

Seibt: Wirtschaftsinformatik, #99

Zusammenfassung des Lernstoffs

18.10.2005

Die Vorlesung folgt dem Script Dr. Seibt's; die Infos hier dienen nur der Ergänzung.

Seitenzahlen entsprechen dem gedruckten Script, nicht den aktuell gezeigten PP-Seiten.

1. Autoproduktion

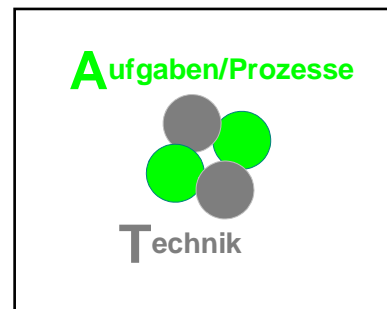
- **BAS** = Betriebliches Anwendungs-System
BIS = Betriebliches Informations-System
- Es gibt immer **weniger eigene DV-Abteilungen**, da **HR** die **größten Kosten** in der IT verursachen und der ständigen Schulung und Weiterbildung bedürfen.
 - ⇒ daher auch: **IDV** (individuelle Datenverarbeitung) **wächst**
 - ⇒ das führt zu **Sicherheitsproblemen**, z.B. durch die Übernahme von Fa.-Daten, und u.U. Ressourcenproblemen wg. unsicherer Verfügbarkeit (MA schreibt eigene Routine und steht – z.B. wg. Krankheit oder Arbeitsplatzwechsel – nicht zur Verfügung)
- **Auto-Film**
 - ⇒ **Computer** und **Roboter** in der Automobilindustrie bestehen als **Leading Computers, Subsysteme** und **Netze**; sie **steuern** und **kontrollieren**
 - ⇒ **Automatisierung** von **Produktionsprozessen** sowie von **Informations-** und **Kommunikationsprozessen**
 - ⇒ **Anteil** der automatisierten Teilprozesse bestimmt **Automatisierungsgrad**
 - ⇒ Es werden trotz übermächtig scheinender Automatisierung viele **Teilprozesse** von Menschen erledigt – zumindest immer dann, wenn Fingerspitzengefühl gefragt ist („Autofenster-Schlag“)
- Die **WI** ist wesentlich am **Handling** der **Komplexität** von **Produktionsprozessen** beteiligt

2. Informatik und Wirtschaftsinformatik

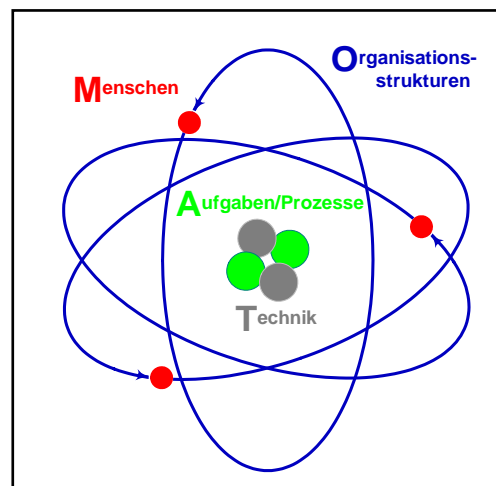
- **Informatik** und **Wirtschaftsinformatik** sind zwei verschiedene **Wissenschaftsdisziplinen**
- Vor- und Nachteile der **Organisationsform Gruppenarbeit** (10 MA + 1 GL)?

- Der Kern der **Informatik** ist die **Auseinandersetzung mit Algorithmen**; in der Praxis gibt es aber auch oft „**Fuzzy-Problems**“ („schmutzige Probleme“, eher: „unscharfe Probleme“), die **nicht durch Algorithmen** beschrieben werden können.
⇒ trotz GOB (Grundlagen ordnungsgemäßer Buchhaltung) sind **nicht alle Buchhaltungsprobleme** klar mathematisch **definierbar** (z.B. wg. Bewertungsproblemen), oftmals sog. „political aspects“
- **Informatik** („Computer Science“) = **„Beschäftigung mit Computern“**
Effizientes lösen von Problemen mit Computern
 - **Theoretische** Informatik (formale Sprachen, ...)
 - **Praktische** Informatik (Programmiertechnologie, ...)
 - **Technische** Informatik (Rechnerorganisation, ...)
 - **Angewandte** Informatik (Medizinische Informatik, ...)
- **Wirtschaftsinformatik** = von manchen als Teilgebiet der Informatik, Bereich angewandte Informatik, betrachtet, ist aber: **Lehre von den 1. Informations- und Kommunikationssystemen** (=„Informationssysteme“), den Mensch-Maschine-Systemen und **2. den IT-Anwendungssystemen**
Analyse der Aspekte:
 - **technische,**
 - **organisatorische,**
 - **wirtschaftliche** und
 - **soziale**
 Dimension
- **Wichtigster Parameter** von Mensch-Maschine-Systemen ist die **Effizienz** (zwecks Gewinnmaximierung)

- Ein **IT-Anwendungssystem** („IT-AS“) definiert sich über
 - die **Aufgaben** und **Prozesse** der **Organisation**, für die es konzipiert wird und
 - die eingesetzte **Informationstechnik** (IT).



