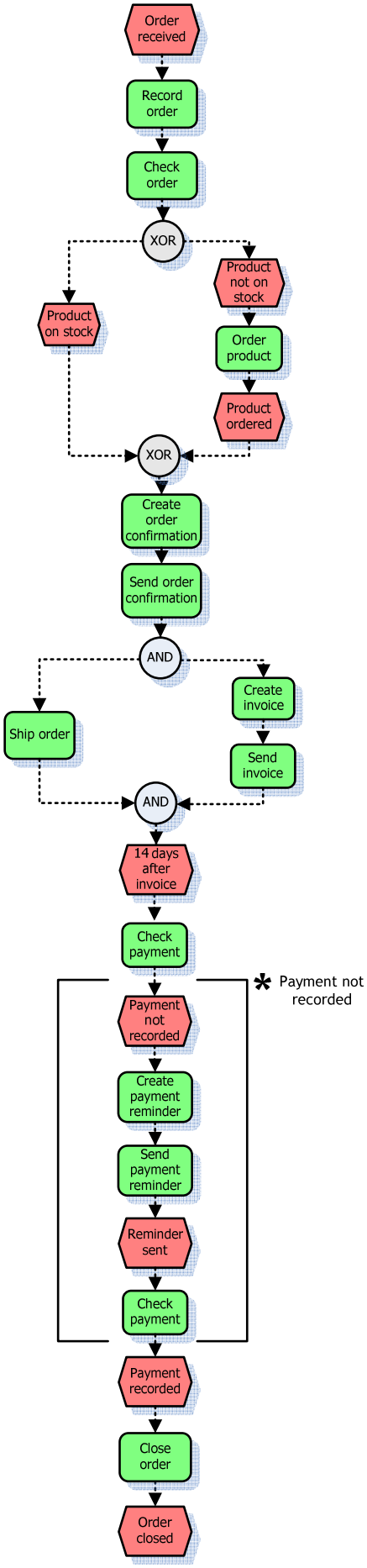
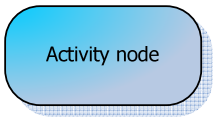
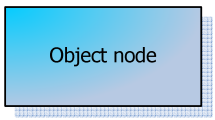
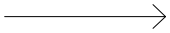
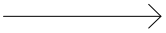


Figure 42: Demonstration of an EPC diagram – Improved notation

3.13. UML activity diagram

An UML activity diagram describes a course with the help of actions and edges. With this kind of diagram, use cases can be described in natural language and illustrated graphically. With natural language, normally only trivial courses can be documented clearly whereas with activity diagrams, it is possible to document very complex courses in a clear and comprehensible way.
The elements of an activity which illustrate actions are called activity nodes. An activity consists of activity nodes and activity edges. The activity edges are directed and connect the nodes. (cf. Staud, 2006:375 ff)

3.13.1. Notation of activity diagrams

Symbol	Naming	Meaning
	Activity	An action is a single step in a process course and is represented by a rectangle with rounded edges and with the action name in the middle. An action normally has ingoing and outgoing edges. Ingoing edges are starting an action. If there are more ingoing edges existent, the action is started when all ingoing edges are activated. At the finalization of an action all outgoing edges become active. Activity nodes can include different behaviors e.g. a call for an operation or an execution. They can also include control flow elements for alternative or combination of edges when nodes are processed parallel.
	Object	Object nodes are representing objects as we understand them from the object-oriented approach. The task of an object node is to model the flow of objects of an activity.
	Control flow edge	The execution of a node leads to the execution of other nodes. The control is passed on from node to node on the edges: If an action is executed, the control is passed on to the next one, etc.
	Object flow edge (Data flow)	We can distinguish two different flows in activity diagrams: Control flows and object (data) flows. Both flows are representing different, but dependent aspects!

