JUSTUS-LIEBIG-UNIVERSITÄT GIESSEN
PROFESSUR BWL – WIRTSCHAFTSINFORMATIK
UNIV.-PROF. DR. AXEL SCHWICKERT

Schwickert, Axel; Dörr, Lea; Schramm, Laura; Schick, Lukas; Hilmer, Sabrina

Systems Engineering – Reader zur WBT-Serie

ARBEITSPAPIERE WIRTSCHAFTSINFORMATIK

Nr. 9 / 2020
ISSN 1613-6667

Arbeitspapiere WI Nr. 9 / 2020

Autoren: Schwickert, Axel; Dörr, Lea; Schramm, Laura; Schick, Lukas; Hilmer, Sabrina

Titel: Systems Engineering – Reader zur WBT-Serie

Zitation: Schwickert, Axel; Dörr, Lea; Schramm, Laura; Schick, Lukas; Hilmer, Sabrina: Systems Engineering – Reader zur WBT-Serie, in: Arbeitspapiere WI, Nr. 9 / 2020, Hrsg.: Professur BWL – Wirtschaftsinformatik, Justus-Liebig-Universität Gießen 2020, 235 Seiten, ISSN 1613-6667.

Kurzfassung: Das vorliegende Arbeitspapier dient als Reader zur WBT-Serie „Systems Engineering“, die im E-Campus Wirtschaftsinformatik online zur Verfügung steht.

Software-Systeme sind nicht mehr aus dem Berufs- oder Privatleben wegzudenken. Die vorliegende WBT-Serie schafft durch die Erläuterung der Grundbegriffe des Software Engineerings sowie der Projektarbeit ein Verständnis für die Entwicklung von IT-Systemen. Der Prozess der Software-Entwicklung wird durch die allgemeinen, sequentiellen, evolutionären und agilen Vorgehensmodelle verdeutlicht. Die zur technischen Entwicklung benötigten Voraussetzungen werden durch die Systemtheorie, die Graphentheorie und verschiedene Modelldarstellungen, wie z. B. die Daten- und Prozessmodellierung veranschaulicht.

Schlüsselwörter: Systems Engineering, Software Engineering, System-Entwicklung, Software-Entwicklung, Modelldarstellungen, IT-Projekte, Vorgehensmodelle, Ergebnismodelle, Prozessmodellierung, Datenmodellierung, ARIS

A Zur Einordnung der WBT-Serie

Die WBT-Serie richtet sich an Interessenten des Themenbereiches „Systems Engineering“.

Für Ihr Selbststudium per WBT müssen Sie einen Internet-Zugang haben – entweder auf Ihren eigenen PCs, auf den PCs im JLU-Hochschulrechenzentrum, in den JLU-Bibliotheken oder dem PC-Pool des Fachbereichs.

B Die Web-Based-Trainings

Der Stoff zu diesem Thema ist in Lerneinheiten zerlegt worden und wird durch eine Serie von Web-Based-Trainings (WBT) vermittelt. Mit Hilfe der WBT kann der Stoff im Eigenstudium erarbeitet werden. Die WBT bauen inhaltlich aufeinander auf und sollten in der angegebenen Reihenfolge absolviert werden.

WBT-Nr.	WBT-Bezeichnung	Bearbeitungs-dauer
1	Grundlegende Begriffe: Engineering und Modellierung	90 Min.
2	Kennzeichen von Projekten	90 Min.
3	IT-Projekte: Organisation und Personal	90 Min.
4	Projektmanagement: Standards und Software	90 Min.
5	Grundlagen zu Vorgehensmodellen	90 Min.
6	Sequentielle und evolutionäre Vorgehensmodelle	90 Min.
7	Agile Vorgehensmodelle	90 Min.
8	Vorgehensmodelle: Historie und Trends	90 Min.
9	Grundlagen der Modelldarstellung	90 Min.
10	Aufgaben- und funktionsorientierte Modellierung	90 Min.
11	Daten- und Prozessmodellierung	90 Min.

Tab. 1: Übersicht der WBT-Serie

Die Inhalte der einzelnen WBT werden nachfolgend in diesem Dokument gezeigt. Alle WBT stehen Ihnen rund um die Uhr online zur Verfügung. Sie können jedes WBT beliebig oft durcharbeiten. In jedem WBT sind enthalten:

- Vermittlung des Lernstoffes,
- interaktive Übungen zum Lernstoff und
- abschließende Tests zum Lernstoff .

Inhaltsverzeichnis

Seite

Seite

Seite

Seite

Seite

Seite

Seite

Seite

Seite

Seite

Seite

Abbildungsverzeichnis

Seite

Seite

Seite

Seite

Seite

Tabellenverzeichnis

1 Grundlegende Begriffe: Engineering und Modellierung

1.1 Historie und Grundlagen

1.1.1 Einleitung

Dieses Modul soll einen Überblick geben, wie Systeme mit Ingenieurstechniken entwickelt werden. Systeme können unterschiedlich beschaffen sein und verschiedene Komponenten haben.

Ein IT-System kann aus den Komponenten Hardware, Software und Daten bestehen. Ein IT-System wird z. B. entwickelt, um die Roboter auf einer Fertigungsstraße zu bedienen. Die Hardware (die Roboter) werden von der Software gesteuert. Deshalb ist es nur sinnvoll, beide Komponenten zusammen zu entwickeln. Die Roboter erzeugen wiederum Daten. Die erzeugten Daten müssen gespeichert und verwertet werden, um beispielsweise Wartungszyklen des Roboters vorhersagen zu können.

Ein derart komplexes System aus verschiedenen Komponenten muss mit Bedacht entwickelt werden.

Es folgt der Schluss: Die Produktion auf der Fertigungsstraße kann nur sichergestellt werden, wenn alle Komponenten fehlerfrei zusammenarbeiten. Um genau diese fehlerfreie Zusammenarbeit der Komponenten sicherzustellen, brauchen Sie Systems Engineering.

1.1.2 Das Software Engineering als Teildisziplin

Die WBT-Serie „Systems Engineering“ bezieht sich im Speziellen auf das Software Engineering als Teildisziplin des Systems Engineering. Es wird die Entwicklung von Hardware aus der Betrachtung ausgeschlossen.

Historisch gesehen, entstanden jedoch zum Beispiel Großrechner zunächst „aus einer Hand“. So entwickelte IBM sowohl die Hardware der Rechner als auch die Software und alle für den Nutzer notwendigen Applikationen. Dies gestaltete sich oft als mühsamer Prozess. Selbst wenn die Nutzer in der Lage waren, eigene Programme zu schreiben, war die Entwicklung anwendungsspezifischer Software aufwendig und schwierig.

Ein Blick in die Vergangenheit wird uns verdeutlichen,:

- wie aufwendig Software-Entwicklung ist und

- wieso Sie standardisierte Methoden und Techniken benötigen.

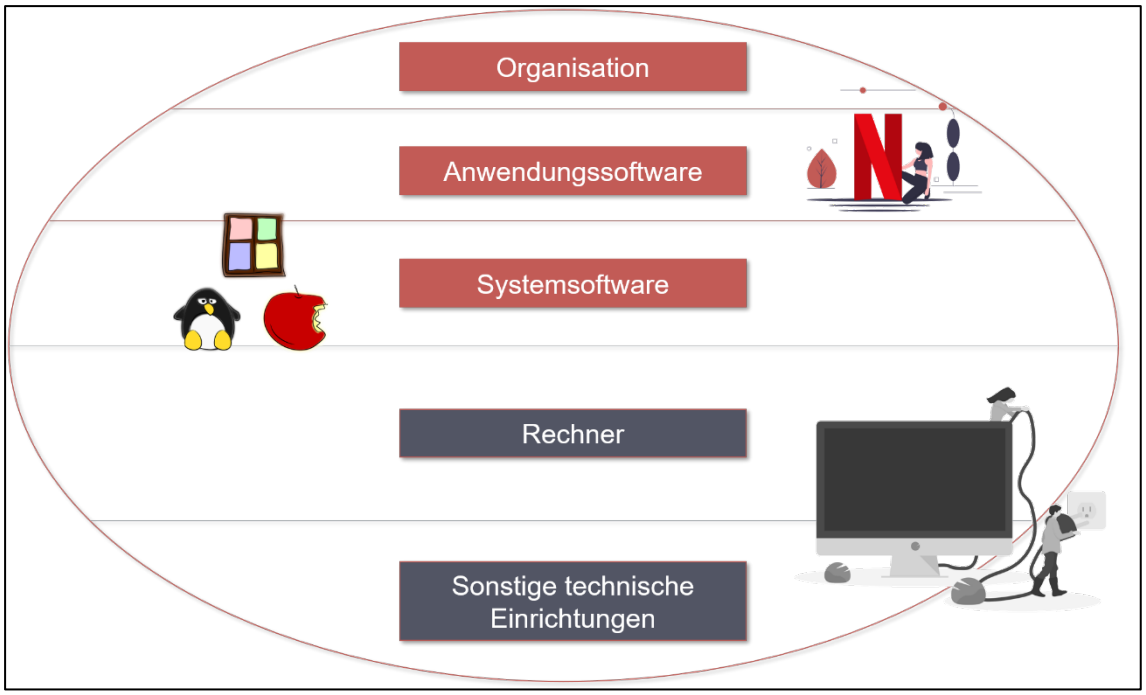


Abb. 1: Einordnungsschemata des System Engineerings

1.1.3 Die Anfänge der Software-Entwicklung – Lochkarten

Zu Anfang der Software-Entwicklung wurden Maschinen durch Lochkarten programmiert. Auf den Bildern ist ein sogenannter Lochkartenlocher der Firma IBM zu sehen.

Auf den Lochkarten sieht man eingestanzte Löcher auf Karton. Ein Loch ist dabei gleichzusetzen mit 1 (= Spannung). Kein Loch ist dabei gleichzusetzen mit 0 (= keine Spannung). Die Programmierung der Maschinen und somit die Speicherung von Daten erfolgte also durch Löcher auf einem Karton, die manuell eingestanzt wurden.

Durch diese einfache Programmierung war es beispielsweise elektrischen Lochkarten-Zähl- und Sortiermaschinen bereits 1890 möglich, die US-Volkszählung von sieben auf zweieinhalb Jahre zu verkürzen.

1.1.4 Die Anfänge der Software-Entwicklung – Binär-Code

Die nächste, aber immer noch aufwendige Evolutionsstufe der Software-Entwicklung ist der Binär-Code. Der Binär-Code ist die digitalisierte Form der Lochkarte und besteht aus einem System aus 0 und 1. In Computern werden Daten durch den Zustand einer Zelle gespeichert, z. B. an oder aus, hoch oder tief, 1 oder 0. Dieses duale System bildet bis heute die Grundlage eines jeden Computers. Alle höheren Programmiersprachen führen auf den Binär-Code zurück.

Binär-Code ist eine maschinelle Programmiersprache der 1. Generation und für Menschen kaum lesbar. Es ist mühsam, in der Abfolge von Nullen und Einsen Fehler zu finden, den Code zu warten oder weiterzuentwickeln.

Fachkräfte, die das Dualsystem beherrschten, verstanden und damit programmieren konnten, waren selten zu finden.

1.1.5 Die Anfänge der Software-Entwicklung – Assembler

Eine deutliche Erleichterung der Programmierung stellte die 2. Generation von Programmiersprachen dar. Die sogenannten Assembler folgten auf den Binär-Code. Der erste Assembler wurde zwischen 1948 und 1950 von Nathaniel Rochester für eine IBM 701 geschrieben.

Assembler bestehen aus bereits fertigen „Binär-Code-Paketen“. Assembler greifen somit auf einen bestimmten Befehlsvorrat zurück. Ein solcher Befehl könnte etwa die Anweisung sein, eine bestimmte Information von A nach B zu bewegen. Der dazu nötige Befehl in Assembler-Sprache wäre „MOV“. Durch die Einführung von lesbarem Text in Programmiersprachen wurde die Programmierung von Software für den Menschen wesentlich einfacher.

Aber auch diese Form der Programmierung war immer noch sehr aufwendig. Die Befehlsvorräte waren an die Prozessorarchitektur der vorliegenden Rechner gebunden. Für den Programmierer bedeutete dies, zunächst die Architektur des Rechners zu verstehen und daraufhin angepasste Software zu programmieren. Die Software-Entwicklung war somit immer noch zeitaufwendig und kostspielig. Software-Entwickler, die die Programmierung beherrschten, waren weiterhin dünn gesät.

1.1.6 Die Anfänge der Software-Entwicklung – Höhere Programmiersprachen

Die Forderung nach Programmiersprachen, die für den Menschen leichter verständlich sind, wurde mit der 3. Generation erfüllt. Zu den Sprachen der 3. Generation gehören unter anderem Fortran, Cobol und Algol.

Diese Programmiersprachen sind durch ihre Lesefähigkeit für den Menschen leichter verständlich und in Folge auch leichter erlernbar. Fehler im Code können leichter entdeckt und behoben werden. Code kann von verschiedenen Programmierern angepasst und weiterentwickelt werden. Die Software-Entwicklung wurde schneller, aber auch komplexer.

Die Möglichkeit der schnelleren Entwicklung von Software zeigte auf, dass es nun nötig war, den Prozess der Software-Entwicklung zu strukturieren. Standardisierte Techniken, die halfen den Code für andere nachvollziehbar zu machen, wurden benötigt. Die Komplexität der Software-Entwicklung musste beherrscht werden. Das Ziel einer schnellen

Anwendungsprogrammierung konnte nicht mehr durch das „drauf-los“-Programmieren erreicht werden, wie es vorher häufig der Fall war.

Der hier gezeigte Code von links nach rechts in Fortran, Cobol und Algol zeigt, wie die Bildschirmanzeige „Hello, World!“ in den verschiedenen Sprachen programmiert wird.

Hier ist der Unterschied und die Vereinfachung zum Binär-Code deutlich ersichtlich. „Hello, World!“ würde sich wie folgt lesen:

0100100001100101011011000110110001101111001011000101011

1011011110111001001101100011001000010000 1

Fortran

```
program helloworld
implicit none
character*13 hello_string
hello_string = „Hello, world!“
write (*,*) hello_string
end program helloworld
```

Cobol

```
HELLO * HISTORIC EXAMPLE OF HELLO
WORLD IN COBOL
       IDENTIFICATION DIVISION.
       PROGRAM-ID. HELLO.
       PROCEDURE DIVISION.
           DISPLAY "HELLO, WORLD".
           STOP RUN.
```

Algol

```
BEGIN
FILE F (KIND=REMOTE);
EBCDIC ARRAY E [0:11];
REPLACE E BY "HELLO WORLD!";
WHILE TRUE DO
  BEGIN
  WRITE (F, *, E);
  END;
END.
```

Abb. 2: Beispiele Höherer Programmiersprachen

1.1.7 Die Anfänge der Software-Entwicklung – Fehleinschätzungen

Die neue Technologie und die Entwicklung von Software wurden gerade am Anfang unterschätzt.

Sie erinnern sich an das berühmte Zitat von Thomas Watson als IBMs Präsident von 1943.:

„I think there is a world market for maybe five computers.“

Bis weit in die 60er Jahre wird der zukünftige Bedarf an Rechnern und damit an Software völlig unterschätzt. Darüber hinaus wird Software meist ohne Grundlage von verallgemeinerten, nachvollziehbaren und zielführenden Techniken entwickelt. Methoden wie SCRUM oder DIN- und ISO-Normen wie man sie aus Ingenieursdisziplinen kennt, wurden nicht für die Software-Entwicklung verwendet.

Unsystematische Software-Entwicklung führte jedoch zu einer Reihe von schwerwiegenden Problemen.

1.1.8 Die Anfänge der Software-Entwicklung

Wir fassen noch einmal die Anfänge der Software-Entwicklung zusammen. Durch Fehleinschätzungen und Mängel mündete dies schlussendlich in den 60er Jahren in einer Software-Krise.

Programmiersprachen der 1. und 2. Generation

- Bis Ende der 50er Jahre wird hauptsächlich in maschinennahen Sprachen wie Binär-Code und Assembler programmiert.
- Der Code ist unübersichtlich und dadurch fehleranfällig.
- Es mangelt an Programmierern.

Programmiersprachen der 3. Generation

- Ab Ende der 50er Jahre werden höhere Programmiersprachen verwendet, die für Menschen verständlicher sind.
- Beispiele sind Fortran, Cobol und Algol.
- Es mangelt weiterhin an Programmierern.

Fehleinschätzungen und Mängel

- Bis weit in die 60er Jahre wird der zukünftige Bedarf an Rechnern und damit an Software völlig unterschätzt.
- Es mangelt an verallgemeinerten, nachvollziehbaren, zielführenden Techniken zur Software-Entwicklung.

1.1.9 Die Software-Krise in den 60er Jahren

Das Vorgehen bei der Programmierung von Software bis in die 60er Jahre hieß „Vom Hirn ins Terminal“. Strukturierte Anforderungsanalysen und Planungen waren selten. Die Programmierer setzten Kundenanforderungen direkt in den Code um. Mit wachsendem Bedarf an Software führte dies zu größeren Problemen.

„Vom Hirn ins Terminal“ or „Code & fix“ or „Quick'n dirty“

- Write some code / Schreibe irgendwelchen Code
- Fix the problems in the code / Behebe Probleme im Code

„Quick 'n dirty“ gehört zu den Software-Entwicklungsmethoden, die unter dem Begriff „cowboy coding“ zusammengefasst werden. Beim Cowboy Coding liegt das Hauptaugenmerk auf dem Schreiben von Code und nicht in der sorgfältigen Planung des Software-Projektes.

Der Prozess des Programmierens wird bei dieser Methode lediglich in zwei Phasen unterteilt: Codieren und fixen. Dieses Prinzip lässt sich mit dem „Trial-and-Error“-Prinzip vergleichen.

Probleme und Folgen

- Unstrukturierter Code, der weder weiterentwickelt werden kann noch Fehler aufzeigt.
- Unstrukturierte Vorgehensweise, die eine Planung des Software-Projekts und der damit verbundenen Kosten unmöglich macht.
- Koordination großer Entwicklerteams durch fehlende Standards und Normen unmöglich.
- Kundenanforderungen können nicht erfüllt werden, da keine Standards für qualitativ-hochwertige Entwicklung existieren.
- Zeit- und Kostenüberschreitungen sind an der Tagesordnung.
- Durch die fehlenden Fachkräfte können nur wenige Anwendungen umgesetzt werden.

1.1.10 Software-Kunst oder Software-Produkt?

Bis Ende der 60er Jahre wurde Software-Entwicklung als eine geistig-kreative Tätigkeit aufgefasst.

Erst dann setzte sich die Erkenntnis durch, dass Software ein komplexes technisches Produkt ist, das mit systematischen, kontrollierten Arbeitsprozessen zu erstellen ist.

In diesem Sinne muss Software „technisch konstruiert” werden – unter Einsatz von ingenieurswissenschaftlichen Methoden, Verfahren und Werkzeugen. Denn die Ingenieurswissenschaften entwickelten bereits Methoden, um komplexe Systeme verständlich aufzubereiten.

Die folgende Seite zieht einen Vergleich zwischen einem Kunstwerk und einem technischen Produkt.

1.1.11 „Technische Konstruktion“ vs. „geistig-kreative Tätigkeit“

	Werkstatt (techn. Produkt)	Atelier (Kunstwerk)
Die geistige Voraussetzung	ist das vorhandene und verfügbare technische Know-how	ist u. a. die Inspiration des Künstlers
Die Termine	sind in der Regel mit genügender Genauigkeit planbar	sind wegen der Abhängigkeit von der Inspiration nicht planbar
Der Preis	ist an den Kosten orientiert und darum kalkulierbar	ist nur durch den Marktwert, nicht durch die Kosten bestimmt
Normen und Standards	existieren, sind bekannt und werden in aller Regel respektiert	sind rar und werden, wenn sie bekannt sind, nicht respektiert
Eine Bewertung, ein Vergleich	kann nach objektiven, quantifizierten Kriterien durchgeführt werden	ist nur subjektiv möglich, das Ergebnis ist umstritten
Der Urheber	bleibt meist anonym, hat keine emotionale Bindung zum Produkt	betrachtet das Kunstwerk als Teil seiner selbst
Gewährleistung und Haftung	sind klar geregelt, können nicht ausgeschlossen werden	sind undefiniert und praktisch kaum durchsetzbar

Abb. 3: Vergleich des Software-Engineering Mindsets

1.1.12 „Software Engineering“ – Wie es begann

„The whole trouble comes from the fact that there is so much tinkering with software. It is not made in a clean fabrication process, which it should be.“

„What we need, is software engineering.“

– Friedrich L. Bauer

1.1.13 Die Softwarekrise

Die Softwarekrise vermittelte den Eindruck, dass Software nicht mit der Entwicklung der physischen Rechner-Anlagen Schritt halten konnte. 1967 beschloss das NATO Science Committee daraufhin eine „Study Group on Computer Science“ einzurichten. Friedrich Bauer wurde gebeten, sich an dieser Arbeitsgruppe zu beteiligen. Er war Professor an der Technischen Universität München und hielt bereits 1967 erste Informatik-Vorlesungen in Deutschland.

Der Auftrag dieser Arbeitsgruppe war unter anderem, das gesamte Gebiet der „Computer Science“ zu analysieren und eine Konferenz mit Fachbeiträgen zu organisieren. Zu einem späteren Zeitpunkt wurde auch bereits über eine Fachdisziplin für „Computer Science“ nachgedacht.

Man begann mit der Ausarbeitung einer größeren Anzahl von Arbeitspapieren für die Arbeitstagung. Die Tagung fand 1968 in Garmisch statt.

Die Tagung selbst hatte das Ziel, zur Auflösung der Software-Krise beizutragen. Viele Fachbeiträge erkannten, dass es in der Software-Entwicklung notwendig sei:

- Anforderungsanalysen durchzuführen,
- Projektmanagement einzuführen und
- Qualität sicherzustellen.

Die Geschichte des Software Engineerings beginnt nach der NATO-Tagung 1968.

1.2 Grundbegriffe

1.2.1 Was ist „Software“?

Software ist ein komplexes immaterielles technisches Produkt.

IEEE Std. 610.2 (1990): Software –

„Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system.“

(1) Software ist ein komplexes, immaterielles, technisches Produkt.

(2) Software wird nicht gefertigt, sondern „nur” entwickelt.

(3) Bei Software sind Original und Kopie völlig gleich.

(4) Software verschleißt nicht.

(5) Software-Fehler entstehen nicht durch Abnutzung.

1.2.2 Software entwickeln, ist anders

Software ist kein materielles Produkt. Sie können Software nicht anfassen. Software ist kein Kunstwerk, sondern ein immaterielles, technisches Produkt, das Sie strukturiert abbilden können.

Daraus ergeben sich weitere Fragestellungen, wie zum Beispiel:

- Wie können Sie den Wert der Software bemessen?
- Wie können Sie den Entwicklungsprozess der Software als Kosten kalkulieren?
- Wie kann man das Vorgehen des Entwicklungsprozesses konstruieren?

Bei der Entwicklung von Software müssen sowohl betriebswirtschaftliche als auch technische Fragestellungen behandelt werden.

1.2.3 Was ist „Software Engineering“?

Software Engineering: Systematische Erstellung von Software

- Software Engineering ist die zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Software-Systemen (nach Balzert).
- Ziel ist es, Software mit einem festgelegten Funktionsumfang, in definierter Qualität und innerhalb vorgegebener Kosten- und Zeitrahmen zu entwickeln.
- Software Engineering umfasst also neben der eigentlichen Entwicklungstätigkeit auch das Projektmanagement, die Qualitätssicherung und die Wirtschaftlichkeitsanalyse der Software-Entwicklung.

1.2.4 SWEBOK – 15 Teildisziplinen des Software Engineering

Das SWEBOK (Software Engineering Body of Knowledge) Handbuch beschreibt allgemein akzeptiertes Software-Engineering-Wissen. Herausgeber ist die IEEE (Institute of Electrical and Eletronics Engineers), ein weltweiter Berufsverband für und von Ingenieuren aus den Bereichen Elektro- und Informationstechnik.

Das SWEBOK definiert 15 Teildisziplinen des Software Engineering (Version 3.0 von 2014):

1. Anforderungsermittlung
2. Entwurf
3. Implementierung
4. Testen
5. Wartung
6. Konfigurationsmanagement
7. Projektmanagement und Metriken
8. Vorgehensmodelle
9. Werkzeuge und Methoden
10. Software-Qualität
11. Professionelle Managementpraktiken / Berufsausübung (Professional Practice)
12. Wirtschaftslehre / Wirtschaftliche Umwelt (Economics Fundamentals)
13. Grundlagen der Informatik
14. Grundlagen der Mathematik
15. Grundlagen der Ingenieurswissenschaft

1.2.5 Was ist ein „Modell“? – Grafische Beispiele

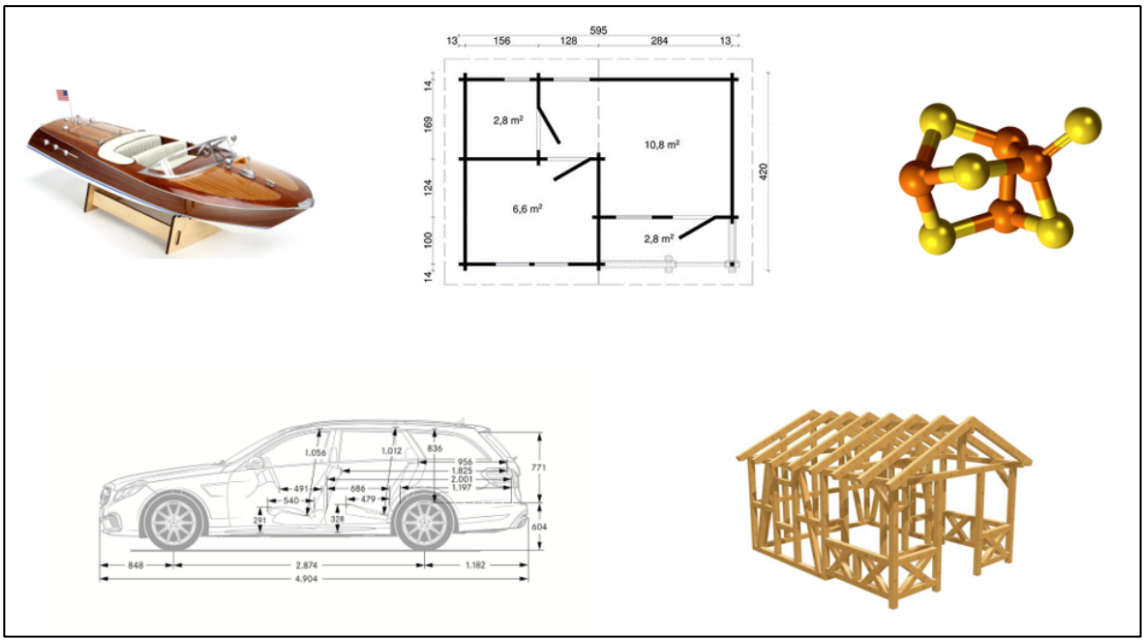


Abb. 4: Grafische Modellbeispiele

1.2.6 Was ist ein „Modell“? – Beispiel einer Spezifikation

Wikipedia: „Schraubenschlüssel“

Ein Schraubenschlüssel ist ein Handwerkzeug zum Anziehen oder Lösen von Schrauben und Muttern mit verschiedenen Antriebsprofilen. Schraubenschlüssel werden auf die Antriebsprofile oder Ausnehmungen an der Mantelfläche der Verbindungselemente gesteckt und im Drehsinn betätigt. Schraubwerkzeuge, die stirnseitig in das Antriebsprofil (kurz Profil) der Verbindungselemente gesteckt und im Drehsinn betätigt werden, nennt man dagegen Schraubendreher.

Die Größe eines Schraubenschlüssels wird durch die Schlüsselweite gekennzeichnet. Bei einem Maulschlüssel mit der Schlüsselweite 17 haben die beiden parallelen Flächen einen Abstand von 17 mm. Die Schlüsselweite ist als Zahl auf der jeweiligen Seite des Schlüssels aufgeprägt. Ein gängiger Typ ist der Gabel- oder Maulschlüssel. Mit diesem können Sechskant- oder Vierkant-Schraubenköpfe bzw. -Muttern oder auch spezielle Verbindungselemente mit nur zwei zueinander parallel angeordneten Schlüsselflächen gedreht werden.

Das Schlüsselmaul ist gewöhnlich um 15° abgewinkelt zur Werkzeugachse angeordnet, um das Ansetzen des Schlüssels bei beengtem Arbeitsraum zu erleichtern. Auch ein Winkel von 75° ist gebräuchlich. Die Größe der Maulöffnung des Schraubenschlüssels ist auf dem Werkzeug in Millimeter-Werten aufgeprägt – bei Verwendung des angloamerikanischen Maßsystems in Inch und Inch-Bruchteilen.

1.2.7 Warum brauchen wir Modelle?

Die Teildisziplinen des Software Engineerings zeigen, dass Software-Entwicklung umfangreich und komplex ist. Sie müssen nicht nur programmieren, sondern auch darauf achten, dass alle vom Kunden gewünschten Funktionen in der Software verfügbar sind.

Das Ingenieurswesen nutzt Modelle, um Komplexität zu beherrschen. Modelle bilden einen Ausschnitt der Realität ab. Modelle sind also Realitätsausschnitte, die einem Zweck dienen und verschiedene Formen annehmen können. Modelle bilden dabei die Realität vereinfacht ab und konzentrieren sich auf das Wesentliche.

Darüber hinaus gibt es verschiedene Modellarten und Darstellungsformen für Modelle. Beim Hausbau entstehen zum Beispiel Modelle für den Grundriss, die Innenausstattung oder die Elektro- und Wasserleitungen klassischerweise auf Papier.

Modelle können aber auch ganz andere Formen annehmen. Aus dem Grundriss eines Hauses kann beispielsweise eine komplette 3D-Simulation im Rechner oder ein Modellbau aus Holz und Pappe entstehen.

1.2.8 Was ist ein „Modell“?

Modelle sind zweckorientiert und bilden Realitätsausschnitte ab.

Das Ingenieurswesen nutzt Modelle, um Komplexität zu beherrschen. Modelle bilden einen Ausschnitt der Realität ab. Modelle sind also Realitätsausschnitte, die einem Zweck dienen und verschiedene Formen annehmen können.

Nach Stachowiak muss ein Modell drei Hauptmerkmale aufweisen:

1. **Abbildungsmerkmal**: Das Modell bildet das Original ab.
2. **Verkürzungsmerkmal**: Modelle bilden nicht alle Eigenschaften des Originals ab, sondern nur eine Auswahl, die vom Modellierer abhängt.
3. **Pragmatisches Merkmal**: Erfolgt die Selektion dieser Eigenschaften nach operationalen Zielsetzungen der Benutzer und werden die Zeiträume der Modellbenutzung mit einbezogen, so ersetzt das Modell das Original zu einem bestimmten Zeitpunkt und zu einem bestimmten Zweck.

Aus den Merkmalen ergeben sich folgende vier Fragestellungen, die ein Modell charakterisieren und beantworten muss:

1. Was ist das Original und was soll davon abgebildet werden?
2. Für wen soll das Modell abgebildet werden?
3. Wann soll das Modell verwendet werden?
4. Wozu soll das Modell verwendet werden?

1.2.9 Was ist ein „Modell“? – Der Hausbau

Um den Nutzen von Modellen zu verdeutlichen, schauen Sie sich das Beispiel Hausbau an.

Stellen Sie sich vor, Sie planen, ein Haus zu bauen. Sie haben bereits das Baugrundstück erworben und eine Idee, wie Ihr Haus später aussehen soll.

Um dem Bauunternehmen einen Plan zu Ihrer Idee zu liefern, beauftragen Sie für die Ausarbeitung einen Architekten.

In einem ersten Schritt beschreiben Sie dem Architekten verbal, wie Sie sich Ihr Haus vorstellen und geben ihm Fotos mit, die Ihre Idee zeigen.

Der nächste Schritt ist nun die Anfertigung eines Grundrisses, der bereits exakte Bemaßungen und technische Spezifikationen enthält. Der Grundriss wird später von der Baufirma benötigt.

Allerdings ist es schwierig, sich das Haus nur anhand des Grundrisses vorzustellen. Sie geben deshalb noch eine 3D-Simulation des Grundrisses in Auftrag.

Die genannten Modelle bauen schrittweise aufeinander auf.

1.2.10 Was ist ein „Modell“? – Die Evolution eines Modells

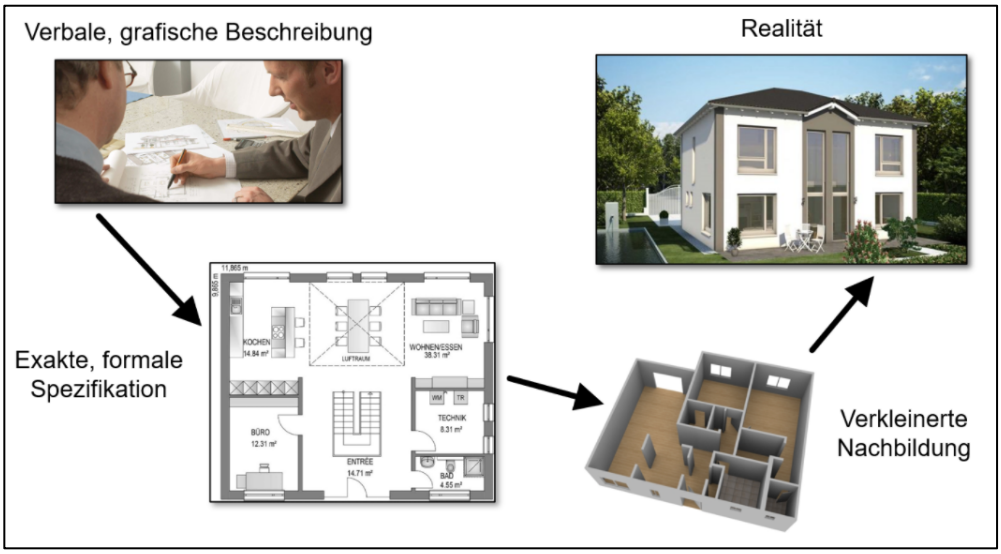


Abb. 5: Modellevolution

1.3 Modelle im Software Engineering

1.3.1 Modelle für das Produkt und den Prozess

Anhand des Beispiels vom Hausbau haben Sie gesehen, dass Modelle helfen, Komplexität zu beherrschen. Das Haus stellt dabei ein reales Produkt dar.

Die Besonderheit im Software Engineering ist, dass nicht nur das Software-Produkt Modelle benötigt. Zusätzlich zu den Modellen des zu entwickelnden Produkts werden im Software Engineering Modelle des Prozesses der Software-Entwicklung verwendet.

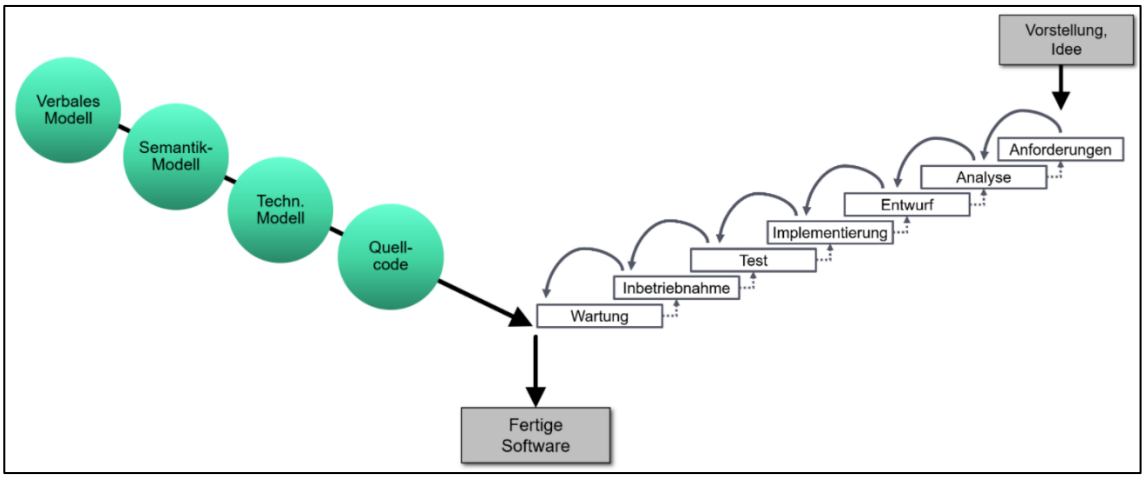


Abb. 6: Zusammenführung der Meilensteine und Projektphasen

1.3.2 Prozess- und Produktmodelle

- Prozessmodelle sind Wegbeschreibungen für das Vorgehen von der Idee hin zum fertigen Produkt.
- Auf diesem Weg werden sukzessive, ständig verfeinerte und konkretisierende „Produktmodelle“ der zu entwickelnden Software erstellt.
- Am Ende des Prozessmodells steht das fertige Produkt – die fertige Software als letztes Modell.

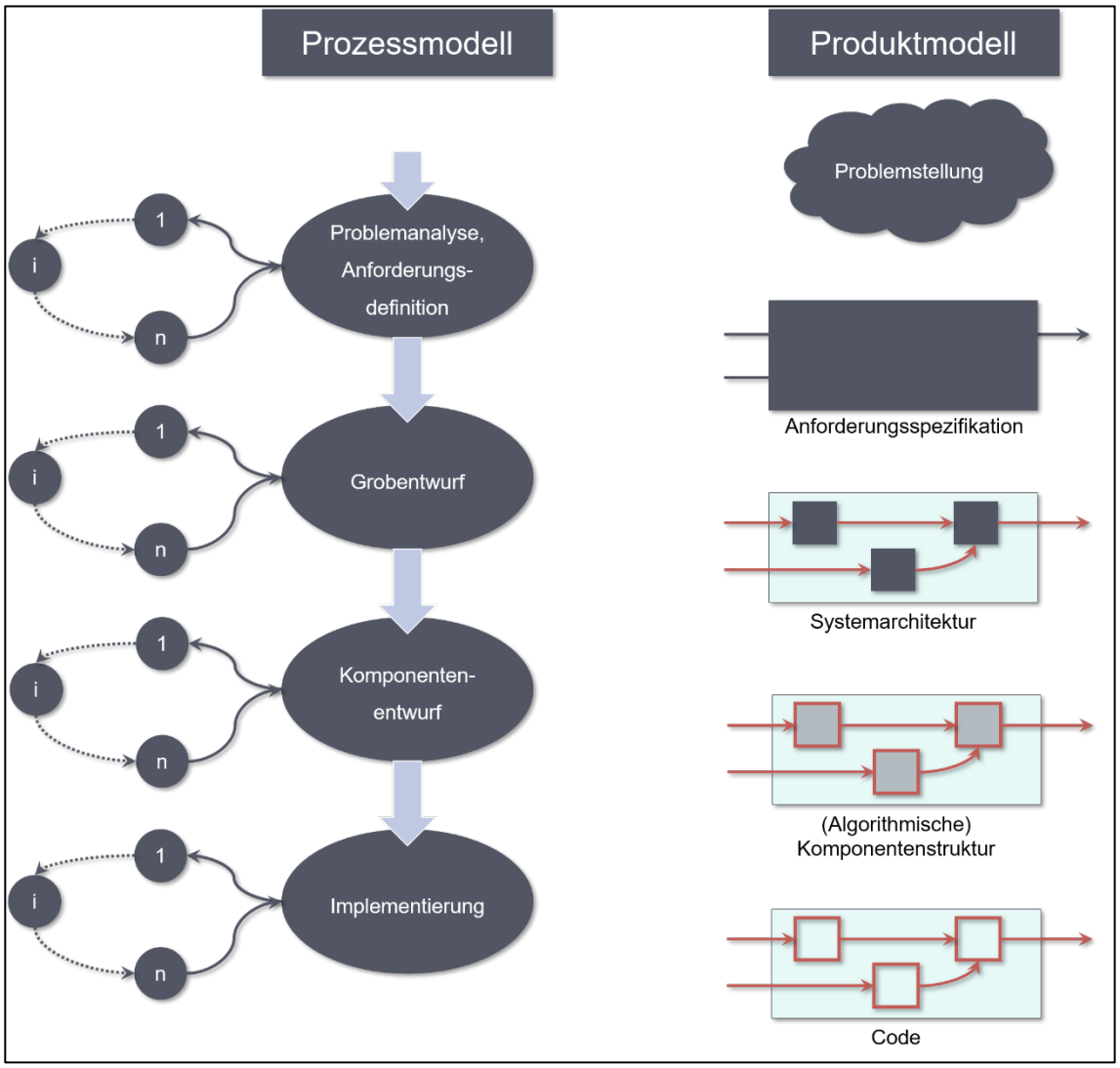


Abb. 7: Prozess- und Produktmodell

1.3.3 Ergebnis-Sicht

Sie konnten festgestellen, dass verschiedene Produktmodelle im Laufe eines Entwicklungsprozesses erstellt werden. Am Ende des Entwicklungsprozesses steht das fertige Produkt – die fertige Software als letztes Modell.

Die Ergebnis-Sicht ist das „Was" der Entwicklung. Die Produktmodelle sind Sichten auf Entwicklungsergebnisse. Produktmodelle können mit verschiedenen Modellierungsansätzen erstellt werden, wie z. B. funktions-, daten-, prozess- oder objektorientierter Modellierung.

Im weiteren Verlauf der WBT-Serie werden Sie die wichtigsten Modellierungsansätze kennenlernen.

Strukturierte Analyse – SA

Die SA als Ausprägung der funktionsorientierten Modellierung hat sich seit ihren Anfängen in der Mitte der siebziger Jahre zu einer wichtigen Standardmethode der Systemanalyse entwickelt. Die Art der grafischen Darstellung der Vernetzung von parallelen Funktionen bzw. Prozessen in Datenflussdiagrammen, die den Kern der strukturierten Analyse darstellen, wurde bereits vor fast 90 Jahren von Forschern auf dem Gebiet des Operation Research verwendet.

Geschäftsprozessmodellierung – GPM

Unternehmen haben viele Möglichkeiten, ihre Geschäftsprozesse darzustellen. Wichtig dabei ist immer die detaillierte Angabe aller Aktivitäten und beteiligten Personen, die Bestandteil der einzelnen Fachabteilungen sind. Diese Darstellung wird Geschäftsprozessmodellierung genannt. Bei der Geschäftsprozessmodellierung wird das „Was“, „Wer“ und „Wie“ der Geschäftsprozesse beschrieben. Eine häufig verwendete Methode zur Modellierung von Geschäftsprozessen ist die BPMN (**B**usiness **P**rocess **M**odel and **N**otation).

Objektorientierte Modellierung mit der UML

Heute hat sich mit der Unified Modeling Language ein Standard zur objektorientierten Modellierung durchgesetzt. Dieser bietet zwar eine einheitliche Modellierungssprache, beschreibt allerdings keinen Prozess zur objektorientierten Modellierung.

Es gibt zwei Modellierungsansätze, die der objektorientierten Modellierung eine besondere Bedeutung beimessen.:

- Semantische Objektmodelle und
- die multiperspektivische Unternehmensmodellierung.

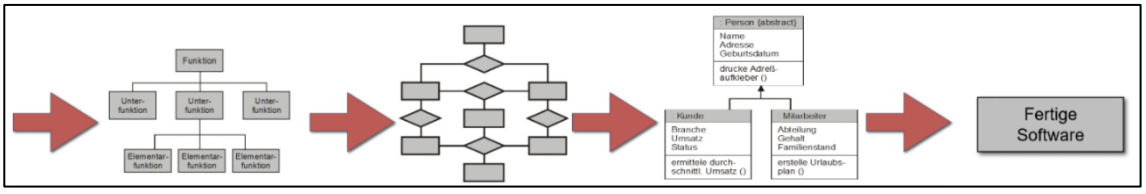


Abb. 8: Evolution der Meilensteine zur fertigen Software

1.3.4 Prozess-Sicht

Um das Software-Produkt zu entwickeln, wird ein geplantes Vorgehen benötigt. Von der Idee des Produktes bis hin zur Benutzung durch den User liegen viele Zwischenschritte.

Diese Zwischenschritte können durch „Meilensteine“ abgegrenzt werden. Ein geplantes Vorgehen von der Idee über Zwischenschritte bis zum fertigen Produkt wird Prozess-Modell genannt.

Die Prozess-Sicht beschreibt also das „Wie“ der Entwicklung – die Gesamtheit der Schritte zur Entwicklung des IT-Systems.