- Da sich die das IT-AS definierenden **Aufgaben** in der Praxis ständig **ändern**, bedarf es eines „**Continuous Improvement**“ (Wartung/ Pflege eines IT-AS aufgrund Änderungen des Umfeldes)
- Durch Hinzufügung der Aspekte „**Bestimmte Organisationsstrukturen**“ sowie „**Beteiligte Menschen**“ erstarkt das IT-AS zum **Informationssystem** („IS“). Das macht auch die stark **praxeologische¹ Orientierung** der WI deutlich, konkrete Problemlösungen anzubieten.



¹ Praxeologie = Wissenschaft vom (rationalen) Handeln, Entscheidungslogik [Duden]


- **Wissensmanagement** in der Auto-Industrie ist eine **Verwaltung** von **Know-How**, insb. auch im Rahmen eines kontinuierlichen Verbesserungsprozeß/ der Aus- und Weiterbildung.
- Die praktische Erfahrung mit dem **Wissensmanagement** war in den letzten ca. 10 J. **enttäuschend**
 - ⇒ Ausnahme: Toyota
 - ⇒ der Erfolg der Einführung von Wissensmanagement hängt wesentlich mit der Eigenschaft der beteiligten Personen zusammen, neue Wege zu gehen
- bspw. eGovernment
 - ⇒ meint: IT in der Verwaltung
- In der WI werden **Begriffe** der **Informatik**, der **Wirtschaftswissenschaften** sowie **eigene Terminologie** eingesetzt
- Was ist der Unterschied zwischen scheinbar eindeutigen Bezeichnungen wie „Daten“ vs. „Informationen“?
 - ⇒ wenn man sich mit solchen Gedankenspielen vertraut macht und entsprechend philosophische Aspekte in einer Klausur (aber dann immer mit Herkunftsnachweis!) verwendet, gibt es zusätzliche Punkte...
- **Definitionen** (hierarchisch sortiert, aufeinander aufbauend)
 - Information** = **zweckorientiertes** Wissen
 - ↑ **Wissen** = **Kenntnisse** über **Tatsachen** und Zusammenhänge
 - ⇒ **implizites** Wissen (**subjektives** W, unbewußt, auf Erfahrung beruhend)
 - auch: Spezialwissen (mannschaftsspezifisches W im Verein)
 - ⇒ **Simultan-Wissen** für ad-hoc-Entscheidungen
 - ⇒ **Explizites** Wissen (**objektives** W, niedergeschrieben, codifiziert)
 - ⇒ **Sequentielles Wissen** für Planerisches und Analytisches
 - ↑ **Daten** = Gebilde aus Zeichen oder kontinuierlichen² Funktionen (=analoge Daten)
 - ↑ **Zeichen** = Bedeutungsträger (z.B. Laut oder Buchstabe)
- Wg. des Schwerpunkts auf explizitem Wissen in der abendländischen Kultur und Wissenschaft haben dort Bücher einen so hohen Stellenwert.
- Die **Semiotik**³ beschäftigt sich mit Problemlösungen bei der Kommunikation zwischen menschlichen Kommunikations-Partnern und gliedert sich in
 - **Syntax**
 - **Semantik**
 - **Pragmatik**
- Die **Art der Kommunikation** bestimmt wesentlich den **Stil** der **Koordination** und **Kooperation**.
- „**Informationstechnik-Systeme** bestehen aus Computern und Netzwerken.“
- Beachte: **Informationstechnologie** (=Lehre von der Informationstechnik)
 - vs.
 - Informationstechnik** (=Technik in der Informatik)
- **Anwendungssystem** („Anwendung“, „Application“)
 - **im engeren Sinne** ist dies **Software** und die **Daten**
 - **im weiteren Sinne** ist dies Software und die Daten sowie **Hardware** (und notwendige **zus. Technik**)
 - ⇒ ein **BAS** ist dann ein **AS** für die **Belange** eines **bestimmten Betriebes**
 - ⇒ für die Buchhaltung, Personalwesen u.a.

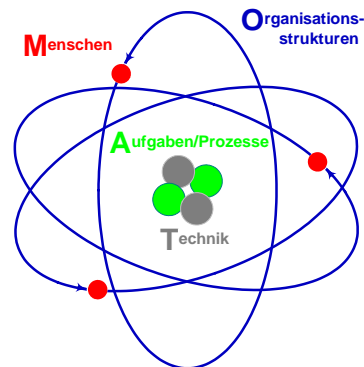
² Eine Funktion ist kontinuierlich, wenn zwischen zwei Punkten ihres Grafen beliebig kleinen Abstands beliebig viele Bildpunkte platziert werden können.