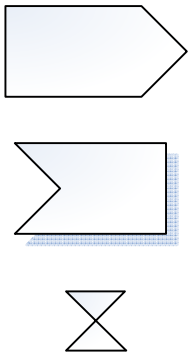
		The distinction between control and data flows is strict: Every edge which is leading to or leading from object nodes has to be object flow edges!
	Signal	A signal represents a notification about an important incident which immediately activates an edge. Signals are illustrated by arrows. Particularly for time events, a sandglass symbol is used. Received signals are illustrated by an edge which points to the action which receives the signal. Sent signals are illustrated by an edge which is leading away from the action which sent the signal.
Swim lanes		Swim lanes in an activity diagram are describing who or what is responsible for a certain activity. Swim lanes are illustrated by vertical lines. The elements of an activity diagram (activities, object nodes) are always exactly situated within one swim lane.
	Decision Node/Merge node (Alternative/Exclusive OR)	Control nodes
	Fork Node/Join Node (AND)	
	End node – Flow Final	An end node marks the end of a course in an activity diagram. They are illustrated by a filled circle with a ring around it.
	End node – Activity Final	An activity end node is illustrated by a circle with a cross in it. Activity end nodes and final end nodes have at least one ingoing, but no outgoing edges.
	Start node – Initial node	A start node marks the start of a course and it has outgoing, but no ingoing edges.

Table 14: Basic Elements of an UML activity diagram (cp. Staud, 2006:374ff, translated by KG)

An activity diagram consists of a start- and an end-node as well as some actions and edges which are connecting those elements together. In- and outgoing objects are called parameters of the activity. Edges are describing the transition of an action to the next one and so the control- and