1.3.5 Ergebnis- und Prozess-Sicht

Zwei Sichten der Planung und Entwicklung von IT-Systemen

- Die Ergebnis-Sicht: Das „Was” der Entwicklung – die Gestaltung und Darstellung des IT-Systems mit Modellierungsansätzen wie z. B. funktions-, daten-, prozess-, objektorientierter Modellierung („Produktmodelle“)
- Die Prozess-Sicht: Das „Wie” der Entwicklung – die Vorgehensweise der Entwicklung – die Prozessschritte zur Entwicklung des IT-Systems („Vorgehensmodelle“)

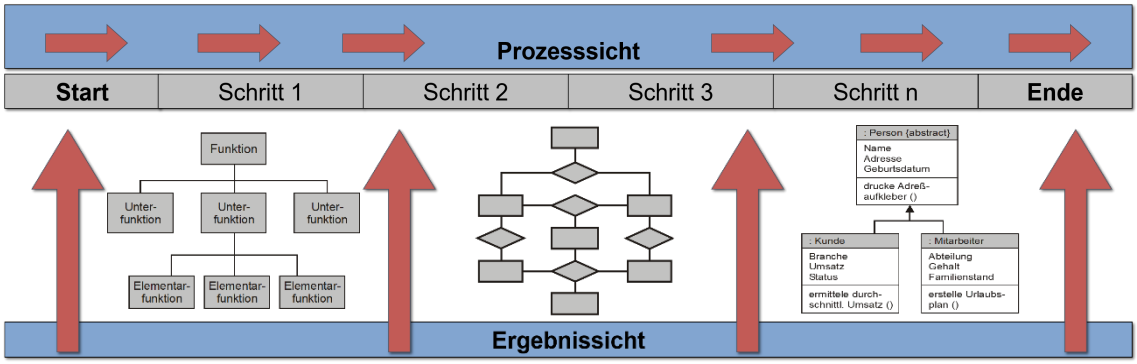


Abb. 9: Prozess- und Ergebnissicht

1.4 Abschlusstest – WBT 01

1.4.1 Abschlusstest

Bitte beantworten Sie die folgenden Fragen durch das Ankreuzen der korrekten Antworten (Tab. 2). Bei einigen Fragen können auch mehrere Antworten richtig sein.

Nr.	Frage	Richtig	Falsch
1	Termine in der Entwicklung einer Software sind nicht planbar.		
2	Software ist ein komplexes, immaterielles und technisches Produkt.		

3	Was umfasst Software Engineering unter anderem?		
	Entwicklungstätigkeiten		
	Projektmanagement		
	Qualitätssicherung		
	Wirtschaftlichkeitsanalyse		
	Subjektive Bewertungsmethoden		
4	Modelle sind zweckorientiert.		
5	Das _____. der Planung und Entwicklung von IT-Systems wird durch die _____. dargestellt. Das _____. der Planung und Entwicklung von IT-Systems wird durch die _____. dargestellt.		
6	Was sind die Hauptmerkmale des allgemeinen Modellbegriffs?		
	Abbildungsmerkmal		
	Standardisierungsmerkmal		
	Pragmatisches Merkmal		
	Messbarkeitsmerkmal		
	Verkürzungsmerkmal		
	Praktisches Merkmal		

Tab. 2: Abschlusstest – WBT 01

1.5 Typische Aufgabenstellungen

Typische Aufgabenstellungen – Grundbegriffe: Engineering und Modellierung:

Zur Bearbeitung dieser Aufgabenstellungen beachten Sie bitte: Verlangt ist eine fachlich zutreffende, inhaltlich nachvollziehbare und kausal zusammenhängende Erörterung aus vollständigen Sätzen in lesbarer Handschrift. Für jede Aufgabe: Maximal zwei Seiten Text.

Aufgabe 1:

Erläutern Sie die Anfänge der Software-Entwicklung in den 1950er und 1960er Jahren.

Aufgabe 2:

Was ist Software?

Aufgabe 3:

Erläutern Sie den Begriff und die Entstehung des Software Engineerings.

Aufgabe 4:

Was verstehen Sie unter Vorgehensmodellen (Prozessmodellen) und Ergebnismodellen (Produktmodellen)?

Aufgabe 5:

Welche zwei Sichten werden bei der Planung / Modellierung von Informations- und Kommunikationssystemen grundsätzlich unterschieden? Erläutern Sie, welchen Zweck diese beiden Sichten erfüllen sollen.

Aufgabe 6:

Was verstehen Sie unter methodischer Durchgängigkeit / Standardisierung?

2 Kennzeichen von Projekten

2.1 Was sind Projekte?

2.1.1 Projekte anstatt routiniertes Tagesgeschäft

Die meisten Software-Entwicklungen werden in Projekten abgehandelt. Dabei tauchen in den Medien immer wieder Schlagzeilen von gescheiterten IT-Projekten auf.

Der aktuelle Abschnitt dieses WBT soll darüber informieren, wie Sie mit Hilfe von Projekt-Arbeit Software erstellen und was Sie unter Projekten verstehen können.

Sie sollen einen Überblick darüber erhalten, warum die Entwicklung von Software nicht im Tagesgeschäft abgewickelt werden kann und welche Besonderheiten zu beachten sind, wenn Sie Projekte durchführen.

Kurzum werden die Fragen beantwortet:

- Wieso benötigen Sie Projekte?
- Wieso bearbeiten Sie Aufgaben in Projekten?
- Wieso kann dies nicht im Tagesgeschäft abgewickelt werden?
- Wofür brauchen Sie Projekte?

Zunächst wird ein einschlägiges Beispiel eines gescheiterten IT-Projektes betrachtet.

2.1.2 1996: IBM zu Olympia

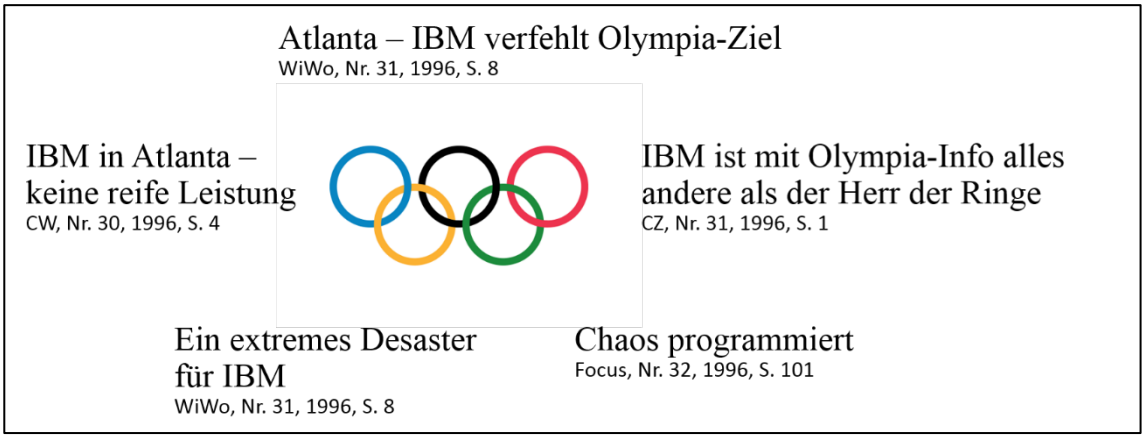


Abb. 10: Info96 Artikel

1996 sollte das Informationssystem „Info96“ der IBM Nachrichtenagenturen über die Olympischen Spiele in Atlanta informieren. Das WNPA (World News Press Agency System) sollte die Nachrichtenagenturen mit aktuellen Informationen über die Olympischen Spiele versorgen. Die Zugangsgebühren für die Nachrichtenagenturen betrugen 10 Tsd. US-Dollar, die Sponsorengebühren für IBM ca. 40 Mio. US-Dollar. Millionen Fernsehzuschauer sollten die technologische Kompetenz von IBM wahrnehmen. Der Fachwelt

sollte die Software Lotus Notes als Lösung für die Integration von Daten ins World Wide Web angepriesen werden. IBM installierte dafür über 7000 Computer, 4000 Web-Pages und 250 lokale Netzwerke, welche die Ergebnisse in Rekordzeit weiterreichen sollten. Auf Grund fehlerhafter Software und mangelnder Koordination endete das Projekt jedoch in einem extremen Desaster.

2.1.3 Leistungen des Systems „Info 96“

Fragwürdige Ergebnisse

- Boxer mit 50 cm Körpergröße kämpfen gegen Riesen mit 6,5 Meter Länge.
- Pferd und Reiter tragen denselben Namen.
- Zwei Bahnradfahrer stellen gleichzeitig einen Weltrekord auf, obwohl ihr Rennen erst am Folgetag startet.

Organisatorische Folgen

- Wettkampfergebnisse wurden aufgrund „technischer Probleme” nicht oder nur schleppend zu den Medienvertretern übertragen.
- Zeitweilig erhielten die ca. 17.000 akkreditierten Journalisten nicht einmal die Listen der Athleten, die sich für die Wettkämpfe gemeldet hatten.
- „Zwei Tage nach Beginn der Spiele war das Olympische Komitee noch immer nicht in der Lage, die Ergebnisse der Wettkämpfe auf die eigene Internet-Seite zu übertragen.”
- „Zwischenzeitlich wurde ein Notsystem eingerichtet, das an die alten Griechen erinnert. Per Fax gingen die Ergebnisse an ein zentrales Büro. Von dort aus übernahmen Boten die weitere Verteilung.“

2.1.4 Gründe für die Fehlleistungen des Systems „Info 96“

Systemkomplexität

- 7000 PCs, 100 Workstations, 4 Großrechner
- 5450 Meilen Kupferkabel, 2000 Meilen Glasfaserkabel verlegt
- 4000 Web-Pages mit 5000 Bildern und 65 Videos
- 1000 Arbeitsplatzdrucker, 150 Systemdrucker
- 250 lokale Netzwerke zur PC-Verbindung
- 100 verschiedene Programme im Einsatz

Mangelhafte Koordination

- „Totale Kommerzialisierung der Spiele”
- Vielzahl von Sponsoren, vor allem Telecom-Unternehmen
- Mangelhaftes Zusammenspiel der Beteiligten

2.1.5 Projektorientierung

Projekte wie das Info 96-Projekt von IBM sind typisch für die Software-Entwicklung. In den letzten Jahrzehnten steigt die Anzahl der durchgeführten Projekte durch verschiedene Einflussfaktoren.

Eine dieser Projektkonstellationen ist die steigende Aufgabenkomplexität. Unsere Systeme werden immer mehr vernetzt. Insel-Systeme müssen nun Schnittstellen zu anderen Systemen herstellen, um eine Prozesskette bestmöglich und effizient gestalten zu können.

Neben der Komplexität tragen auch der Technologiefortschritt und die damit einhergehende Dynamisierung der Märkte dazu bei, dass vermehrt Projekte durchgeführt werden. Während früher eine Software über mehrere Monate oder gar Jahre entwickelt wurde, stehen heute erste (Beta-)Versionen bereits nach wenigen Wochen oder Tagen zur Verfügung.

In zwingender Folge bedeutet dies auch eine Änderung der Methoden, mit denen Sie Software entwickeln.

Projektorientierung durch...

- Komplexitätssteigerung anstehender Aufgaben
- Dynamisierung der Märkte
- Technologiefortschritte
- Wertewandel und Paradigmenwechsel

2.1.6 Was kennzeichnet ein Projekt?

Sie wissen nun, wieso Sie Projekte benötigen und warum diese immer häufiger in der Praxis angewandt werden. Doch welche Eigenschaften unterscheiden ein Projekt von der täglichen Routinearbeit?

Im Wesentlichen unterscheidet sich ein Projekt durch:

- die Einmaligkeit der Bedingungskonstellation,
- die exakte Zielvorgabe,
- zeitliche, finanzielle, personelle und inhaltliche Begrenzungen,
- die Abgrenzung gegenüber anderen Vorhaben und
- die projektspezifische Organisation.

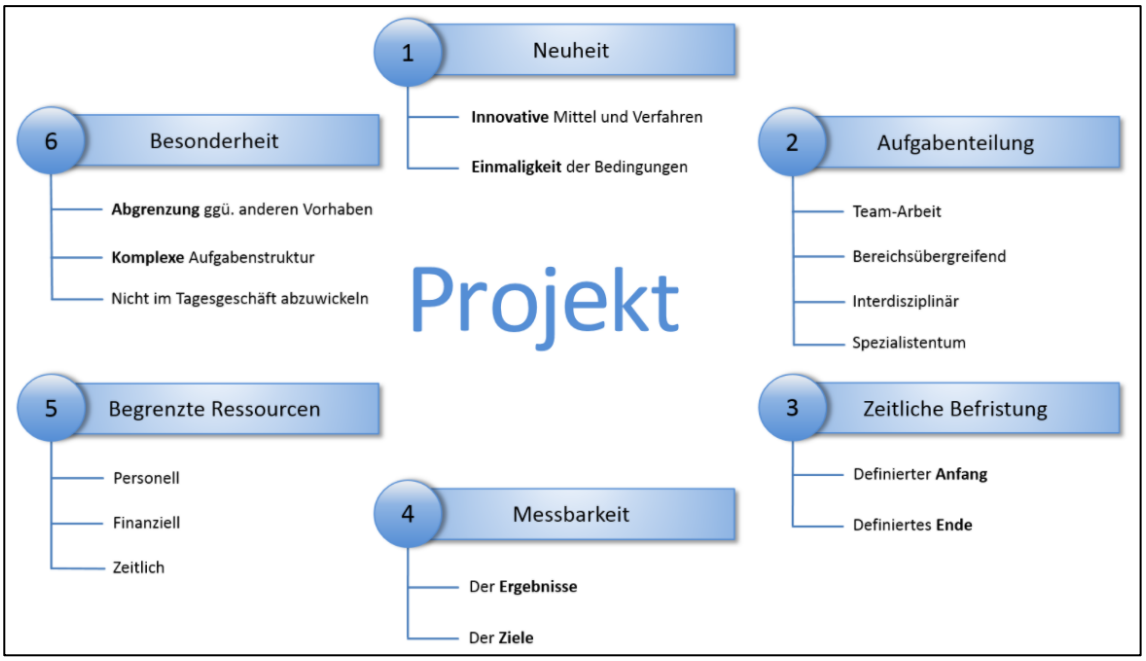


Abb. 11: Projekteigenschaften

2.1.7 Bedingungskonstellation für Projekte

Projektorientierung ist durch bestimmte wirtschaftliche und technologische Rahmenbedingungen notwendig geworden. Projekte unterliegen aber auch Herausforderungen. Hier sind einige Rahmenbedingungen beispielhaft aufgeführt.:

1. Innovative Mittel und Verfahren
2. Komplexe Aufgabenstrukturen
3. Arbeitsteilung, Spezialistentum, Teamwork
4. Großprojekte mit hohem Risiko
5. Terminzwang:
   Jedes Projekt hat einen definierten Start- und Endzeitpunkt. Alle notwendigen Aufgaben zur Fertigstellung des Projekts müssen in einem festen zeitlichen Rahmen vollendet werden. Ein Projekt unterliegt immer einen Terminzwang. Der Kunde erwartet dabei unter Umständen nicht nur den einzuhaltenden Liefertermin des Produktes, sondern auch Zwischenergebnisse (Meilensteine). Auch die Meilensteine sind zu einem bestimmten Zeitpunkt zu liefern.
6. Integration von Planung, Steuerung und Kontrolle:
   Ein Projekt kann nicht im operativem Tagesgeschäft abgehandelt werden. Deshalb werden für Projekte eigene Organisationsformen aufgesetzt, die von einem Projektleiter geführt werden. Der Projektleiter plant, steuert und kontrolliert das

Projekt; er vereint somit Positionen aus mehreren Bereichen eines Unternehmens, z. B. das strategische und mittlere Management durch die Maßnahmenplanung, die Controlling-Abteilung durch die Kennzahlensteuerung und die Finanzabteilung durch die Budgetplanung.

2.1.8 Beispiele für Projekte

- Euro-Einführung, Y2K, Toll Collect, nPA, eGK
- Relaunch einer Web Site einer Organisation
- Einführung einer betriebswirtschaftlichen Standardanwendungssoftware
- Aufbau einer VR-Umgebung zum Produkt-Support

2.2 Erfolgsfaktoren in IT-Projekten

2.2.1 Projekterfolg kann gemessen werden

Wir haben geklärt, was ein Projekt ist und wieso Sie gerade in der jetzigen Zeit Projekte benötigen. In diesem Kapitel wird erläutert,

- was ein Projekt erfolgreich macht,
- was ein Projekt nicht erfolgreich macht und
- welche Einflussfaktoren auf ein Projekt einwirken.

In der Software-Entwicklung müssen Sie nicht nur auf eine korrekte Abarbeitung der fachlichen Aufgaben achten. Sie müssen sich auch der möglichen Herausforderungen bewusst sein, um ein Projektdesaster wie beim IBM-Projekt „Info 96“ zu vermeiden.

Hierzu wurden in Langzeitstudien anhand von spezifischen Messgrößen einige wichtige Faktoren herausgearbeitet.

2.2.2 GPM 2003: Untersuchungsgegenstand, -methode

Bereits im Jahr 2003 führte die GPM eine Studie zum Thema Erfolgsfaktoren im Projektmanagement durch. Dabei wurden Projekte wie IT, Produktentwicklung, Organisationsveränderung oder Standortwechsel unter die Lupe genommen.

Die Umfrage basiert auf 76 führenden Unternehmen aus den Branchen der Finanzdienstleistung, Informations- und Kommunikationstechnologie, der Fertigung und weitere Branchen. Siebzig Prozent der Unternehmen konnten einen Umsatz mit mehr als 100 Millionen Euro vorweisen.

Wichtigste Erkenntnisse waren:

- Bei 50 % der Unternehmen machen Projektkosten mehr als 10 % der jährlichen Gesamtkosten des Unternehmens aus.

- Bei 22 % der Unternehmen machen Projektkosten mehr als 50 % der jährlichen Gesamtkosten des Unternehmens aus.

Das Ergebnis wurde durch standardisierte Interviews und schriftliche Befragungen nach Projekterfolg und verantwortlichen bzw. begleitenden Faktoren erfragt.

Die Bewertung „Projekterfolg“ durch gleichmäßige Gewichtung der Erreichungsgrade bei Projekt-Kosten, -Zeit und -Zielen erlangt.

2.2.3 GPM 2003: Ergebnisse zum Erfolg von IT-Projekten

- Mit der Größe des Projektes nimmt der durchschnittliche Erfolg ab.
- Alle IT-Projekte realisierten nur 60-80 % des vollen Erfolgs.

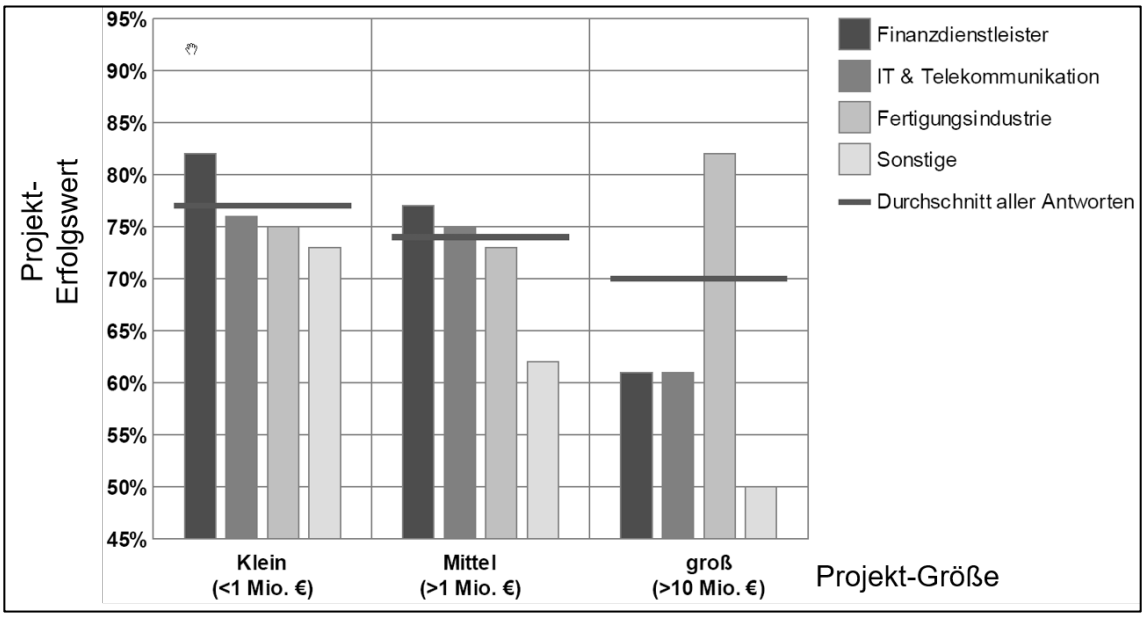


Abb. 12: GPM Untersuchungsergebnisse: Einfluss der Projektgröße

2.2.4 GPM 2003: Wesentliche „Erfolgsfaktoren“

Faktoren nach Häufigkeit der Nennung in den Befragungen:

1. Vorliegen von Geschäftsmodell und Zustimmung der Führungskräfte
2. Verfügbarkeit geeigneter Mitarbeiter in ausreichender Anzahl
3. Nutzung von Techniken und Instrumenten des Projektmanagements
4. Unterstützung des Projektmanagements durch effektives Controlling
5. Aktives Management der Veränderungen von Projektzielen und Projektanforderungen
6. Aktives Betreiben von Stakeholder-Management
7. Schulung von Projektleiter und -mitarbeiter

8. Anerkennung der Projektleistung durch Bonus, Gehaltserhöhung oder Beförderung

2.2.5 GPM 2003: Wesentlicher „Erfolgsfaktor“ von IT-Projekten

- In IT-Projekten scheinen wesentliche Erfolgsfaktoren die Vereinbarkeit mit dem Geschäftsmodell und
- die vorliegende Zustimmung der Führungskräfte zu sein.
- Bei ca. einem Drittel der IT-Projekte fehlen Geschäftsmodell und Zustimmung der Führungskräfte.

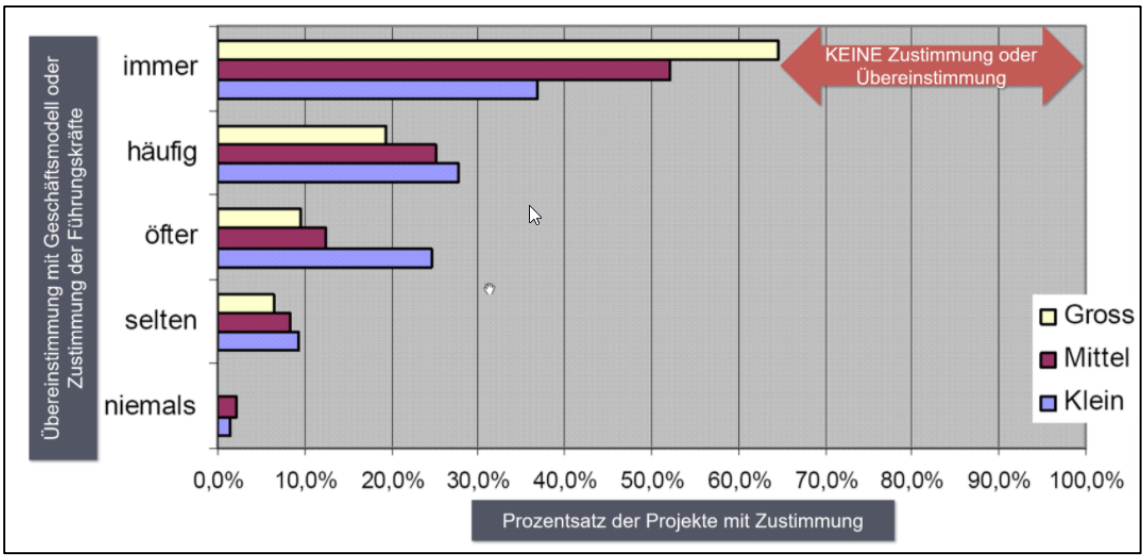


Abb. 13: GPM Untersuchungsergebnisse: Zustimmung der Führungskräfte

2.2.6 GPM 2003: Rangliste der „Misserfolgsfaktoren“

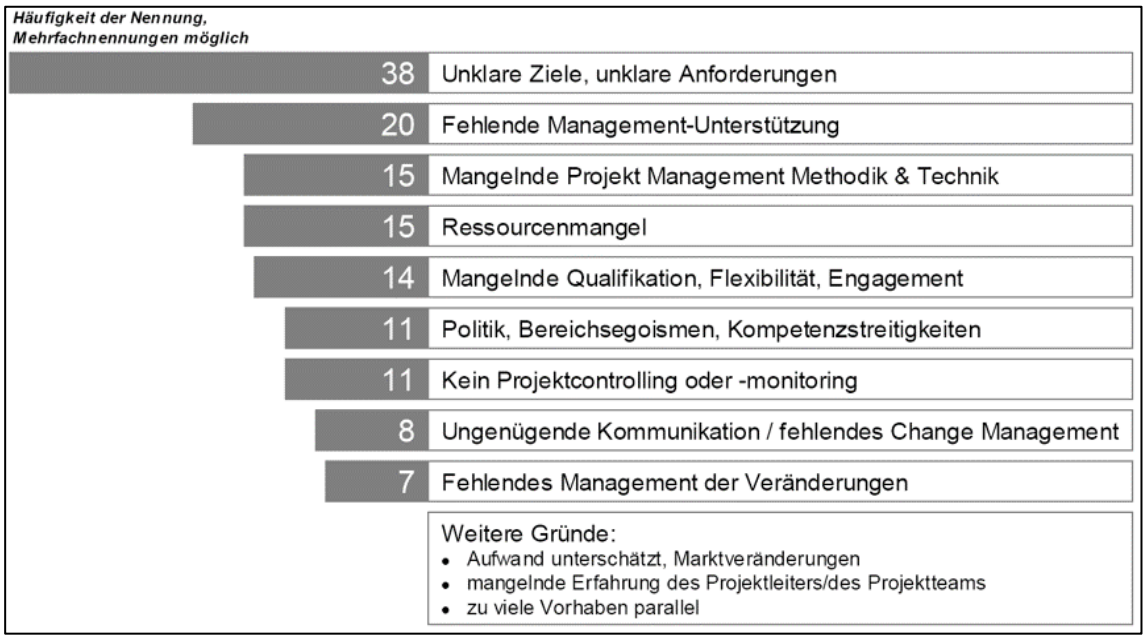


Abb. 14: GPM Untersuchungsergebnisse: Misserfolgsfaktoren

2.2.7 GPM 2003: Ursachen für erfolgreiche Projekte

Sind Erfolgsfaktoren benannt, ist es natürlich sinnvoll der Fragestellung nachzugehen, ob diese in ihrer Wichtigkeit unterschiedlich sind. Was haben die 10 besten Unternehmen signifikant anders gemacht?

	Nennungen bei den **10 besten** Unternehmen	Nennungen im **Durchschnitt** der Grundgesamtheit
1. Betreiben mehr Stakeholder Management	60%	30%
2. Schulen ihre Projektmitarbeiter intensiver	100%	77%
3. Bereiten ihre Projektleiter häufiger auf den Einsatz vor	80%	59%
4. Führen stärker Controlling und Reporting durch	85%	70%

Abb. 15: GPM: Erfolgreiche Unternehmen bei der Projektumsetzung

2.2.8 Standish Group 2013: „The CHAOS Manifesto“

Die bekannteste Langzeitstudie über Erfolgsfaktoren in IT-Projekten ist das sogenannte „CHAOS Manifesto“ der Standish Group. Seit 1994 wurden mehr als 40.000 IT-Projekte ausgewertet.

Die IT-Projekte werden dabei in drei Gruppen unterteilt: Erfolgreich, nicht erfolgreich und teilweise erfolgreich.

Die Studie untersucht Erfolgs- und Misserfolgsfaktoren und prüft eine Korrelation zwischen Erfolgswahrscheinlichkeit und Projektgröße.

RESOLUTION

	2004	2006	2008	2010	2012
Successful	29%	35%	32%	37%	39%
Failed	18%	19%	24%	21%	18%
Challenged	53%	46%	44%	42%	43%

Project resolution results from CHAOS research for years 2004 to 2012.

Abb. 16: Standish Untersuchungsergebnisse: Projekterfolgsquote

2.2.9 Standish Group 2013: Messgrößen

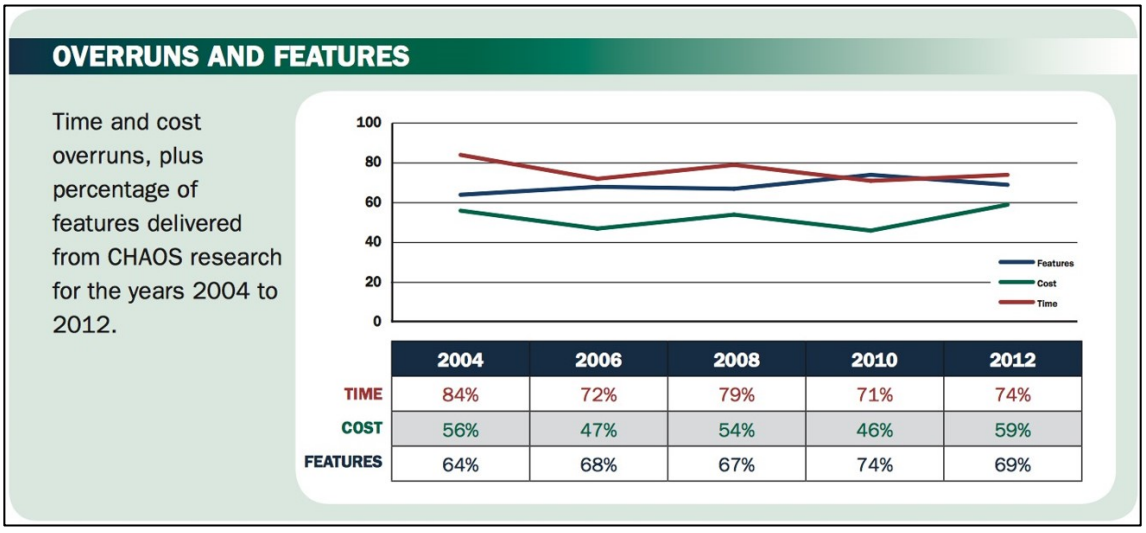
Die Messgrößen, die laut Studie ein Projekt erfolgreich machen sind dabei.:

- Zeit,
- Kosten und
- eingebaute Features, die den Anforderungen entsprechen.

Über die Jahre hinweg kann man keinen nennenswerten Trend bei der Einhaltung der Projektzeit erkennen.

Die veranschlagten Kosten eines Projektes werden auch im Jahr 2012 bei mehr als 40 % aller Projekte überschritten.

Die realisierten Funktionen bleiben weitgehend konstant. Circa 30 % der Anforderungen werden nicht umgesetzt.



	2004	2006	2008	2010	2012
TIME	84%	72%	79%	71%	74%
COST	56%	47%	54%	46%	59%
FEATURES	64%	68%	67%	74%	69%

Abb. 17: Standish Untersuchungsergebnisse: Zeit, Kosten, Features

2.2.10 Standish Group 2013: Projektgröße

Die Studie überprüft dabei nicht nur, ob ein Projekt erfolgreich war, sondern unterteilt dabei nochmal in Projektgröße mit:

- kleinen Projekten kleiner einer Millionen Dollar an Arbeitsaufwand und
- großen Projekten mit mehr als 10 Millionen Dollar Arbeitsaufwand.

Es lässt sich feststellen, dass kleinere Projekte wesentlich erfolgreicher durchgeführt werden als große Projekte in Bezug auf Einhaltung:

- der Zeit,
- der veranschlagten Kosten und
- der realisierten Anforderungen.

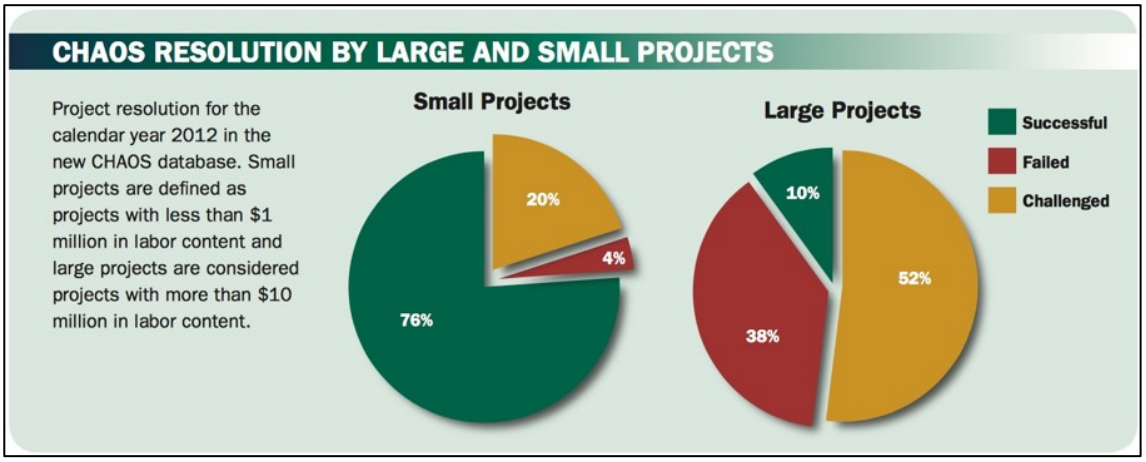


Abb. 18: Standish Untersuchungsergebnisse: Projektgröße 2013

2.2.11 Standish Group 2015: Projektgröße

Die Standish Group veröffentlichte auch 2015 einen Bericht, der Aussagen über die Erfolgswahrscheinlichkeit von IT-Projekten auf Basis ihrer Größe trifft.

Je größer ein Projekt wird, desto geringer ist die Wahrscheinlichkeit, dass das Projekt innerhalb der veranschlagten Zeit, Kosten und mit allen Features abgeschlossen wird.

	Successfull	Challenged	Failed
Grand	2 %	7 %	17 %
Large	6 %	17 %	24 %
Medium	9 %	26 %	31 %
Moderate	21 %	32 %	17%
Small	62 %	16 %	11 %

Abb. 19: Standish Untersuchungsergebnisse: Projektgröße 2015

2.2.12 Standish Group 2015: Projektmethodik

Ab 2015 wurde darüber hinaus auch die Projektmethodik in die Betrachtung miteinbezogen. Dabei ist auffällig, dass die sogenannte „Wasserfall-Methodik“ einen wesentlich geringeren Erfolg verspricht als agile Vorgehensmodelle.

Was Vorgehensmodelle sind, erfahren Sie in den WBT 05 – 07 dieser WBT-Serie.

Size	Method	Successfull	Challenged	Failed
All Size Projects	Agile	39 %	52 %	9 %
	Waterfall	11 %	60 %	29 %
Large Size Projects	Agile	18 %	59 %	23 %
	Waterfall	3 %	55 %	42 %
Medium Size Projects	Agile	27 %	62 %	11 %
	Waterfall	7 %	68 %	25 %
Small Size Projects	Agile	58 %	38 %	4 %
	Waterfall	44 %	45 %	11 %

Abb. 20: Standish Untersuchungsergebnisse: Projektmethodik und -größe

2.2.13 Standish Group 2013: Wesentliche „Erfolgsfaktoren“

Basierend auf den Daten der Standish Group sind für alle Projektarten folgende Faktoren besonders wichtig, die hier auf Basis der Wichtigkeit geordnet sind:

1. Unterstützung des höheren bzw. Senior-Managements
2. Einbindung der späteren Nutzer
3. Optimierung
4. Fachkräfte
5. Projektmanagement-Expertise
6. Agile Prozesse
7. Klar formulierte Geschäftsziele
8. Emotionale Reife
9. Durchführung
10. Tools und Infrastruktur

2.2.14 Erfolgsfaktoren für IT-Projekte

Zusammenfassend lässt sich feststellen, dass vor allem die unten gelisteten Erfolgsfaktoren für IT-Projekte wichtig sind. Es fällt auf, dass die zwischenmenschlichen Aspekte die technischen Faktoren überlagern.

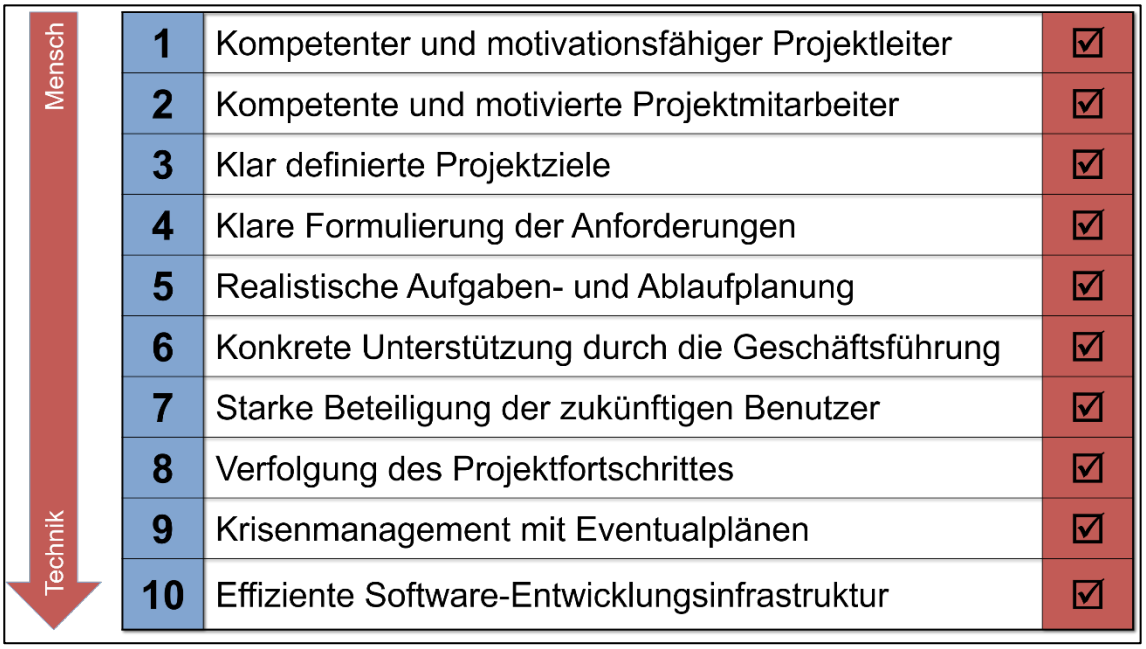


Abb. 21: Erfolgsfaktoren in IT-Projekten

2.2.15 Messgrößen für IT-Projekte

Der Erfolg eines IT-Projektes wird, wie bereits durch die Studien erwähnt, in drei Kategorien ermittelt:

- Zeit – Wie viel ist benötigt?
- Kosten – Wie viele werden verursacht?
- Qualität – Sind die Anforderungen erfüllt?

Dieses „magische Dreieck“ aus den Faktoren Zeit, Kosten und Qualität ist jedoch nicht die einzige Messgröße.

Am Rande werden Aspekte wie qualitativer Nutzen, Zukunftsfähigkeit, Wettbewerbsfähigkeit, Innovationsgrad, persönliches Engagement oder technische Eleganz zur Kenntnis genommen.

2.3 Projektmanagement

2.3.1 Projektmanagement (PM): Definition und Ziele

Projektmanagement (PM) ist das Planen, Steuern und Kontrollieren von Projekten.

Es umfasst:

- alle **Definitions**-, **Planungs**-, **Steuerungs**- und **Kontrollaktivitäten** und
- somit auch **Personengruppen**, die für diese Aktivitäten verantwortlich sind.

Die Ziele des Projektmanagements umfassen:

1. die vollständige Realisierung der Anforderungen, d. h. die Sicherstellung der **Qualität** des Projektes,
2. den **termingerechten** Abschluss des Projektes im vereinbarten Zeitraum und
3. die **kostengerechte** Durchführung des Projektes.

2.3.2 Projektmanagement: Aufgaben

Aus den Zielen des Projektmanagements lassen sich dessen Aufgaben ableiten. Unter anderem ist das Projektmanagement ein Instrument, um:

- …**Ziele eindeutig zu formulieren**. Wissen Sie nicht, was der Kunde mit seinem Projekt erreichen möchte, sind Anforderungen an die Entwicklung nur schwer zu stellen.
- …**transparente** und **aktuelle Planungen** zur Verfügung zu stellen. Ohne einen Projektplan ist es nicht möglich, das Projekt zu leiten oder nachzuvollziehen, wie es sich entwickelt.
- …die **Ressourcen anforderungsgerecht einzusetzen**. Entwickler haben viele verschiedene Fähigkeiten und sind aber für manche Aufgaben wie z. B. die Kundenkommunikation nicht geeignet.
- …**Probleme zu erkennen** und **zu beseitigen**. Stakeholder-Management ist eine wichtige Aufgabe für jeden Projektleiter, damit sich z. B. die Projektmitarbeiter auf ihre Aufgaben konzentrieren können.
- …das Projekt **permanent zu kontrollieren** und Eventualpläne zur Krisenbewältigung vorzubereiten. Beispielsweise muss eine Abweichung von den Kundenwünschen durch falsches Verständnis der Entwickler ihrer Aufgaben sofort entgegengewirkt werden.
- …das Projekt zu **berichten** und **zu dokumentieren**. Das Wissen aus einem Projekt muss erhalten werden.
- …durch **Projektklima** den **Teamgeist** zu fördern. Ein Team, welches nicht zusammenarbeitet, gefährdet den Projekterfolg.

2.3.3 Voraussetzungen für die Projektabwicklung: Organisation

Um ein Projekt erfolgreich durchführen zu können, müssen bestimmte Voraussetzungen erfüllt sein. Diese lassen sich in vier Kategorien zusammenfassen: Organisation, Führung, Zieldefinition und Verfahren.

Organisation

- Konstituierung des „Projektes“ innerhalb der Unternehmensorganisation durch exakte Definition einer Organisationseinheit „neben der Linie“ zur Strukturierung und Administration der Beteiligten.

Sie haben festgestellt, dass ein Projekt nicht im Tagesgeschäft abbildbar ist. Deshalb ist es nötig, eine Aufbauorganisation für das Projekt und die Eingliederung des Projektes ins Unternehmen herzustellen.

2.3.4 Voraussetzungen für die Projektabwicklung: Führung

Um ein Projekt erfolgreich durchführen zu können, müssen bestimmte Voraussetzungen erfüllt sein. Diese lassen sich in vier Kategorien zusammenfassen: Organisation, Führung, Zieldefinition und Verfahren.

1. **Führungskonzept und Team**

- Definition von Führungsorganisation, -aufgaben, -techniken und -mittel für die Abwicklung eines Projektes
- Projektmanagement-System zur Definition/Planung/Steuerung von Beteiligten/Einflussfaktoren und zur Kontrolle der Erfolgskriterien

Sie haben festgestellt, dass ein Projekt hauptsächlich durch menschliche Faktoren bestimmt ist. Ein Projekt benötigt folglich eine Leitung benötigt, um den Projekterfolg sicherzustellen.

Um die Erfolgskriterien zu erreichen, benötigen Sie Management-Methoden und -Techniken. Diese stellen zum Beispiel den Teamzusammenhalt sicher.

2.3.5 Voraussetzungen für die Projektabwicklung: Zieldefinition

Um ein Projekt erfolgreich durchführen zu können, müssen bestimmte Voraussetzungen erfüllt sein. Diese lassen sich in vier Kategorien zusammenfassen: Organisation, Führung, Zieldefinition und Verfahren.

2. **Zieldefinition**

- Über die Problemerkennung, die Situationsstudie und die Zielanalyse erfolgt die Formulierung fachlicher Ziele.
- Auf Basis der fachlichen Ziele werden die Projektziele konkretisiert.

Sie haben festgestellt, dass die Kundenanforderungen an ein IT-System bzw. IT-Projekt definiert werden müssen. Häufig können Sie dabei feststellen, dass die Ziele auch beim Kunden nicht konkret vorliegen. Diese Herausforderung ist im Kontext der Zieldefinition zu lösen und muss für die fachliche Ausarbeitung weiter heruntergebrochen werden. Meistens wird in diesem Zusammenhang von sogenannten Arbeitspaketen gesprochen.

2.3.6 Voraussetzungen für die Projektabwicklung: Verfahren

Um ein Projekt erfolgreich durchführen zu können, müssen bestimmte Voraussetzungen erfüllt sein. Diese lassen sich in vier Kategorien zusammenfassen: Organisation, Führung, Zieldefinition und Verfahren.

3. Verfahren und Planung

- Festlegung der Aktivitätenfolge, die das Problem einer Lösung zuführt
- Problemspezifisches Konzept unter Berücksichtigung aller Einflussfaktoren

Sie haben festgestellt, dass nicht nur das „Was“ eines Projektes in fachliche Arbeitspakete zu unterteilen ist, sondern auch das „Wie“. Folgende Fragestellungen stehen im Fokus der Planung und sollten vor Beginn des Projektes geklärt werden:

- Wie gehen Sie das Projekt an?
- Was bearbeiten Sie als Erstes und was danach?
- Wie binden Sie den Kunden mit in das Projekt ein?