³ Semiotik = 1. Semiologie (Lehre von den Zeichen, Zeichentheorie), 2. Wissenschaft vom Ausdruck, Bedeutungslehre

- **Software** läßt sich klassifizieren:
 - **Anwendungssoftware** (Gesamtheit der Programme zur Abwicklung bestimmter DV-Aufgaben)
 - **Systemsoftware** (Betriebssystem, Bridge-Programme⁴, ...)
 - **Werkzeugsoftware** (Compiler, Softwareengineering, CASE-Systeme⁵, ...)
- Unterscheide **Anwender** als **organisatorische Einheit**, die AS einsetzt
vs.
Benutzer als **Person**, die AS einsetzt

3. BAS/ BIS

-  Ein **BAS** ist ein **spezifisches IT-Anwendungssystem** zum Einsatz für bestimmte(n) Betrieb(en) – dessen Komponenten sind **Aufgaben/Prozesse** und **Technik** (sowie den betrieblichen Daten) bestehen
- Ein **BIS** ist ein **spezifisches Informationssystem**, das für die Belange eines bestimmten Betriebs oder -typs geschaffen wurde; es besteht aus einem oder mehreren BAS und kann nicht gekauft werden. Sein **Aufbau** folgt dem **ATOM-Modell**. In einem Betrieb können **mehrere BIS parallel** existieren.



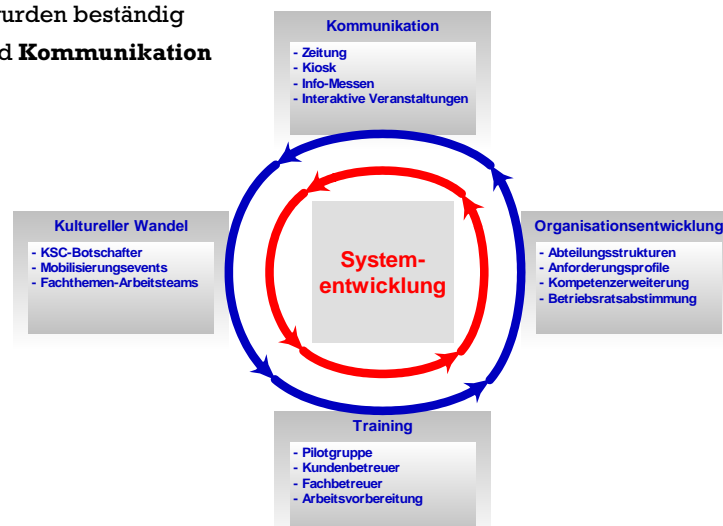
4. Fallstudie Hamburg-Mannheimer

- Die **Änderung** von **Kundenschnittstellen** zieht meistens viele Änderungsnotwendigkeiten bei anderen Prozessen nach sich
- Bei der **Optimierung** des **BIS** der **Hamburg-Mannheimer** war es wichtig, **Dateninseln** (Schadensachbearbeiter – Vermittler – Vertragsbearbeiter) zu **vermeiden** oder zu beseitigen, indem eine **gemeinsame Datenbank** genutzt wird.

⁴ Bridge-Programme setzen bestehende Anwendungen im Falle eines Betriebssystem-Wechsels um

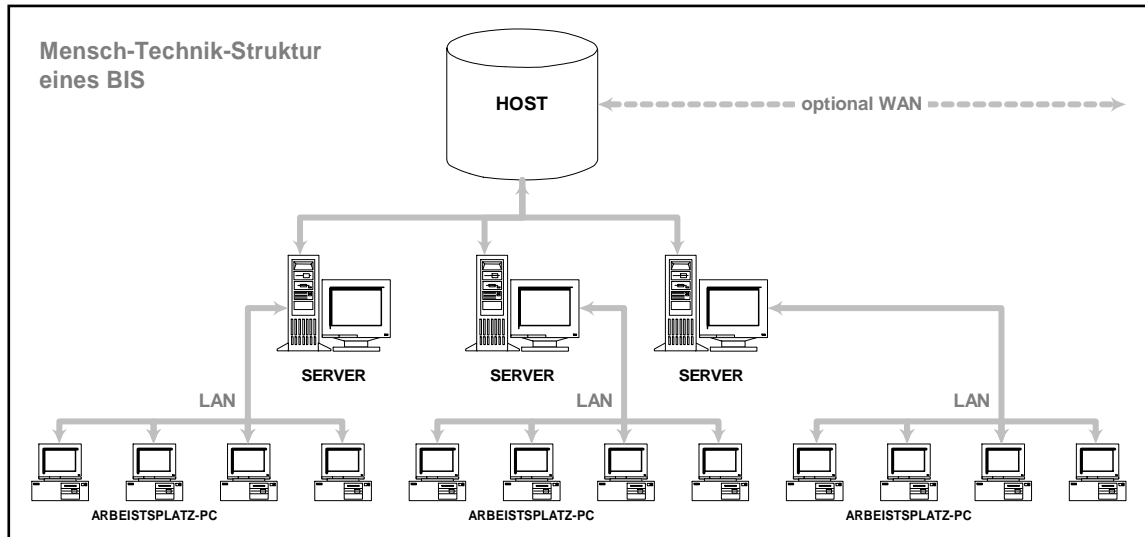
⁵ CASE („Computer-Aided Software Engineering“) steht für die Computer-seitige Unterstützung im gesamten Prozess der Software-Entwicklung. CASE-Tools sind Programme, die den Software-Ingenieur bei der Planung, dem Entwurf, der Implementierung und der Dokumentation unterstützen. Ein wichtiger Bestandteil der meisten modernen CASE-Tools ist die Metasprache UML, die zur Visualisierung der Artefakte eines Software-Systems dient. Teilweise sind sie in moderne IDEs (Integrated Development Environment) integriert, teilweise sind es eigenständige Applikationen, deren Fokus vollständig auf CASE liegt (ohne dabei die anderen typischen Elemente einer Entwicklungsumgebung anzubieten). Klassische CASE-Tools unterstützen anstelle der objektorientierten UML die sogenannten strukturierten Techniken Strukturierte Analyse und Strukturiertes Design (SA/SD), sowie Entity-Relationship-Modellierung (ERM).