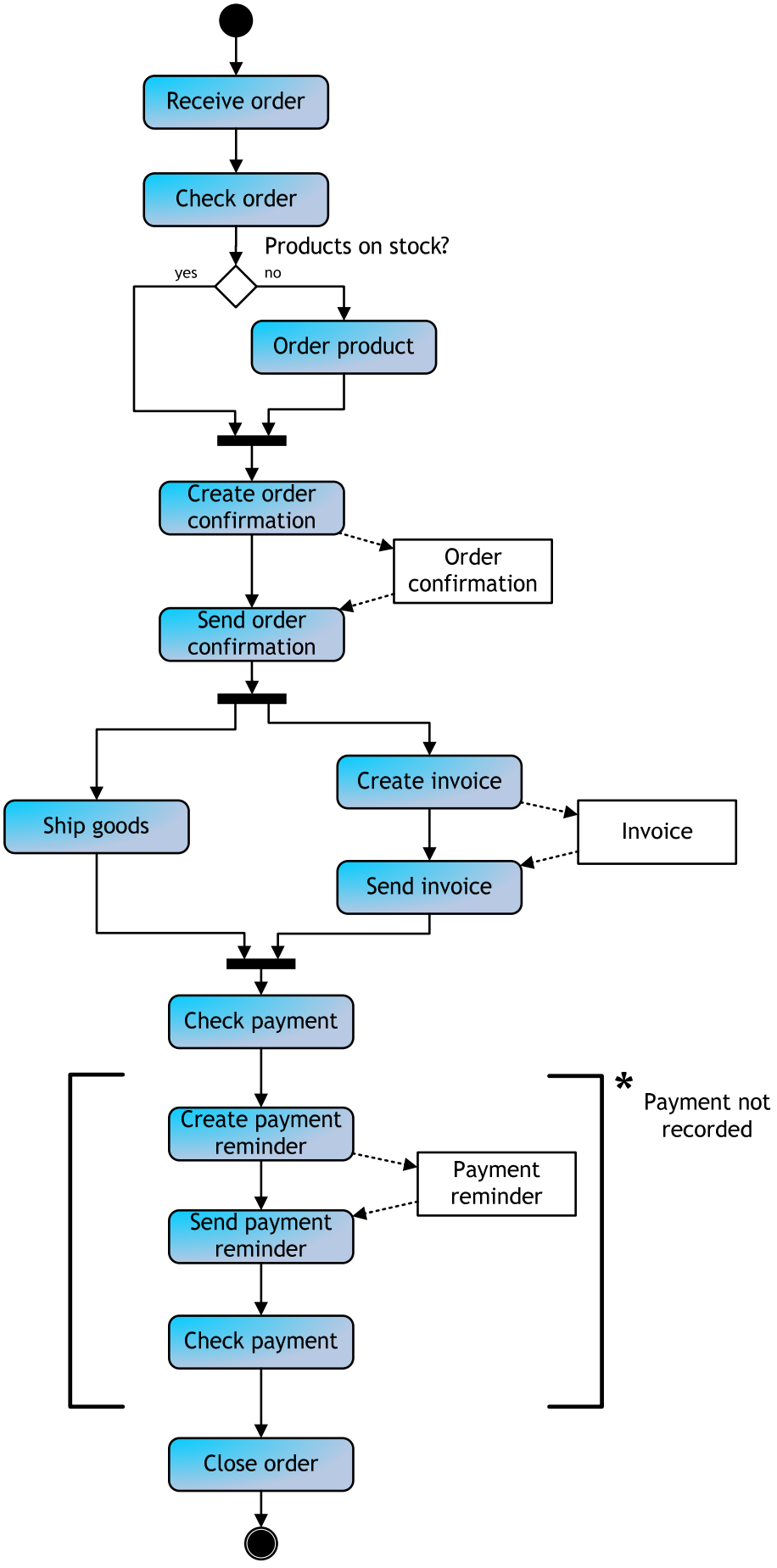


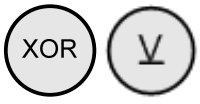
Figure 48: Demonstration of an UML activity diagram - Improved notation

3.14. Extended event driven process chains (eEPC)

The original EPC notation is recommended for a simple illustration of the control flow. For a more advanced illustration, the connection of all views on the business process is missing. In 1998 the original EPC notation got some advancement and is called extended EPC. Functions require not only events which are triggering courses, but they also need inputs (especially information, but material as well). For the analysis and the design of business processes, not only the static structure of data is important, also the dynamic view, especially when we are talking about information. Also essential for the optimal information supply is information about where and from which position or function in the company the information objects are accessed or changed.
(cp. Gadatsch, 2005:82f, translated by KG)

This is the reason why the extended event driven process chain includes information objects in the notation. Information objects show which information in the process course is used by which function. The information objects are illustrated by rectangles. A drawn through arrow, directed on a function means the usage (e.g. reading) of information by a function. An arrow, directed on an information object means the processing or the changing of the information object by a function.
(Heinrich, 2003:405, translated by KG)

3.14.1. Notation of eEPC diagrams

Symbol	Naming	Meaning
	Event	Description of a occurred state from which the rest of the process course is dependent
	Function	Description of the transformation of an input state to an output state
	Logical operator ‘exclusive OR’	Logical linking operators describe the logical linking of events and functions
	Logical operator ‘OR’	
	Logical operator ‘AND’	

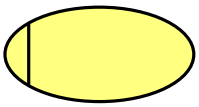
	Organizational unit	Description of the structure of a company
	Information object	Representation of information objects
	Control flow	Course of events and functions
	Data flow	Description of a function if it is read, written, or changed
	Assignment	Assignment of resources/organizational units
	Process guidepost	Process refinement

Table 15: Basic Elements of eEPCs (Gadatsch, 2005:83, translated by KG)