2.4 Abschlusstest – WBT 02

2.4.1 Abschlusstest

Bitte beantworten Sie die folgenden Fragen durch das Ankreuzen der korrekten Antworten (Tab. 3). Bei einigen Fragen können auch mehrere Antworten richtig sein.

Nr.	Frage	Richtig	Falsch
1	Welche Erfolgsfaktoren sind in IT-Projekten wichtig?		
	Ein Projektleiter mit fachlicher Expertise		
	Unterstützung durch die Geschäftsführung		
	Einbindung von zukünftigen Nutzern in der Entwicklungsphase		
	Projektcontrolling		
	Projektreporting		

2	Das magische Dreieck ist bei IT-Projekten zu vernachlässigen.		
3	Die Projektgröße stellt keinen zu beachtenden Faktor bei der Projektplanung dar.		
4	Projekte werden durch den Technologiefortschritt bedingt.		
5	Projekte können neben dem Tagesgeschäft durchgeführt werden.		
6	Projektmanagement umfasst...		
	die Funktion		
	die Organisationsform		
	die technische Ausführung		
	die Methoden		
	die technische Fachkonzeption		
7	Projekte zeichnen sich durch die _____ der Bedingungskonstellation aus. Die _____ Zielvorgabe ist dabei besonders wichtig, um die Projektziele „in time, on budget, an on _____“ zu erreichen.		

Tab. 3: Abschlusstest – WBT02

2.5 Typische Aufgabenstellungen

Typische Aufgabenstellungen – Kennzeichen von Projekten:

Zur Bearbeitung dieser Aufgabenstellungen beachten Sie bitte: Verlangt ist eine fachlich zutreffende, inhaltlich nachvollziehbare und kausal zusammenhängende Erörterung aus vollständigen Sätzen in lesbarer Handschrift. Für jede Aufgabe: Maximal zwei Seiten Text.

Aufgabe 1:

Welche Aufgabe übernimmt das Projektmanagement?

Aufgabe 2:

Welche Aufgaben übernehmen Vorgehens- und Ergebnismodelle?

Aufgabe 3:

Erläutern Sie die Merkmale eines Projektes im Vergleich zur Arbeit in der konventionellen Linienorganisation.

Aufgabe 4:

Erläutern Sie Komplexitätssteigerung, Dynamisierung, Technologiefortschritte, Wertewandel als Ursachen für das steigende Aufkommen von „Projekten“. Nennen Sie einige typische Großprojekte, typische Projekte in Unternehmen, typische Projekte aus Ihrem persönlichen Bereich.

Aufgabe 5:

Welche herausgehobenen Erfolgsfaktoren für Projekte werden in Längsschnittstudien ermittelt? Welche nicht?

Aufgabe 6:

Erläutern Sie die vier wesentlichen Voraussetzungen für die Projektabwicklung.

3 IT-Projekte: Organisation und Personal

3.1 Was ist eine PM-Software?

3.1.1 Einleitung in die Aufbauorganisation

Sie konnten bisher feststellen, dass ein Projekt nicht mit den Routineaufgaben in der Linienorganisation durchgeführt werden kann. Deshalb ist es notwendig darüber nachzudenken,

- wie sich Projekte innerhalb der Organisation abbilden lassen und
- wie sich das Projekt selbst organisiert ist.

Auf den folgenden Seiten werden Sie verschiedene formale Organisationsformen kennenlernen. Die Organisationsformen spiegeln die Aufbauorganisation eines Projektes wider. Die Organisation ist für das jeweilige Projekt individuell auszuwählen und anzupassen.

Aber auch die personale Organisation des Projekt-Teams sollte mit Bedacht gewählt werden. Dies wird Ihnen in Kapitel zwei dieses WBT genauer erläutert.

3.1.2 Projekt-Organisation: Definition und Zielsetzung

Projekt-Organisation: Definition

Formale Regelung der Zusammenarbeit der am Projekt Beteiligten, sowie deren Einbindung in die Unternehmensorganisation

Projekt-Organisation: Zielsetzung

Optimaler Einsatz der verfügbaren Personal- und Sachmittelkapazitäten bei Erfassung der ganzen Projektbreite und des gesamten Projektablaufs

3.1.3 Projekt-Organisation: Grundformen

Für die Aufbauorganisation eines Projektes gibt es verschiedene Grundformen, die Sie sich auf den nächsten Seiten näher ansehen.

Bei der Aufbauorganisation geht es um die „Statik“ eines Projektes. Bei der Ablauforganisation geht es um die „Dynamik“ eines Projektes.

3.1.4 Der Arbeitskreis / Kommission

Der Arbeitskreis oder auch Kommission oder Gremium ist eine

> *„zur Erfüllung einer bestimmten Aufgabe gebildete Gruppe von Expert(inn)en, eine beschlussfassende Körperschaft oder ein Ausschuss“.*

Solche Gremien sind also Entscheidungswerkzeuge bestehend aus verschiedenen Experten eines Fachgebiets. Es sind kleinere Zusammenkünfte, die selten Entscheidungen treffen oder Entscheidungen vorbereiten. Im Projekt-Umfeld bedeutet dies etwa einen Unternehmensbedarf oder konkrete Handlungsalternativen mit Expertise zu besprechen. Die Expertise darf hierbei aus verschiedenen Abteilungen kommen, um sich ein umfassendes Bild der Situation zu machen.

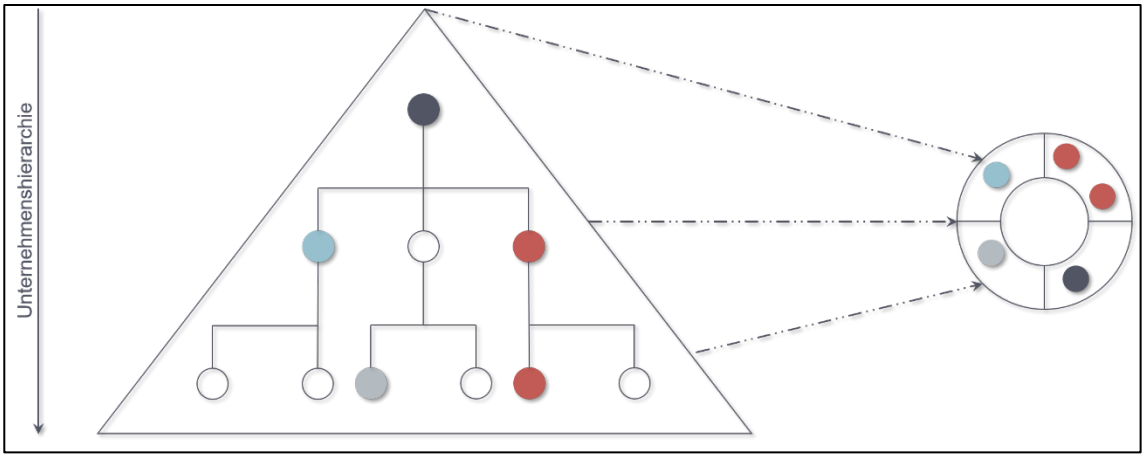


Abb. 22: Arbeitskreis

3.1.5 Der Arbeitskreis

Der Arbeitskreis oder auch Kommission oder Gremium ist eine

> *„zur Erfüllung einer bestimmten Aufgabe gebildete Gruppe von Expert(inn)en, eine beschlussfassende Körperschaft oder ein Ausschuss“.*

Solche Gremien sind also Entscheidungswerkzeuge bestehend aus verschiedenen Experten eines Fachgebiets. Es sind kleinere Zusammenkünfte, die selten Entscheidungen treffen oder Entscheidungen vorbereiten. Im Projekt-Umfeld bedeutet dies etwa einen Unternehmensbedarf oder konkrete Handlungsalternativen mit Expertise zu besprechen. Die Expertise darf hierbei aus verschiedenen Abteilungen kommen, um sich ein umfassendes Bild der Situation zu machen.

Eingliederung in die Unternehmensorganisation:

Arbeitskreise treffen sich losgelöst von der Unternehmensorganisation in kleineren Gruppen mit hierarchieübergreifenden Experten eines Fachgebietes.

Beispiele:

Gremien bei Datev eG: Der Vertreterrat

Mit Beirat und Vertreterrat hat DATEV freiwillig zwei weitere Gremien geschaffen, die dem Vorstand beratend zur Seite stehen. Der Vertreterrat der DATEV berät den Vorstand aus Anwendersicht. Er erörtert mit ihm die Wünsche, Anregungen und Beschwerden der Mitglieder und berät ihn auf dem Gebiet der Software, der Dienstleistungen und der Mitgliederbetreuung.

Arbeitskreis „Wertorientierte Führung in mittelständischen Unternehmen“ der Schmalenbach-Gesellschaft e.V.

Ein Ergebnis dieses Arbeitskreises sind 10 Thesen für mehr Effizienz in der Planung mittelständischer Unternehmen.

Ziel war es, Anregungen zur Weiterentwicklung der in mittelständischen Unternehmen bereits existierenden Planungssysteme hin zu stärkerer Wertorientierung zu geben, aber nicht ein vollständiges, neues, normatives Planungskonzept vorzulegen.

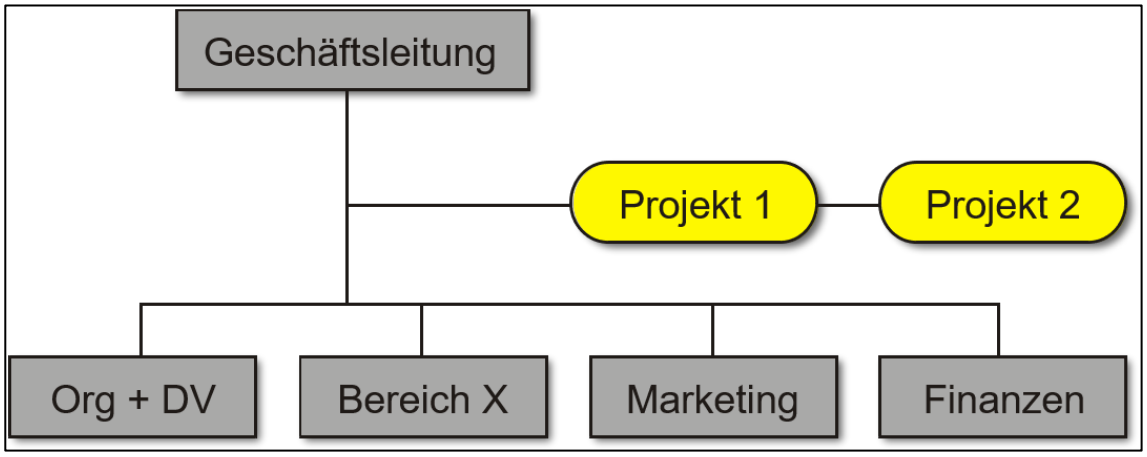


Abb. 23: Einfluss-Projekt-Organisation

3.1.6 Die Einfluss-Projekt-Organisation

- Projektleitung und Mitarbeiter bleiben in ihren Linienpositionen.
- Projektleitung und Mitarbeiter in Stabsstelle zusammengefasst.
- Projektleitung verfügt über keine Entscheidungs- und Weisungsbefugnis.
- Projektleitung verfolgt fachlich, terminlich, kostenmäßig den Projektverlauf.

- Projektleitung schlägt Linieninstanzen Maßnahmen vor.

3.1.7 Einfluss-Projekt-Organisation

In einer Einfluss-Projekt-Organisation werden sowohl der Projektleiter als auch die Mitarbeiter in einer Stabsstelle zusammengefasst. Gleichzeitig bleiben alle Beteiligten in Ihren Linienpositionen tätig. Die Entscheidungs- und Weisungsbefugnisse eines Projektleiters sind dadurch begrenzt. Er verfolgt lediglich den fachlichen, terminlichen und kostenmäßigen Projektverlauf und schlägt den Linieninstanzen Maßnahmen vor.

Eingliederung in die Unternehmensorganisation:

Die Projekt-Stabsstelle wird als Sekundärorganisation gebildet. Mitarbeiter haben somit eine zusätzliche Funktion zu ihren Routineaufgaben. Der Projektleiter kann hier lediglich Maßnahmen vorschlagen, aber nicht durchsetzen.

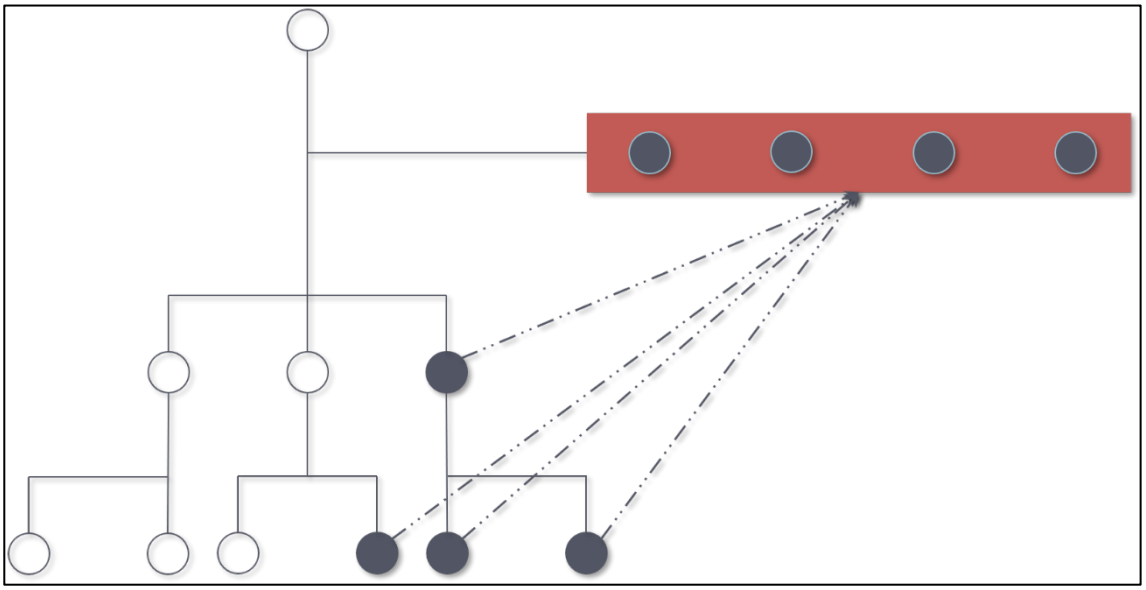


Abb. 24: Einfluss-Projekt-Organisation als Stabsstelle

Beispiel: Das PMO

Das Projekt-Management-Office (PMO)

Eine heute beliebte Stabsstelle ist das sogenannte Projekt-Management-Office (PMO). Das PMO leitet nicht selbst Projekte, aber stellt Projekten Tools zur Verfügung und nimmt Koordinierungsaufwand ab.

Das PMO ist vor allem in größeren Unternehmen und Beratungshäusern vertreten.

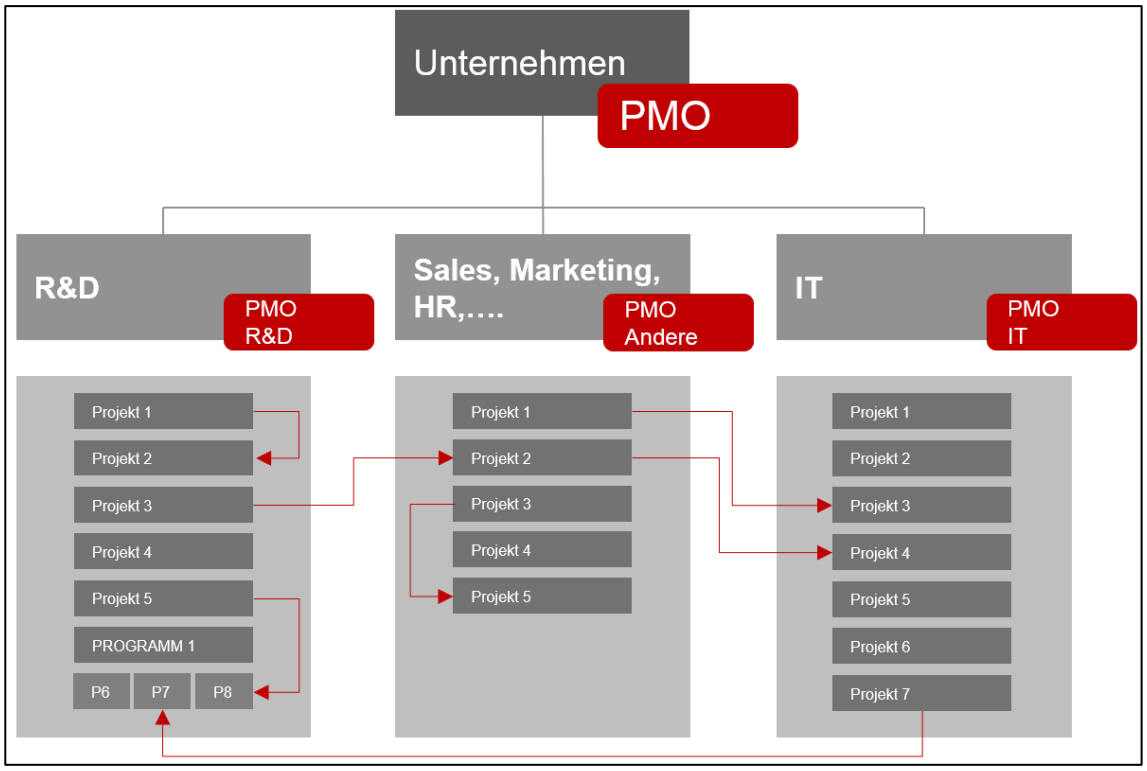


Abb. 25: Das PMO

3.1.8 Vor- und Nachteile EPO

Vorteile:

- Flexibilität bei Personaleinsatz, durch Mehrfacheinsatz
- Keine zusätzliche Organisationseinheit benötigt
- Projekterfahrung bleibt in der Fachabteilung

Nachteile:

- Niemand fühlt sich voll für das Projekt verantwortlich.
- Es kann kein Team entstehen.
- Lange Entscheidungswege
- Kommunikationsprobleme wegen Abteilungsgrenzen

3.1.9 Die „reine Projekt-Organisation“

- Projektleitung und Mitarbeiter nicht mehr auf ihren Linienpositionen
- Alle Mitarbeiter unter Leitung des Projektleitung
- Projektleitung besitzt alle formalen Kompetenzen
- Nur Projektleitung ist weisungsbefugt (Vorgesetzter auf Zeit)
- Mitarbeiter arbeiten ausschließlich für das Projekt

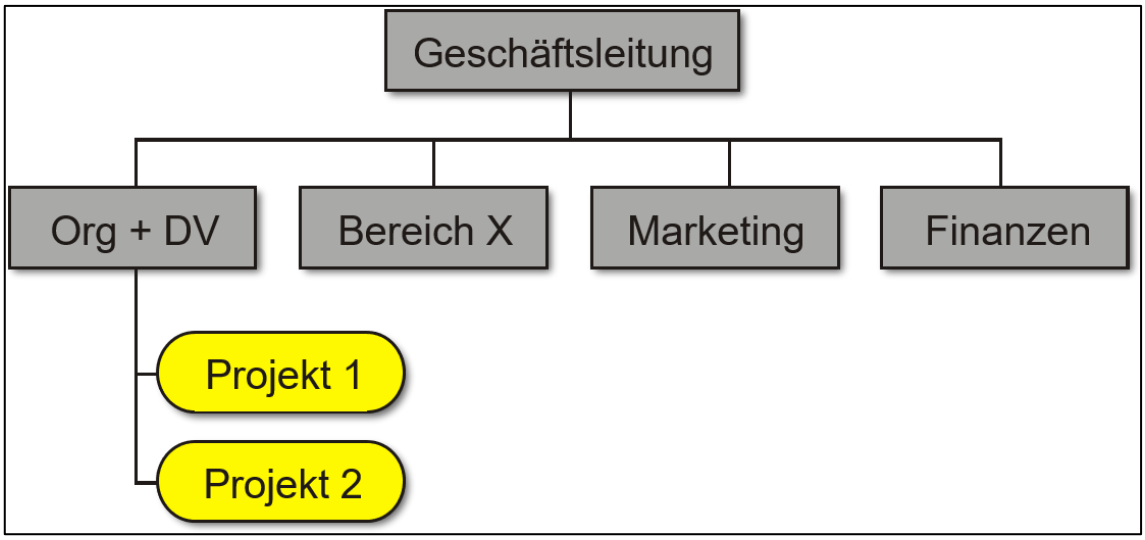


Abb. 26: Reine Projektorganisation

3.1.10 Die reine Projektorganisation

Die reine PO zeichnet sich dadurch aus, dass der Projektleiter und die Mitarbeiter nicht mehr in ihren Linienpositionen, sondern ausschließlich für das Projekt arbeiten.

Dabei stehen alle Mitarbeiter unter der Leitung des Projektleiters, der alle formalen Kompetenzen besitzt und als einziger weisungsbefugt ist. Der Projektleiter ist bis Ende des Projektes ein Vorgesetzter auf Zeit.

Während der Projektlaufzeit sind „reine Projekte“ also feste Organisationseinheiten.

Eingliederung in die Unternehmensorganisation:

Die Projekte sind in der reinen Projektorganisation häufig einer Abteilung zugeordnet. Schnittstellen zu anderen Abteilungen, die nicht an Projekten beteiligt sind, bestehen kaum.

Beispiel 1:

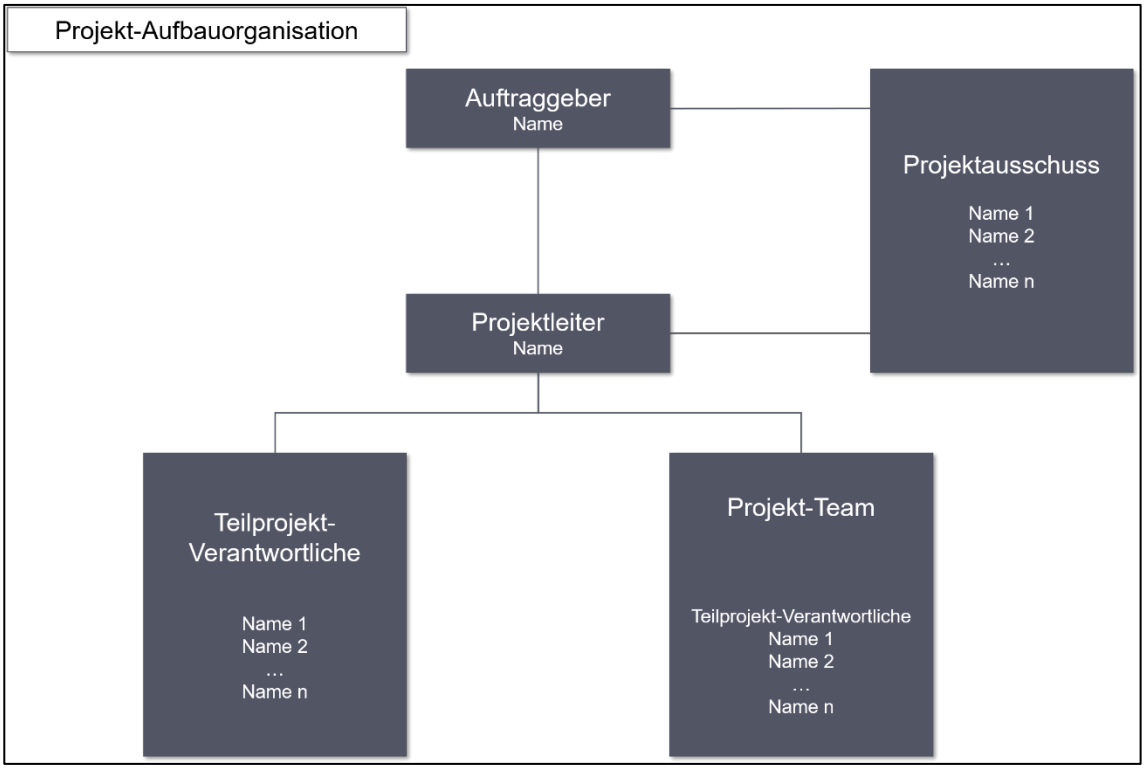


Abb. 27: Projektaufbau-Organisation: Beispiel

Beispiel 2:

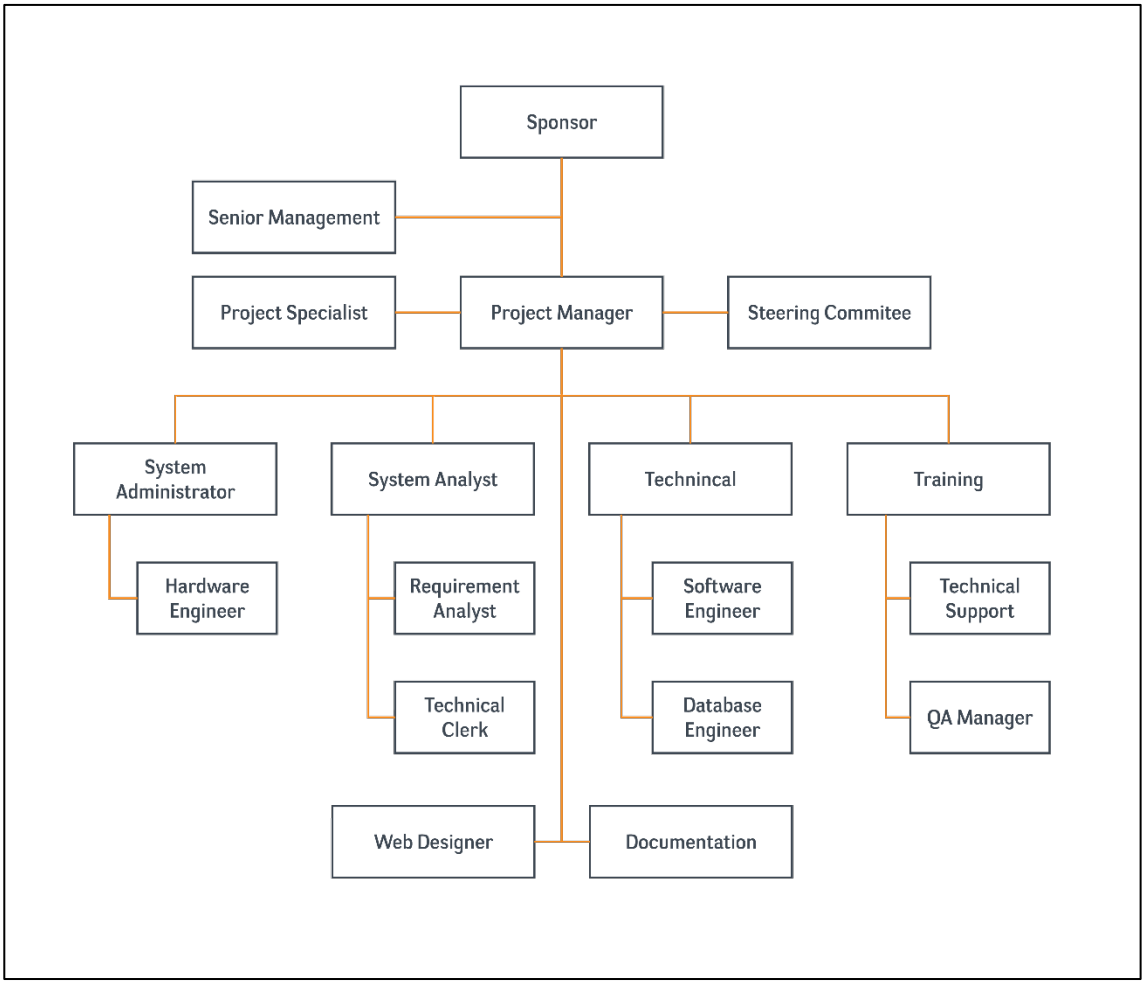


Abb. 28: Beispiel der reinen Projektorganisation

3.1.11 Vor- und Nachteile RPO

Vorteile:

- Kurze Entscheidungswege
- Schnelle Reaktionszeiten
- Starke Identifikation mit dem Projekt

Nachteile:

- Einbindung kurzzeitig benötigter Spezialisten problematisch
- Team-Besetzung problematisch (Abzug aus Fachabteilung)
- Team-Auflösung problematisch (Reintegration)

3.1.12 Die Matrix-Projektorganisation

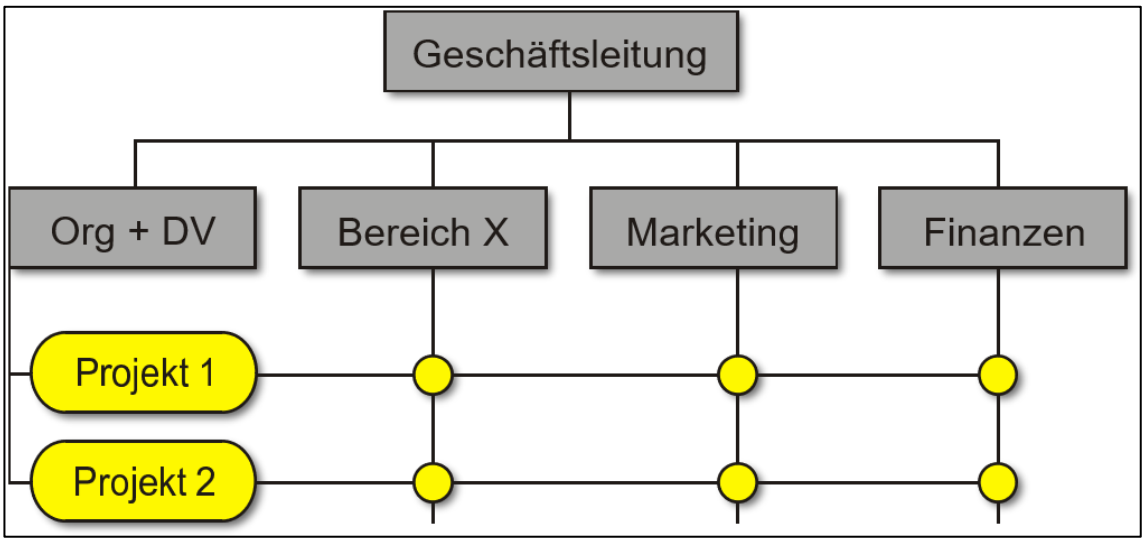


Abb. 29: Matrix-Projektorganisation

- Projektleitung und Mitarbeiter in Linien- und Projektpositionen
- Kompetenzaufteilung zwischen Projektleitung und Linienvorgesetzten
- Benötigt hochentwickeltes Führungsverständnis
- Mitarbeiter arbeiten für das Projekt und für die Linie.
- Am häufigsten anzutreffende Projektorganisationsform

3.1.13 Matrix-Projektorganisation

Bei der Matrix-Projektorganisation sind der Projektleiter und die Mitarbeitergleichzeitig in ihren Linien- und Projektpositionen tätig. Dabei arbeiten sie für das Projekt und gleichzeitig für ihre Linienposition zu bestimmten Teilen. Hierbei kommt es zur Kompetenzaufteilung zwischen dem Projektleiter und dem Linienvorgesetzten. Dazu ist ein hochentwickeltes Führungsverständnis notwendig. Dies ist die am häufigste anzutreffende Projektorganisationsform.

Eingliederung in die Unternehmensorganisation:

Die Primärorganisation („Linie“) wird mit zusätzlichen projektbezogenen Weisungsrechten überlagert. Der Projektleiter verfügt über informale und formale Befugnisse über die beteiligten Abteilungen hinweg.

Beispiel der Matrix-Projektorganisation:

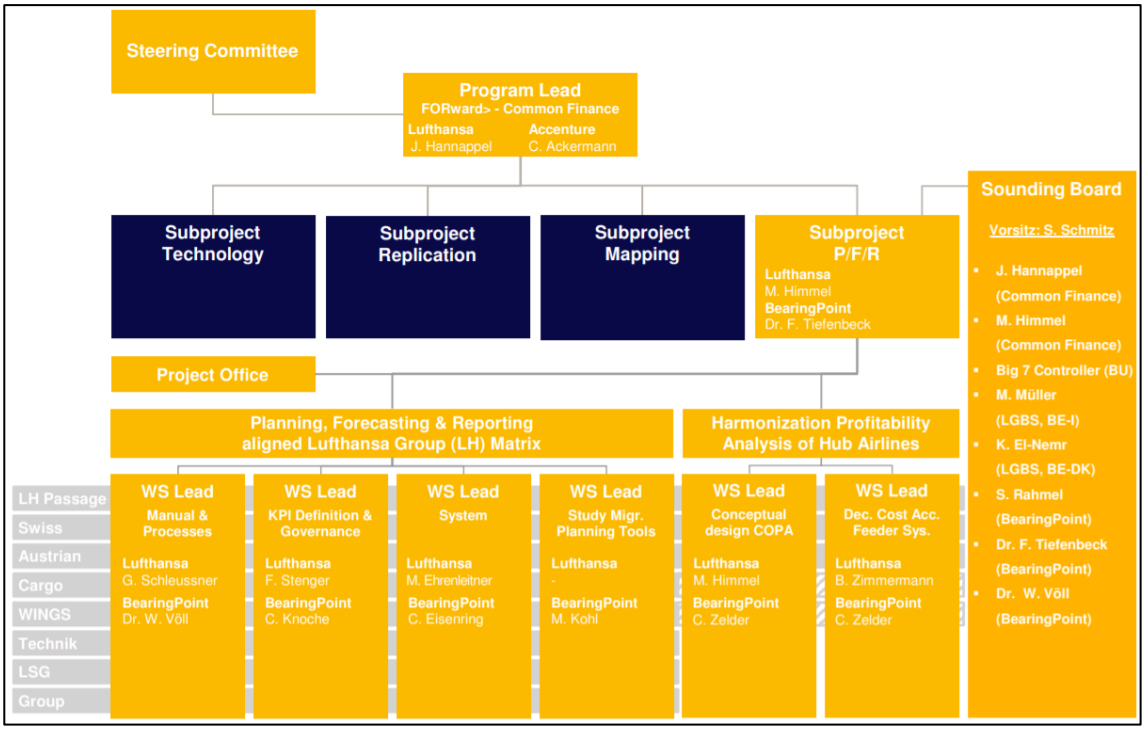


Abb. 30: Beispiel der Matrix-Projektorganisation bei der Lufthansa

3.1.14 Vor- und Nachteile MPO

Vorteile:

- Flexibler Personaleinsatz
- Fachabteilung muss sich durch die Aufbauorganisation in die Projektgestaltung miteinbringen
- Kontinuität der Mitarbeiter-Laufbahn gewahrt
- Interdisziplinäre Betrachtungsweise

Nachteile:

- Interessenkonflikte institutionalisiert
- Kompetenzkonflikte zwischen der Projektleitung und Linienvorgesetzte
- Verunsicherung der Mitarbeiter durch zwei Vorgesetzte
- Gefahr des Projekt-Boykotts durch Fachabteilung
- Unter Umständen lange Entscheidungszeiten

3.1.15 Einfluss- und Kontrollspanne

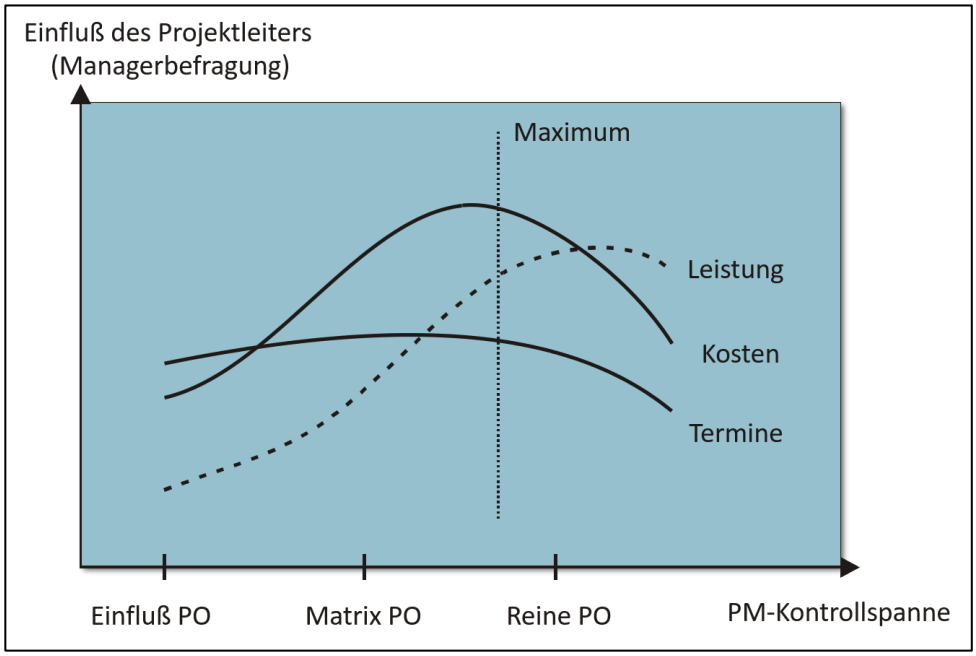


Abb. 31: Einfluss- und Kontrollspanne des Projektleiters

Durch die verschiedenen Projektorganisationsformen wird ersichtlich, dass der Einfluss auf das Projektgeschehen durch die Projektleitung von der Organisationsform abhängt. Somit variiert auch der Einfluss auf die drei Projekt-Messgrößen Zeit, Kosten und Qualität. Die Projektorganisation beeinflusst diese Größen in unterschiedlicher Weise. Eine reine Projektorganisation kann beispielsweise die bestmögliche Qualität liefern, da das Team dediziert nur für das Projekt arbeitet und keine Tätigkeiten „in der Linie“ vollbringen muss. Die Matrix-Organisation hingegen nutzt die Ressource „Personal“ optimal aus und verursacht somit geringere Kosten. Daraus ergeben sich Kriterien für die optimale Wahl der Projektorganisation. Je nachdem welche Einflussfaktoren eine Rolle in einem Projekt spielen, sollte eine bestimmte Organisationsform gewählt werden.

3.1.16 Kriterien für die Projekt-Organisation

Kriterien des IT-Projektes	Einfluss-Projekt-organisation	Reine Projekt-organisation	Matrix-Projekt-organisation
Bedeutung für das Unternehmen	gering	sehr groß	groß
Umfang	gering	sehr groß	groß
Unsicherheit (Ziele)	gering	sehr groß	groß
Technologie	Standard	neu	kompliziert
Zeitdruck	gering	hoch	mittel
Dauer	kurz	lang	mittel
Komplexität	gering	hoch	mittel
Bedürfnis zentraler Steuerung	mittel	sehr groß	groß
Mitarbeitereinsatz	nebenamtlich	vollamtlich	Teilzeit
Projektleiter Persönlichkeit	wenig relevant	sehr fähig	qualifiziert

Abb. 32: Kriterien für verschiedene Projektorganisationsformen

3.2 Führungskonzept und Projekt-Team

3.2.1 Einleitung in personenbezogenes Projektmanagement

Im vorigen Kapitel eins sind Sie auf die formale Aufbauorganisation eines Projektes eingegangen. Das Kapitel zwei „Führungskonzepte und Projekt-Team“ befasst sich mit den Mitarbeitern und dem Führungspersonal in einem Projekt.

In einem Projekt arbeiten verschiedene Menschen miteinander und füreinander: zum einen das Team der Projektmitarbeiter und Projektleiter, welche das Projekt ausführen. Hierbei muss auf eine optimale Team-Zusammensetzung geachtet werden.

Zum anderen können auch Projektbeteiligte involviert sein, die ein berechtigtes Interesse an dem Projekt und dessen Umsetzung haben. Dies kann beispielsweise der Auftraggeber sein, der das Projekt durch ein sogenanntes „Steering Committee“ koordiniert, überwacht und durch diese Instanz auch Entscheidungen trifft.

3.2.2 Ein gemeinsames Rollenverständnis

In einem Projekt gibt es i. d. R. einen Auftraggeber und einen Auftragnehmer, der das Projekt ausführt. Ausführende sind z. B. das Projekt-Team oder auch ein einzelner Entwickler. In einem Bauprojekt ist der Auftraggeber der Bauherr, der Auftragnehmer der Architekt. Beide Rollen haben spezifische Verantwortungen, die sie erfüllen müssen. Zu Beginn eines jeden Projektes ist es für die erfolgreiche Durchführung wichtig, die Verantwortungen der zwei verschiedenen Rollen zu klären und festzuhalten.

Verantwortung des „Bauherrn“ (Auftraggeber, Benutzer, Anwender)

- Entscheidung über Einsatz und Nutzen
- Vorgabe der fachlichen Systemanforderungen
- Nutzenbegründung und Abnahme des Systems

Verantwortung des „Architekten“ (Entwickler, Projektteam)

- Beratung der Anwender über Nutzenpotential
- Durchführung von Situationsanalysen
- Unterstützung der Anwender bei der Erarbeitung der fachlichen Anforderungen
- Beratung der Anwender zu Konsequenzen von Vorgaben und Änderungen
- Auswahl, Entwicklung und Implementierung von Systemen

3.2.3 Beispiel: Verantwortungsmatrix

Projekt-phase \ Projekt-partner	Benutzer-Seite				Entwickler-Seite		
	Auftrag-geber	Betrof-fene	Ent-scheider	Projekt-leiter	Fach-berater	System-analytiker	System-entwickler
1. Situationsstudie	P, A	M	K, E	P	M	---	---
2. Fachkonzeption							
2.1 Anforderungs-ermittlung	K, E	M	I	P	M	A	---
2.2 Anforderungs-spezifizierung							
3. Systemkonzeption							
3.1 Entwurf des Programmsystems							
3.2 Entwurf des Datensystems							
3.3 Entwurf des Hard-ware-Systems							
4. Systemrealisierung							
4.1 Programmierung							
4.2 Integration							
5. Systemanwendung							
5.1 Installation							
5.2 Wartung							

P = Planung
E = Entscheidung
A = Ausführung
K = Kontrolle
M = Mitwirkung
I = Information

Abb. 33: Verantwortungsmatrix in Projekten

3.2.4 Verteilung der Projektverantwortung

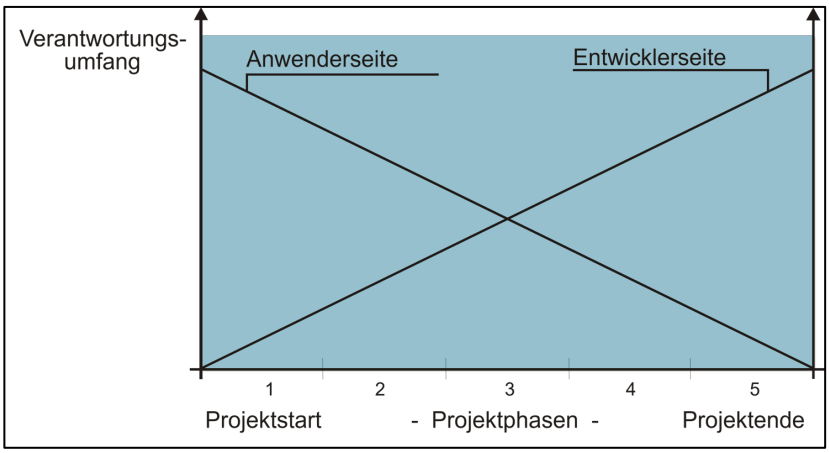


Abb. 34: Projektverantwortung über die Projektphasen

Es ist deutlich erkennbar, dass gerade zu Projektstart die Anwenderseite die größere Verantwortung trägt. Der Bauherr muss hier beispielsweise dem Architekten verdeutlichen, wie sein Traumhaus aussehen soll. Welchen Zweck soll das Haus erfüllen? Wird eine Familie einziehen oder ein Senioren-Paar? Überträgt man dies auf ein Software-Entwicklungsprojekt muss der spätere Software-Nutzer dem Projekt-Team verdeutlichen, welchem Zweck die Software dienen soll.