- **Schwächen bei der HM vor dem Reengineering**
 - **Spartenorganisation** erschwert Kundenbetreuung
 - **Fehlende Erfassung** von **Kundenwünschen/-beschwerden**
 - **Nicht** in den Kundenservice **integrierte Zugänge**
 - **Fehlendes Logging** mündlicher Kontakte
 - **Service kennt Kundendaten nicht**
 - **Kein Fallabschluß** beim ersten Kundenkontakt
 - Durchschnittliche **Erledigungsdauer zu hoch**
- Durch **Logfile-Analysen, Unternehmensberater** und **Kundenbefragungen** („Schwachstellenanalyse“) wurden 40-50 **Kennziffern** festgestellt, die die Mängel **operational** dokumentierten
- **Nach dem Reengineering** war realisiert
 - „**One face to the customer**“ (i.S. fehlender Delegation zu den Sparten)
 - KSC (Kundenservicecenter) ist **Koordinator** zwischen den Sparten
 - KSC ist **zentrale Schnittstelle**
 - KSC hat **ggf. Lotsenfunktion** zur fachlichen oder vertrieblichen Erledigung
 - Die **Mitarbeiter** des **KSC** erlangen durch Trainings **Sozialkompetenz** (Teamgeist, Einfühlungsvermögen, ...), **Fachkompetenz** (Kaufmännische Ausbildung, Computerkenntnisse, ...) und **persönliche Kompetenz** (Belastbarkeit, Eigenmotivation, ...)
 - Flachere **Hierarchie** von
Abteilungsleiter → Büroleiter → Gruppenleiter → **Team**
zu
Abteilungsleiter → **Teamleiter** und **Team**
 - Statt 3270-Terminals gibt es **PC-Arbeitsplätze** (Arbeitsplatz-individuell konfiguriert)
 - Die Entwicklung arbeitet für das **Frontend** mit **MFC C++** und setzt auf **Middleware** in **Cobol** auf, diese arbeitet in Singlerequests auf den Datenbanken
 - Der **KSC-Mitarbeiter identifiziert** den Kunden, **berät** und **erkennt** den **Handlungsbedarf**, **erledigt** die **Anfrage** und arbeitet damit **alleine** schon **fallabschließend**
 - Schriftliche **Vorgangszeiten** wurden durch den **Scanworkflow** von **3 auf 2 Tage reduziert**
- **Middleware** setzt **hardware-individuelle Datenströme** in genormte Anfragen- und Ergebnisströme eines gemeinsamen Datenpools **um** – sei es wg. neuer Frontends oder einem Wechsel des Betriebssystems
- Das **gesamte Projekt** (vom Strategieauftrag bis zum Abschluß Hauptprojekt) dauerte von **1995 ... 2001**. Während der **Weiterentwicklung** und Implementierung von 1998 bis 1999 **stieg** die **Kunden-zufriedenheit** in allen Bereichen um ca. **0,2 Pkte.** (Skala 1-5), wobei die durchschnittliche Abweichung von Versicherung zur Versicherung bei 0,16 Pkte. (bei 20 untersuchten Versicherungen) lag.
- Während des Projekts wurden beständig **Systementwicklung** und **Kommunikation** **parallel** betrieben



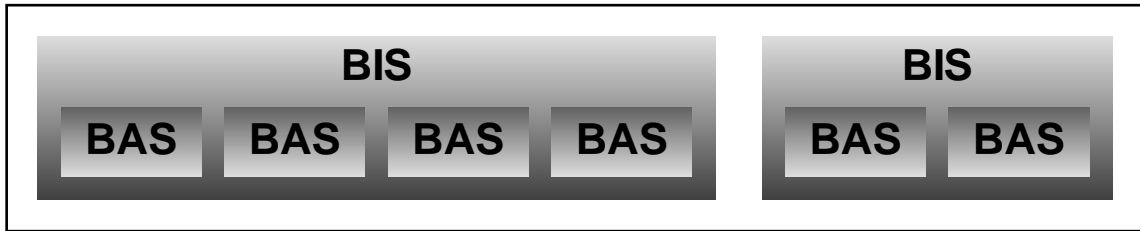
5. Beispiele für BAS/ BIS

- Je nach Aufgabengebiet **unterscheidet** sich die **technische Realisation** der in einem Unternehmen eingesetzten **BAS**, das darauf aufsetzende **BIS** hat aber **grundsätzlich** eine **vergleichbare Struktur**:

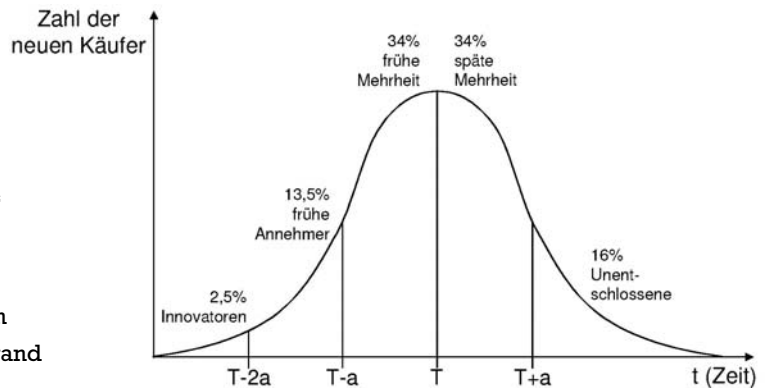


- **BAS** lassen sich nach MERTENS **klassifizieren**:
 - **Administrationssysteme** zur **Rationalisierung der Massendatenverarbeitung**
 - Abrechnungssysteme
 - Rechnungsschreibungssysteme
 - **Dispositionssysteme** zur **Vorbereitung/ Ersetzung menschlicher Entscheidungen** (auf häufige, **regelmäßige Entscheidungen** gerichtet)
 - Bestelloptimierungssysteme
 - Materialdispositionssysteme
 - **Planungssysteme** zur **Erweiterung von Dispositionssystemen um Planungsfaktoren** (auf seltene, **unregelmäßige Entscheidungen** gerichtet)
 - Investitionsplanungssysteme
 - Produktionsprogrammplanungssysteme
 - **Kontrollsysteme** zur **Kontrolle der Einhaltung von Plänen** und Einleitung von Maßnahmen (als **Pendant zum Planungssystem**)
 - Finanzkontrollsysteme
 - Produktionskontrollsysteme
- **Anwendungssysteme** sind zu **unterscheiden** nach
 - **Administrationssystemen** als branchenspez. oder -unspez. Anwendungen
 - **Führungssystemen** als Informations- oder Planungssysteme
 - **Querschnittssystemen** als Büro-, Multimedia- oder Wissens-Systeme„Querschnittssysteme“ werden in dieser Vorl. nicht behandelt, trotzdem sind sie wichtig; hier intensiver behandelt: Admin.- und Führungssysteme

- In jedem Unternehmen gibt es **zeitgleich mehrere BIS'** mit **mindestens gleich großer Zahl** von **BAS'**.



- Jedes **BAS** hat einen dem **Produktlebenszyklus** vergleichbaren **Lebenszyklus**;
Lebensdauer von BAS/BIS-Systemen ist technisch und organisatorisch bedingt
 - „don't change a running system“
 - evtl. Schnittstellen nötig für alte und neue Systeme
 - Ende der Lebenszeit kündigt sich z.B. durch stark steigenden Aufwand für Wartung und Pflege an



- **electronic commerce**
 - INet-basierter Vertrieb
 - EDI (electronic data exchange)
 - Anbieter und Käufer bedienen sich des INets (=„elektronische Märkte“)
- **Supply-Chain-Management**
 - Hersteller – Distributor – Logistiker – Einzelhandel – Endkunden
 - JIT = Just In Time
 - Autobahn wird zum größten Lager

6. IDV und ODV

- Zur Erstellung eines **PIS** (personal information system) reicht schon die **Konfiguration** eines PC-Systems – das PIS muß nicht zwingend programmiert werden
 - Ggs.: „Customizing“ (=für alle Anwender des Betriebs)
- Das **PIS** kann als **individuelle BAS/BIS-Variante** betrachtet werden
 - Unterschied zum „echten“ BAS/BIS ist die **Unverbindlichkeit**, da die Prozesse nur für den Ersteller gelten und **größere Spielräume** möglich sind
 - generell **viel Spielraum** für ein PIS bei **Planungs- und Controlling-Systemen**, **kein Spielraum** bei **produktionsunterstützenden Systemen**
- „Benutzer-Service“ zur Unterstützung der **PIS's** / der **IDV** war **getrennt** von der **zentralen DV** angesiedelt
- **Keine IDV** („Personal Computing“) bei z.B. **Personaldatenverwaltung**, da es eine Vielzahl von **Einschränkungen/ gesetzlichen Vorgaben** gibt