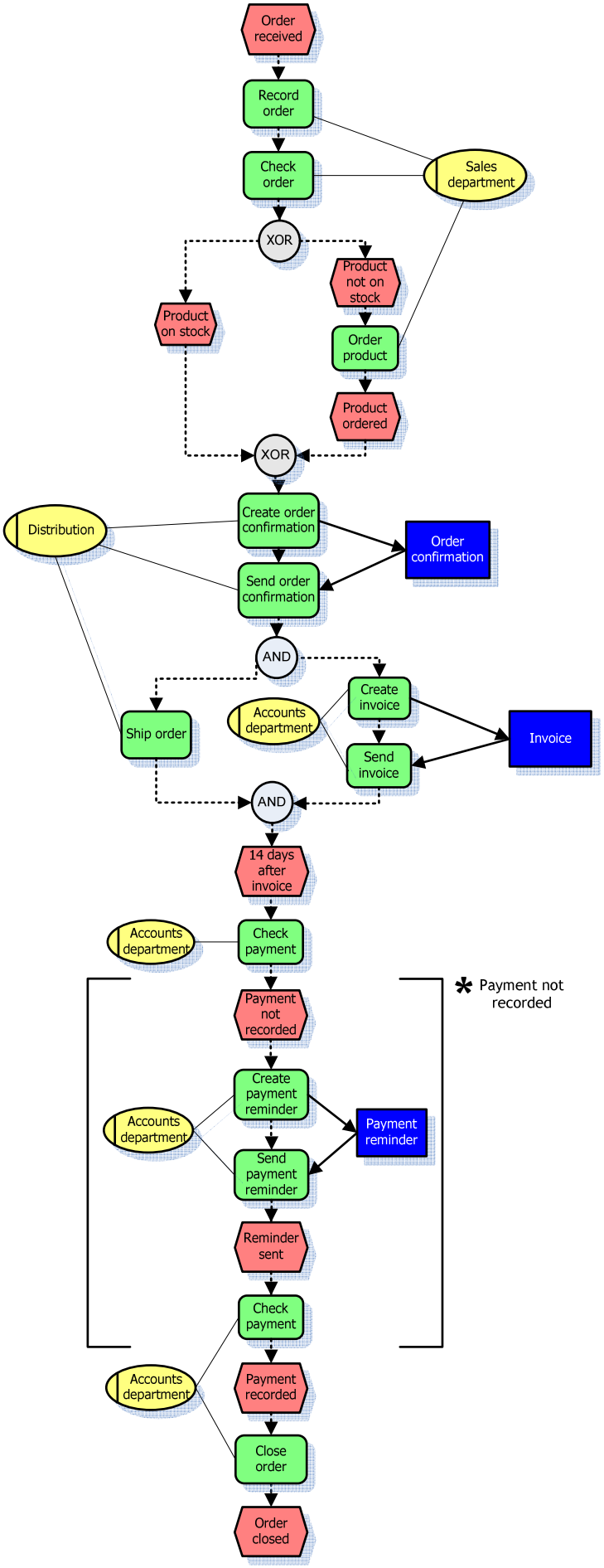


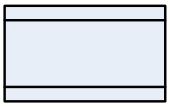
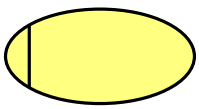
Figure 51: Demonstration of an extended event driven process chain (eEPC) – Improved notation

3.16. Object oriented event driven process chain (oEPC)

In 1997, Scheer developed an object oriented EPC notation for reaching two goals:
Application of the event driven concept of object interaction in object-oriented, component-based information systems, and
Integrated description of processes and their objects.

The object-oriented paradigm requires a modification of the term "business process", which describes the event driven processing of business objects for goods and services. Business objects capsulate functions (methods) and data (instance variables), which are required for creating benefits.
Events are not starting functions, like we learned in the definition of the classical EPC (cp. Section 3.12). In the object-oriented approach of EPCs, events start a call for methods of the instantiated objects. The notation is only based on the classical EPC notation! New symbols for business objects (business class), data (instance variables) and functions (methods) are introduced.
(cp. Gadatsch, 2005:97, translated by KG)

3.16.1. Notation of oEPCs

Symbol	Naming	Meaning
	Object class	Business object which encapsulates relevant functions (methods) and data (instance variables) for processing
	Event/message	Description of a occurred state from which the rest of the process course is dependent
	Method/ function	Function (method) of an object for manipulating data (instance variable). Private methods are not visible outside of the object.
	Instance var. /Attribute	Data (Instance variable) which are manipulated by the methods of an object
	Organizational unit	Description of the structure of a company