3.2.5 Projektorganisation – funktional

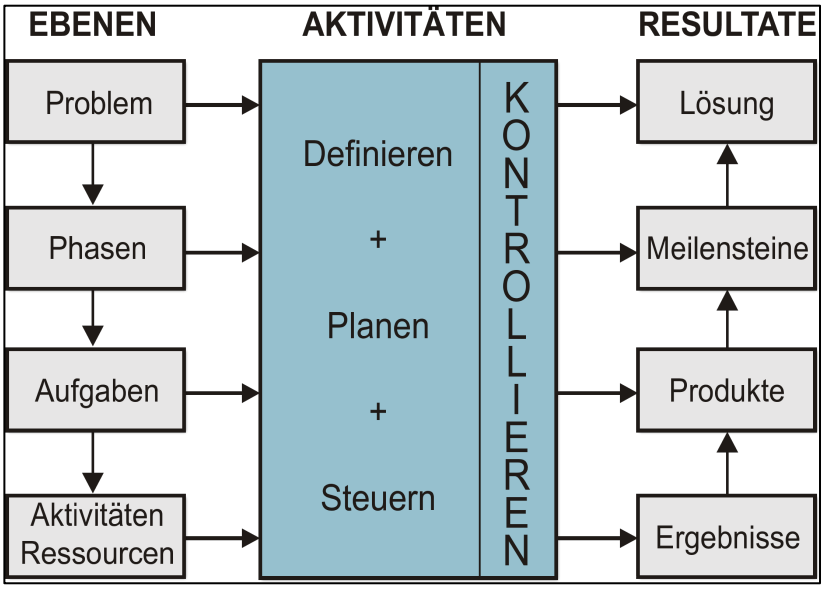


Abb. 35: Funktionale Projektorganisation

3.2.6 Koordinierung und Lenkung im Unternehmensumfeld

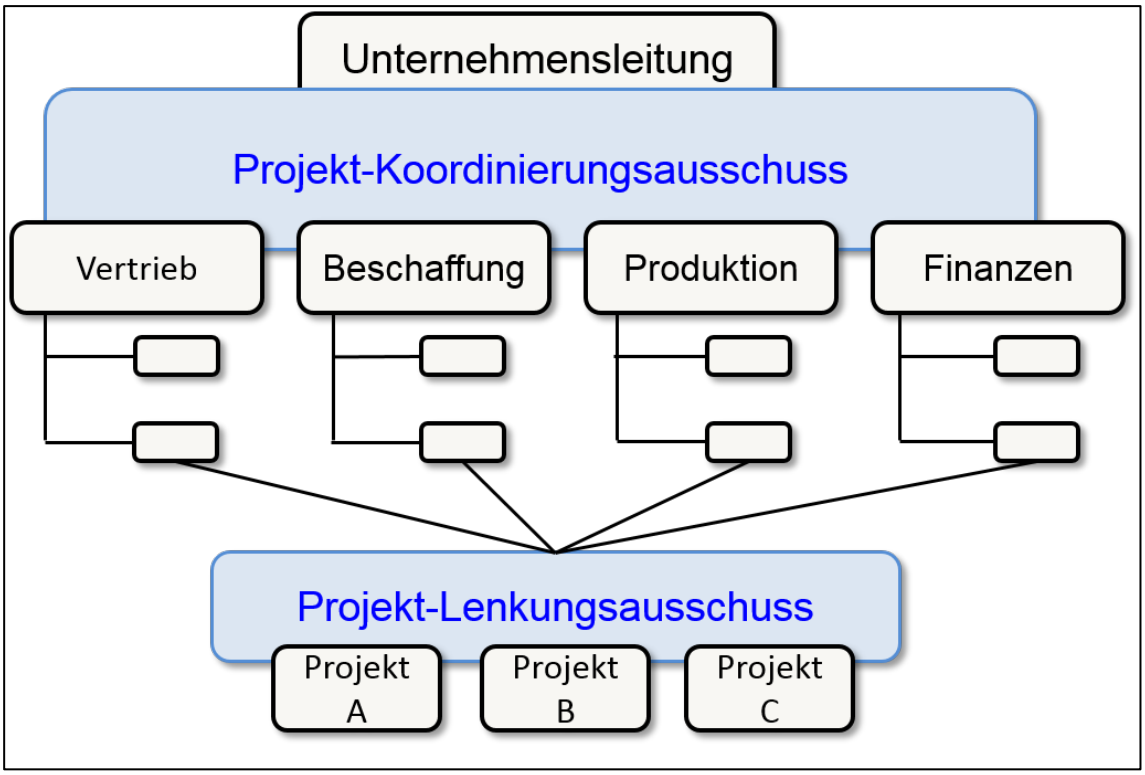


Abb. 36: Koordinierungs- und Lenkungsausschuss

Jedes Projekt ist in ein Unternehmensumfeld eingebettet. In diesem Umfeld können zum Beispiel Koordinierungsgremien und Lenkungsausschüsse gebildet werden.

Der Projekt-**Koordinierungsausschuss** entscheidet in der Praxis über einen Projektantrag. Dieser Ausschuss ist mit Personen aus der Geschäftsleitung und weiteren Vertretern des Managements besetzt.

Der Projekt-**Lenkungsausschuss** ist der verbindende Ausschuss zwischen der Projekt- und Linienorganisation. Dieser Ausschuss soll dafür sorgen, dass die Interessen aller Projektbeteiligten in geeigneter Weise vertreten werden. Leiter dieses Ausschusses ist meist der Auftraggeber des IT-Projektes.

3.2.7 Das Projekt-Team in seiner Umwelt

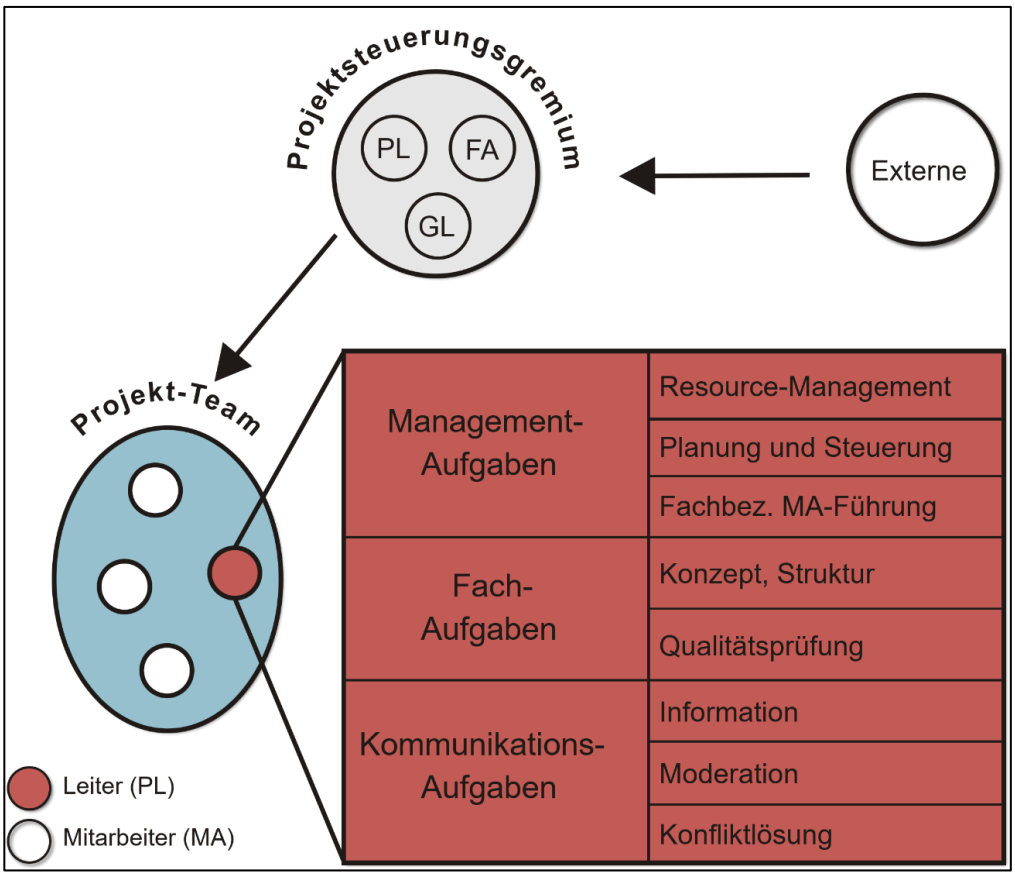


Abb. 37: Personale Umfeld- und Aufgabenbetrachtung

Ein Projekt muss in seiner Umwelt betrachtet werden. Verschiedene unternehmensinterne Akteure und Richtlinien können Einfluss nehmen.

Das eigentliche Projekt-Team besteht aus der Projektleitung und verschiedenen fachspezifischen Mitarbeitern. Die Projektleitung muss das Umfeld beobachten und einbeziehen. Die Kommunikation mit den Projekt-Mitarbeitern und anderen Projektinteressenten sowie die Planung, Steuerung und Kontrolle des Projektes gehören zum Aufgabenbereich des Projektleiters.

Die Projektleitung muss dementsprechend nicht nur fachliche Expertise besitzen, sondern auch soziale und emotionale Intelligenz mitbringen.

3.2.8 Team-Zusammensetzung

Bei der Zusammensetzung eines Projekt-Teams darf die Projektleitung nicht nur auf fachliche Expertise achten. Im Idealfall sollte das Projekt-Team auch aus unterschiedlichen Persönlichkeiten bestehen. Dies ist gerade in der Generierung von Lösungsansätzen von großem Vorteil. Allerdings sollte der Projektleitung auch klar sein, dass verschiedene

Persönlichkeiten auch miteinander im Konflikt stehen können. Die Projektleitung sollte daher auch in der Lage sein, auch interne Konflikte zwischen Team-Mitgliedern zu lösen.

Untenstehende Grafik ist der Hermann-Miller Denkstil-Richtung nachempfunden. Diese unterteilt menschliche Denkweisen in vier Kategorien. Sich diagonal gegenüberstehende Denkstile haben der Erfahrung nach Konfliktpotenzial.

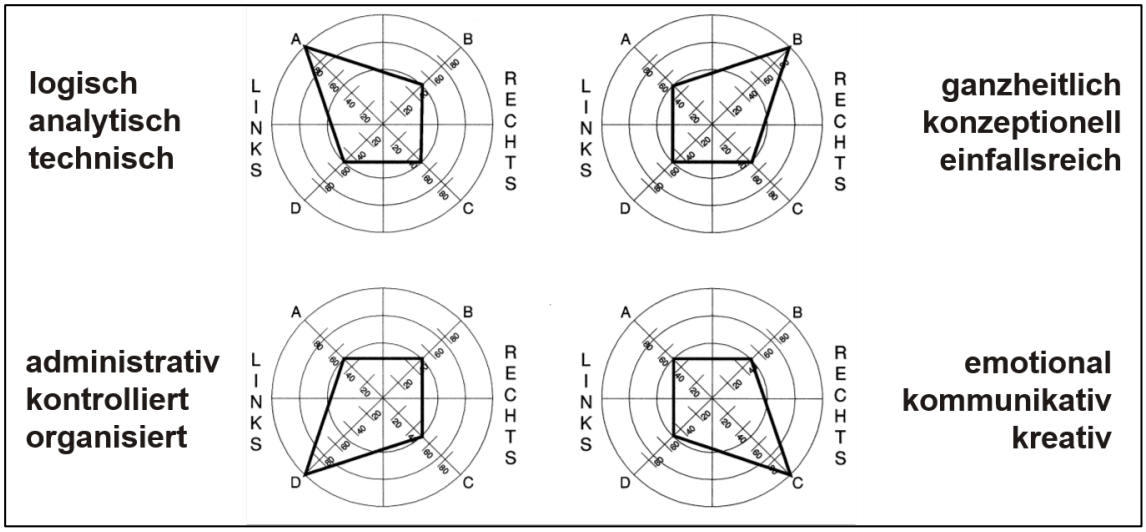


Abb. 38: Beispiel verschiedener Denkstile nach Hermann-Miller

3.3 Abschlusstest – WBT 03

3.3.1 Abschlusstest

Bitte beantworten Sie die folgenden Fragen durch das Ankreuzen der korrekten Antworten (Tab. 4). Bei einigen Fragen können auch mehrere Antworten richtig sein.

Nr.	Frage	Richtig	Falsch
1	Die Projektorganisation stellt keinen Einflussfaktor bei der Aufsetzung eines Projektes dar.		
2	Die Einfluss-Projektorganisation unterstellt der Projektleitung dediziertes Personal mit voller Entscheidungsbefugnis.		
3	Welche Projekt-Organisationsformen gibt es?		
	Einfluss-Projektorganisation		
	Reine Projektorganisation		
	Matrix-Projektorganisation		
	Arbeiterkreis		

4	Die Projektleitung muss keine besonderen Bedürfnisse der Mitarbeiter beachten.		
5	Die Projektleitung muss sich im Projekt durch Fachkenntnisse einbringen.		

Tab. 4: Abschlusstest – WBT03

3.4 Typische Aufgabenstellungen

Typische Aufgabenstellungen – IT-Projekte: Organisation und Personal

Zur Bearbeitung dieser Aufgabenstellungen beachten Sie bitte: Verlangt ist eine fachlich zutreffende, inhaltlich nachvollziehbare und kausal zusammenhängende Erörterung aus vollständigen Sätzen in lesbarer Handschrift. Für jede Aufgabe: Maximal zwei Seiten Text.

Aufgabe 1:

> Welche sind die Grundformen der Projektorganisation? Beschreiben sie diese Organisationsformen und nennen sie Vor- und Nachteile.

Aufgabe 2:

> Beschreiben Sie die „Einfluss-Projektorganisation“ und erläutern Sie deren Vor- und Nachteile.

Aufgabe 3:

> Beschreiben Sie die „Reine Projektorganisation“ und erläutern Sie deren Vor- und Nachteile.

Aufgabe 4:

> Beschreiben Sie die „Matrix-Projektorganisation“ und erläutern Sie deren Vor- und Nachteile.

Aufgabe 5:

> Erläutern Sie das gemeinsame Rollenverständnis von „Bauherr“ (Auftraggeber) und „Architekt“ (Software-Entwickler) in IT-Entwicklungsprojekten.

4 Projektmanagement: Standards und Software

4.1 Projektmanagement-Standards

4.1.1 Das Hauptproblem im Projektmanagement

Ein Hauptproblem

Bei IT-Projekten handelt es sich um ein relativ junges Anwendungsgebiet, welches noch erforscht werden muss. Es kann noch keine verlässlichen Zahlen, Daten und Fakten liefern. Ein Problem, was sich daraus ergibt, ist, dass die Planung im Projektmanagement auf Schätzungen beruht. Schätzungen beruhen wiederum auf Erfahrungswerten einzelner Personen.

Die Folgen

Bei der realen Projektdurchführung treten immer wieder Planabweichungen auf wie z. B.:

- Personelle Veränderungen ziehen Budget- oder Terminänderungen nach sich, sollte ein Mitarbeiter für längere Zeit ausfallen oder kündigen.
- Lieferverzögerungen lassen einen Vorgang nicht wie geplant starten.
- Preisschwankungen bringen die Kostenstruktur durcheinander.

Schnelle und adäquate Reaktionen auf diese Abweichungen sind erforderlich. Denn nicht beachtete Planabweichungen können für das Projekt weitreichende Folgen haben.

4.1.2 Projektmanagement: Standards und Software – Die Lösung?

Die Lösung

PM-Software kann gerade zur Planung hilfreich sein, indem sie einen hohen Detailgrad dennoch transparent abbilden kann. Darüber hinaus hilft sie:

- die Planung exakt durchzukalkulieren,
- Alternativen zu suchen und
- das Projekt permanent zu verfolgen.

Standards hingegen sind Werkzeuge, die einen Planungsrahmen vorgeben. Im Projektmanagement werden hier die „Best Practices“ von verschiedenen Organisationen zusammengetragen. Dabei können sich Standards und Software als Projektmanagement- Toolbox ergänzen. Bevor Sie sich genauer mit PM-Software beschäftigen, schauen Sie sich zunächst einige wichtige Standards im Projektmanagement an.

4.1.3 Was ist ein Standard?

„Ein Standard ist ein Dokument, das durch eine Behörde, eine Gewohnheit oder allgemeinen Konsens als Modell oder Beispiel eingeführt ist. [...]

Der Standard kennzeichnet diejenigen Prozesse, die in den meisten Fällen für die meisten Projekte als gute Praktiken gelten und benennt zudem die Eingangs- und Ausgangswerte, die in der Regel mit diesen Prozessen in Zusammenhang gebracht werden.“

4.1.4 PMBOK: Project Management Body of Knowledge

Das PMBOK stammt vom PMI (Project Management Institute) aus den USA. Es ist eines der beschriebenen „Best Practices“ für das Projektmanagement. Dabei unterteilt es ein Projekt in Phasen und innerhalb der Phasen beschreibt es Prozesse.

Die Phasen werden unterteilt in: Initiierung, Planung, Ausführung, Abschluss und Überwachung.

Wie die unten abgebildete Grafik aufzeigt, wurden die „Best Practices“ mit jeder Edition umfangreicher. Die Prozesse haben sich jedoch nicht wesentlich verändert. Dies deutet darauf hin, dass die vorhandenen Phasen lediglich detaillierter beschrieben werden konnten.

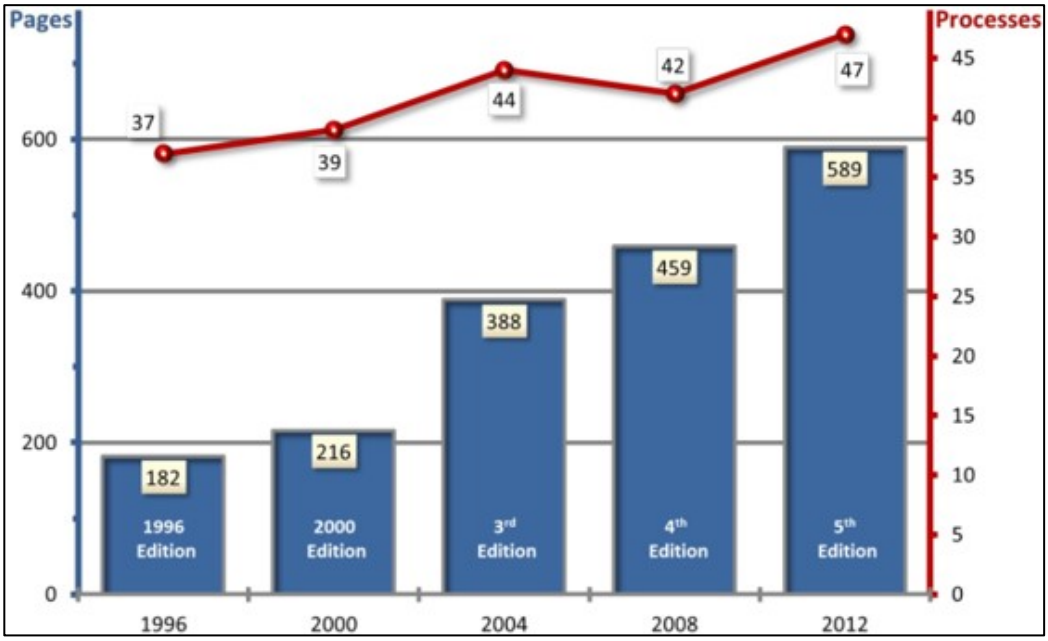


Abb. 39: Seitenanzahl PMBoK

4.1.5 PMBOK: Inhalt und Struktur

Inhalt

- Umfangreiche Sammlung von Projektmanagementmethoden
- Sammlung von Vorgehensweisen, die sich in der Praxis bewährt haben (engl. „best practice“).
- PMBOK ist prozessorientiert, Durchführung eines Projekts erfolgt anhand von Prozessen.
- Beschreibt für jeden der 49 Einzelprozesse In- und Outputs, Techniken, Werkzeuge und Verfahren des Projektmanagements.

Struktur

PMBOK-Guide besteht aus drei Teilen:

- PM-Rahmen (enthält eine Einführung in die Nutzung des Guides und Begriffsdefinitionen)
- PM-Prozessgruppen (teilt die 49 Einzelprozesse in die Prozessphasen ein und ordnet sie einem Wissensgebiet zu)
- und PM-Wissensgebiete (bilden mit 10 Kapiteln den Schwerpunkt des Guides).

4.1.6 PRINCE2: Projects in Controlled Environments

PRINCE2 entstand aus einer britischen Regierungsinitiative, dem „Office of Government Commerce“ und wird heute von AXELOS weiterentwickelt. PRINCE2 gilt als eine der anpassungsfähigsten Projektmanagement-Standards und wird wie das PMBOK aktualisiert.

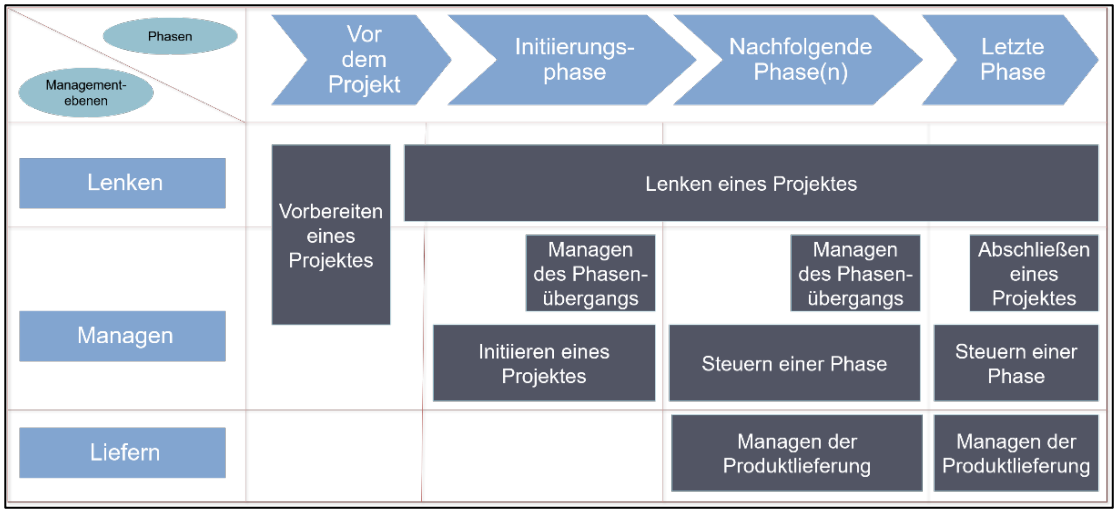


Abb. 40: PRINCE2 Managementebenen und Phasen

PRINCE2 beschreibt 7 Grundprinzipien, 7 Themen und 7 Prozesse zur Anpassung des Projekts an dessen Umgebung und damit dessen Einflussfaktoren.

PRINCE2 beschreibt insgesamt 9 Rollen und ihre Verantwortlichkeiten im Projekt. Drei Gruppen von Stakeholdern werden jedoch besonders herausgehoben:

1. **Executive:** Der Auftraggeber ist entscheidende Autorität innerhalb des Projekts und Eigentümer des Business Cases.
2. **Senior User:** Der Benutzervertreter vertritt die Interessen der späteren Anwender des Projektergebnisses.
3. **Senior Supplier:** Der Lieferantenvertreter vertritt die Interessen der Ersteller der Projektprodukte.

Die Themen umfassen:

- Business Case
- Organisation
- Qualität
- Pläne
- Risiken
- Änderungen
- Fortschritt

Die Prozesse umfassen:

- Vorbereiten eines Projekts
- Initiieren eines Projekts
- Lenken eines Projekts
- Steuern einer Phase
- Managen der Produktlieferung
- Managen eines Phasenübergangs
- Abschließen eines Projekts

4.1.7 PRINCE2: Zusammensetzung und Hauptmerkmale

PRINCE2 (ein Akronym für **PR**ojects **IN C**ontrolled **E**nvironments) ist eine de-facto-projektbasierte Methode für das effektive Projektmanagement“.

Zusammensetzung aus:

- Dem PRINCE2 manual (Managing Successful Projects with PRINCE2 – 2017 edition)
- Einem Zertifikations-Schema mit mehreren Levels
- Einer AXELOS-Mitgliedschaft, die zusätzlichen Support bietet

Hauptmerkmale:

- Fokus auf gewerbliche Ausrichtung
- Reduzierung der Komplexität durch Aufteilung des Projekts in kontrollierbare Stufen
- Hohe Flexibilität macht PRINCE2 anpassungsfähig an einzelne Projekte.
- Produktbasierter Planungsansatz
- Definierte Organisationsstruktur für das Projektmanagement-Team

4.1.8 ICB: Individual Competence Baseline

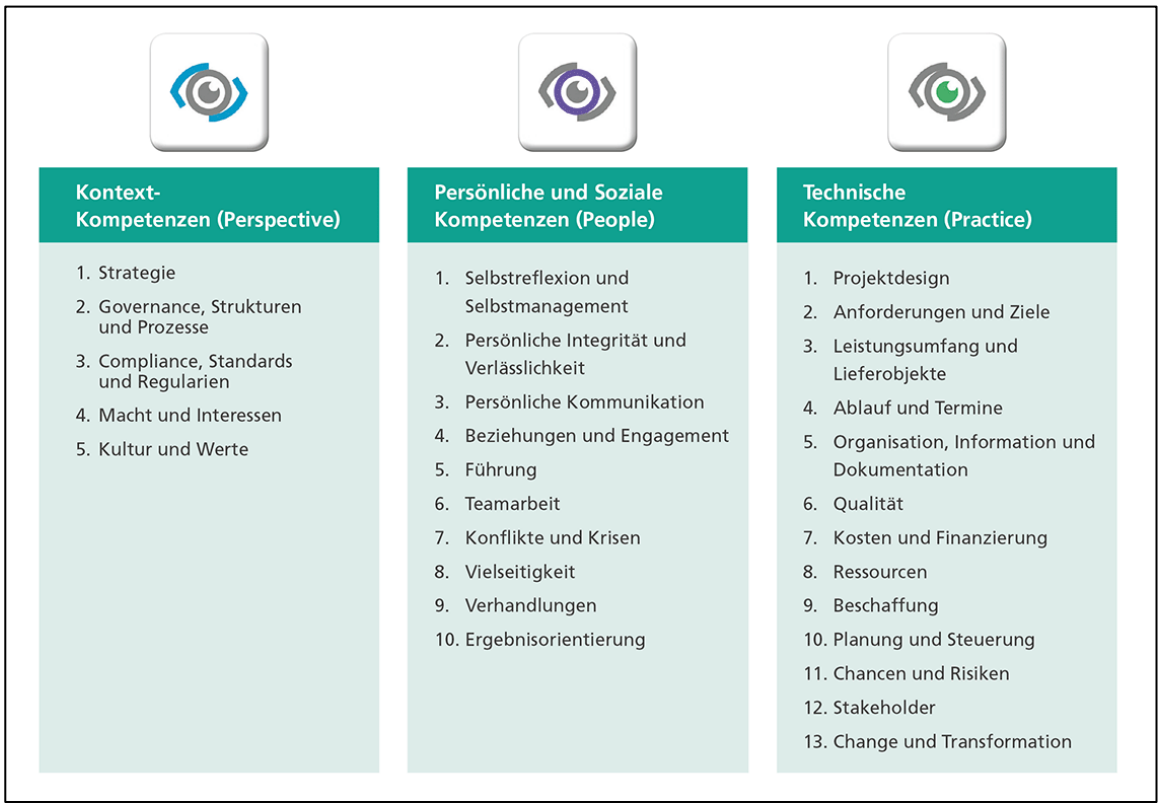


Abb. 41: ICP Kompetenzen

Entstehung:

- Entwickelt von IPMA: Ein internationaler Dachverband für Projektmanagement-verbände
- In Deutschland vertreten durch GPM
- Bis 2005: IPMA Competence Baseline (ICB 3)
- Ab 2015: Individual Competence Baseline (ICB 4)

Hauptmerkmale:

- Verfolgt nicht den klassischen PM-Ansatz (PM-Techniken), sondern konzentriert sich auf den Menschen im Projekt (Kompetenzen).
- Drei Kompetenzfelder bilden das Gerüst des ICB.
- Werden symbolisch zum „Eye of Competence“ zusammengeführt.

4.1.9 IPM, AXELOS, IPMA – Standards im Vergleich

Kriterium	PMBOK	PRINCE2	ICB
Struktur	• Definiert Prozesse in einer Matrix von Prozessgruppen und Wissensgebieten. • Beschreibt für jeden Prozess Input, Output, Werkzeuge und Verfahren	• Bietet eine übergeordnete, sofort umzusetzende Rahmenstruktur. • Vorgehensmodell mit 1 Projektumfeld, 7 Grundprinzipien, 7 Themen, 7 Prozessen	• Definiert Kompetenzelemente in drei Kompetenzfeldern
Schwerpunkt, Orientierung	• Prozessorientiert • Fokus auf Vereinheitlichung von Prozessen und Terminologie	• Phasen- und prozessorientiert; achtet besonders auf Führungsstrukturen und Prozesse.	• Kompetenzorientiert • Ziel ist eine ganzheitlichere Qualifikation (inkl. Soft Skills)
Besonderheiten	• Standardisierte Vorgehensweise • Gilt als internationaler Marktführer • Setzt englische Sprachkenntnisse voraus	• Produktbasierte Planung • Lässt sich gut mit anderen Standards kombinieren	• Branchenspezifische oder länderspezifische Vorgehensmodell • Länderorganisationen übersetzen das Regelwerk.
Verbreitung	• Weltweit dominierend • Über 650.000 Zertifikate weltweit (Stand 2017)	• Beliebt in Großbritannien, Niederlanden; steigende Verbreitung in Europa • Rund 950.000 Zertifikate weltweit (Stand 2012)	• Weltweit vertreten - Über 170.000 Zertifikate weltweit (Stand 2012)
Branchen	• Branchenübergreifend	• Ursprünglich Vorgehensmodell für IT-Projekte • Seit 2009 branchenübergreifend	• Branchenübergreifend

Abb. 42: Vergleich der drei größten Standards

4.1.10 Weitere Projektmanagement-Standards

Es existieren eine Vielzahl von anderen Projektmanagement-Standards z. B.:

- IPMA – International Project Management Association – Verband von über 40 Projektmanagement-Vereinigungen (z. B. GPM in Deutschland)
- HERMES: In der Schweiz seit 1986 für alle IT-Projekte des Bundes verbindlich; aktuell Version 5.1, seit 2015 für alle Projekte des Bundes
- ISO 21500 „Leitfaden zum Projektmanagement“: seit Februar 2016 unter DIN ISO 21500:2016-02 als deutsche Norm akzeptiert

4.2 Projektmanagement-Software

4.2.1 Microsoft Project

In der WBT-Kategorie „IT-Projektmanagement" des E-Campus Wirtschaftsinformatik können Sie einen vertiefenden Einblick zum Thema Projektmanagement-Software erhalten. Sowohl „MS Projects" als auch die frei zugängliche Software-Lösung „ProjectLibre" werden genauer dargestellt.

Im Folgenden schauen Sie sich beispielhaft Standard-Aufgaben mit einer PM-Software an. Sie verwenden iScitor, welches grafisch überholt erscheint, aber die Sachverhalte gut darstellt.

4.2.2 Typisches Vorgehen mit einer PM-Software

Sie sehen die typischen Schritte, die Sie in einer PM-Software vor und während des Projekts durchführen. Dabei wird die Planung des Projekts nicht nur im Zeitverlauf detaillierter, sondern auch in der Projektmanagement-Software. Die logische Konsequenz ist: Je weiter fortgeschritten das Projekt, desto detaillierter werden Pläne und Modelle in der PM-Software. Im Folgenden werden Sie die Schritte einzeln betrachten.

Abb. 43: Ausführungsschritte in einer PM-Software

4.2.3 1. Projekt neu anlegen

Um Projektmanagement-Software zu nutzen, wird zunächst ein neues Projekt angelegt. Sie legen in unserem Beispiel das Projekt „Hausbau" an. Jedes Projekt wird in einer Datei gespeichert.

4.2.4 2. Vorgänge erfassen

Nun müssen die nötigen Vorgänge zum Hausbau gesammelt werden. Es müssen beispielsweise Wände hochgezogen werden und das Dach errichtet und abgedichtet werden. Die so entstehende Liste sollte vollständig sein, muss aber noch keine richtige Reihenfolge haben.

Sind die Vorgänge vollständig erfasst, können sie mit der jeweiligen Vorgangsdauer versehen werden.

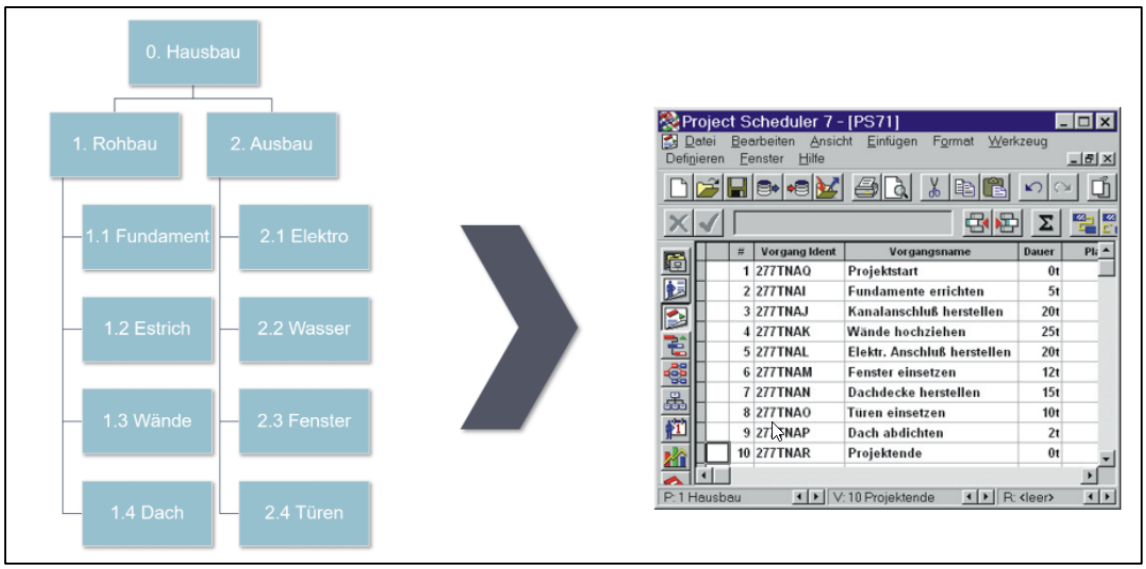


Abb. 44: iScitor Projektvorgänge erfassen

4.2.5 Vorgangsliste Hausbau

In der nebenstehenden Tabelle können Sie sehen, welche Vorgänge für das Projekt „Hausbau“ benötigt werden.

Zugeordnet zu den Vorgängen finden Sie in der nächsten Spalte die Vorgänger, also die Vorgänge, die abgeschlossen sein müssen, bevor der aktuelle Vorgang gestartet werden kann.

In der vierten Spalte sind die Dauern für den Vorgang verzeichnet. Dies beschreibt die Zeit, die für den Vorgang voraussichtlich benötigt wird. Die Dauer basiert auf Schätzwerten.

In der letzten Spalte werden Nachfolger i. d. R. durch die Projektmanagement- Software automatisch generiert.

Nr = Bez..	Beschreibung	Direkte Vorgänger	Dauer	Direkte Nachfolger
1	**Start (Projektanfang)**	---	0	2, 3
2	Fundamente errichten	1	05	4, 5
3	Kanalisationsanschluß herstellen	1	20	6, 7, 8
4	Wände hochziehen	2	25	6, 7, 8
5	Elektr. Hauptanschluß herstellen	2	20	7
6	Fenster einsetzen	3, 4	12	10
7	Dachdecke herstellen	3, 4, 5	15	9
8	Türen einsetzen	3, 4	10	10
9	Dach abdichten	7	02	10
10	**Ziel (Projektende)**	6, 8, 9	0	---

Abb. 45: Vorgangsliste erfassen

4.2.6 3. Vorgangsreihenfolge erfassen

Nachdem Sie bestimmt haben, welche Vorgänge Sie benötigen, um das Haus zu bauen, müssen Sie die Reihenfolge der Vorgänge bestimmen. Die Wände können beispielsweise nicht hochgezogen werden, bevor das Fundament errichtet ist. Andere Vorgänge können jedoch gleichzeitig erfolgen.

Darüber hinaus sollte auch die Dauer und Abhängigkeiten der Vorgänge erfasst werden, um im nächsten Schritt einen Netzplan erstellen zu können.

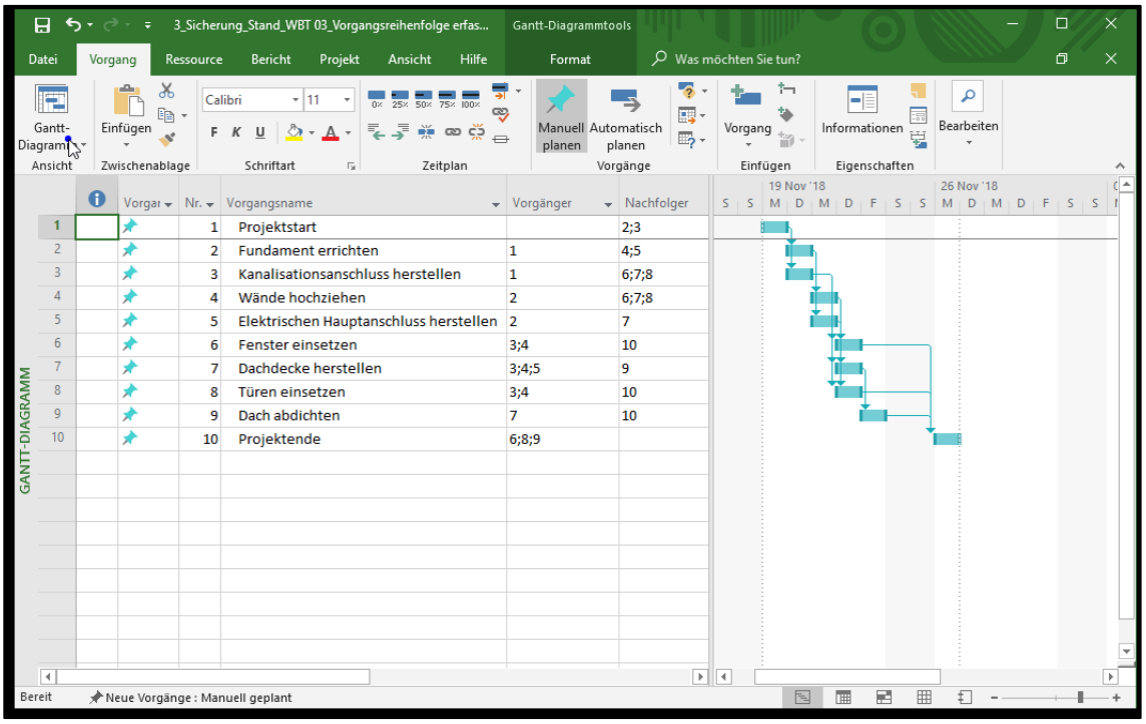


Abb. 46: MS Project Vorgangsliste und Gantt-Diagramm

4.2.7 4. Netzplan erstellen

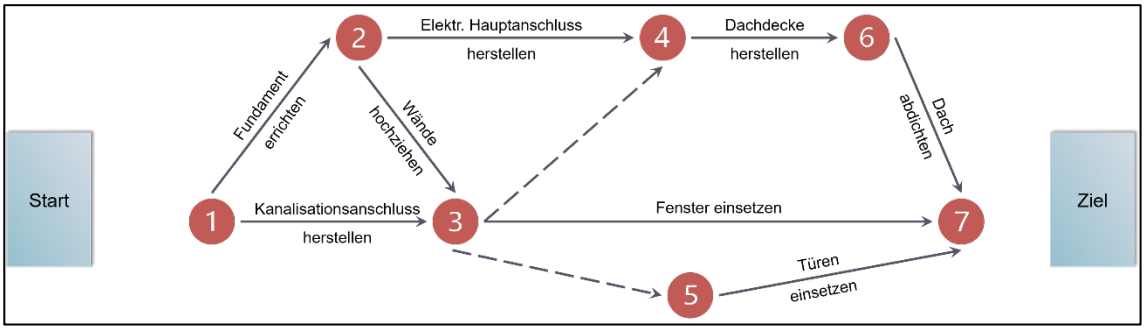


Abb. 47: Beispiel eines Netzplanes beim Hausbau

Die Netzplantechnik dient im Wesentlichen der zeitlichen Analyse, Planung und Überwachung von Projekten. Gerade Projektabläufe lassen sich gut durch Netzpläne abbilden.

Anhand der Kreise (Ereignisse) und der Pfeile (Vorgänge) werden nun Abhängigkeiten der einzelnen Vorgänge deutlich. Es ist erkennbar, welcher Vorgang abgeschlossen sein muss, damit der darauffolgende Vorgang beginnen kann. Eine Projektmanagement-Software kann einen Netzplan auf Basis der Eingabe der Vorgangsreihenfolge automatisch berechnen und visuell aufbereiten. Die Netzplantechnik ermittelt den sogenannten kritischen Pfad. Dieser zeigt alle Vorgänge ohne zeitlichen Puffer auf. Eine Verzögerung der Vorgänge auf dem kritischen Pfad führt zur Verlängerung des Projekts. Auf der nächsten Seite sehen Sie ein Beispiel eines Netzplans.

4.2.8 Beispiel Netzplantechnik: Vorgangspfeilnetz

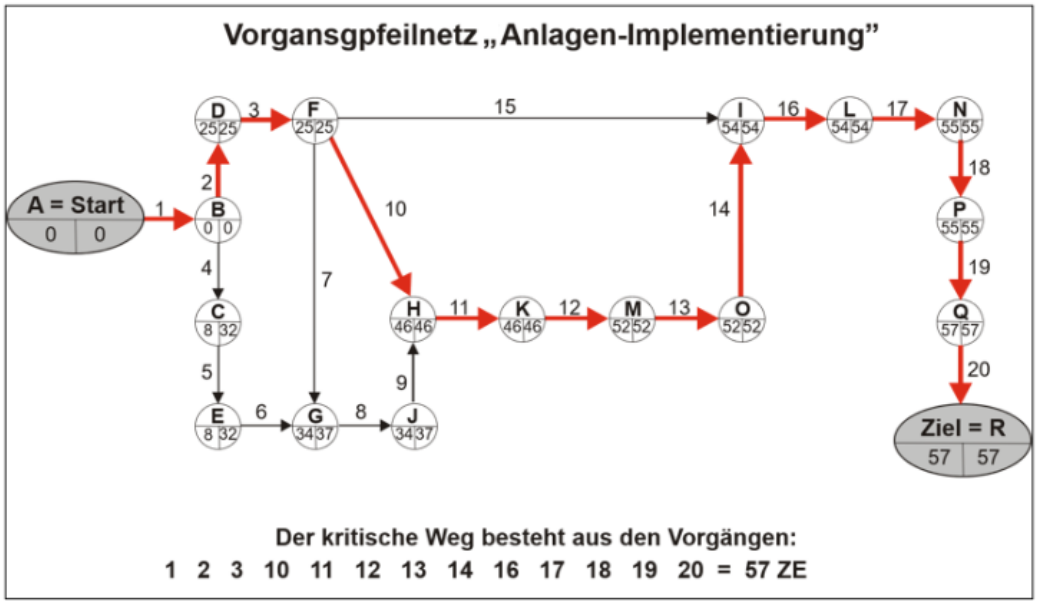
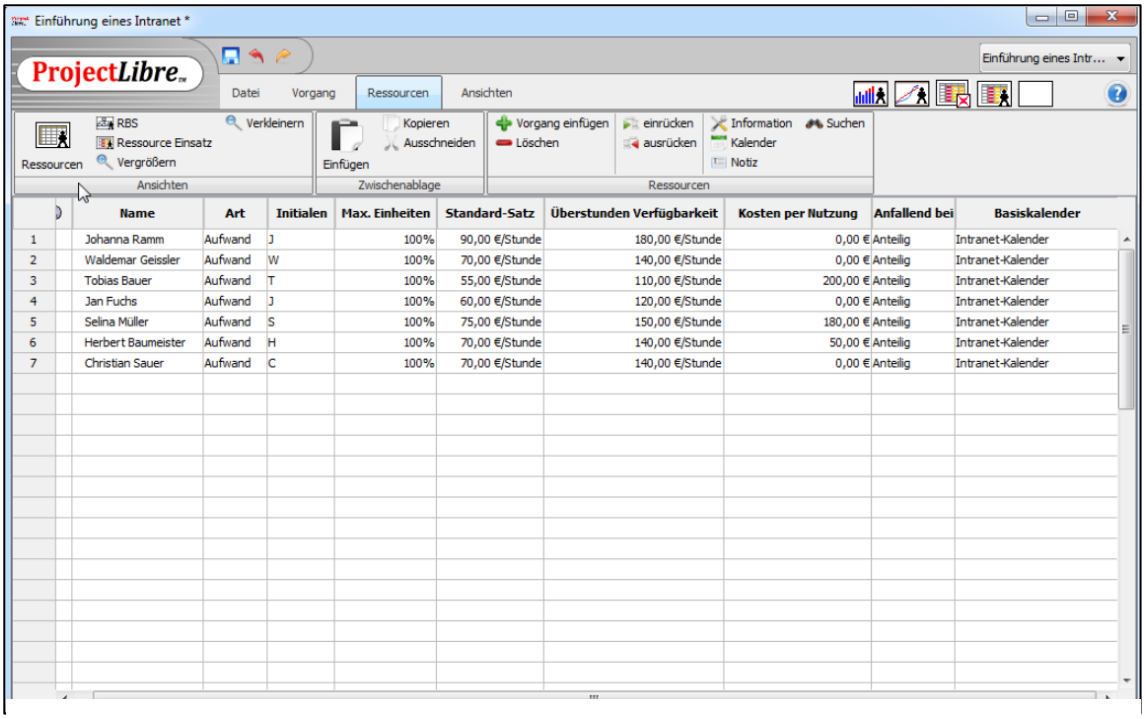


Abb. 48: Vorgangspfeilnetz mit kritischem Pfad

4.2.9 5. Ressourcen-Planung

Nachdem Sie durch die Netzplantechnik die Abhängigkeiten der Vorgänge visualisiert haben, müssen Sie die Vorgänge auch mit Ressourcen bestücken. Ressourcen können



	Name	Art	Initialen	Max. Einheiten	Standard-Satz	Überstunden Verfügbarkeit	Kosten per Nutzung	Anfallend bei	Basiskalender
1	Johanna Ramm	Aufwand	J	100%	90,00 €/Stunde	180,00 €/Stunde	0,00 €	Anteilig	Intranet-Kalender
2	Waldemar Geissler	Aufwand	W	100%	70,00 €/Stunde	140,00 €/Stunde	0,00 €	Anteilig	Intranet-Kalender
3	Tobias Bauer	Aufwand	T	100%	55,00 €/Stunde	110,00 €/Stunde	200,00 €	Anteilig	Intranet-Kalender
4	Jan Fuchs	Aufwand	J	100%	60,00 €/Stunde	120,00 €/Stunde	0,00 €	Anteilig	Intranet-Kalender
5	Selina Müller	Aufwand	S	100%	75,00 €/Stunde	150,00 €/Stunde	180,00 €	Anteilig	Intranet-Kalender
6	Herbert Baumeister	Aufwand	H	100%	70,00 €/Stunde	140,00 €/Stunde	50,00 €	Anteilig	Intranet-Kalender
7	Christian Sauer	Aufwand	C	100%	70,00 €/Stunde	140,00 €/Stunde	0,00 €	Anteilig	Intranet-Kalender

Abb. 48: ProjectLibre Ressourcenliste

dabei Mitarbeiter, Geräte oder Dienstleistungen sein. Dies ist wichtig für die Kapazitätsplanung eines Projektes.