- Volumen der **IDV** muß **Waage halten** – übermäßige Nutzung der individuellen Möglichkeiten kann zu **Wildwuchs** und der **Verschwendung** von **Arbeitszeit** führen
- Im Aufkeimen der IDV (vor 10-15 Jahren) gab es bei einige großen Fa. **Belohnungen** für die „**beste IDV**“ (RWE, Daimler, Siemens)
- **Vorteile** bei der **IDV** können sein:
 - **Endbenutzer** bekommt exakt **gewünschtes System**, **Erhöhung** der **Zufriedenheit**
 - **Verringerung** der **Abhängigkeit** von **zentraler DV**
 - **Beschleunigung** der **System-Entwicklung**
 - **Verringerung** des **Kommunikationsaufwands** während der Entwicklung
 - **Erhöhung** der **Entwicklungskapazitäten**
- **Probleme** bei der **IDV** sind:
 - **Zeitabhängigkeit** der **Entwicklung** und der **Aufgaben**
 - **Integrations-/Koordinations-Probleme**
 - Wachstum in **Bereiche ohne Gestaltungsspielraum**
 - Probleme bei **potentieller Umwandlung** einer **IDV**- in eine **ODV**-Lösung
- **Denzentrale Anwendungsentwicklung** (=ODV, organisatorisch verbindliche DV) wird in den **Fachabteilungen** betrieben, das Ergebnis sind **fachabteilungsspezifische BAS/ BIS**; die **ODV-Entwicklung** wird von den **Mitarbeitern** der Abteilung durchgeführt
- **ODV** (organisatorisch verbindliche DV) muß zumindest (organisatorisch) **abteilungsweit verbindlich** sein, damit sie Sinn macht
→ ODV als Ggs. zur IDV
- Damit die **Fachabteilungen** wirklich **autark** werden, wird auch **eigenes Know-How** entwickelt – u.U. in denselben Weiterbildungskursen, in denen die Kollegen aus der IT-Abteilung sitzen
- **Vorteile** der **ODV** können sein:
 - Gutes **fachliches Know-How** bei der Aufgabenstellung
 - **Unabhängigkeit** von **zentraler DV** bei der **Wartung** des Systems
 - **Zeitlicher** Vorteil
 - **Raschere Nutzung** des Systems und damit **Kostenreduktion**
 - **Höhere Motivation** der Fachmitarbeiter
- **Probleme** bei der **ODV** sind:
 - Aufbau **eigener Entwickler-Kapazitäten**
 - **Budget- und Zeitbelastung** durch Weiterbildung
 - **Folgeaufwand** für Wartung
 - Kommunikationsprobleme zwischen den Fachabteilungen („**Rad wieder neu erfinden**“)
 - **Verlust** von **Synergieeffekten**, die eine zentrale DV hätte
 - **Inkompatibilitäten** zu anderen Systemen im Betrieb
- **ODV-Entwicklung** in zentraler Anwendungsentwicklung **dann**, wenn Fachabteilung **nicht genügend Budget** hat

7. SAS

- Gründe für **IS** (Individuelle Anwendungssoftware) statt **SAS** (Standard-Anwendungssoftware):
 - **Zeitprobleme** bei Eigenentwicklung
 - **spezifische Anforderungen**
- **Vorteile von SAS:**
 - **hohe Verfügbarkeit**
 - **kostengünstig**
→ Kostenvorteile werden u.U. durch notwendiges Customizing aufgeessen
 - **Erfahrungen des Herstellers** nutzen
 - **Positiver Einfluß** auf **Workflow**, wenn sich die MA den Software-Vorgaben anpassen
- **Nachteile von SAS:**
 - **keine Deckung** der **Aufgaben** im Betrieb mit den **Lösungen** der Software
 - kostenintensives **Customizing**
 - psychologische Probleme in den Fachabteilungen, **Akzeptanzprobleme**
 - **Kompatibilitätsprobleme**
 - **Abhängigkeit** vom Hersteller
- „**Schrankware**“ = Gekaufte Software, deren Funktionalität sich als unpassend erwies und die unbenutzt bleibt
- **Deutsche** Unternehmen legen Wert darauf, daß **individuelle Betriebsspezifika** in der Software abbildbar sind, **amerikanische U.** **akzeptieren** ein größeres Maß an **Standardisierung**
- **Customizing** kann durchgeführt werden durch
 - **Parametrisierung:** Setzen von Parametern ändert Programmabläufe und -funktionalität
 - **Konfigurierung:** bei modular aufgebautem System werden nur benötigte Module installiert
 - **Individualprogrammierung:** Entwicklung betriebsindividueller Ergänzungen zur SAS

8. **BAS-/BIS-Gestaltung**

- Prozeß der **Entwicklung** eines **BAS** in **5 Stufen:**
 1. **Anforderungen** ans zukünftige System **festlegen** („**Requirements**“) nach einem Business-Modell (z.B. ARIS, vgl. Script v. 5.10.)
 2. **Analyse** (der Anforderung) und **Design** (der Lösung) mit Definition DB, Screen, Aufgabenabwicklung („**Functions**“), Gesamtdesign („**Architektur**“)
 3. Programm(e) **herstellen**
 4. Module anhand ...
 - Logik (der Funktionen)
 - Screens
 - DB I/O
 - etc.
 ... **testen** und **gesamtintegrieren**
 5. Übernahme in **Realumgebung** und mit **echten Daten Produktion** und **Nutzung**
...
Pflege, Wartung, Weiterentwicklung
→ führt dann evtl. wieder zu **Pkt. 1.**