XOR ∧ ∨	Connector	Logical linking operator for the linking of business objects and events.
	Control flow	Temporal-logical dependency of events and business objects
	Service relation	Event driven messages between business objects
	Edge	Assignment of methods, instance variables and organizational units to objects

Table 17: Notation of oEPC (Gadatsch, 2005:98, translated by KG)

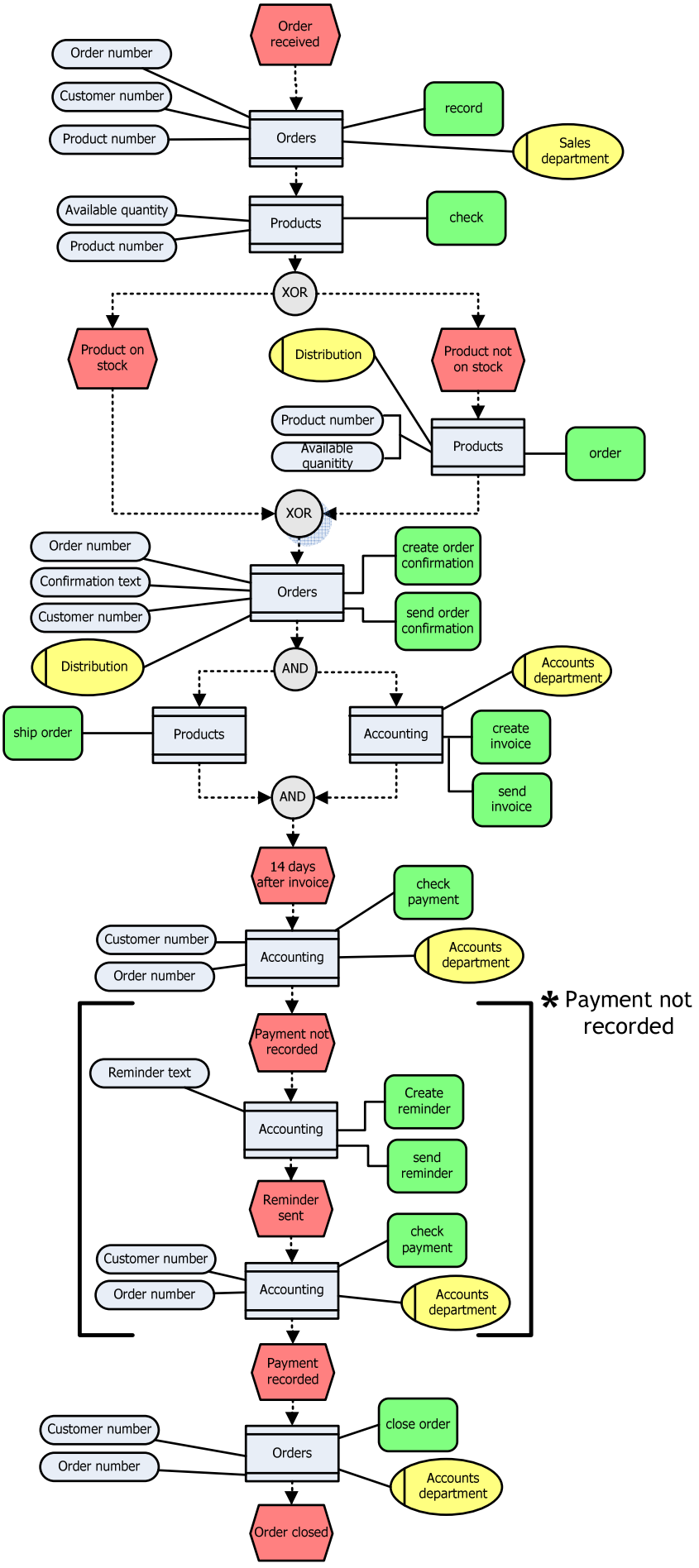


Figure 57: Demonstration of an object oriented EPC – Improved notation