Durch die Eingabe der verfügbaren Ressourcen und das entsprechende Mapping auf die Vorgänge, können Sie erfahren, in welchen Vorgängen Engpässe oder Überlastungen entstehen können.

In dem Beispiel des Hausbaus können Engpässe entstehen, wenn Sie vor Projektstart nicht genug Beton für das Fundament bestellt haben. Auch alle Baustellen-Mitarbeiter dürfen eine maximale Arbeitszeit nicht überschreiten. Sie müssen deshalb genügend Mitarbeiter für den Bau einplanen. Auf den folgenden Seiten sehen Sie, wie die Ressourcen-Planung in iScitor aussieht.

4.2.10 iScitor: Ressourcen erfassen

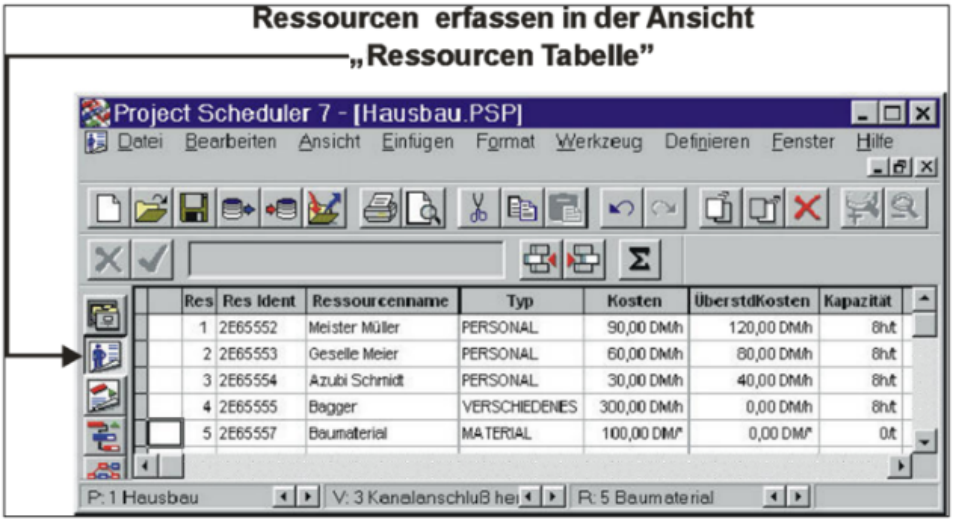


Abb. 50: iScitor: Ressourcen-Tabelle

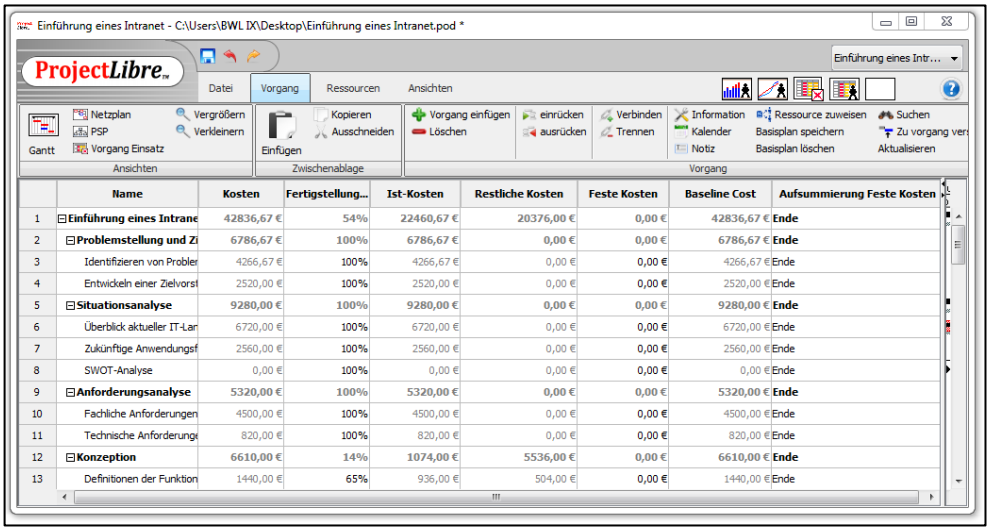


Abb. 49: ProjectLibre: Ressourcen-Tabelle

4.2.11 iScitor:Ressourcenhistogramm

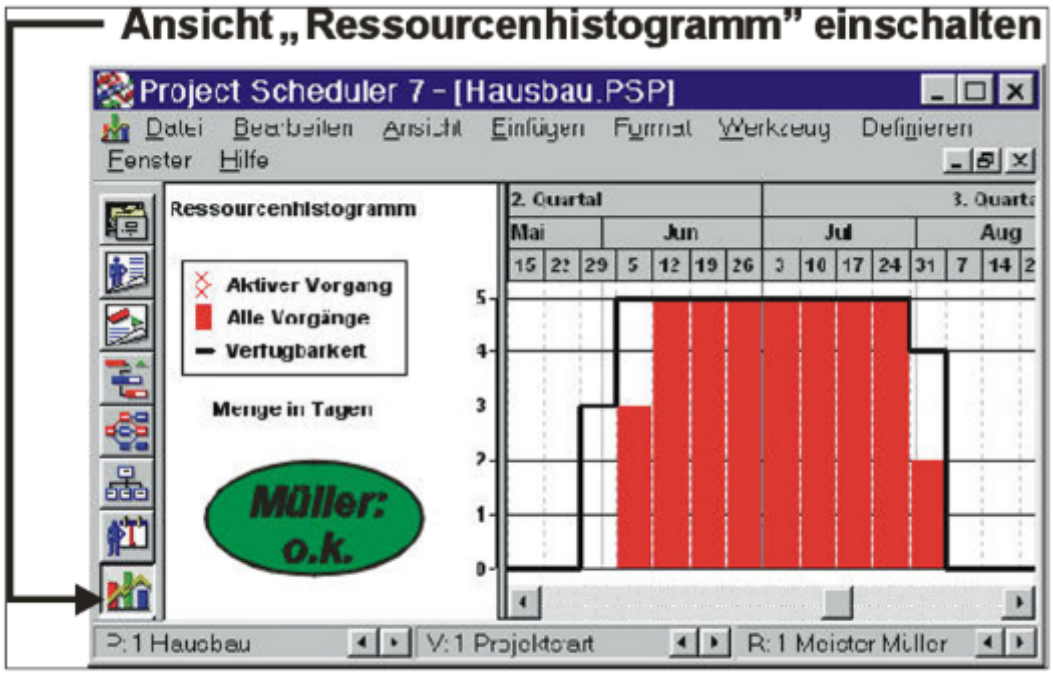


Abb. 52: iScitor: Ressourcenhistogramm

4.2.12 iScitor: Ressourcenüberlastung

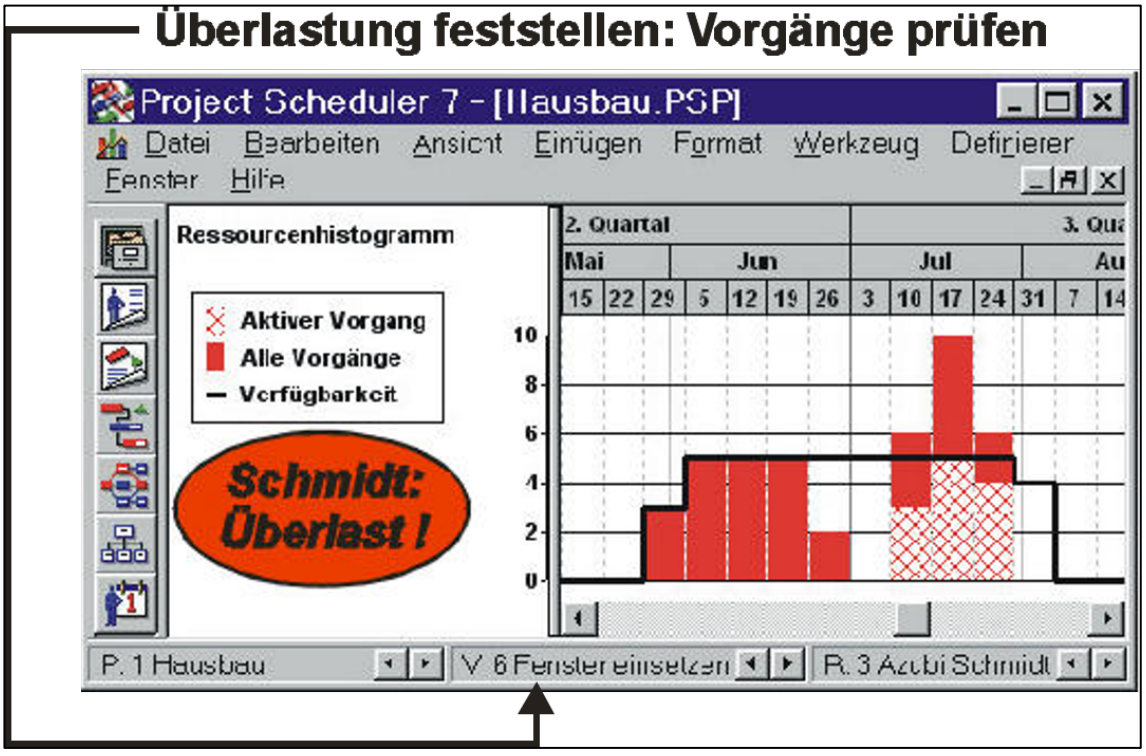


Abb. 50: iScitor: Ressourcenüberlastung

4.2.13 6. Projektverfolgung und Berichte

Mit Projektmanagement-Software können Sie nicht nur planen, sondern auch transparent auf Probleme in der Planung reagieren. Das Überlastungshistogramm ist ein erstes Indiz, dass die personelle Planung einer Anpassung bedarf. Mit Berichten können Sie z. B. den kritischen Pfad verfolgen oder für potenzielle Verzögerungen mit Eventualplänen entgegenwirken. Projektberichte sind vor allem für Personengruppen wichtig, die nicht aktiv am Tagesgeschäft des Projekts teilnehmen, aber trotzdem einen Überblick brauchen. Dies können bspw. Stakeholder oder die Geschäftsleitung sein.

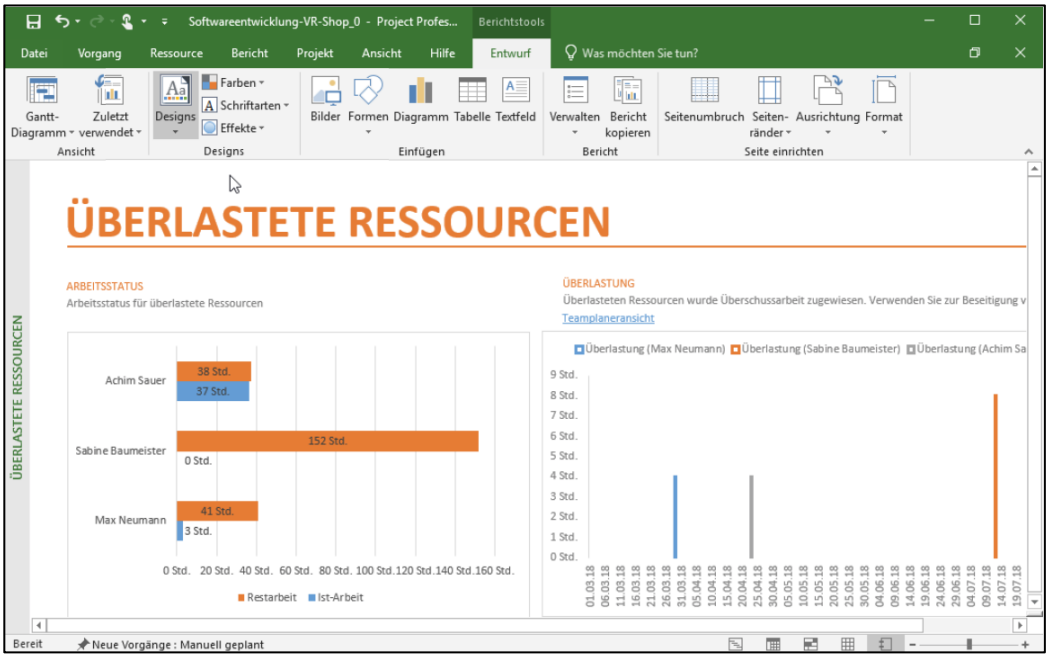


Abb. 51: MS Project Ressourcenüberlastungsbericht

4.2.14 iScitor: Basisplan im Vergleich

Projektmanagement-Software bietet die Möglichkeit, den Ist-Status eines Projektes mit dem Basisplan zu vergleichen.

Der Basisplan ist der Soll-Zustand eines Projektes. Weicht der Ist-Zustand davon ab, ist es notwendig einzugreifen.

Eine Anpassung an den Plan kann auf Grund von drei Möglichkeiten geschehen:

- Anpassung der Qualität,
- Anpassung des Budgets oder
- Anpassung der Projektdauer.

Die folgenden Seiten zeigen, wie diese Dateneingabe in PM-Software erfolgt.

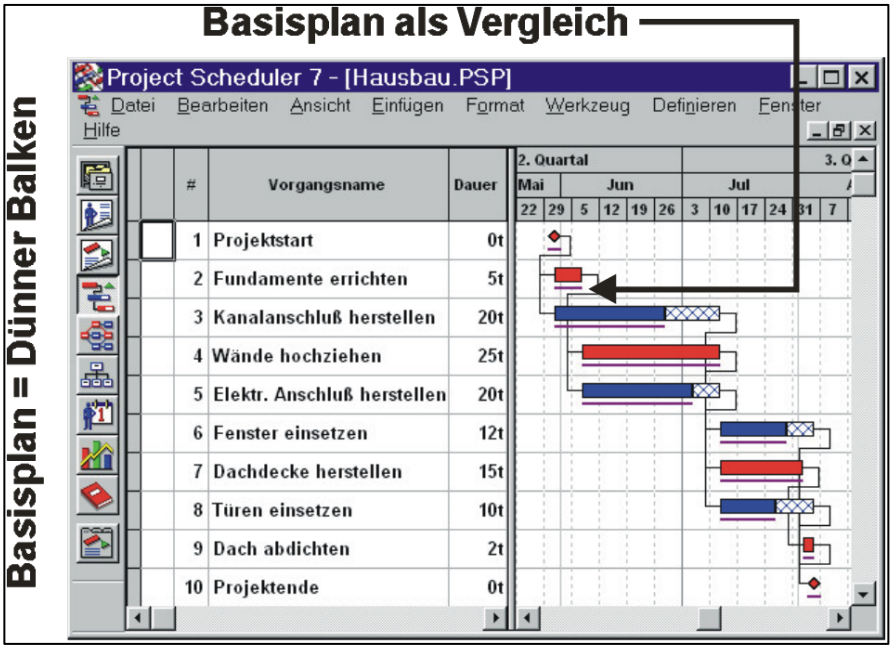


Abb. 52: iScitor Basisplan-Abweichung

4.2.15 iScitor: Projektverfolgung

Projektmanagement-Software bietet unter anderem die Möglichkeit, den Zeitplan eines Projektes zu verfolgen.

Es lässt sich erfassen,:

- wie der Fortschritt des einzelnen Vorganges ist,
- wann der geplante Start des Vorganges ist und
- wann der Vorgang tatsächlich gestartet ist.

4.2.16 iScitor und MS Project: Ressourcenverteilung

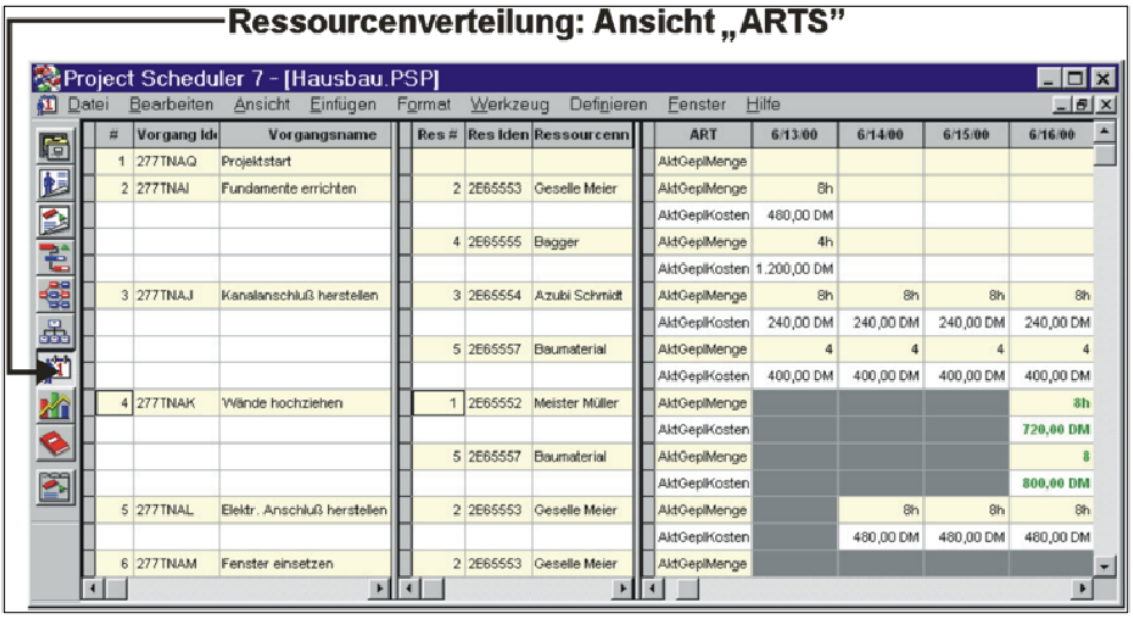
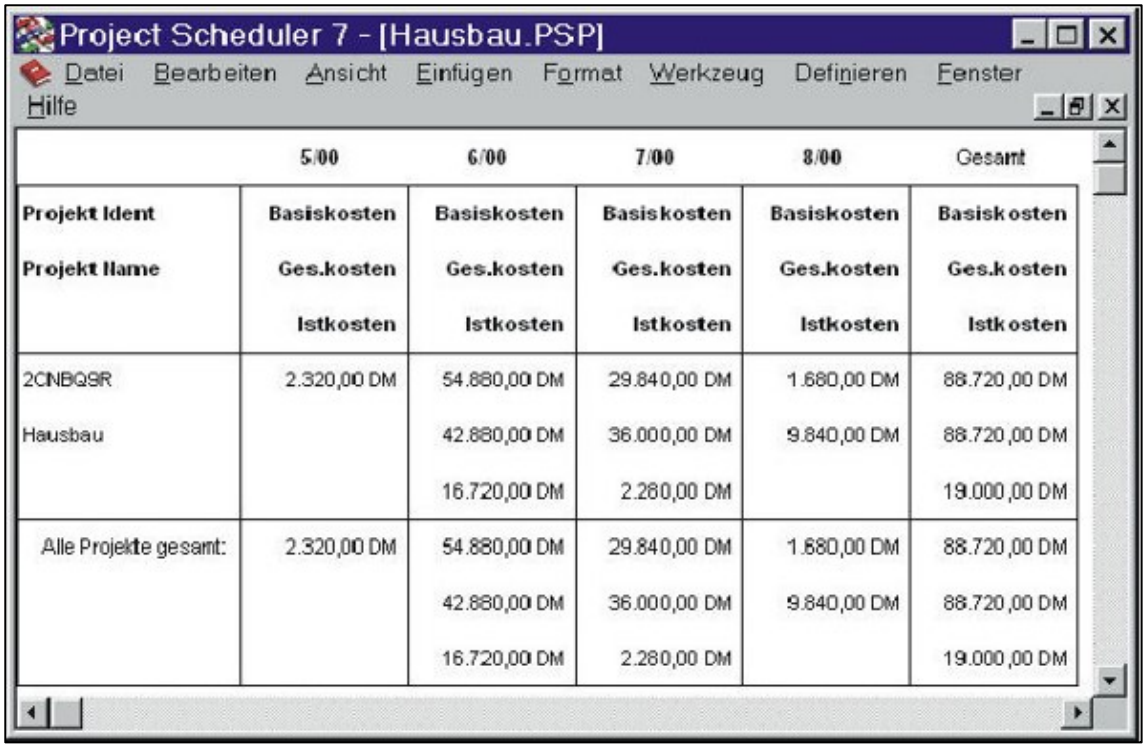


Abb. 53: iScitor: Ressourcenverteilung

4.2.17 iScitor und MS Project: Projekt-Bericht



	5/00	6/00	7/00	8/00	Gesamt
Projekt Ident	Basiskosten	Basiskosten	Basiskosten	Basiskosten	Basiskosten
Projekt Name	Ges.kosten	Ges.kosten	Ges.kosten	Ges.kosten	Ges.kosten
	Istkosten	Istkosten	Istkosten	Istkosten	Istkosten
2CNBQ9R	2.320,00 DM	54.880,00 DM	29.840,00 DM	1.680,00 DM	88.720,00 DM
Hausbau		42.880,00 DM	36.000,00 DM	9.840,00 DM	88.720,00 DM
		16.720,00 DM	2.280,00 DM		19.000,00 DM
Alle Projekte gesamt:	2.320,00 DM	54.880,00 DM	29.840,00 DM	1.680,00 DM	88.720,00 DM
		42.880,00 DM	36.000,00 DM	9.840,00 DM	88.720,00 DM
		16.720,00 DM	2.280,00 DM		19.000,00 DM

Abb. 54: iScitor Bericht der Kosten

Es gibt verschiedene Typen von Berichten. Berichte dienen dabei nicht nur der Projektverfolgung, sondern auch als Kontrollwerkzeug für Projekt-Stakeholder.

Ein Lenkungsgremium hat so beispielsweise Einsicht in das bisher verbrauchte Budget eines Projektes, kann Schwachstellen nachvollziehen und eventuell eingreifen.

Berichte ermöglichen es, die zuvor geschätzten Werte des Projekts mit den tatsächlich eingetretenen Zahlen zu vergleichen und daraus Konsequenzen für andere Projekte abgeleitet.

4.3 Abschlusstest – WBT 04

4.3.1 Abschlusstest

Bitte beantworten Sie die folgenden Fragen durch das Ankreuzen der korrekten Antworten (Tab. 5). Bei einigen Fragen können auch mehrere Antworten richtig sein.

Nr.	Frage	Richtig	Falsch
1	Berichte arbeiten nur mit Schätzwerten, die zu Beginn des Projektes geplant wurden.		
2	Projektmanagement-Standards sind „Best Practices“, die für jedes Projekt ohne Anpassung gültig sind.		
3	Welche Schritte der Projektplanung können durch eine PM-Software abgebildet werden?		
	Vorgänge des Projektes erfassen		
	Dauer eines Vorganges graphisch abbilden		
	Ist- und Soll-Zustände vergleichen		
	Kapazitätsüberlastungen anzeigen		
	Einen Netzplan erstellen		
4	Das PMBOK und PRINCE2 sind Projektmanagement-Standards, die weit verbreitet als „Best Practices“ gelten.		
5	Projektmanagement-Software bieten automatisch generierte Eventualpläne bei Projektabweichung.		
6	Projektmanagement-Standards bieten...		
	Analysewerkzeuge		
	Vorgangserfassung		
	die Abbildung von Eventualplänen		

	Methoden, die auf das Projekt angepasst werden können		
	technische Fachkonzeptionen		
7	Das Hauptproblem bei Projekten ist, dass häufig _____ verwendet werden.		

Tab. 5: Abschlusstest – WBT04

4.4 Typische Aufgabenstellungen

Typische Aufgabenstellungen – IT-Projekte: Standards und Software

Zur Bearbeitung dieser Aufgabenstellungen beachten Sie bitte: Verlangt ist eine fachlich zutreffende, inhaltlich nachvollziehbare und kausal zusammenhängende Erörterung aus vollständigen Sätzen in lesbarer Handschrift. Für jede Aufgabe: Maximal zwei Seiten Text.

Aufgabe 1:

> Erläutern Sie, was Projektmanagement-Standards und PM-Software für ein Entwicklungsprojekt leisten können.

Aufgabe 2:

> Beschreiben Sie die wesentlichen Schritte bei der konkreten Nutzung von PM-Software in einem Entwicklungsprojekt.

5 Grundlagen zu Vorgehensmodellen

5.1 Zwei Perspektiven im Software Engineering

5.1.1 Gestaltung des Software Engineering

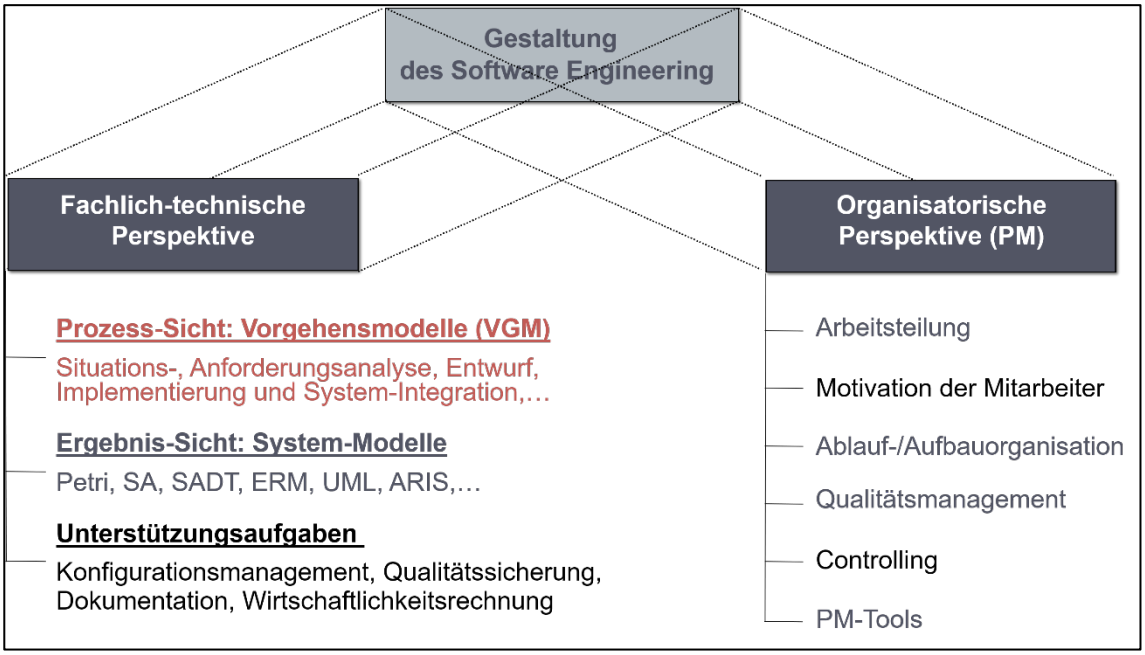


Abb. 55: Fachlich-technische und organisatorische Perspektive

Das Software Engineering kann aus zwei verschiedenen Perspektiven betrachtet werden. In diesem Kapitel betrachten Sie die fachlich-technische Perspektive und genauer die Prozess-Sicht. Sie gehen der Frage nach, „wie“ Software entwickelt werden kann. Anhand verschiedener Vorgehensmodelle schauen Sie sich unterschiedliche Möglichkeiten des Vorgehens bei der Software-Entwicklung an.

5.1.2 Prozess-Sicht und Ergebnis-Sicht

Ergebnis-Sicht: Die fachliche Modellierung

Das „Was” der Entwicklung – die Gestaltung und Darstellung des IT-Systems mit Modellierungsansätzen wie z. B. funktions-, datenfluss-, daten-, prozess- oder objektorientierter Modellierung.

Prozess-Sicht: Der dynamische Zeitablauf

Das „Wie” der Entwicklung – die Vorgehensweise der Entwicklung – die Prozessschritte zur Entwicklung des IT-Systems.

Der Zusammenhang:

Die beiden Sichten stehen eng miteinander in Verbindung, denn Ziel eines Vorgehens oder Prozesses ist immer ein Ergebnis. Man kann dies leicht anhand eines Beispiels erläutern: Der teilweise mühselige Prozess einer Bergbesteigung wird vorher genau geplant und in kleinere Etappen mit Zwischenzielen unterteilt. So scheint der Aufstieg leichter und das Ziel der schönen Aussicht kann leichter erreicht werden, als wenn man planlos darauf losläuft.

Die beiden Sichten ergänzen sich unweigerlich und greifen wie Zahnräder ineinander. Bevor Sie Software entwickeln, benötigen Sie einen Plan, der vorgibt, wie sie entwickelt werden muss. Sie benötigen aber auch Zwischenschritte bzw. Ergebnisse wie die fachliche Beschreibung, damit Sie dem Ziel Schritt für Schritt näherkommen können.

5.1.3 Prozess-Sicht der Planung und Entwicklung von IT-Systemen

Vorgehensmodelle:

Wie z. B. sequentielle Phasenmodelle, evolutionäre Spiralmodelle, und agile Vorgehensmodelle

Methodische Durchgängigkeit:

Die einzelnen Prozessschritte werden durch aufeinander abgestimmte Ergebnisssichten (Analyse- und Darstellungstechniken, z. B. durch ein Bündel von UML-Konzepten) unterstützt.

Durch die Prozess- und Ergebnis-Sicht geht man im Software Engineering also der Frage nach: Welche Teilaufgaben sind in welcher Reihenfolge zur Entwicklung eines IT-Systems zu bewältigen?

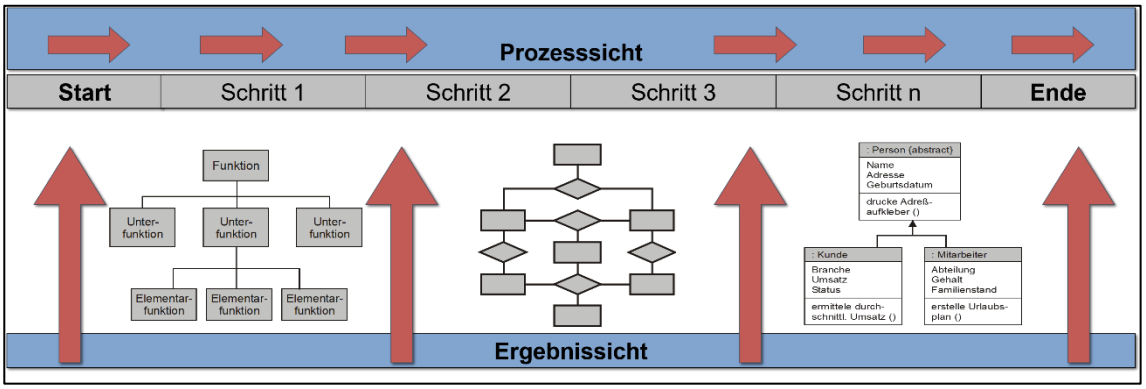


Abb. 56: Prozess- und Ergebnis-Sicht

5.1.4 Beispiel: Zusammenhang zwischen Ergebnis- und Prozess-Sicht

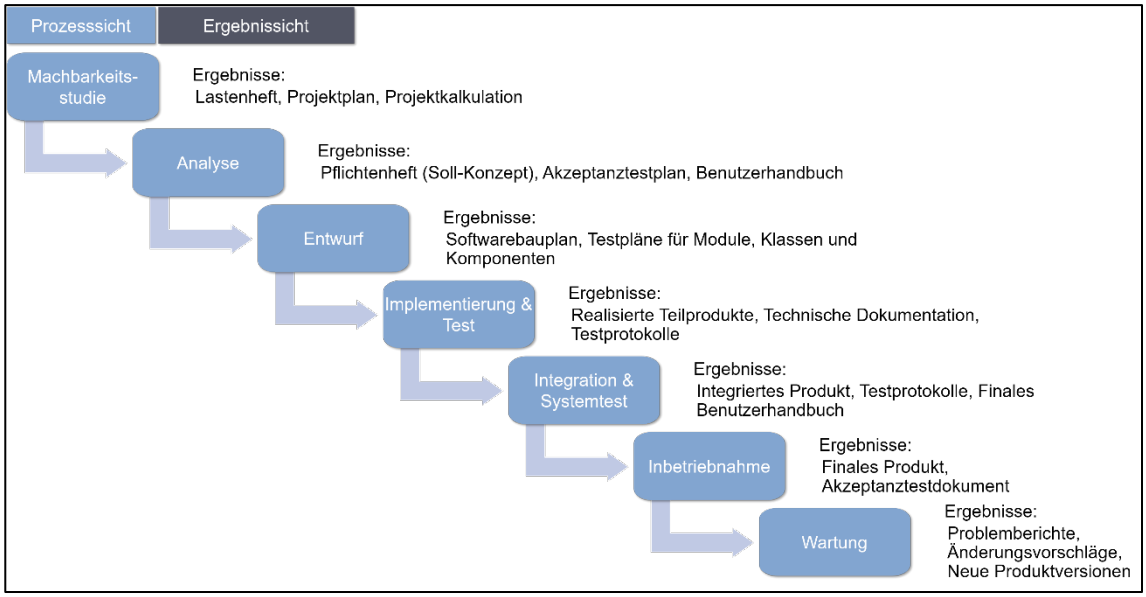


Abb. 57: Phasen und Meilensteine

Die Rechtecke der Grafik (vgl. Royce) beschreiben die Phasen eines typischen Software-Projektes. Sie beziehen sich auf die Prozess-Sicht eines Projektes. Die nebenstehenden Ergebnisse sind mögliche Gegenstände bzw. Meilensteine, die eine solche Phase abschließen. Die Meilensteine beziehen sich dabei auf die Ergebnis-Sicht.

5.1.5 Methodische Durchgängigkeit am Beispiel Hausbau

Beim Hausbau kommt es wie bei so vielen Projekten auf die möglichst genaue und vollständige Anforderungsbeschreibung an. Der beauftragte Architekt muss durch verbale, schriftliche und visuelle Hilfsmittel herausfinden, was die zukünftigen Hausbesitzer wünschen. Sind die ersten Anforderungen geklärt, skizziert der Architekt das Haus in der konzeptuellen Zeichnung. Wird diese von den Bauherren akzeptiert, können, darauf aufbauend, genauere Pläne angefertigt werden. Diese werden dann wiederum benötigt, um das geplante Haus beim Bauamt genehmigen zu lassen und erste Kostenschätzungen zu tätigen. Während des Baus muss sich dann jeder Mitarbeiter auf der Baustelle darauf verlassen, dass der angefertigte Plan des Hauses alle Einzelheiten enthält, die sie für ihre methodische Durchgängigkeit am Beispiel Software Engineering Arbeit benötigen.

Aus dem mündlichen Konzept der Bauherren entsteht also im Laufe der Zeit erst ein grober Plan und schließlich der exakte Grundriss mit technischen Einzelheiten. Der Detaillierungsgrad nimmt zu. Die methodische Durchgängigkeit stellt die Voraussetzung für eine reibungslose Übersetzung in die nächste Detaillierungsebene sicher.

5.1.6 Methodische Durchgängigkeit am Beispiel Software Engineering

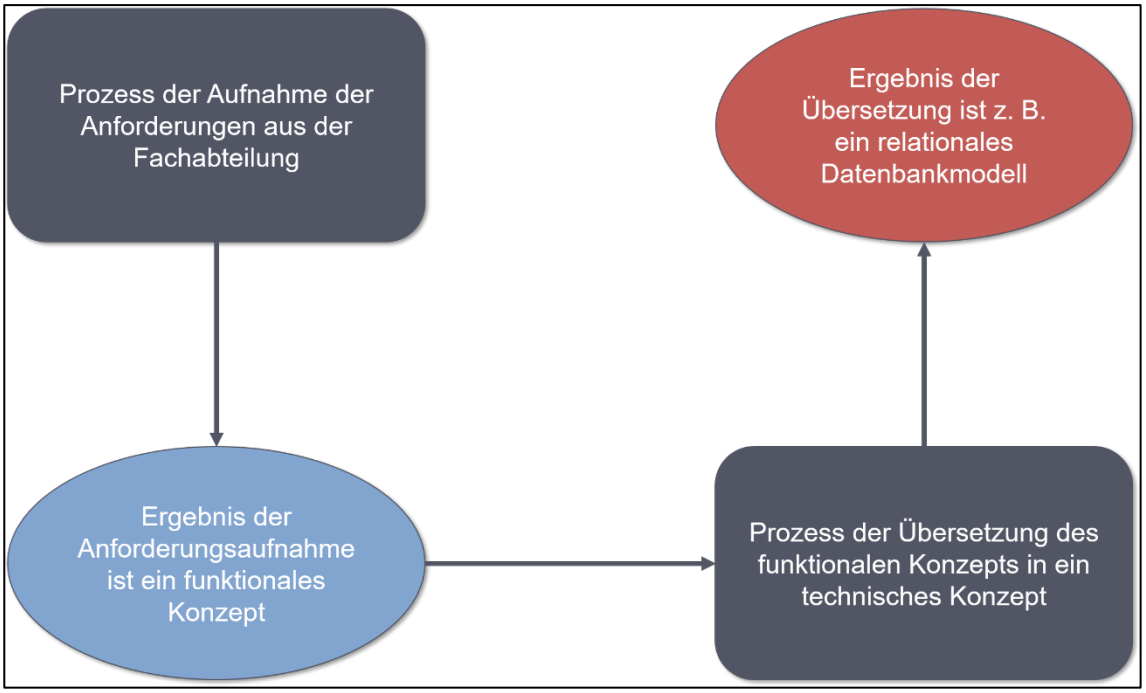


Abb. 58: Methodische Durchgängigkeit im Software Engineering

Wenn die Methoden zur Entwicklung eines Software-Produkts aufeinander aufbauen (1), kann die Übersetzung der Anforderungen an ein Produkt bis zur technischen Entwicklung in Code erfolgen.

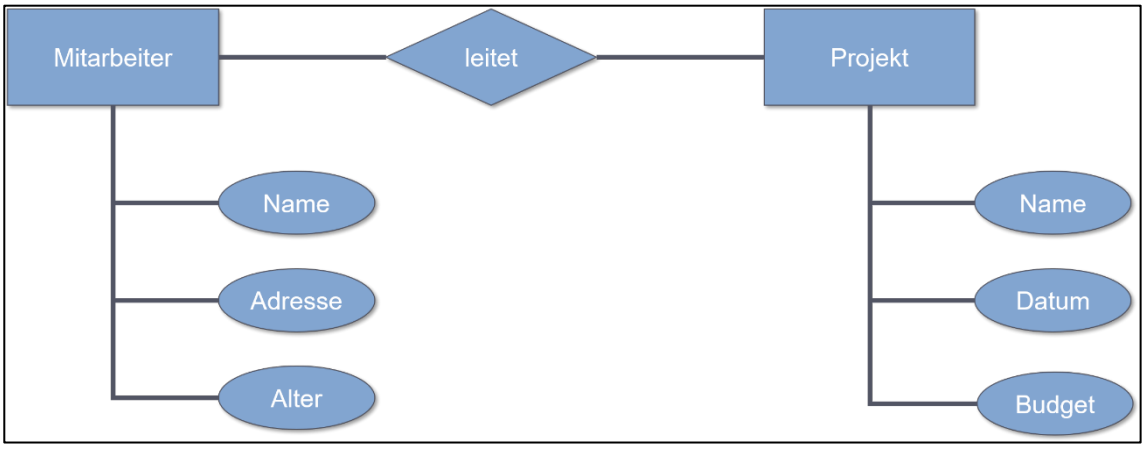


Abb. 59: ER-Modell Beispiel

In diesem Beispiel wird angenommen, dass mit Hilfe des Entity Relationship Model (2) die mündlich überlieferten Anforderungen des Kunden umwandeln (3) werden können. Basierend auf diesem ERM, ist es möglich ein Datenbank-Konzept, ein Relationen-Modell abzubilden.

Um diese methodische Durchgängigkeit zu erzielen, müssen alle Vorgänge gut aufeinander abgestimmt sein. Dies kann am besten geschehen, wenn das Vorgehen anhand eines Vorgehensmodelles geplant wird. Was ein Vorgehensmodell ist, sehen Sie auf der nächsten Seite.

Personalschlüssel	Vorname	Nachname	Geburtstag	Straße	Hausnummer	PLZ	Ort
1	Antje	Musterfrau	05.06.1990	Gartenweg	3	36648	Musterstadt
2	Herrmann	Mustermann	02.01.1965	Licherweg	96	35548	Musterdorf

Projektschlüssel	Personalschlüssel (Projektleitung)	Startdatum	Enddatum	Budget	Währung
PJ-102	2	01.01.2021	31.12.2021	3.000.000	€
PJ-103	1	05.08.2020	01.11.2022	1.000.500	$

Abb. 60: Relationenmodell Beispiel

5.2 Vorgehensmodelle

5.2.1 Was sind Vorgehensmodelle?

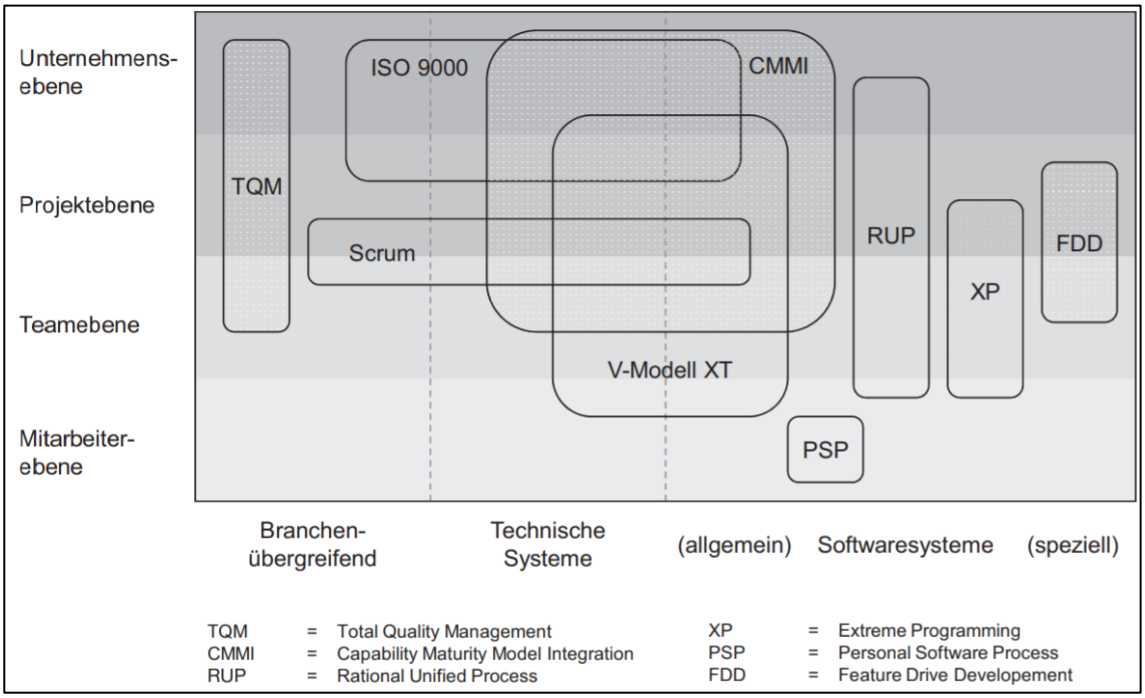


Abb. 61: Übersicht über Vorgehensmodellen

Die für eine effiziente Durchführung von Software-Projekten notwendige (Ablauf-)Organisation des Entwicklungsprozesses wird über so genannte Vorgehensmodelle strukturiert.