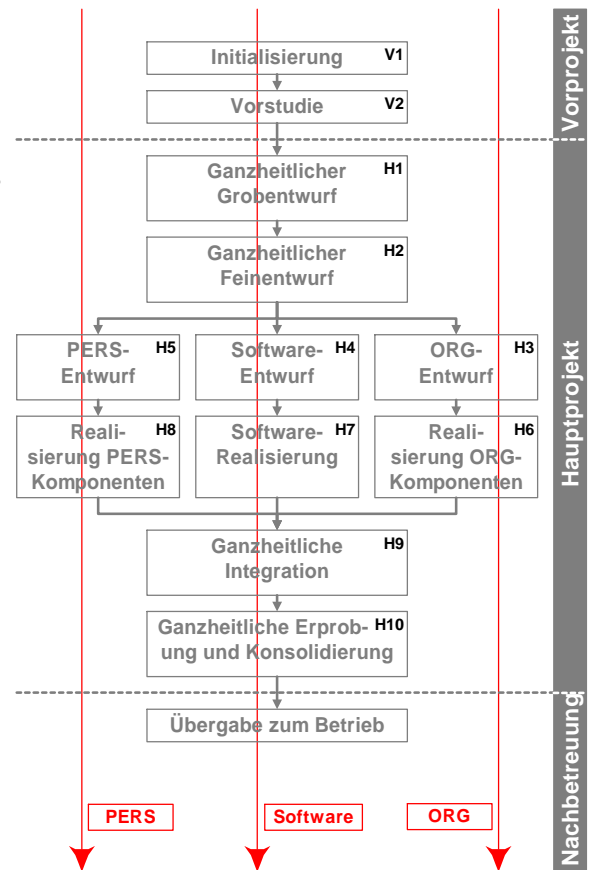
- Dem **ATOM**-Modell folgend werden bei der Programmentwicklung (auch so, wie sie in vielen Lehrbüchern dargestellt wird) die **AUFGABEN/ PROZESSE** sehr wohl **berücksichtigt** – es müssen aber dazu auch die **ORGANISATION** und die **MENSCHEN** Berücksichtigung finden.
- **Phase 1** auch: „Blaupause“, „**Pflichtenheft**“ (zumindest bei der BAS-Entwicklung)
- Strenge **Phasenhaltigkeit** geht **nicht**; die mehr praxisgerechte „**Meilensteinlogik**“ hilft
⇒ jedes **Vorgehensmodell** („VM“) muß **projektindividuell** erstellt werden
- Problem beim **Wissensmanagement**: viele behalten das **Know-How** nur **im Kopf**, weil das Dokumentieren „nur Zeit kostet“
- **Konsolidierung** muß die **Menschen/ die Organisation** einbeziehen
- Am **Phasenkonzept** (bzw. herkömmlichen Vorgehensmodellen) ist **kritisch**:
 - **Konzentration** auf **technische Systemkomponenten**
 - **Vernachlässigung** der **Implementierungsproblematik**
 - **Vernachlässigung** der **Erprobung/ Konsolidierung**
 - **Vernachlässigung** des **Systemlebenszyklus**

- Die Grafik stellt **kein Phasenmodell** dar
⇒ bei H3 – H8 **Parallelität**
- „**Initialisierung**“ = (Hinweis auf) **Notwendigkeit, Änderungen** vorzunehmen
- „**Vorstudie**“ = **Nutzen- und Kostenfeststellung** als Grundlagen für eine Go/No-Go-Entscheidung
- „**Ganzheitlicher Grobentwurf**“ = u.a. **Prüfung von Details** und **Feststellung der Realisierbarkeit**

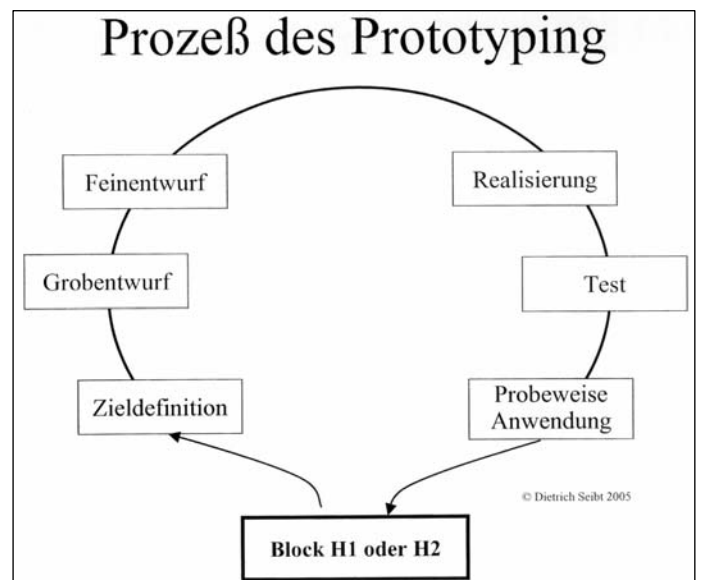
Nach der Vorstudie bilden

der	ORG-	Entwurf	einen „ Feinstentwurf “
	Software-		
	PERS-		

- Häufig **fehlen** in der Praxis **ganzheitliche Integration** und **ganzheitliche Erprobung/ Konsolidierung**
- Der **Grobentwurf** dient auch dazu, „**Sichtweisen zu vereinheitlichen**“ – erst durch eine **Normierung der Notation** wird ein Feinentwurf möglich
- Die **roten Linien** der Grafik bezeichnen die **drei Prozesse** der **Software-**, der **ORG-** sowie der **PERS-Entwicklung**, mit denen man sich **vom Anfang bis zum Ende beschäftigen** muß
- „**IKT/IS**“ = Informations- und Kommunikationstechnik/ Informationssysteme
- Der **Projekterfolg** hängt wesentlich auch davon ab, ob es gelingt, **Kommunikationsprobleme** zwischen verschiedenen Fachleuten **auszuräumen**