An der Entwicklung von Software sind meistens mehrere Personen beteiligt, die jeweils verschiedene Rollen einnehmen und deren Aktivitäten aufeinander abgestimmt werden müssen. Mithilfe von Prozess- bzw. Vorgehensmodellen lässt sich beschreiben, welche

Vorgänge in welcher zeitlichen Abfolge erledigt werden müssen und wer für sie verantwortlich ist.

Vorgehensmodelle helfen dabei, die Software-Entwicklung übersichtlicher zu gestalten und die Beteiligten sinnvoll zu koordinieren.

Die nebenstehende Grafik nimmt eine Einteilung der Vorgehensmodelle nach Bezugsebene und Entwicklungsgegenstand vor.

5.2.2 Aufgabe der Vorgehensmodelle

Ein Vorgehensmodell organisiert einen Prozess der gestaltenden Produktion in verschiedene, strukturierte Abschnitte, denen wiederum entsprechende Methoden und Techniken zugeordnet sind.

Aufgabe eines Vorgehensmodells ist es, die allgemein in einem Gestaltungsprozess auftretenden Aufgabenstellungen und Aktivitäten in einer logischen Reihenfolge darzustellen. Mit ihren Festlegungen sind Vorgehensmodelle organisatorische Hilfsmittel, die für konkrete Aufgabenstellungen individuell angepasst werden können und sollen.

5.2.3 Ohne Vorgehensmodelle: Unstrukturierte Vorgehensweise

Vorgehensmodelle unterstützen uns also dabei, die Prozesse in einem Projekt zu steuern. Darüber hinaus verleihen sie komplexen Systemen des Software Engineerings einen organisatorischen Rahmen.

Würden Sie bei der Projektdurchführung keine Vorgehensmodelle benutzen, hätte dies schwerwiegende Konsequenzen bei der Organisation des Projektes und dessen Durchführung.

The „Hacker“ or „Code & fix“ or „Quick´n dirty“

- Write some code
- Fix the problems in the code

Probleme und Gefahren

- Unstrukturierter Code, der weder weiterentwickelt werden kann noch Fehler aufzeigt.
- Unstrukturierte Vorgehensweise, die eine Planung des Software-Projekts und die damit verbundenen Kosten undurchsichtig macht.
- Koordination großer Entwicklerteams durch fehlenden Standard oder Normen unmöglich.
- Kundenanforderungen können nicht erfüllt werden, da keine Standards für qualitativ-hochwertige Entwicklung existieren.

- Zeit- und Kostenüberschreitungen sind an der Tagesordnung.
- Durch die fehlenden Fachkräfte können nur wenige Anwendungen umgesetzt werden.

5.2.4 Grundformen von Vorgehensmodellen

Es gibt eine Vielzahl von Vorgehensmodellen...

Nachfolgend werden folgende Grundformen unterschieden:

- Allgemeine Vorgehensmodelle
- Sequentielle Vorgehensmodelle
- Evolutionäre Vorgehensmodelle
- Agile Vorgehensmodelle

...die sich u. a. durch Folgendes unterscheiden:

- Der zeitlichen Abfolge der Bearbeitung von Aufgaben
- Der Annahmen über die Form der Arbeitsteilung
- Der Abstimmung von Teilaufgaben und der Aufbauorganisation
- Der Empfehlungen zu Dokumentation, Erzeugung von Zwischenprodukten und bestimmten Hilfsmitteln

5.2.5 Beispiel: Allgemeines Vorgehensmodell

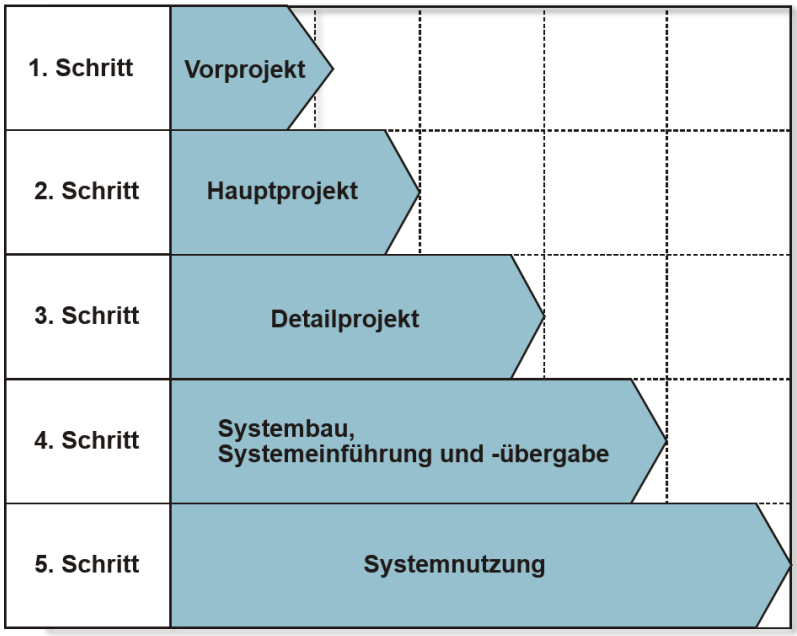


Abb. 62: Allgemeines Vorgehensmodell

Mithilfe von Prozess- bzw. Vorgehensmodellen lässt sich beschreiben, welche Vorgänge in welcher zeitlichen Abfolge erledigt werden müssen und wer für sie verantwortlich ist. Vorgehensmodelle helfen dabei, die Software-Entwicklung übersichtlicher zu gestalten und die Beteiligten sinnvoll zu koordinieren.

Das allgemeine Vorgehensmodell sieht eine sequenzielle Abarbeitung aller Projekt-Vorgänge in verschiedenen Phasen vor. Diese Phasen können bspw. aus einem Vorprojekt, einem Hauptprojekt, einem Detailprojekt, dem Systembau, der Systemeinführung und -übergabe und schlussendlich aus der Nutzung bestehen.

Jede Phase endet jeweils mit einem fest definierten Meilenstein bzw. Ergebnis. In der Vorphase könnte ein solches Ergebnis beispielsweise die detaillierte Problemstellung für das zu lösende Problem darstellen.

5.3 Das allgemeine Vorgehensmodell

5.3.1 Beispiel: Phasen eines allgemeinen Vorgehensmodells

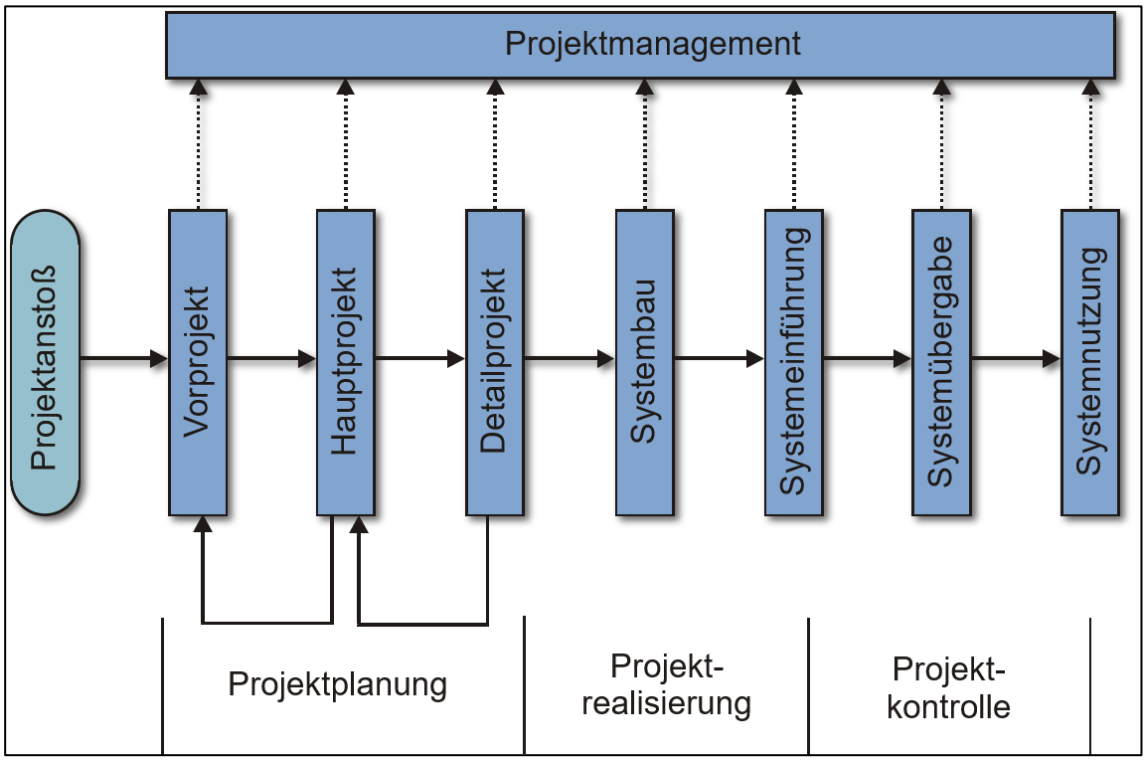


Abb. 63: Projektphasen im allgemeinen Vorgehensmodell

5.3.2 Mögliche Phasen, Aktivitäten und Ergebnisse eines allgemeinen Vorgehensmodells

Phasen	Aktivitäten	Ergebnisse
Projektanstoß	• Projektauftrag formulieren • Projektorganisationsform wählen • Projektpriorität bestimmen	• Problem, Idee • Projektwürdigkeit • Projektauftrag
Vorprojekt	• Problemstellung überprüfen • Untersuchungsbereich abgrenzen • Situationsanalyse bzw. Standortbestimmung vornehmen • Gestaltungsmöglichkeiten abklären • Ziele erarbeiten • Lösungsprinzipien erarbeiten • erste Wirtschaftlichkeits- und Nutzenüberlegungen anstellen • Projektplanung erstellen (weiteres Vorgehen planen)	• Lösungsprinzipien • Vorgehenskonzept
Hauptprojekt	• Zielsetzung überarbeiten • Gesamtkonzept (eventuell mit Varianten) erarbeiten • Wirtschaftlichkeit überprüfen • Planung aktualisieren	• Gesamtkonzept • Masterplan
Detailprojekt	• realisierungsreife Lösungen ausarbeiten • detaillierte Wirtschaftlichkeitsrechnung erstellen • Unterhaltsorganisation planen • Schulungs- und Einführungskonzept erstellen	• Detailpläne
Systembau	• System bauen (Lösungen benutzungsreif machen) • System testen und abnehmen • Kosten und Wirtschaftlichkeit überprüfen	• einführungsbereites System
Systemeinführung	• Schulung • Unterhaltsorganisation aufstellen	• eingeführtes System

Abb. 64: Aktivitäten und Ergebnisse in den Projektphasen

5.3.3 Vorgehensmodelle: Der gemeinsame Nenner

Gemeinsamer Nenner der allgemeinen Vorgehensmodelle sind die „Phasen“

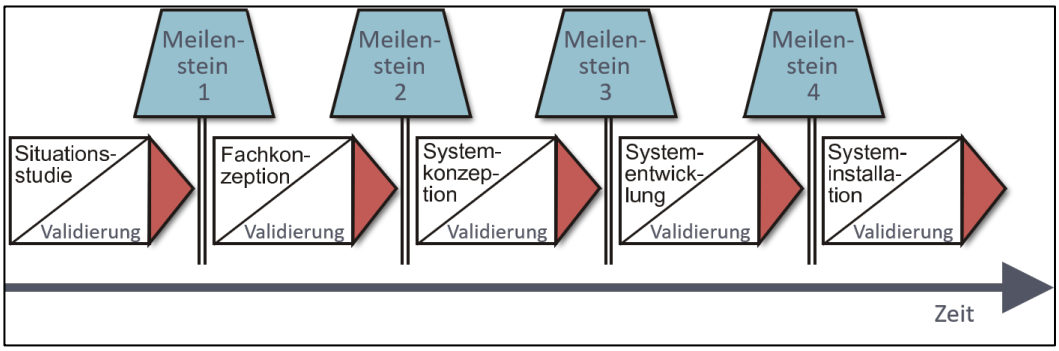


Abb. 65: Sequentielles Vorgehensmodell und Meilensteine

- Gestuftes, phasenorientiertes Vorgehen
- Unterteilt in Phasen mit Meilensteinen und/ oder Zwischenergebnissen
- Abbildung oben: Streng sequentielle Ordnung der Phasen mit Meilensteinen als Zwischenergebnisse

5.3.4 Beispiele: Phasenkonzepte – Teil 1

Verschiedene Autoren (blau) entwickelten unterschiedliche Phasenkonzepte (grau) für die Software-Entwicklung.

Dabei ist auffällig, dass alle Autoren eine definierte Anforderungsanalyse vor Projektstart fordern. Dies stellt einen deutlichen Unterschied zu den evolutionären und agilen Vorgehensmodellen, wie Sie in der nächsten Lerneinheit erfahren werden.

BOEHM (1986)	NAGEL (1990)	HESSE u. a. (1992)
System-Realisierbarkeit	Initialisierung	Analyse
Pläne und Anforderungen	Vorstudie	Definition
Produktentwurf	Org. Untersuchung	System-Entwurf
Feinkonzeption	System-Konzeption	Komponenten-Entwurf
Codierung	Detail-Festlegung	Modul-Implementierung
Integration	Programmierung	Subsystem-Integration
Implementierung	Systemtest	System-Integration
Betrieb und Wartung	Inbetriebnahme	Installation
	Nutzung	Betrieb und Wartung

HOFFMANN (1984)	WALTER (1992)	ZEHNDER (1991)
Ist-Analyse	Projektdefinition	Idee, Vorabklärung
Zielfestlegung	Ist-Analyse	Projektumriss
Durchführbarkeitsstudie	Soll-Konzeption	Konzept mit Varianten
Soll-Konzeption	Realisierungsplan	Realisierung
Systementwurf	Software-Entwicklung	Systemtest
Systemimplementierung	Systemeinführung	Einführung
Systembetrieb		Betrieb

Abb. 66: Phasenkonzepte 1986 - 1991

5.3.5 Beispiele: Phasenkonzepte – Teil 2

BALZERT (1982)	END u. a. (1990)	HEINRICH (1994)
Planung	Projektvorschlag	Strategische IS-Planung
Definition	Planung I	Vorstudie
Entwurf	Planung II	Feinstudie
Implementierung	Realisierung I	Grobprojektierung
Abnahme und Einführung	Realisierung II	Feinprojektierung
Wartung und Pflege	Einsatz	Installation

KARGL (1989)	SUHR u. a. (1993)	SCHÖNTHALER u. a. (1992)
Situationsstudie	Problemanalyse	Vorstudie
Fachkonzeption	Funktionale Analyse	Anforderungsanalyse
Systemkonzeption	Softwaretechn. Entwurf	Entwurf
Systemrealisierung	Implementierung	Implementation
Systemanwendung	Testen und Installation	Systemtest

Abb. 67: Phasenkonzepte 1982 - 1992

5.4 Abschlusstest – WBT 05

5.4.1 Abschlusstest

Bitte beantworten Sie die folgenden Fragen durch das Ankreuzen der korrekten Antworten (Tab. 6). Bei einigen Fragen können auch mehrere Antworten richtig sein.

Nr.	Frage	Richtig	Falsch
1	Die Prozess-Sicht ist die organisatorische Projektmanagementperspektive auf das Software Engineering.		
2	Das allgemeine Vorgehensmodell ist auch allgemein gültig und kann auf jedes Projekt angewendet werden.		
3	Das allgemeine Vorgehensmodell besteht aus folgenden Phasen....		
	Vorprojekt		
	Hauptprojekt		
	Detailprojekt		
	Systembau, Systemeinführung und -übergabe		
	Systemnutzung		

4	Durch die Vorgehensmodelle werden unstrukturierte Vorgehensweisen, die z. B. zu mangelnder Softwarequalität führen systematisch durchbrochen.		
5	Die Grundformen der Vorgehensmodelle sind...		
	die allgemeinen Vorgehensmodelle.		
	die sequentiellen Vorgehensmodelle.		
	die evolutionären Vorgehensmodelle.		
	die agilen Vorgehensmodelle.		
	die strukturierten Vorgehensmodelle.		
6	Die allgemeinen Vorgehensmodelle liefern eine strukturierte Vorgehensweise und halten _____, _____ und _____ fest.		

Tab. 6: Abschlusstest – WBT05

5.5 Typische Aufgabenstellungen

Typische Aufgabenstellungen – Grundlagen zu Vorgehensmodellen

Zur Bearbeitung dieser Aufgabenstellungen beachten Sie bitte: Verlangt ist eine fachlich zutreffende, inhaltlich nachvollziehbare und kausal zusammenhängende Erörterung aus vollständigen Sätzen in lesbarer Handschrift. Für jede Aufgabe: Maximal zwei Seiten Text.

Aufgabe 1:

Was versteht man im Allgemeinen unter einem Vorgehensmodell?

Aufgabe 2:

Was ist die Hauptaufgabe eines Vorgehensmodells?

6 Sequentielle und evolutionäre Vorgehensmodelle

6.1 Sequentielle Vorgehensmodelle

6.1.1 Wiederholung: Die Grundformen von Vorgehensmodellen

Es gibt vier verschiedene Grundformen von Vorgehensmodellen: Allgemeine, sequentielle, evolutionäre und agile Vorgehensmodelle. Sie unterscheiden sich im Wesentlichen durch die zeitliche Abfolge der Bearbeitung von Aufgaben, die Annahmen über die Form der Arbeitsteilung, die Abstimmung von Teilaufgaben und der Aufbauorganisation sowie der Empfehlung von Dokumentation, Erzeugung von Zwischenständen und bestimmten Hilfsmitteln.

Dieses WBT behandelt sequentielle und evolutionäre Vorgehensmodelle. Agile Vorgehensmodelle sind Teil des folgenden WBT.

6.1.2 Beispiel: Streng sequentielles Vorgehensmodell

In einem streng sequentiellen Vorgehensmodell durchläuft das Projekt nacheinander Phase für Phase. Es sind **keine Rücksprünge in vorherige Phasen** vorgesehen. Eine Phase mit allen Arbeitspaketen muss somit komplett abgeschlossen sein, bevor die nächste Phase beginnen kann.

Dies entspricht gerade bei der Erstellung von Software nicht der Realität, da sich beispielsweise durch technische Neuerungen oder Anpassungen der Anforderungen immer Änderungen ergeben können, die auch rückwirkend in das Projekt integriert werden müssen. Streng sequentielle Vorgehensmodelle gelten deshalb als **wirklichkeitsfremd**.

Ein Phasenabschluss wird durch einen Meilenstein definiert. Ein Meilenstein kann beispielsweise ein fertiger Anforderungskatalog sein, der in einer nächsten Phase genau analysiert werden muss.

6.1.3 Beispiel: Wasserfallmodell

Das Wasserfallmodell gehört zu den sequentiellen Vorgehensmodellen mit vordefinierten Phasen, die alle durchlaufen werden müssen. Rücksprünge auf vorherige Phasen sind erlaubt, die Phasen müssen jedoch nacheinander durchlaufen werden.

Das Wasserfallmodell ist vor allem in den klassischen Ingenieursdisziplinen vertreten, wie z. B. im Anlagenbau. Die Phasen unterteilen sich in Anforderungsaufnahme, Anforderungsanalyse, Systementwurf, Systemimplementierung, Test, Inbetriebnahme und Wartung.

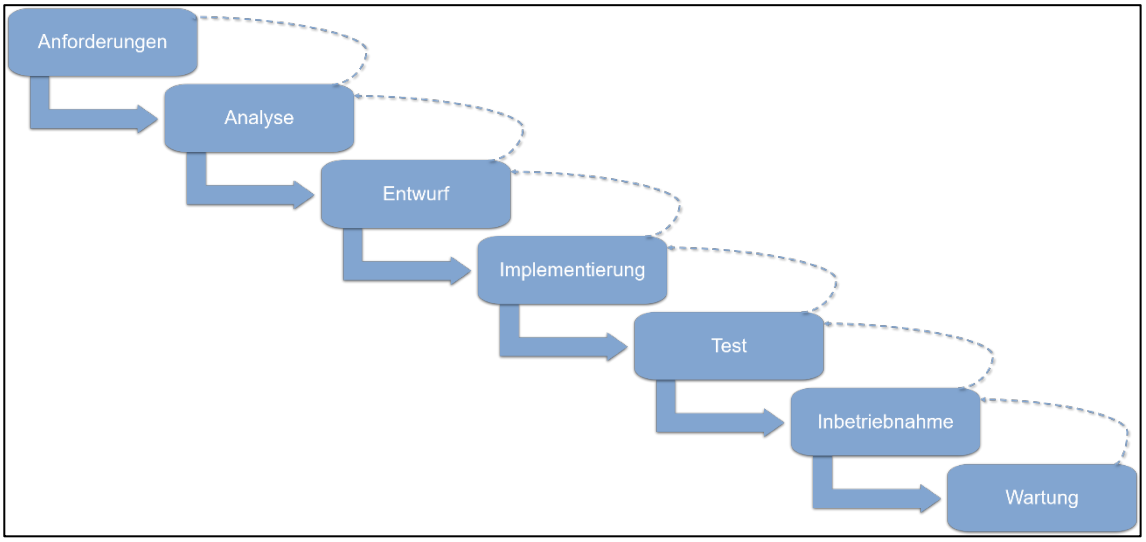


Abb. 68: Wasserfallmodell

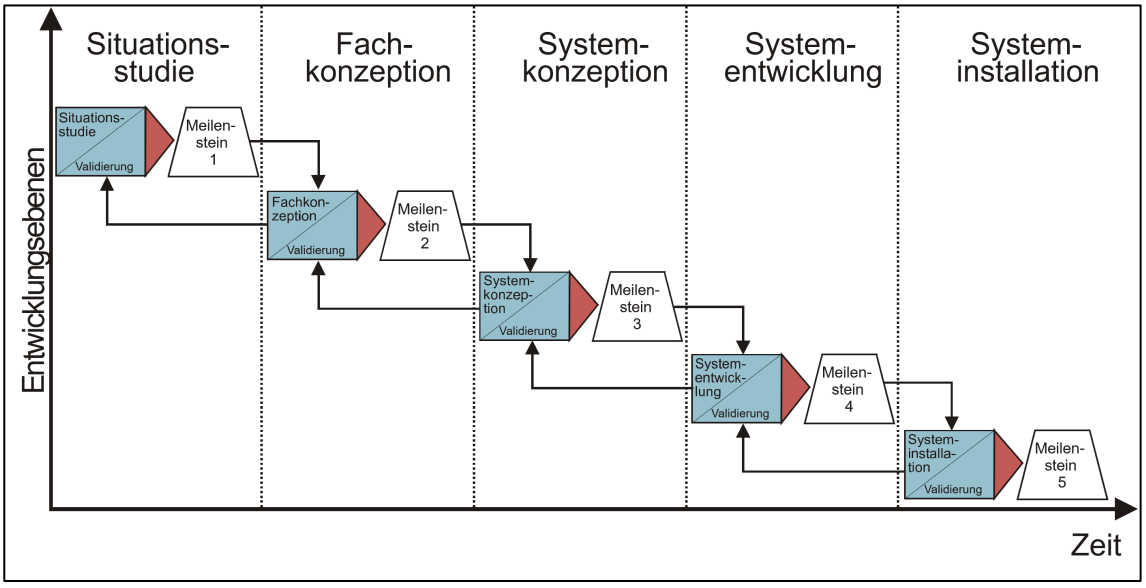


Abb. 69: Beispiel: Wasserfallmodell in den klassischen sequentiellen Phasen

6.1.4 Beispiel: Wasserfallmodell in den sequentiellen Phasen

Das Wasserfallmodell entspricht den klassischen streng sequentiellen Phasen des allgemeinen Vorgehensmodells. Beim Wasserfallmodell „fallen“ die Ergebnisse von einer Phase in die nächste. Das heißt, der erreichte Meilenstein in der Phase der Situationsstudie wird als Basis für die Fachkonzeption genutzt. Erst wenn dieser erreicht ist, kann mit der

Phase der Fachkonzeption begonnen werden. Das Wasserfallmodell bildet heute meistens die Basis für komplexere Modelle, wie z. B. das V-Modell.

6.1.5 Beispiel: V-Modell

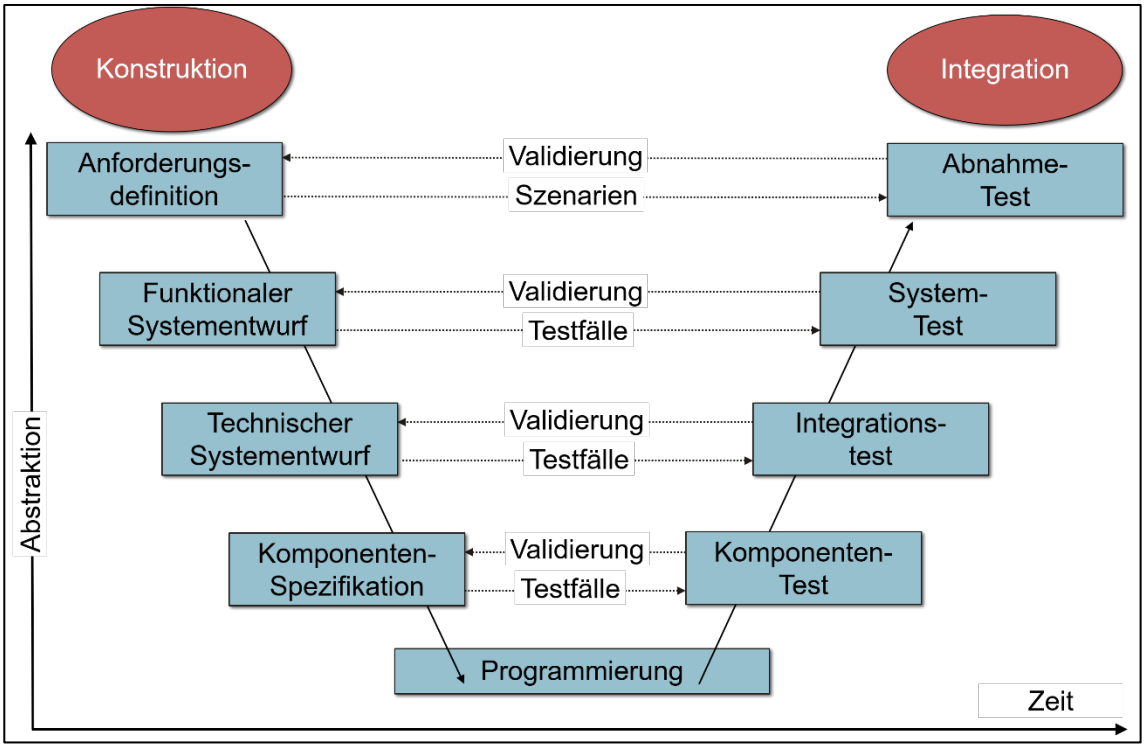


Abb. 70: V-Modell

Das V-Modell wurde in den 90er Jahren als eine Art Weiterentwicklung des Wasserfallmodells konzipiert. Es besteht aus denselben Phasen, wie das Wasserfallmodell, folgt aber einem anderen Aufbau.

Zusätzlich zu den jeweiligen Konstruktionsphasen eines Projekts definiert das V-Modell parallel die begleitenden Vorgehensweisen zur Qualitätssicherung. Die Integrationsphasen beschreiben, wie diese einzelnen Phasen miteinander interagieren können.

Die Grundform des V-Modells ist unten abgebildet. Auf der linken Seite wird in einem wasserfallartigen Prozess herausgearbeitet, was entwickelt werden soll. Dies mündet in der Programmierung. Auf der rechten Seite wird die Entwicklung durch Testfälle und Szenarien verifiziert. Dies entspricht der Qualitätssicherung durch ein systematisches Gegenüberstellen.

Zusätzlich werden den verschiedenen Phasen und Tätigkeiten Methoden zugeordnet. Diesen Methoden werden dann wiederum Werkzeuge zugeordnet. Diese Vorgaben machen das V-Modell relativ starr und unflexibel, weil Phasen, Abläufe, Methoden und Werkzeuge fest vorgegeben sind.

6.1.6 Merkmale und Eignung sequentieller Vorgehensmodelle

Sequentielle Vorgehensmodelle: **Merkmale**

- Auch „Phasenkonzepte“ genannt
- Folgen dem Prinzip der „schrittweisen Verfeinerung“
- Phasenergebnisse („Meilensteine“) pro Phase zu definieren
- Folgephase beginnt, wenn vorhergehende Phase vollständig abgeschlossen ist
- Wasserfallmodell sieht (die in der SW-Entwicklung gängigen) Rücksprünge vor

Sequentielle Vorgehensmodelle: **Eignung**

- Anforderungen an zu entwickelndes Informations- und Kommunikationssystem liegen präzise vor
- Umfangreiche, aber wenig komplizierte Informations- und Kommunikationssysteme
- Informations- und Kommunikationssysteme mit relativ stabilem Projektumfeld
- Informations- und Kommunikationssysteme, die relativ viel Arbeitsteilung erfordern

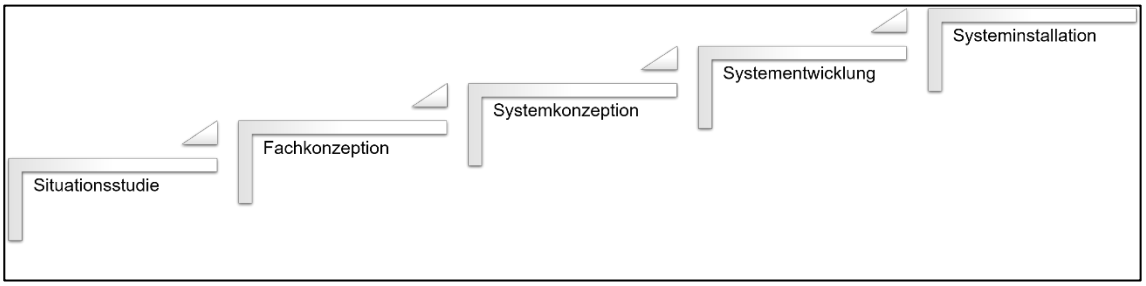


Abb. 71: Sequentielle Phasen

6.2 Parallel-sequentielle Vorgehensmodelle

6.2.1 Parallel-sequentielles Vorgehensmodell: Arbeitspakete

Das parallel-sequentielle Vorgehensmodell stellt eine weitere Fortentwicklung des sequentiellen Vorgehensmodells dar. Es greift dabei einen wichtigen Effizienz-Gedanken auf, indem gewisse Arbeitsabläufe „parallel“ durchlaufen werden. Wenn Aktivitäten gleichzeitig bearbeitet werden können, verkürzt dies die Projektdauer und das Projekt kann schneller fertiggestellt werden.

Parallel-sequentielle Vorgehensmodelle ermöglichen dem Projektmanager, Phasen überlappend anzuordnen. Dies ist jedoch nur möglich, wenn die Arbeitspakete (AP) in sich geschlossene Aufgaben darstellen. Arbeitspakete aus einer vorherigen Phase müssen somit noch nicht vollkommen abgeschlossen sein, um mit dem Hauptprojekt zu beginnen. Es ist aber wichtig, dass keine Abhängigkeiten zwischen den Arbeitspaketen bestehen.

Die vollständige Abarbeitung eines Arbeitspaketes vor dem Beginn einer neuen Phase, wird somit explizit vernachlässigt, um Ressourcen effizienter nutzen zu können.

„Ein Arbeitspaket beschreibt eine zu erbringende Leistung innerhalb eines Projekts, die von einer einzelnen Person oder organisatorischen Einheit bis zu einem festgelegten Termin mit vereinbartem Aufwand geliefert werden kann. Ein Arbeitspaket kann eine interne Struktur aufweisen, deren Elemente aber aus Sicht der Projektleitung nicht einzeln betrachtet werden müssen, sondern als „Paket“ gehandhabt werden können. [...]“

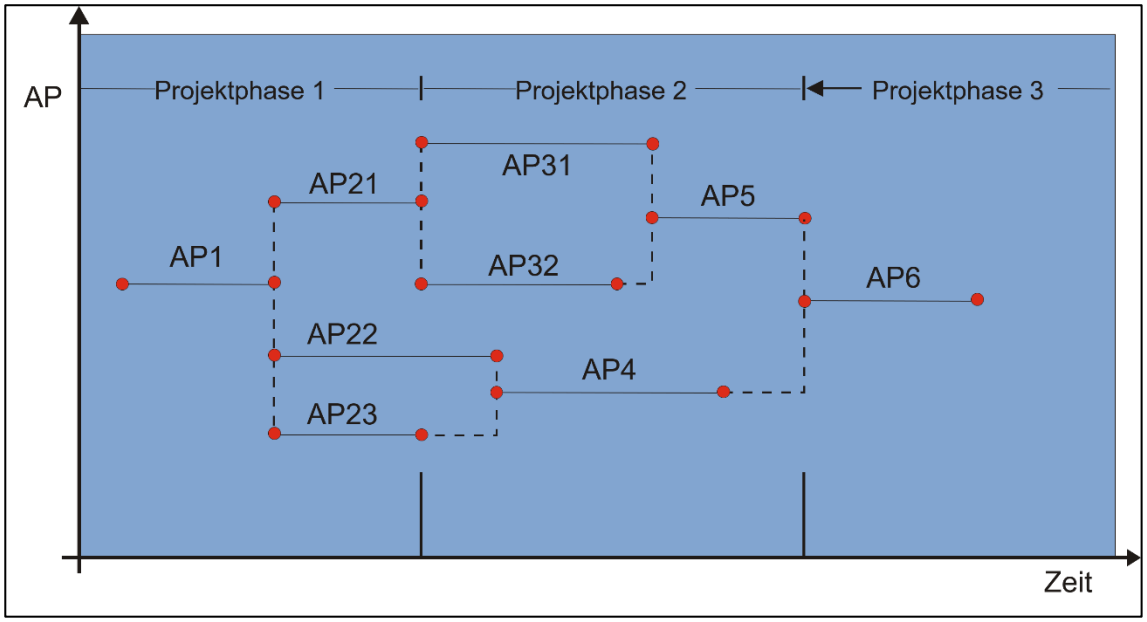


Abb. 72: Arbeitspakete und simultanes Arbeiten

6.2.2 Merkmale und Eignung parallel-sequentieller Vorgehensmodelle

Abb. 73: Parallel-sequentielles Vorgehensmodell

Parallel-sequentielle Vorgehensmodelle: **Merkmale**

- Überlappte, zeitlich versetzte Arbeitsschritte phasenintern und phasenübergreifend
- Keine Fixierung mehr auf umfassende, monolithische Phasenresultate
- Stattdessen kleinere Leistungseinheiten („Arbeitspakete”)
- Im V-Modell können die Arbeitspakete durch Bündel aus Submodellen dargestellt werden (Projektmanagement, Qualitätsmanagement, Konfigurations-Management, Software-Erstellung). Dadurch ist eine Phasenüberlappung möglich.

Parallel-sequentielle Vorgehensmodelle: **Eignung**

- Für komplexere Projekte (V-Modell gültig im öffentlichen Sektor)
- Realitätsnäher als rein sequentielle Vorgehensmodelle
- Kleinere, einzeln abprüfbare Teilkomponenten
- Änderungen im Projektumfeld können flexibler berücksichtigt werden
- Nachteil: Vergleichsweise hoher Koordinationsaufwand

6.2.3 Beispiel: Simultaneous Engineering / Concurrent Development

Der Begriff Simultaneous Engineering wird seit 1989 in den USA verwendet. Von Simultaneous Engineering ist hauptsächlich die Rede, wenn die Wettbewerbsfähigkeit gesteigert werden soll, indem man die Produktionszeit eines Produktes verringert, aber dennoch dessen Qualität und Kosten verbessert. Die Verbesserung wird dadurch erlangt, dass die Phase der Produktentwicklung mit der Phase des Designs und der Produktion integriert wird.

Dieser Ansatz lässt sich wie unten beschrieben, auch in das Software Engineering überführen.

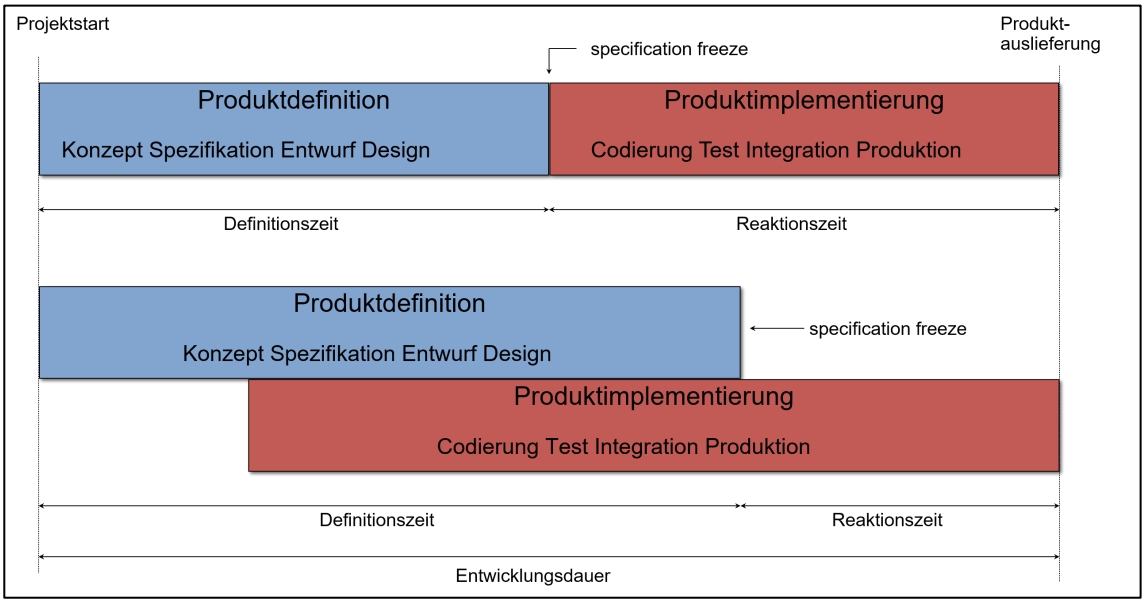


Abb. 74: Concurrent Development

Ein Stück Software kann, wie ein Produkt, immer weiterentwickelt werden. Während beispielsweise noch am Design der Software gearbeitet wird, kann schon mit der Codierung der Software begonnen werden. Dadurch verlängert sich die Definitionszeit, aber die Reaktionszeit verkürzt sich, da schnell auf Änderungen eingegangen werden kann. In manchen Projekten ist es sogar notwendig erste Entwürfe zu codieren, um die Software „greifbar“ zu machen und sie auf ihre Eignung zu testen.

6.2.4 Merkmale und Eignung Simultaneous Engineering

Simultaneous Engineering: **Merkmale**

- Entwicklung kann begonnen werden, bevor Anforderungen feststehen.
- Entwicklung muss begonnen werden, bevor Anforderungen feststehen.
- Geeignet, wenn Produktarchitektur frühzeitig festgelegt werden kann.
- Hilfreich, wenn sich Anforderungen während der Entwicklungszeit verändern.

Simultaneous Engineering: **Eignung**

- Kundennähe: In welchem Maß werden Kunden(anforderungen) in den Prozess eingebunden?
- Innovations- und Reaktionsfähigkeit: In welchem Maß ist der Prozess geeignet, mit Veränderungen umzugehen?
- Reaktionsschnelligkeit: Fähigkeit, möglichst schnell auf (unvorhergesehene) Ereignisse reagieren zu können.
- Roll-Out-Fähigkeit: Das zu entwickelnde Produkt ist zu (fast) jedem beliebigen Zeitpunkt während der Entwicklung auslieferbar.

6.3 Evolutionäre Vorgehensmodelle

6.3.1 Allgemeines Spiralmodell

Evolutionäre Vorgehensmodelle unterscheiden sich von sequentiellen Vorgehensmodellen dahingehend, dass auf vordefinierte Meilensteine und Sequenzen verzichtet wird. Von Beginn an wird direkt am Prototypen gearbeitet, der sich über das gesamte Projekt stetig weiterentwickelt, bis er am Ende das fertige Produkt darstellt. Die Phasen, hier Zyklen genannt, werden genutzt, um den Prototypen zu verbessern. Rückschritte sind dabei gewollt, sobald ein voriger Prototyp näher am gewünschten Ergebnis liegt. Beispiele für dieses sogenannte „Prototyping" sind:

- das Spiralmodell von Boehm,
- Rapid Development
- oder das Clustering.

Alle evolutionären Vorgehensmodelle haben gemein, dass die Zwischenergebnisse aufeinander aufbauen und dem Kunden schneller „greifbare" Ergebnisse geliefert werden können.

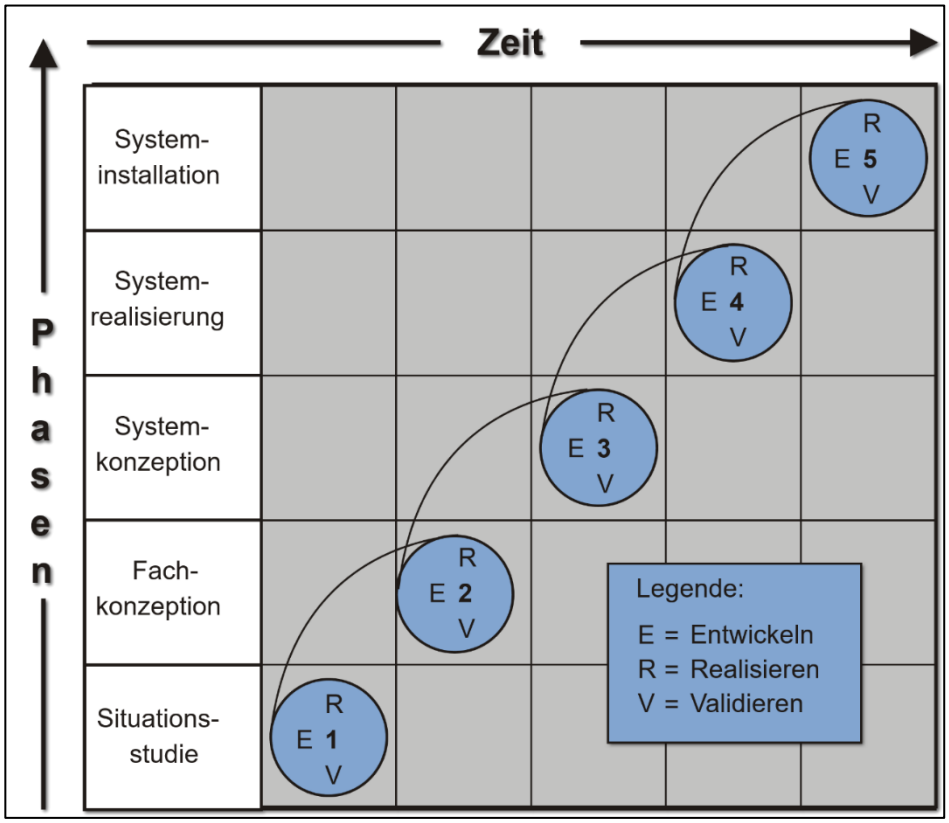


Abb. 75: Schema des Vorgehens im Spiralmodell

6.3.2 Beispiel: Spiralmodell nach Boehm

Das Spiralmodell ist es ein risikogetriebenes Vorgehensmodell explizit für die forschende Entwicklung. Dies bedeutet, dass das Ergebnis nicht von Anfang an voll definiert wird und explorativ bzw. zyklisch erarbeitet wird. Die erarbeiteten Zwischenergebnisse werden auf Basis ihres analysierten Risikos weiterentwickelt. Ein Zwischenergebnis heißt im Spiralmodell „Prototyp", die Arbeitsphasen „Iterationen". Das Spiralmodell beinhaltet dabei vier Schritte:

1. **Risikoanalyse:** Welchen Rahmenbedingungen unterliegt das Projekt? Welche Ziele, Lösungen und eventuelle Alternativen müssen Sie planen?
2. **Evaluierung**: Welche Alternative ist unter Berücksichtigung der Risiken zu erstreben? Welche Strategien helfen bei der Risikominimierung?
3. **Vorgehen:** Welche Schritte sind in der nächsten Iteration sinnvoll?
4. **Review und Planung:** Waren unsere Schritte sinnvoll und wie sollten wir die nächste Iteration gestalten?

Diese einzelnen Schritte werden dabei so lange durchlaufen, bis ein Prototyp dem gewünschten Endprodukt entspricht.

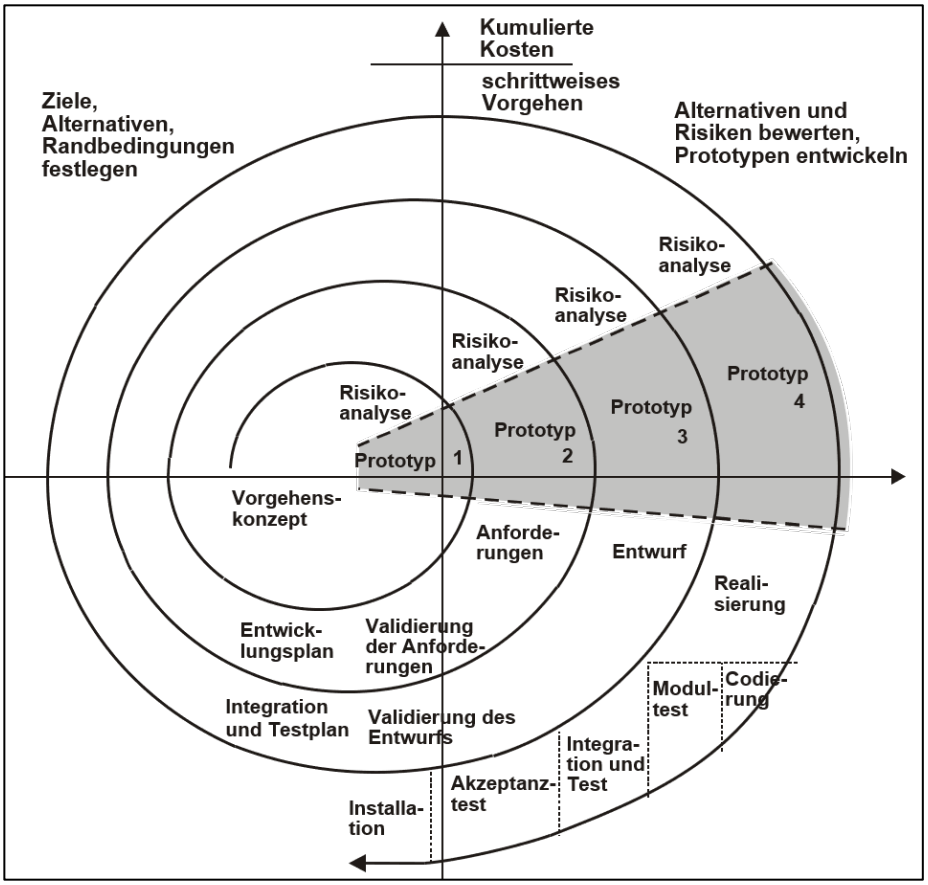


Abb. 76: Beispiel eines Spiralmodell mit Prototyping

6.3.3 Beispiel: Spiralmodell im Software Engineering

Das Spiralmodell ist ein iterativer Prozess, der die Kontrolle und Minimierung der Projektrisiken zum Ziel hat. Iterativ bedeutet, dass die Schritte Entwerfen, Implementieren und Validieren immer wieder durchlaufen werden. Das eigentliche Vorgehen beim Spiralmodell ist daher nicht ausgestaltet, sondern ergibt sich aus den Schritten der vorherigen Iteration.

Durch die iterative Vorgehensweise wird das zu entwickelnde System schrittweise ausgebaut. Ein anfänglicher Prototyp wird so kontinuierlich weiterentwickelt. Beispielsweise könnte in einem ersten Schritt eine App auf Papier skizziert werden.

Die Skizze wird kurzerhand umgesetzt und dem Kunden präsentiert, der dann sein Feedback dazu gibt. Auf Basis dieses Feedbacks kann die Weiterentwicklung der App erfolgen, bis ein erstes Produkt, ein Minimal Viable Product (MVP), entwickelt wurde. Bis ein MVP entsteht, werden mehrere Feedback-Zyklen durchlaufen.

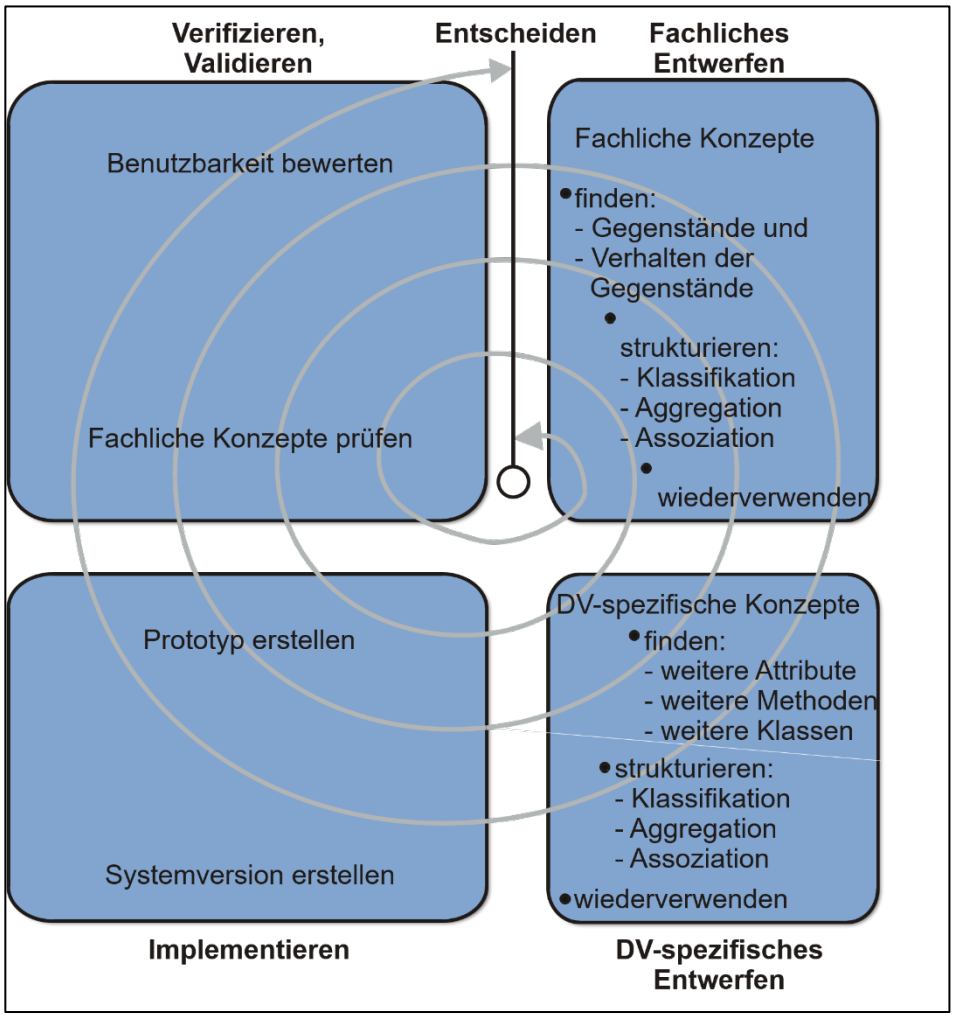


Abb. 77: Spiralmodell im Software Engineering

6.3.4 Beispiel: Cluster-Bildung

Das Clustern leitet sich aus der Notwendigkeit ab, große Software-Systeme in kleinere Subsysteme zu unterteilen. Dies bricht die Komplexität herunter und macht das Projekt beherrschbarer. In der unteren Grafik wird deutlich, dass das Clustering nach dem Entwurf der Grobarchitektur ansetzt.

Im Software Clustering werden aus dem Gesamtprojekt eines Software-Systems Teilprojekte („Cluster“) herausgebildet, die unabhängig voneinander entwickelt werden können. Diese Teilprojekte durchlaufen danach die bereits bekannten Phasen des allgemeinen Vorgehensmodelles.

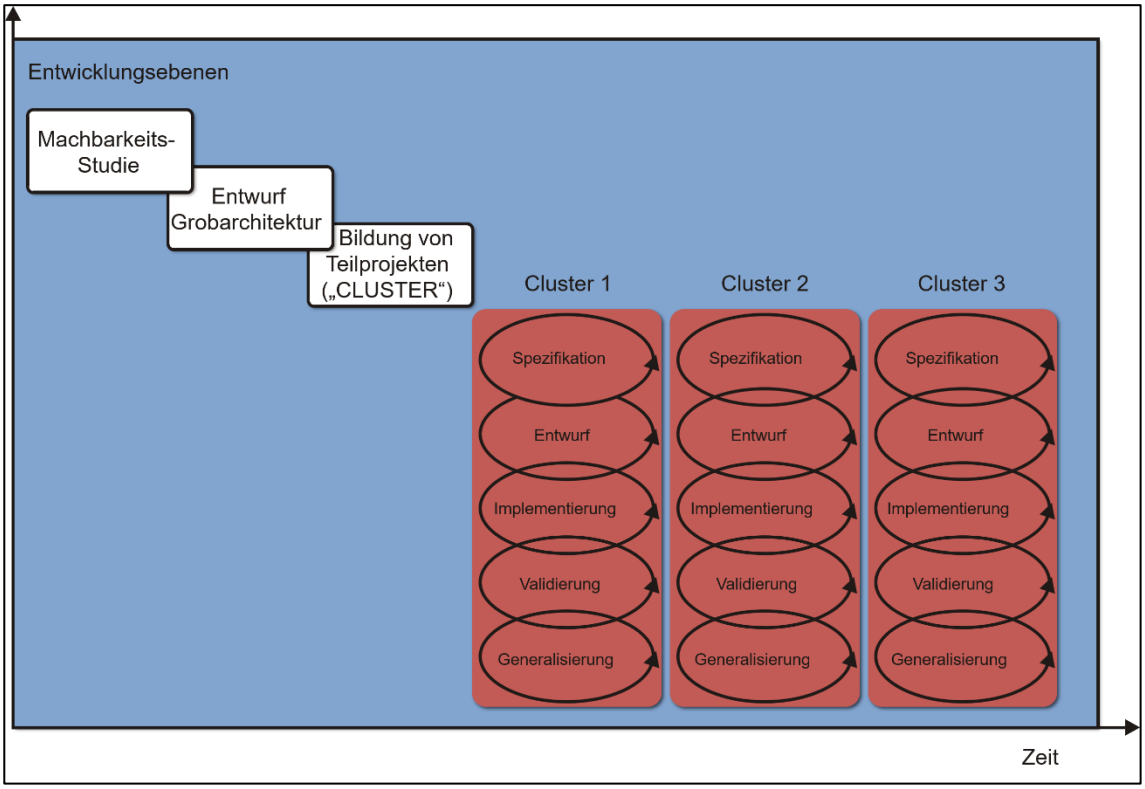


Abb. 78: Clustering als evolutionäres Vorgehensmodell

6.3.5 Merkmale evolutionärer Vorgehensmodelle

Evolutionäre Vorgehensmodelle: **Merkmale**

- Weitgehender Verzicht auf Sequentialisierung und vordefinierte Zwischenergebnisse
- Zwischenresultate werden durch „systematisches Probieren” in zyklisch gestufter Abfolge von Entwerfen, Realisieren und Validieren erzeugt.
- Grundlage „Prototyping”: explorativ, experimentell, evolutionär
- Spiralmodell (Böhm): inkrementell-iteratives Vorgehen

Evolutionäre Vorgehensmodelle: **Eignung**

- Innovative, komplexe Informations- und Kommunikationssysteme
- Im Voraus schwierig zu strukturierenden Informations- und Kommunikationssystemen
- Rapid Application Development
- Nachteil: Meilenstein-Zäsuren verschwimmen

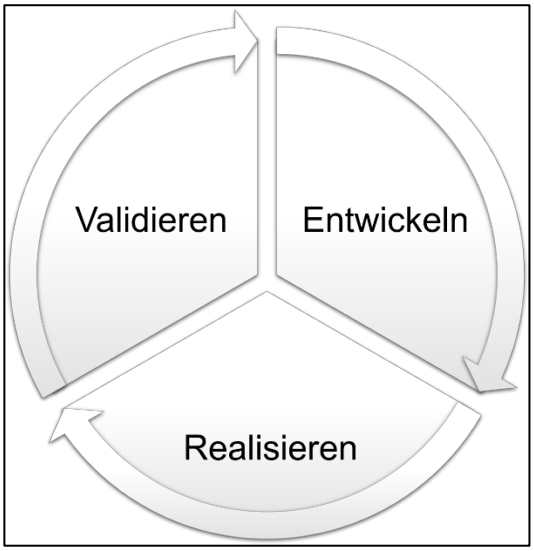


Abb. 79: Vorgehen bei evolutionären Vorgehensmodellen

6.3.6 Prototyping als evolutionäre Praktik

Prototyping: **Merkmale**

- Explorativ: Spezifikations- und Kommunikationshilfe zwischen Fachabteilung und Entwicklern
- Experimentell: Technisch-funktionsfähige Baumuster für Tests
- Evolutionär: Sukzessive Weiterentwicklung von „Versionen“
- d. R.: Punktuelles Prototyping in anderen Vorgehensmodellen
- Prototyping ist kein Vorgehensmodell, sondern eine Technik/Praktik.

Prototyping: **Eignung**

- „Ich kann Ihnen nicht sagen, was ich will, aber ich sage es Ihnen, wenn ich es sehe.“
- Es werden frühzeitig funktionsfähige Systemteile entwickelt.
- Auftraggeber bekommt frühzeitig einen realistischen Einblick in das Endprodukt
- Problem: Gesamtarchitektur

6.4 Abschlusstest – WBT06

6.4.1 Abschlusstest

Bitte beantworten Sie die folgenden Fragen durch das Ankreuzen der korrekten Antworten (Tab. 7). Bei einigen Fragen können auch mehrere Antworten richtig sein.

Nr.	Frage	Richtig	Falsch
1	Sequentielle Vorgehensmodelle sind bei Standard-Software-Einführungen zu bevorzugen.		
2	Clustering ist unter die parallel-sequentiellen Vorgehensmodelle einzusortieren.		
3	Das allgemeine Vorgehensmodell besteht aus folgenden Phasen....		
	Vorprojekt		
	Hauptprojekt		
	Detailprojekt		
	Systembau, Systemeinführung und -übergabe		
	Systemnutzung		
4	Evolutionäre Vorgehensmodelle sind mit agilen Vorgehensmodellen identisch.		
5	Die Grundformen der Vorgehensmodelle sind...		
	die allgemeinen Vorgehensmodelle		
	die sequentiellen Vorgehensmodelle		
	die evolutionären Vorgehensmodelle		
	die agilen Vorgehensmodelle		
	die strukturierten Vorgehensmodelle		

Tab. 7: Abschlusstest – WBT06

6.5 Typische Aufgabenstellungen

Typische Aufgabenstellungen – Sequentielle und Evolutionäre VGM

Zur Bearbeitung dieser Aufgabenstellungen beachten Sie bitte: Verlangt ist eine fachlich zutreffende, inhaltlich nachvollziehbare und kausal zusammenhängende Erörterung aus vollständigen Sätzen in lesbarer Handschrift. Für jede Aufgabe: Maximal zwei Seiten Text.

Aufgabe 1:

> Erläutern Sie die wesentlichen Merkmale von sequentiellen VGM. Erläutern Sie, für welche Entwicklungsprojekte sich diese VGM besonders eignen.

Aufgabe 2:

> Erläutern Sie die wesentlichen Merkmale von evolutionären VGM. Erläutern Sie, für welche Entwicklungsprojekte sich diese VGM besonders eignen.

Aufgabe 3:

> Erläutern Sie das Problem „Dokumentation“ bei sequentiellen und evolutionären VGM.

Aufgabe 4:

> Stellen Sie die Merkmale und Eignungsbereiche von sequentiellen und evolutionären Vorgehensmodellen zur Software-Entwicklung kritisch vergleichend gegenüber.

Aufgabe 5:

> Welchem VGM-Typ entsprechen die Anleitungen beim Lego-Modellbau und die Aufbauanleitungen für Ikea-Möbel?

Aufgabe 6:

> Mit welchem VGM-Typ erstellen Sie Ihre Thesis? Warum?

Aufgabe 7:

> Erläutern Sie die Zusammenhänge zwischen den beiden VGM-Typen „sequentiell“ und „evolutionär“ auf der einen Seite und den Ihnen bekannten Formen der Arbeitsteilung (Fließband, Werkstatt) auf der anderen Seite.

7 Agile Vorgehensmodelle

7.1 Prinzipien der agilen Vorgehensmodelle

7.1.1 Die Forderung nach Agilität

Die Forderung nach mehr Agilität in der Projektvorgehensweise kam besonders Ende der 90er Jahre auf. Man verwies darauf, dass Softwareentwicklung keine Fließbandarbeit ist, sondern das Wissen in den Mittelpunkt stellt. Dementsprechend sind nicht nur andere Management-Methoden, sondern auch eine andere Unternehmenskultur notwendig.

Das Thema effektives Wissensmanagement bildete deshalb auch bei der Entwicklung von agilen Methoden wie Scrum einen zentralen Punkt. Viele agile Methoden legen ein Augenmerk auf den bestmöglichen Transfer von Wissen: Wissen zwischen Kunde und Consultant oder zwischen den Teammitgliedern selbst.

Darüber hinaus stellen die agilen Methoden die Frage, ob eine vollkommene Anforderungsanalyse in Software-Projekten sinnvoll ist. Ist es wirklich möglich, durch eine einzige Aufnahme der Anforderungen, eine Software zu entwickeln, die vollkommen die Wünsche des Kunden erfüllt?

In diesem WBT lernen Sie einige agile Praktiken kennen, die sich alle auf das agile Manifest berufen.

7.1.2 Die Entstehung der agilen Projektdenkweise

Beruhend auf den Forderungen nach einem Wandel in der Software-Entwicklung wurde das Agile Manifest 2001 von 17 Software-Entwicklern verfasst. Bis heute ist es eine wichtige Grundlage für die Vorgehensweise in Software-Projekten.

Die bekanntesten Vertreter sind Jeff Sutherland und Ken Schwaber, bekannt geworden durch die Scrum Methodik, Ken Beck, Erfinder des Extreme Programming und Alistair Cockburn, der die Crystal-Methodenfamilie für IBM entwickelte.

Das Agile Manifest ist jedoch nicht als starre Leitlinie und Regelwerk zu verstehen, sondern als Rahmen. Das Manifest betont, was in der Software-Entwicklung besonders beachtet werden sollte und welche Schwerpunkte für eine erfolgreiche Projektumsetzung gesetzt werden müssen. Dies geschieht durch die Vorgabe von vier Leitlinien und zwölf Prinzipien.

7.1.3 Das Agile Manifest

Abb. 80: Das Agile Manifest

7.1.4 Die 12 Prinzipien des Agilen Manifests

- Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufrieden zu stellen.
- Heiße Anforderungsänderungen sind selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.
- Liefere funktionierende Software regelmäßig innerhalb weniger Wochen oder Monate und bevorzuge dabei die kürzere Zeitspanne.
- Fachexperten und Entwickler müssen während des Projektes täglich zusammenarbeiten.
- Errichte Projekte rund um motivierte Individuen. Gib ihnen das Umfeld und die Unterstützung, die sie benötigen und vertraue darauf, dass sie die Aufgabe erledigen.
- Die effizienteste und effektivste Methode, Informationen an und innerhalb eines Entwicklungsteams zu übermitteln, ist im Gespräch von Angesicht zu Angesicht.
- Funktionierende Software ist das wichtigste Fortschrittsmaß.

- Agile Prozesse fördern nachhaltige Entwicklung. Die Auftraggeber, Entwickler und Benutzer sollten ein gleichmäßiges Tempo auf unbegrenzte Zeit halten können.
- Ständiges Augenmerk auf technische Exzellenz und gutes Design fördert Agilität.
- Einfachheit - die Kunst, die Menge nicht getaner Arbeit zu maximieren - ist essenziell.
- Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams.
- In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann und passt sein Verhalten entsprechend an.

7.1.5 Übersicht agiler Praktiken

Nachdem Sie nun wissen,

- warum das agile Manifest entwickelt wurde,
- welchen Schwerpunkt das Manifest setzt und
- welche Prinzipien sich daraus ableiten,

schauen Sie sich nun die verschiedenen Ausprägungen agiler Vorgehensmodelle an. Dazu gehören unter anderem:

- Xtreme Programming (XP)
- Scrum
- Crystal

7.2 Agile Praktiken

7.2.1 Wieso wurden agile Vorgehensmodelle entwickelt?

Ende der 90er Jahre wurde eine Tendenz weg von starren und schwergewichtigen Prozessen hin zu leichgewichtigen und flexiblen Prozessen sichtbar. Ursprünglich adaptiert von der Ingenieurskunst, die genaue Spezifikationen plangetreu umzusetzen, wurde sichtbar, dass die streng sequenzielle Vorgehensweise in Software-Projekten nicht sinnvoll ist.

Software-Entwicklungsprojekte sind meist komplex und durch Interdependenzen zunächst schwer planbar. Auch die Anforderungen der Kunden ändern sich schnell und technische Spezifikationen sind für Fachabteilungen beim Kunden meist nur schwer verständlich. Durch eine flexible, nutzergetriebene Entwicklung, wie es die agile Software-Entwicklung vorsieht, kann der Projekterfolg sichergestellt werden. Durch die frühe Einbindung von Kunden und die Beschränkung auf die Kernfunktionalität kann das Projekt

im geplanten Zeitraum, im geplanten Budget und mit den geplanten Anforderungen in der gewünschten Qualität umgesetzt werden.

Die agilen Methoden versuchen also, die Probleme der Komplexität durch eine ganz einfache Vorgehensweise zu lösen, um den Projekterfolg sicher zu stellen.

7.2.2 Extreme Programming (XP)

Xtreme Programming ist vor dem Hintergrund entstanden, die Qualität der Software zu steigern und in einem volatilen Projektumfeld zu entwickeln.

Besonders beachtet wurden bei der Konzeption die sich ständig ändernden Anforderungen der Kunden, welche schnell in ein Projekt eingearbeitet werden müssen.

Die Vorgehensweise ist also auf kurze Veröffentlichungszyklen ausgerichtet.

Im Folgenden werden die Praktiken beschrieben, die sich vor allem von den sequentiellen und evolutionären Vorgehensmodellen unterscheiden und sich von der Fließband-Mentalität abgrenzen lassen.

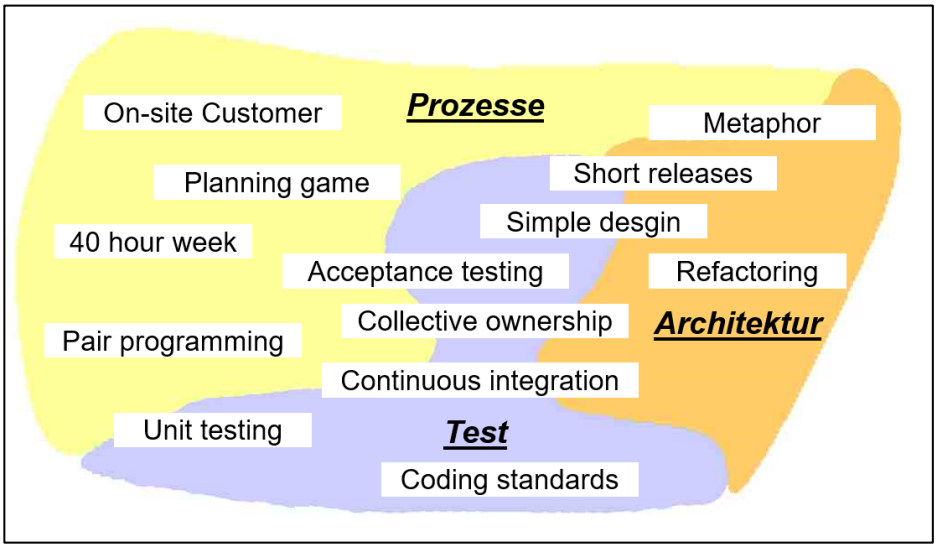


Abb. 81: eXtreme Programming: Prozesse, Architektur und Test

Werte und Prinzipien

- Kommunikation und Kundeneinbindung
- Feedback und inkrementelle Entwicklung
- Einfachheit und Mut zur Entscheidung

Praktiken

- Verstärken sich gegenseitig
- Zusammenspiel von Prozess, Test und Architekturfragen

7.2.3 XP: Planungssitzung

Anforderungen auf User Story Cards sammeln

- Kunde schreibt die Karten.
- Skizzen, Texte, auch Umgangssprache

Entwickler schätzen Kosten pro Karte

- Schätzung auf Erfahrungsbasis (Ähnlichkeit zwischen Karten)
- Abstrakte Bemaßung der Kosten (z. B. 1 bis 10 Punkte)
- Ab-/Rücksprache mit Kunden
- Fortschreibung der Karten mit Kostenvermerken (Story Points)

Auswahl von Karten für nächstes Release durch Kunden

- Nach Wichtigkeit, Nutzen priorisiert
- Entwickler-Team wird limitiert durch seine Produktivität

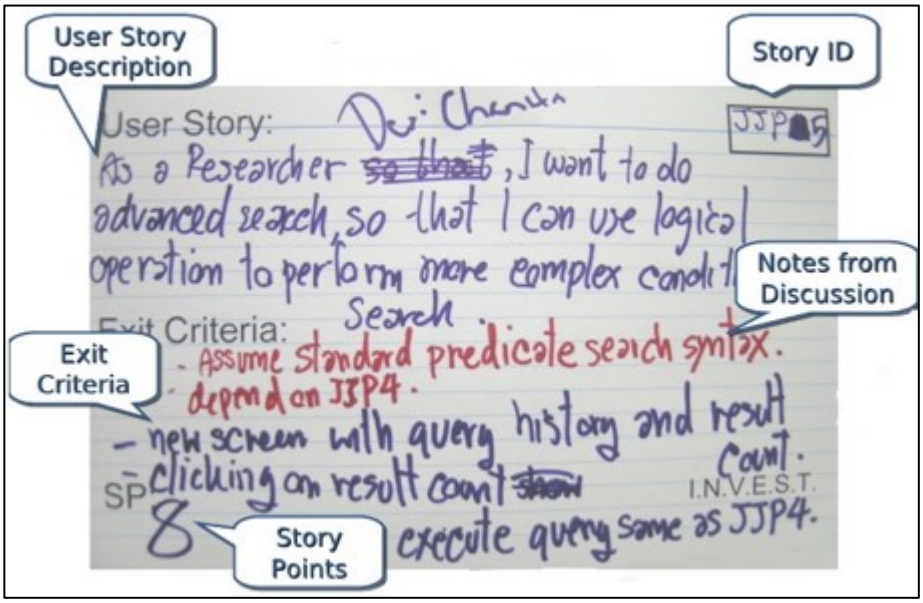


Abb. 82: eXtreme Programming User Story Cards

7.2.4 XP: Kunde vor Ort

Kunde ist über die gesamte Projektlaufzeit verfügbar

- Idealerweise direkt vor Ort ...
- ... oder zumindest jederzeit und schnell erreichbar.

Kunde schreibt und überprüft User Stories

- Anforderungen (Karten) werden ständig weiterentwickelt.
- Fragen der Entwickler werden sofort beantwortet.

Kunde trifft Entscheidungen

- Zeitverluste vermeiden durch direkte Verfügbarkeit
- Keine Allein-(Fehl-)Entscheidungen durch Entwickler

7.2.5 XP: Metapher

Vermeidung von latenten Missverständnissen

- In Softwareprojekten besteht häufig ein latentes Missverständnis zwischen Kunden und Entwickler-Team.
- Der Entwickler hat Schwierigkeiten mit der Fachsprache des Kunden und umgekehrt.

Metapher – ein rhetorisches Stilmittel

- Mit bildhaften Beschreibungen ausdrücken
- Bezüge zu konkreten Dingen

Schaffung einer gemeinsamen Terminologie

- Alle Metaphern, Bezeichnungen, Notationselemente etc. und jeweils deren Aussagen sind Inhalte eines Projekt-Glossars.
- Regeln einer gemeinsamen Sprache von Kunden und Entwicklern

7.2.6 XP: Kurze Release-Zyklen

Releases in Monats-/Wochen-Schritten

- Kurze Iterationen, um dem Kunden in regelmäßigen Zeitabständen einen lauffähigen Zwischenstand des Produkts zu liefern.
- Kurze Iterationen erlauben schnelle Feedbackschleifen zwischen Entwicklern und Kunden.

Projektfortschritte gemäß fertigen User Stories

- Visualisierung und Messbarkeit der Fortschritte
- Selbst bei Projektabbruch ist ein Nutzen sichtbar.

Nach dem Release ist vor dem Release.

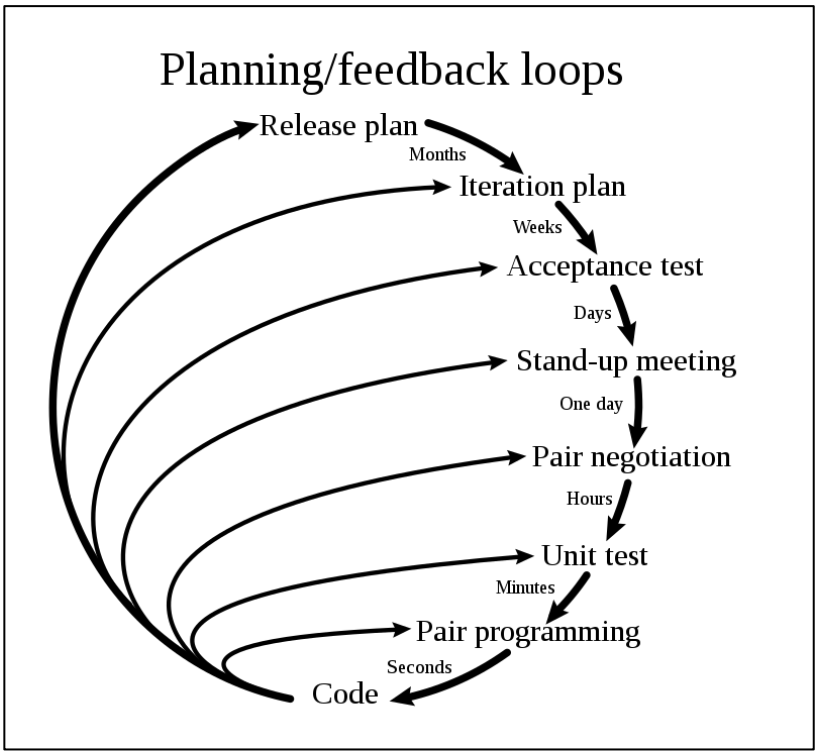


Abb. 83: eXtreme Programming Release Zyklen

- Nach einem Release: Neues Planning Game für das nächste Release
- Erstellung, Auswahl neuer Story Cards; evtl. neue Kostenschätzungen

7.2.7 XP: Einfacher Entwurf

Einfacher Entwurf

- Ist einfach zu verstehen, implementieren, ändern, testen
- Aber: Was ist „einfach“?

Keine konstruktive Vorausschau

- Es soll die einfachste Lösung angestrebt werden, also diejenige, die genau das Gewünschte erreicht (und nicht mehr).
- Bewusst allgemein (generisch) gehaltene Lösungen oder vorbereitende Maßnahmen für potentiell zukünftige Anforderungen werden vermieden.
- Zum Thema Einfachheit sind die umgangssprachlichen Akronyme KISS („Keep it small and simple“, „Keep it simple and stupid“) und YAGNI („You Ain't Gonna Need It“) verbreitet.

7.2.8 XP: Programmierer-Paare

Metapher: Fahrer und Beifahrer

- Zwei Programmierer arbeiten an einer Story Card.
- Einer codiert, der andere prüft.
- Regelmäßiger Tausch (täglich oder öfter)

Ständiges Miteinanderlernen

- Steigert Wissenstransfer, Wissensverteilung
- Anfänger lernen schneller von Spezialisten; Niveau-Angleichung
- Kann Ausfälle von Einzelnen besser kompensieren

Ständiger Code Review

- Vier-Augen-Prinzip
- Verhindert und findet Fehler

7.2.9 XP: Code Verantwortung und Standards

Gemeinsame Code-Verantwortung im Paar

- Jeder darf den Quellcode des anderen bearbeiten.
- Arbeitsteilung in Story Cards
- Kompensation der möglichen temporären, personellen Ausfälle

Code-Standards schaffen gemeinsames Verständnis

- Voraussetzung für gemeinsame Code-Verantwortung
- Leserlichkeit, „sprechender“ Quellcode
- Strukturstandards wie Klammerungen, Einrückungen, Namenskonventionen, Groß-/Kleinschreibung etc.

7.2.10 XP: Die Woche hat 40 Arbeitsstunden

Grundidee: Überlastung vermeiden

- Persönliche Limits für effektive Arbeitsleistung
- Mehr arbeiten, kann kontraproduktiv sein.
- Überlastung ist demotivierend und beeinträchtigt die Arbeitsqualität.

Überlastung führt zu ...

- ... Nachlässigkeiten im Planungsprozess
- ... Fehlern bei der Codierung
- ... Unzufriedenheit im Team

Überlastung ist Ergebnis falscher Planung

- Arbeit außerhalb der regulären Arbeitszeit wird im Einzelfall zwar geduldet, aber auf keinen Fall besonders entlohnt oder erwartet.

7.2.11 XP: Weitere Praktiken

Kollektives Eigentum

- Team im Vordergrund; Einzelne haben keine Wissensmonopole

Permanente Integration

- Komponenten in kurzen Zeitabständen lauffähig zusammenbauen

Permanentes Testen

- Anwendungstestfälle definiert vor Codierung; ständiges Testen

Refactoring

- Ständiges Verbessern des Codes; ständige Code-Reviews

Diverse „Begleitpraktiken“

- Team-Konstanz, tägliches Deployment, Kundeneinbeziehung etc.

7.2.12 Schematischer Vergleich Xtreme Programming

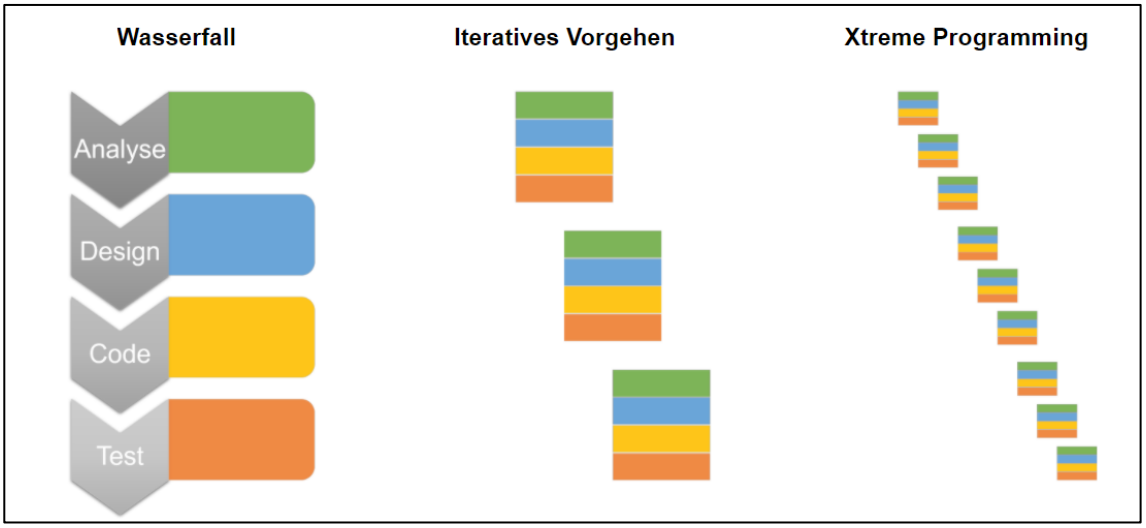


Abb. 84: Vergleich eXtreme Programming, Wasserfall und iteratives Vorgehen

Im Xtreme Programming wird im Vergleich zu sequentiellen und iterativen Vorgehensmodellen sehr viel kleinschrittiger vorgegangen. Die Phasen der Analyse, des Designs, der Codierung und des Testens werden beibehalten. Jedoch ist der Zyklus der Durchführung der Aktivitäten sehr viel engmaschiger als beispielsweise im Wasserfall-Modell. Dies bedingt natürlich eine sehr enge Zusammenarbeit mit dem Kunden.

7.2.13 Scrum – Das geordnete Gedränge

Scrum – die englische Bezeichnung für das angeordnete Gedränge im Rugby-Sport. Der Bezug zu Scrum als Prozess für Projektmanagement und Entwicklung liegt darin, dass das Team im Mittelpunkt steht und jeweils vor dem nächsten Scrum den geplanten Spielzug bespricht. Ohne diese Selbstorganisation des Teams funktioniert Scrum nicht.

7.2.14 Scrum – klein, kreativ, schnell

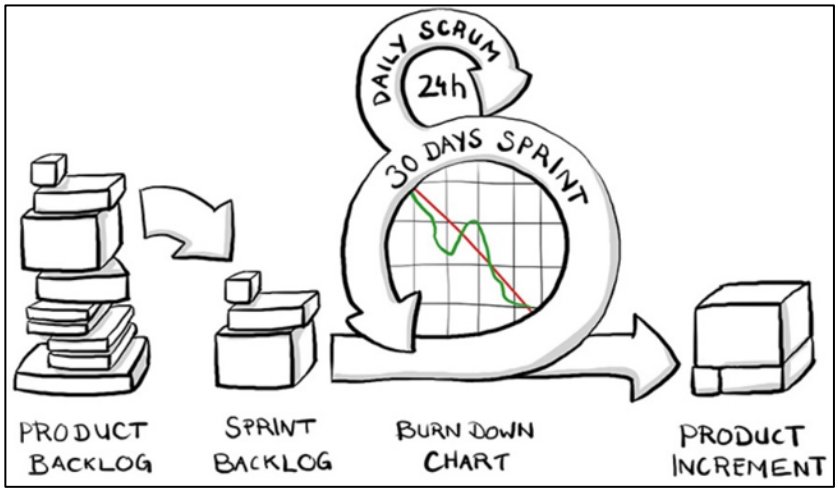


Abb. 85: Scrum Vorgehen

7.2.15 Scrum – Das Scrum-Team

Ein Scrum-Team besteht aus dem Product Owner, dem Entwicklungsteam und einem Scrum Master.

Das Entwicklungsteam ist selbstorganisiert und entscheidet ohne Projektleiter und äußeren Einfluss selbst, wie die Arbeit erledigt wird. Maximale Produktivität, Flexibilität und Verantwortung sind das Ergebnis.

Der Product Owner ist die Rolle, die alle Schnittstellen zu weiteren Kunden / Anforderungsträgern / Stakeholdern bildet. Er liefert dem Scrum-Team eine priorisierte Liste von Anforderungen.

Die Aufgabe des Scrum Masters ist es, dem Team ungestörte Arbeit zu ermöglichen und die Einhaltung der Scrum-Regeln zu überwachen. Der Scrum Master interagiert insofern auch mit dem produktverantwortlichen Product Owner, welcher in der Planungsphasen die Anforderungen an das Team heranträgt, und achtet auf dessen Zusammenarbeit mit dem Entwicklungsteam.

7.2.16 Scrum – Artefakte

Die Scrum Artefakte sind ein wesentlicher Bestandteil eines jeden Projektes, dass mit Scrum durchgeführt wird. Sie sind mit den Praktiken aus XP zu vergleichen und geben dem Projekt somit einen Ordnungsrahmen vor.

Product Backlog

In einem Product Backlog werden alle Anforderungen in Form von User Stories einer zu entwickelnden Software aufgenommen.

Sprint Backlog

In einem Sprint Backlog werden die User Stories aus dem Product Backlog eingeplant, die in einem Sprint zu entwickeln sind.

Thirty Day Sprint

Im 30-Tages Sprint werden nur die Anforderungen bzw. User Stories umgesetzt, die auch in einem Sprint eingeplant wurden. Neue Anforderungen werden in den Product Backlog überführt.

Daily Scrum

Der Daily Scrum findet jeden Tag während eines Sprints statt. Dort wird berichtet, welche Aufgaben gestern erledigt wurden, heute anstehen und welche Probleme bestehen.

Product Increment

Ein Produktinkrement besteht aus einem oder mehreren Sprints und bildet die Kernfunktionalität einer Software dar. Es ist das Ergebnis eines Sprintes.

Burn Down Chart

Das Burn Down Chart zeigt die Geschwindigkeit an, mit der ein Team die bepunkteten User Stories abarbeitet. Ein schnelles Team kann bspw. in einem Sprint mehr Punkte umsetzten.

7.2.17 Scrum – Ablauf von Scrum

Am Anfang eines jeden Sprints wird ein Sprint Planning Meeting abgehalten. In diesem befragt das Entwicklerteam den Product Owner so lange zu dessen Anforderungen höchster Priorität, bis klar wird, wie viele davon das Entwicklerteam zur Realisierung in den neuen Sprint aufzunehmen in der Lage ist. Der Product Owner wird auch zu möglichen Kostenersparnissen bei Änderung der Prioritäten beraten. Das Ergebnis ist die Liste der für den Sprint eingeplanten Anforderungen.

Die zweite Zeremonie ist das tägliche Abhalten des Daily Scrum. Dies ist ein ca. 15-minütiges Standup Meeting (Meeting im Stehen) des Entwicklerteams, bei dem jedes Teammitglied den vergangenen Tag resümiert, mögliche Arbeitshemnisse benennt und ausstehende Aufgaben formuliert. Jede eventuell entstehende Diskussion wird vom Scrum Master verhindert, um die Meetingzeit im Plenum zu minimieren.

Am Ende eines jeden Sprints findet das Sprint Review statt. Hier werden die Sprint-Ergebnisse vom Entwicklerteam präsentiert und vom Product Owner entweder abgenommen oder verworfen. Weitere anwesende Stakeholder liefern wertvolles zusätzliches Feedback. Der Scrum Master hält dieses für den Product Owner fest und moderiert im Anschluss noch eine Retrospective des Entwicklungsteams zur Entdeckung von Optimierungspotentialen.

7.2.18 Die Bedeutung von Stimmung in agilen Projekten

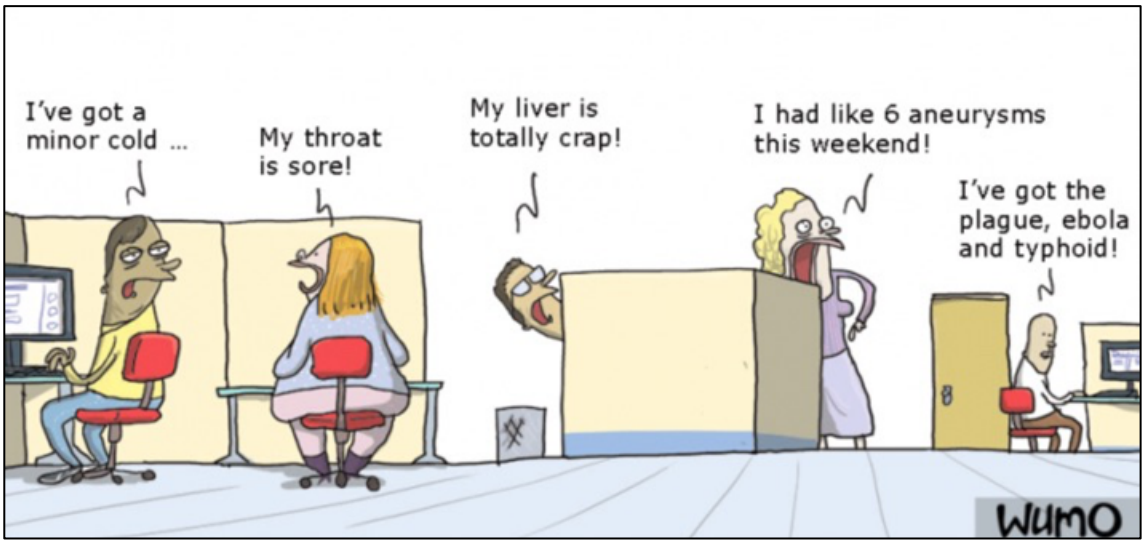


Abb. 86: Scrum und Stimmung zwischen Teammitgliedern

Die meisten agilen Methoden und Praktiken setzen auf ein selbstorganisiertes Team, das eng zusammenarbeitet. Die Aufgabenpakete sind nicht mehr stark getrennt wie in den sequentiellen Vorgehensmodellen, sondern werden zusammen abgearbeitet. Die Leistung eines Teams ist auch stark von der Stimmung im Team abhängig. Ein agiles Projekt muss deshalb besonders auf Team-Dynamiken achten.