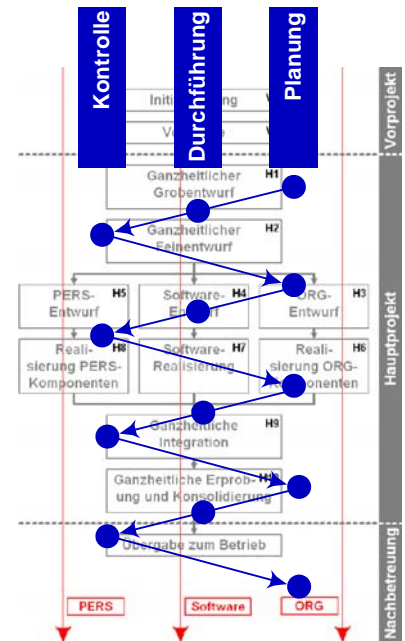
- Auf die **Veränderungen/ Ziele** hin ist im **Projektverlauf kontinuierlich zu kontrollieren** (z.B. nach jedem Block bzw. Meilenstein)
 - **Achtung: Vor Beginn des Hauptprojekts** müssen die **Gesamtziele** sowie die **Teilziele** mit deren Beitrag zur Gesamtziel-Erreichung **definiert** werden (und zwar **operational**); nötigenfalls ist auch eine **Methode zur Überprüfung der Zielerreichung** festzulegen
- **Beim Vorprojekt** gibt es noch **keine Projektleitung**, aber ab dem Hauptprojekt wird ein Projektleiter festgelegt – dies ist oftmals der Initiator des Projekts
- **Frühzeitige Übergabe** der neu entwickelten Software an die Kunden **spart Zeit** und Geld (für die Entwicklung bzw. die QS)
 - „Bananensoftware“
 - **entspricht aber nicht dem Win-Win-Prinzip** und spricht nicht für Firmenkultur
 Stattdessen: Auslieferung ausschließlich an β -Kunden, die wissen, daß sie mit unfertiger Software arbeiten, aber für ihre Mitarbeit (Testing, ...) Entgegenkommen erwarten dürfen
- Die **Bereiche H3 –H8** sind ggf. um die **Einbettung von SAS zu erweitern**
- **Zweck des Prototypings** ist **Veranschaulichung**; es dient als **Akzeptanz-Test**, der **Visualisierung** technischer Vorgaben, der **Transparenz der Aufgaben** und der **Verständigungsoptimierung** zwischen Technikern und Nutzern
 - **Wichtig**, da die **Bereinigung von Fehlern** und Mißverständnissen im **Laufe eines Projekts immer teurer** werden (!!)
- Die **Lenkungsorgane** eines Projekts sind der **Projektleiter** und die **Projektkommission** (incl. „Bauherren“ und Management des Auftragnehmers)
- Falls ein **Projektabbruch** sinnvoll erscheint, muß die Möglichkeit zum Zeitpunkt der Erkenntnis (i.d.R. am Meilenstein) auch genutzt werden – Argumente wie „nun haben wir schon soviel investiert“ u.ä sind irrelevant
- **Meilensteinlogik ergänzt und verfeinert** die Phasen-/Blocklogik und dient der **Kontrolle des Projekts** sowie der Herbeiführung eines **Konsens' zwischen allen Beteiligten** (Teilabnahme!)
- An einem **Meilenstein** kann passieren:
 - **Freigabe** des SG-(Systemgestaltungs-)Blocks
 - **Ergänzung folgender SG-Blöcke**
 - **Ergänzung vorheriger SG-Blöcke, Nachbesserung** mit Rücksprung zum vorherigen Meilenstein
 - **Abbruch** des Projekts



9. Querschnittsaktivitäten in der BIS-Entwicklung

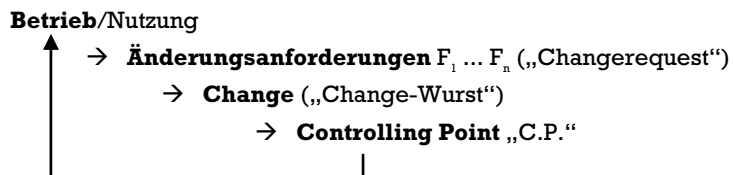
- **Lenkungsaktivitäten** sind **permanent** notwendig und bei individueller Notwendigkeit durchzuführen (sind aber nicht immer zwingend Aufgabe des Lenkungsausschuß')

- Im Rahmen der **System-Implementierung** wird das neue BAS/ BIS in sein **Ziel-Umsystem** (Umgebung für Subsystem) **eingebettet**; dabei müssen Rückkopplungsprozesse berücksichtigt werden
- Bereich **PERS** spiegelt die menschlichen Aspekte wider
- Bei der **Projektplanung** wird **geplant, durchgeführt** und **kontrolliert**; diese Schritte **wiederholen** sich **ständig** durch das gesamte Projekt hinweg



10. Systemlebenszyklus und Systemlebensdauer

- Die **steuerliche Lebensdauer** ist 5 J für ein BAS/BIS; in der Praxis beträgt sie aber auch 20–25 J.
- Der **System-Lebenszyklus** ist die **zyklische Abfolge** von **Betriebs-** und **Wartungsabschnitten**; er wiederholt sich während der System-Lebensdauer mehrfach:



- **Achtung:** Die **Systemlebensdauer beginnt** nicht mit der Indienststellung, sondern **mit dem Beginn der Entwicklung**, also beim **Start des Hauptprojekts**