7.2.19 Crystal-Methodenfamilie – Teil 1

- Entworfen von Alistair Cockburn Ende der 90er Jahre
- Software wird alle drei Monate (ein Inkrement) ausgeliefert.
- Team prüft Prozess mindestens 2 x pro Inkrement.
- Familie mit vier verschiedenen, nach Farben benannten Methoden
- Beispiel „E20“:
  Für Projekte mit 7 - 20 Entwicklern und mit möglichen essenziellen finanziellen Verlusten ist die Methode „Crystal Yellow“ zu wählen.

		Crystal Methodenfamilie				
		Crystal Clear	Crystal Yellow	Crystal Orange	Crystal Red	Crystal Maroon
Kritikalität des Projektes	Verlust von Menschenleben (L)	L6	L20	L40	L80	L200
	Essenzielle finanzielle Verluste (E)	E6	E20	E40	E80	E200
	Mäßige finanzielle Verluste (D)	D6	D20	D40	D80	D200
	Verlust an Komfort (C)	C6	C20	C40	C80	C200
		1 bis 6	7 bis 20	21 bis 40	41 bis 80	81 bis 200
		Anzahl der Personen, die am Projekt beteiligt sind				

Abb. 87: Crystal-Methodenfamilie

Die Crystal-Methodenfamilie besteht aus fünf Teilen.

Die Familienteile werden aufgeteilt in die Projektkritikalität und die Anzahl der am Projekt beteiligten Personen.

Das Projektrisiko wird dabei in vier verschiedene Kategorien geordnet: Verlust an Komfort, mäßige bis essenzielle finanzielle Verluste und der Verlust von Menschenleben.

7.2.20 Crystal-Methodenfamilie – Teil 2

- Cockburn extrahierte aus Dutzenden von Projekten 7 Prinzipien, die bei erfolgreichem Projekt berücksichtigt werden müssen.
- Die 7 Prinzipien bilden das Rückgrat der Crystal-Methoden.

Prinzipien	Neuartige Erkenntnis	Gesunder Verstand
1. Direkte Kommunikation zwischen zwei anwesenden Personen ist der preisgünstigste und schnellste Kanal, um Informationen auszutauschen.	?	X
2. Je größer das Team wird, umso mehr Regeln braucht das Team zur Koordination und um eine akzeptable Konformität der Ergebnisse zu erzielen.	?	X
3. Überflüssige Regeln in einer Methode kosten überproportinal viel Geld.	?	X
4. Je höher die Kritikalität eines Projektes ist, umso höher muss der Zeremonie-Anteil sein (von außen vollziehbare Interaktionen, Entscheidungen).	?	X
5. Formalismus | Prozess | Dokumentation ist nicht gleichzusetzen mit Disziplin | Fähigkeit | Verständnis.	?	X
6. Mehr Feedback und Kommunikation vermindern die Notwendigkeit von Zwischenprodukten.	?	X
7. Sinngemäß: „Schlechte" Team-Mitglieder sollen den anderen wenigstens helfen, dass deren Aufgaben schneller erledigt werden.	?	X

Abb. 88: Crystal: Sieben Prinzipien

7.2.21 Eignung der Crystal-Methoden

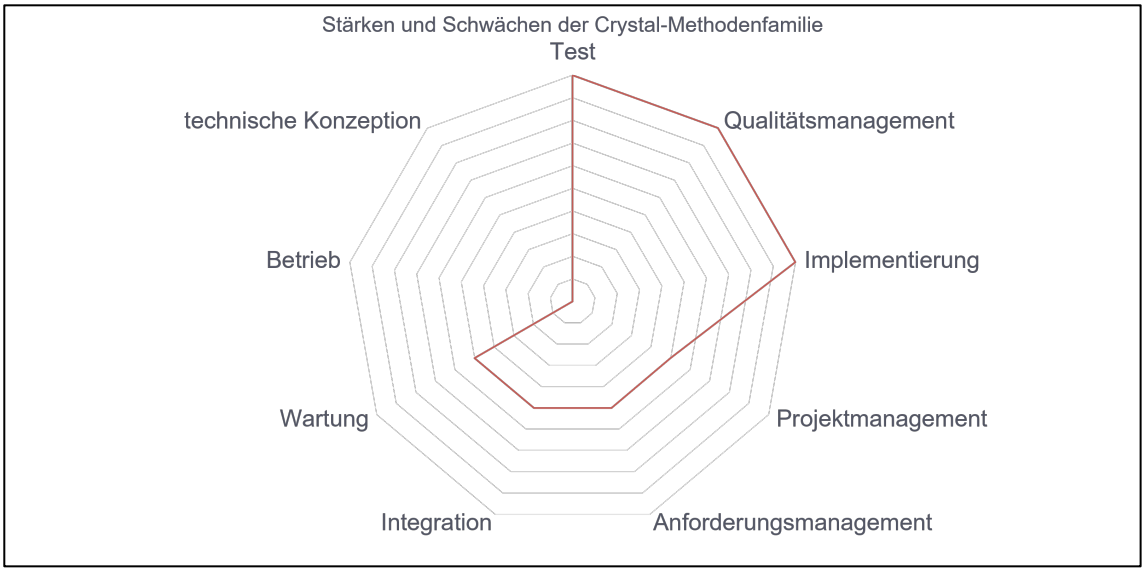


Abb. 89: Stärken und Schwächen der Crystal-Methodenfamilie

Crystal ist keine einzelne Methode zur Software-Entwicklung, sondern eine Sammlung bewährter Vorgehensweisen.

Ursprünglich wurde die Crystal-Methodenfamilie für die IBM Consulting Group von Altair Cockburn entwickelt. Cockburn hatte dabei die Erkenntnis, dass bei den analysierten

Projekten vor allem ein hoher Freiheitsgrad bei der Vorgehensweise zum Projekterfolg führte.

Jedes Projekt und jedes Team ist einzigartig. Aufgrund dessen erfolgt in Crystal eine grobe Abgrenzung von Software-Entwicklungsprojekten durch die beschriebenen Kriterien. Durch Crystal werden vor allem die Aufgaben des Qualitätsmanagements sowie Test und Implementierung gut abgedeckt. Der spätere Betrieb und das Systemdesign rücken bei Crystal jedoch stark in den Hintergrund.

7.2.22 Exkurs: Adaptive Software Development

Es sei an dieser Stelle erwähnt, dass es noch sehr viel mehr agile Vorgehensmodelle gibt. Mit Scrum und XP sind die beiden bekanntesten Modelle gezeigt worden. Mit der Crystal-Methodenfamilie und mit Adaptive Software Development sind zwei übergreifende Modelle benannt, die lediglich ein Rahmenkonstrukt bilden. Folgende Charakteristika sind typisch für das Adaptive Software Development:

- Schrittfolge: „Speculate, Collaborate, Learn“
- Durchlauf der Schrittfolge in Inkrementen von wenigen Wochen
- Speculate: Grober Plan des kommenden Inkrements
- Collaborate: Entwickler und Anwender produzieren gemeinsam
- Learn: Team lernt gemeinsam aus dem zu Ende gehenden Inkrement

Adaptive Software Development ist ähnlich wie die Crystal-Methodenfamilie kein Katalog an definierten Regeln und Aufgaben. Adaptive Software Development ist ein Framework für Konzepte, Praktiken und Guidelines.

Die Eckpfeiler von ADS bildet die oben beschriebene Schrittfolge aus Speculate, Collaborate und Learn. Darüber hinaus werden Ergebnis-Sichten wie das Adaptive Conceptual Model, Adaptive Development Model und Adaptive Management Model eingeführt sowie das CAS-Konzept (Concept of Complex Adaptive System) und das Adaptive Development Life Cycle Konzept.

Auch hier wird wieder deutlich, dass Vorgehensmodelle Verhaltensschablonen sind, die auf die jeweiligen Projektumstände angepasst werden müssen.

7.2.23 Gemeinsame Merkmale aller „agilen Verfahren“

- Kurze Inkremente (Time Boxes) von wenigen Wochen
- Enge Kommunikation im Team und mit dem Kunden
- Gute Kommunikation ersetzt einen Teil der Dokumentation.
- Hoheit über Entwicklungsprozess liegt beim Team.
- Prioritäten werden pro Inkrement mit dem Kunden erneut festgelegt.

- Auf Änderbarkeit wird großer Wert gelegt.
- Besonders geeignet für die Entwicklung von Web-basierten Systemen
- Alle in der Praxiserprobung; kaum wissenschaftlich aufgearbeitet

7.2.24 Einordnung agiler Verfahren

Neben den Gemeinsamkeiten der agilen Verfahren, gibt es dennoch auch wesentliche Unterschiede.

Der Grad der Detaillierung und Festlegung auf bestimmte Regeln variiert stark. Dies hat zur Folge, dass der Fokus des einzelnen Vorgehens sich verschiebt.

Während Adaptive Software Development und Scrum stark den Fokus auf das selbstorganisierte Team legen, welches weitgehend frei und flexibel agieren darf, setzt XP auf festgelegte Verhaltensnormen bzw. Praktiken. XP umfasst folglich eine starke Reglementierung des Verhaltens.

Die Crystal-Methodenfamilie jedoch setzt den Fokus auf die Effizienz. Nach dieser Überlegung hat jedes Projekt, abhängig von der Kritikalität, einen vorgegebenen Handlungsrahmen bzw. es werden andere Methoden empfohlen.

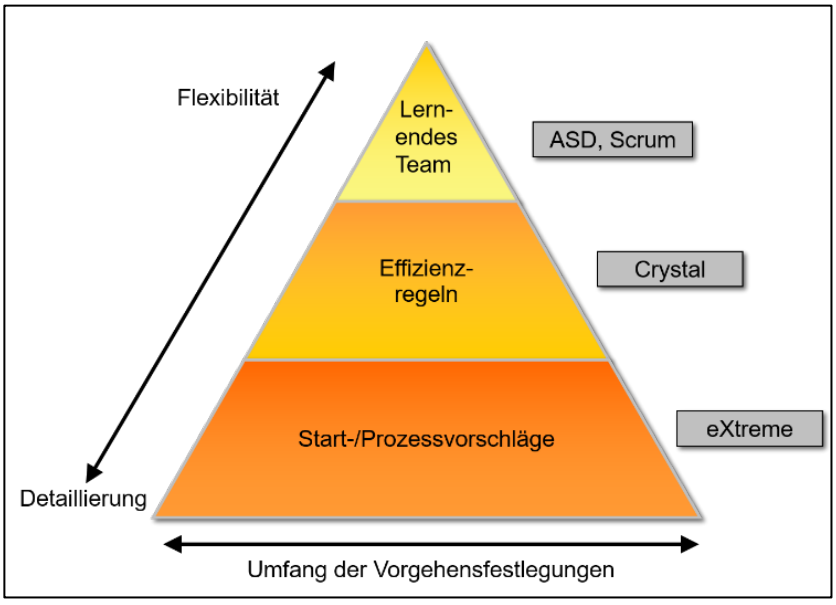


Abb. 90: Einordnung der agilen Vorgehensmodelle

7.3 Abschlusstest – WBT07

7.3.1 Abschlusstest

Bitte beantworten Sie die folgenden Fragen durch das Ankreuzen der korrekten Antworten (Tab. 8). Bei einigen Fragen können auch mehrere Antworten richtig sein.

Nr.	Frage	Richtig	Falsch
1	Agile Vorgehensmodelle haben eine flexible Sichtweise und unterliegen einer hohen Dokumentationslast.		
2	Die agilen Vorgehensmodellen besinnen sich zurück auf den Ansatz „Code and Fix“.		
3	Die Vorgehensweise in agilen Projekten ist...		
	inkrementell.		
	als Verhaltensschablone für alle Projektbeteiligten gedacht.		
	unspezifisch.		
	mit Methoden und Praktiken angereichert.		
	für jedes Projekt übernehmbar.		
4	Durch die Vorgehensmodelle werden unstrukturierte Vorgehensweisen, die z. B. zu mangelnder Softwarequalität führen systematisch durchbrochen.		
5	Die Grundformen der Vorgehensmodellen sind...		
	die allgemeinen Vorgehensmodelle.		
	die sequentiellen Vorgehensmodelle.		
	die evolutionären Vorgehensmodelle.		
	die agilen Vorgehensmodelle.		
	die strukturierten Vorgehensmodelle.		
6	Die Trendentwicklung seit den 90er Jahren zeigt deutlich, dass _____.		

Tab. 8: Abschlusstest – WBT07

7.4 Typische Aufgabenstellungen

Typische Aufgabenstellungen – Agile Vorgehensmodelle

Zur Bearbeitung dieser Aufgabenstellungen beachten Sie bitte: Verlangt ist eine fachlich zutreffende, inhaltlich nachvollziehbare und kausal zusammenhängende Erörterung aus vollständigen Sätzen in lesbarer Handschrift. Für jede Aufgabe: Maximal zwei Seiten Text.

Aufgabe 1:

> Erläutern Sie die Grundwerte / Prinzipien des Agilen Manifests.

Aufgabe 2:

> Warum entstanden Ende der 1990er Jahre die Tendenzen hin zur „Agilen Software-Entwicklung“?

Aufgabe 3:

> Erläutern Sie die Strukturelemente und die Funktionsweise von Xtreme Programming.

Aufgabe 4:

> Erläutern Sie die vier prozessbezogenen Praktiken von Xtreme Programming.

Aufgabe 5:

> Erläutern Sie den prinzipiellen Zusammenhang von XP und Scrum.

8 Vorgehensmodelle: Historie und Trends

8.1 Trends bei Vorgehensmodellen

8.1.1 Löst agil sequentiell ab?

Sequentielle und evolutionäre VGM beruhen auf einer ausgefeilten Dokumentation der Vorgehensweise, den Spezifikationen und den zu erstellenden Zwischenergebnissen eines Software-Produktes. Die Zeit, die beispielsweise in eine Anforderungs- bzw. Situationsanalyse oder Machbarkeitsstudie investiert wird, ist deutlich höher als in agilen Projekten.

In der agilen Methode „Scrum“ hingegen werden z. B. User Stories angefertigt, auf deren Basis die Grundfunktionen von Software ohne gesonderte Situationsanalyse entwickelt werden.

Der Fokus liegt bei agilen Methoden auf der Kommunikation und Interaktion mit dem Kunden und nicht in der Dokumentation der Software, wie es bei „schwergewichtigen“ Software-Planungsprojekten, also bei sequentiellen VGM der Fall ist.

Die Auswahl eines bestimmten Vorgehensmodells für ein Projekt muss deshalb immer genau abgewägt werden. Viel Planungsaufwand bedeutet letztendlich auch einen hohen Dokumentationsaufwand. Darüber hinaus müssen in planungsaufwändigen Projekten auch viele Spezifikationen zu Anfang abgestimmt werden. Die zentrale Frage, die gestellt werden muss, lautet: **Wie viel Planung braucht ein Projekt**?

8.1.2 Zeitliche Einordnung der Vorgehensmodelle

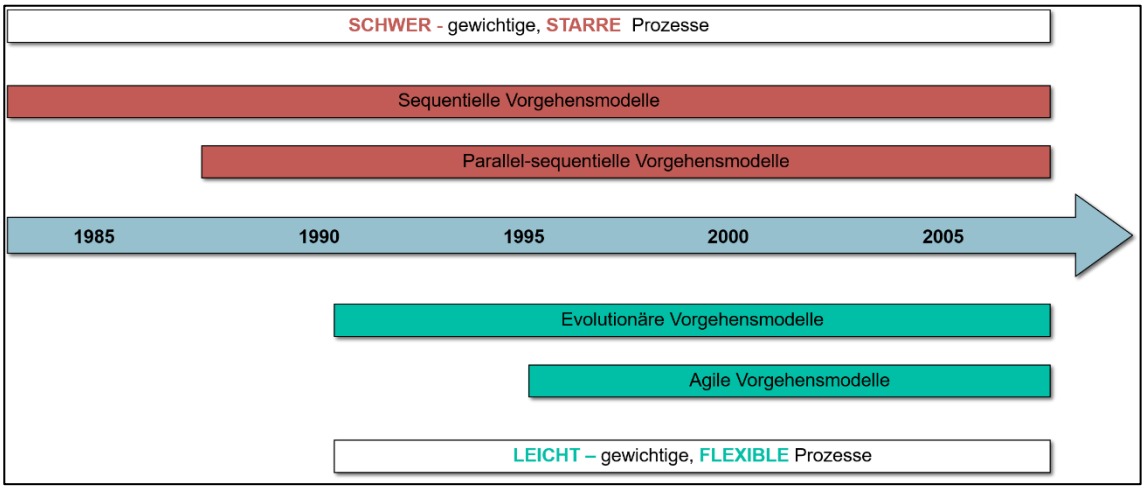


Abb. 91: Zeitliche Einordnung starre und flexible Prozesse

8.1.3 Welches Vorgehensmodell ist für mein Projekt geeignet?

Nicht jedes Vorgehensmodell ist für jedes Projekt geeignet. Man muss unter anderem entscheiden, wie viel Zeit für die Dokumentation veranschlagt wird, wie komplex ein Projekt ist oder ob es sich um eine Standard-Implementation handelt oder eine individuell entwickelte Software.

Generell kann man jedoch die Unterscheidung zwischen starren bzw. schwergewichtigen Vorgehensmodellen und agilen bzw. flexiblen Vorgehensmodellen treffen.

Im Folgenden werden die Unterschiede benannt und auf die Planungstiefe der einzelnen Modelle eingegangen.

Ausprägung / Merkmale	**Leichtgewichtig**	**Schwergewichtig**
Regelungs- und Dokumentationsdichte	Gering	Hoch
Regelung	Allgemeines, viel Eigenverantwortung	Ablauf- und Ergebnisvorschriften des VGM
Abläufe und Ergebnisse	Gestaltbar, Ergebniszeitpunkte und Produkte	Fix vordefiniert
Beispiel	XP, Scrum, Crystal	Wasserfall, V-Modell

Abb. 92: Merkmale leicht- und schwergewichtiger Vorgehensmodelle

8.1.4 Schwer- und leichtgewichtige Vorgehensmodelle

Gewicht + Agilität: „SCHWER + STARR“

- Hohe Regelungs- und Dokumentationsdichte („Schrank-Ware“)
- Regelung: Ablauf- und Ergebnisvorschriften des Vorgehensmodells
- Dokumentation: Ergebnisbeschreibungen (Spezifikationen, Zwischenergebnisse, Modelle etc.)
- Abläufe und zu erzeugende Ergebnisse fix vordefiniert
- Beispiel: V-Modell mit extrem umfangreichem Regelwerk

Gewicht + Agilität: „LEICHT + FLEXIBEL“

- Geringe Regelungs- und Dokumentationsdichte
- Regelung: Allgemeines, Rahmen / viel Eigenverantwortung
- Dokumentation: Beschreibungen nur der „wichtigen“ Sachverhalte, Ergebnisse

- Gestaltbare Abläufe, Ergebniszeitpunkte und Produkte
- Beispiel: eXtreme Programming mit wenigen Regeln

8.1.5 Planungstiefe: Ad-hoc oder Feingranular

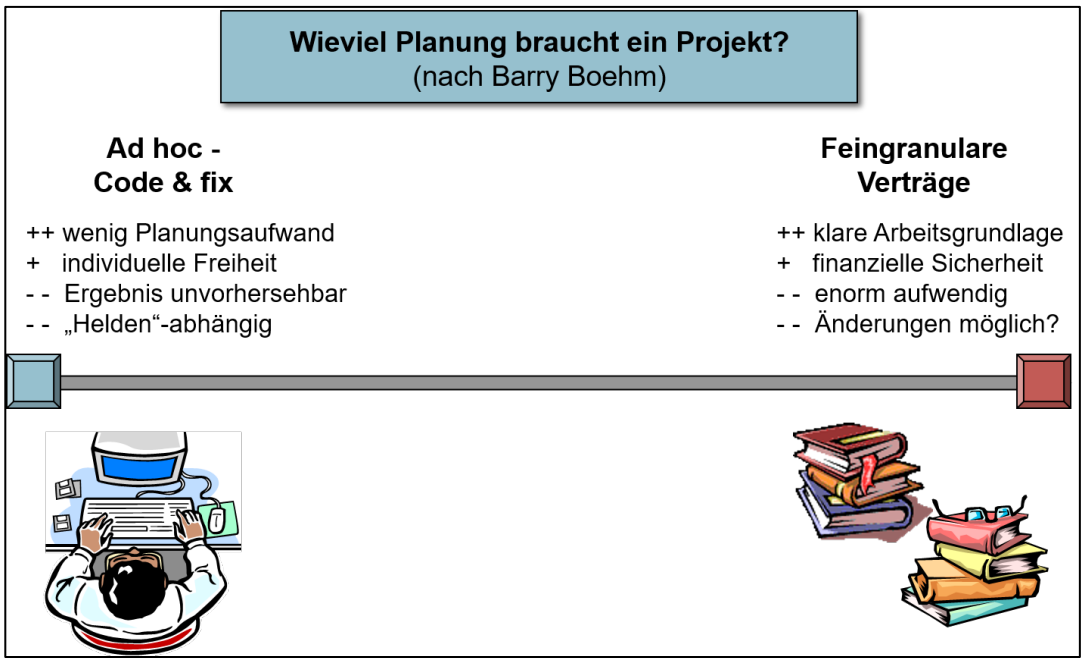


Abb. 93: Planungstiefe Ad-hoc und feingranulare Verträge

8.1.6 Planungstiefe: Plan-getrieben

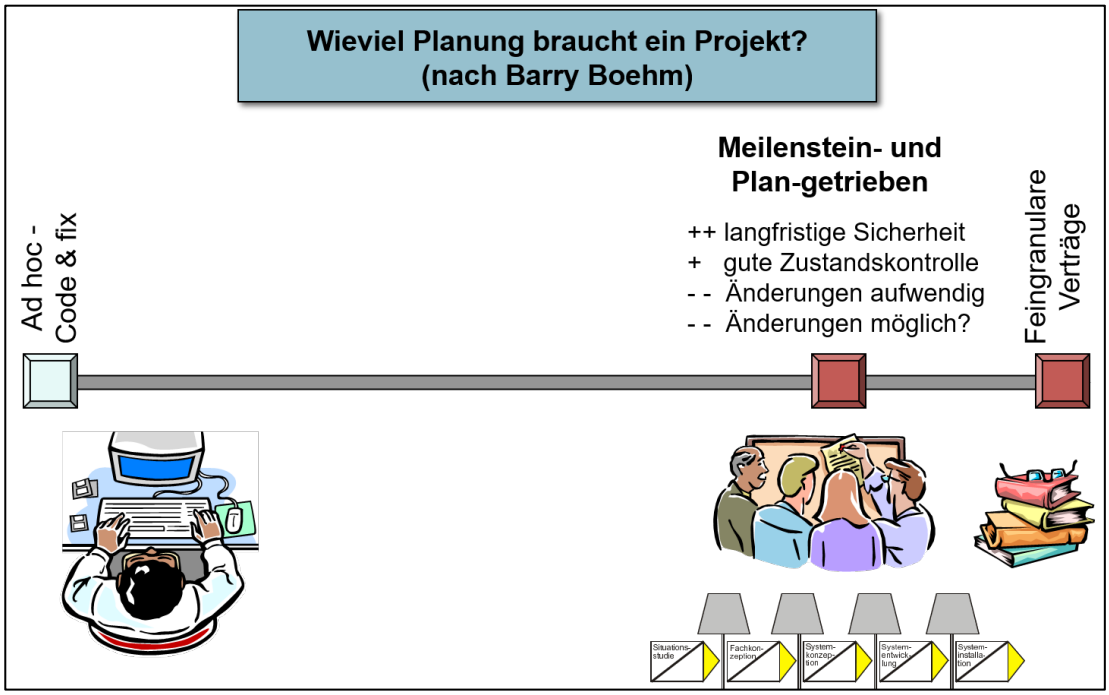


Abb. 94: Planungstiefe: Meilenstein- und plangetriebene Vorgehensmodelle

8.1.7 Planungstiefe: Risiko-gesteuert

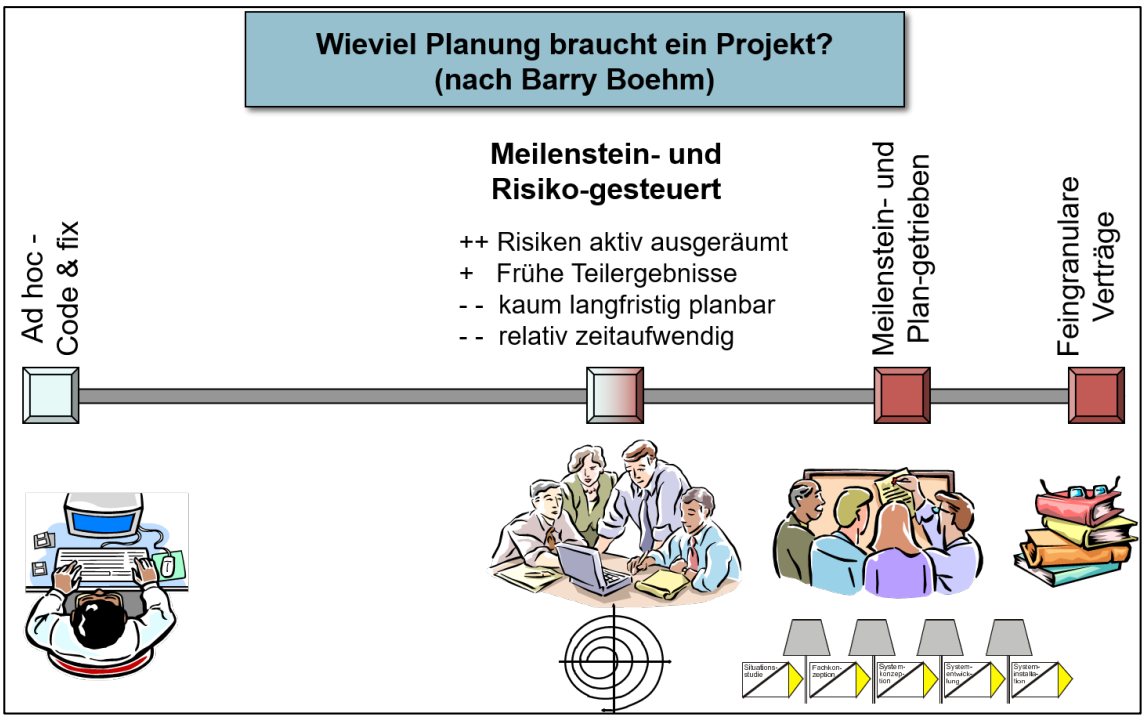


Abb. 95: Planungstiefe: Risikogesteuerte Vorgehensmodelle

8.1.8 Planungstiefe: eXtreme Programming

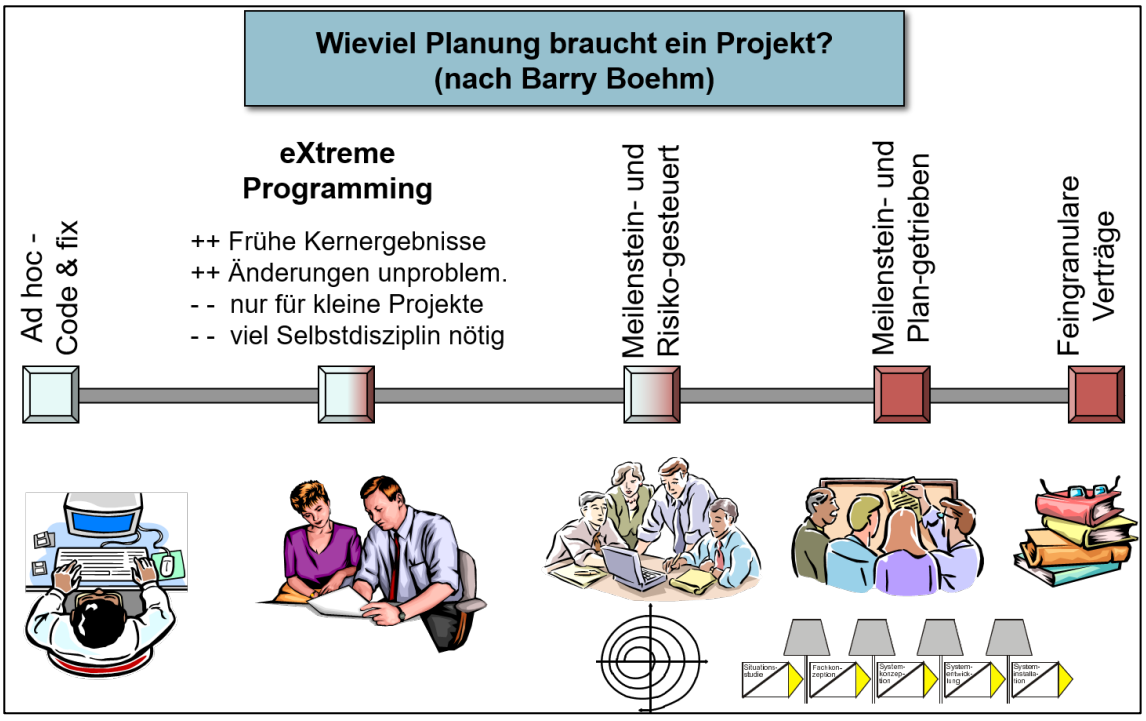


Abb. 96: Planungstiefe: eXtreme Programming

8.1.9 Planungstiefe: Agil vs. starr

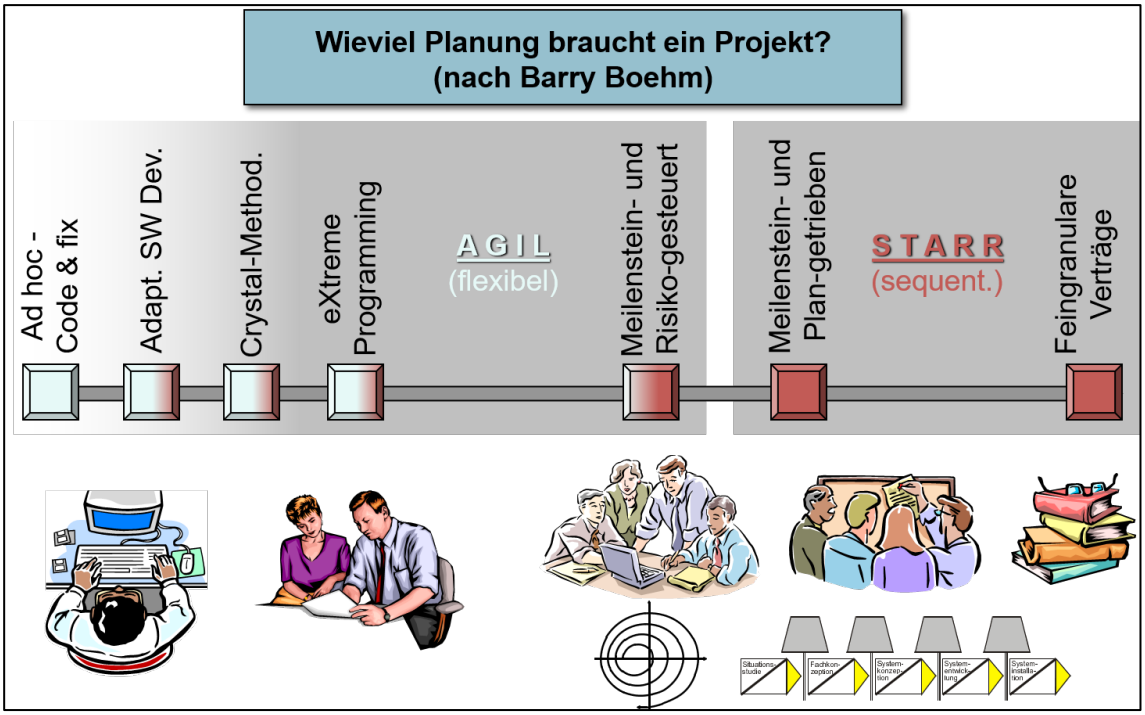


Abb. 97: Planungstiefe: Agile vs. starre Vorgehensmodelle

8.1.10 Agil vs. starr: Prinzip und Denkweise im Vergleich

	Konventionell, starr	Innovativ (?), agil
Mitwirkung des Kunden	Wenig, erzwungen	Kritischer Erfolgsfaktor
Etwas Nützliches wird geliefert	Erst nach längerer Zeit	Mindestens alle sechs Wochen (Inkremente)
Das Richtige entwickeln durch	Langes Spezifizieren, Planen	Entwickeln, zeigen, verbessern
Nötige Disziplin	Wenig, formal	Viel, informell
Änderungen	Erzeugen Widerstand	Erwartet und toleriert
Kommunikation	Über Dokumente	Zwischen Menschen
Vorsorge für Änderungen	Versuch der Vorausplanung	Durch „flexibel Bleiben“

Abb. 98: Eigenschaften konventioneller und innovativer Denkweisen

Sie können agile und starre Vorgehensmodelle anhand von sieben Faktoren unterscheiden. Die Faktoren beschreiben, welche Ansprüche und Erwartungen an die „Verhaltensschablone“, das Vorgehensmodell, gestellt werden. Zum Beispiel erwarten wir bei agilen Projekten eine aktive Rolle des Kunden. Das Verhalten, das vom Kunden erwartet wird,

spiegelt sich also in den Praktiken wider. In Scrum sind dies die Sprint-Planungen und in Xtreme Programming die Planungssitzungen und der Kunde vor Ort.

8.1.11 Zeitliche Einordnung: Sequentiell und Concurrent

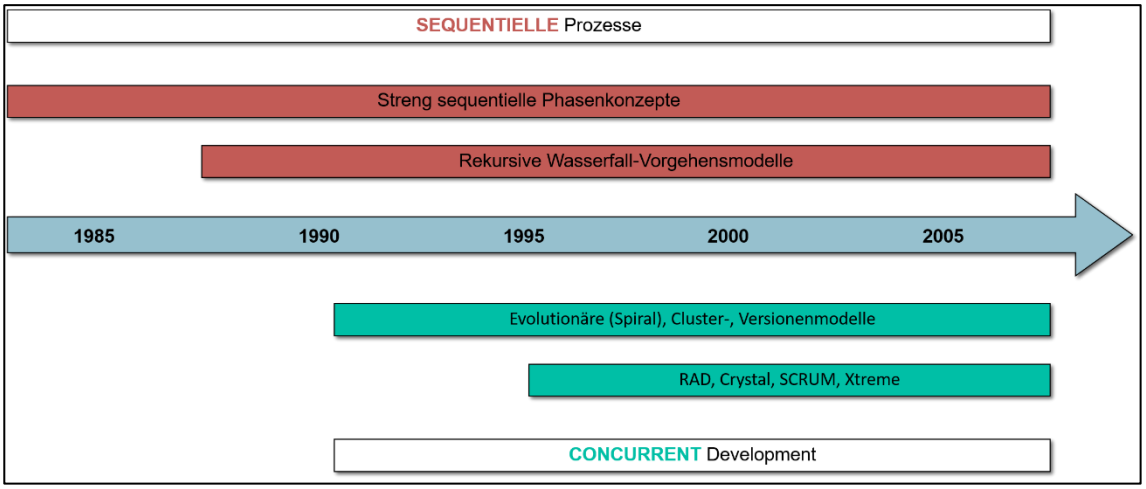


Abb. 99: Zeitstrahl zeitliche Einordnung der Vorgehensmodelle

8.1.12 Überlappung: Sequentialität

- Grundaufgaben „Analyse“ und „Realisierung“ **ohne** Überlappung
- Alle Spezifikationen müssen vorliegen (freeze), bevor die Programmierung (technische Realisierung) beginnt.
- Strikte „Trennung von Essenz und Inkarnation“
- Auf der Spezifikationsgrundlage kann eine detaillierte Gesamtplanung von Aktivitäten, Ressourcen und Terminen für die Realisierung vorab erfolgen.
- Die erzeugte Software kann gegen die Spezifikationen geprüft werden.
- Voraussetzung 1: Das zu erzeugende System kann vorausschauend vollständig spezifiziert werden. Das System ist vollständig bekannt.
- Voraussetzung 2: Keine Änderungen der System-Umwelt während der Laufzeit des Entwicklungsprozesses.
- Voraussetzung 3: Der Entwicklungsprozess muss ausschließlich in der Lage sein, die vorgegebenen System-Anforderungen zu erfüllen (nicht: Umwelt- oder Anforderungsänderungen abzufangen).

8.1.13 Überlappung: Parallelität

- Grundaufgaben „Analyse“ und „Realisierung“ **überlappen**.
- Später Spezifikations-Freeze nach Beginn der technischen Realisierung
- Bewusste/r „Vermischung / Wechsel von Essenz und Inkarnation“
- Grobplanung von Aktivitäten, Ressourcen und Terminen mit geringer Vorausschau rollierend modifizieren

- Häufige Evaluierung von Zwischenständen, um bestehende Zustände abzusichern und weiterführende Informationen zu sammeln.
- Voraussetzung 1: Das zu erzeugende System ist neuartig (Architektur, Anwendung, Technologie).
- Voraussetzung 2: Dynamische, unsichere System-Umgebung
- Voraussetzung 3: Der Entwicklungsprozess muss in der Lage sein, Umwelt- oder Anforderungsänderungen abzufangen und die frühe (schnelle) Einsatzfähigkeit der System-Versionen effizient zu gewährleisten (Zeit- und Kostendruck bei unsicheren Erwartungen).

8.1.14 Fazit: Schwer vs. Leicht

Fazit: „SCHWER + STARR + SEQUENTIELL“

- Bei geringer Unsicherheit = Umwelt, System, Anforderungen bekannt
- Dann sollte das Vorgehensmodell gewährleisten:
  - Schwerpunkt auf Spezifikation, Planung, Kontrolle
  - Klare zeitliche und inhaltliche Trennung von Aufgaben
  - Überwiegend fach- und aufgabenbezogene Qualifikationen
- „Gut geplant, ist halb gewonnen!“

Fazit: „LEICHT + FLEXIBEL + PARALLEL“

- Bei hoher Unsicherheit = Umwelt, System, Anforderungen offen
- Dann sollte das Vorgehensmodell gewährleisten:
  - Schwerpunkt auf Basis-Architektur (anpassungsfähig)
  - Frühzeitiges „System Level Feedback“ (Prototypen, Tests, Benutzerpartizipation)
  - Detailliertes, reagibles Qualitäts-, Projekt-, Prozessmanagement
  - Überwiegend kreative, technische Umsetzungsqualifikationen
- „Nur mit „guten Leuten“ („Helden“) kann man gewinnen!“

8.2 Die Xtreme-Programming-Scrum-Kombi

8.2.1 Perspektiven des Software Engineering

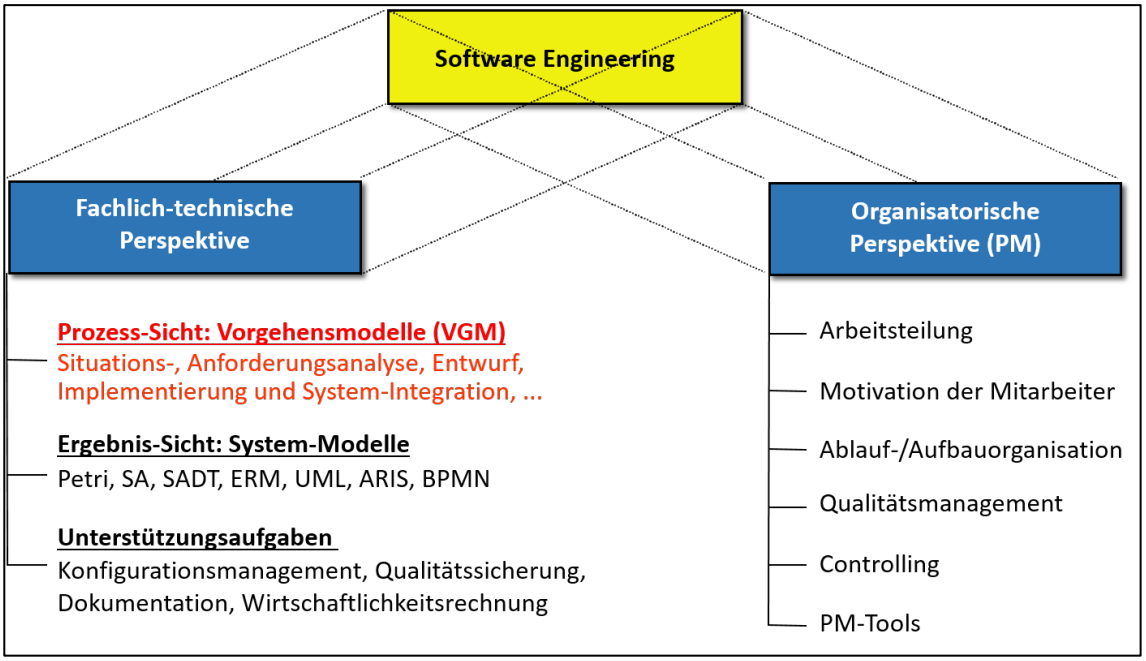


Abb. 100: Perspektiven des Software Engineering

8.2.2 Die Xtreme-Programming-Scrum-Kombi

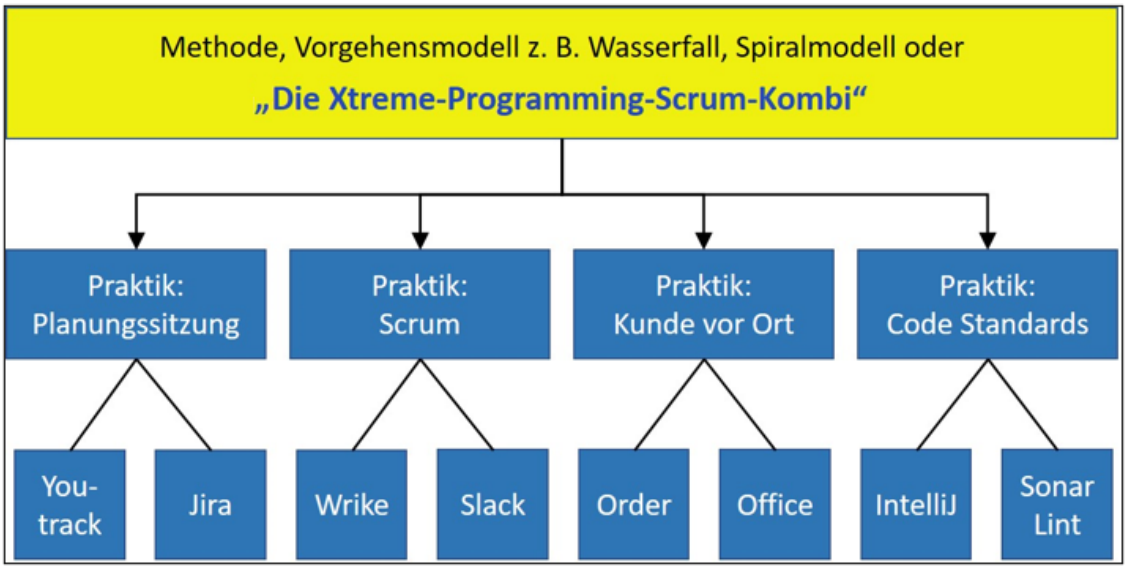


Abb. 101: Die eXtreme-Programming-Scrum-Kombi

Eine Methode ist ein Bündel aus abgestimmten Techniken (Praktiken). Techniken werden von Werkzeugen (Tools) ausgeführt.

8.2.3 Das Agile Manifest (Wdh.)

Abb. 102: Das Agile Manifest (Wdh.)

8.2.4 Extreme Programming (XP)

Xtreme Programming ist vor dem Hintergrund entstanden, die Qualität der Software zu steigern und in einem volatilen Projektumfeld zu entwickeln.

Besonders beachtet wurden bei der Konzeption die sich ständig ändernden Anforderungen der Kunden, welche schnell in ein Projekt eingearbeitet werden müssen.

Die Vorgehensweise ist also auf kurze Veröffentlichungszyklen ausgerichtet.

Im Folgenden werden die Praktiken beschrieben, die sich vor allem von den sequentiellen und evolutionären Vorgehensmodellen unterscheiden und sich von der Fließband-Mentalität abgrenzen lassen.

Werte und Prinzipien

- Kommunikation und Kundeneinbindung
- Feedback und inkrementelle Entwicklung
- Einfachheit und Mut zur Entscheidung

Praktiken

- Verstärken sich gegenseitig
- Zusammenspiel von Prozess, Test und Architekturfragen

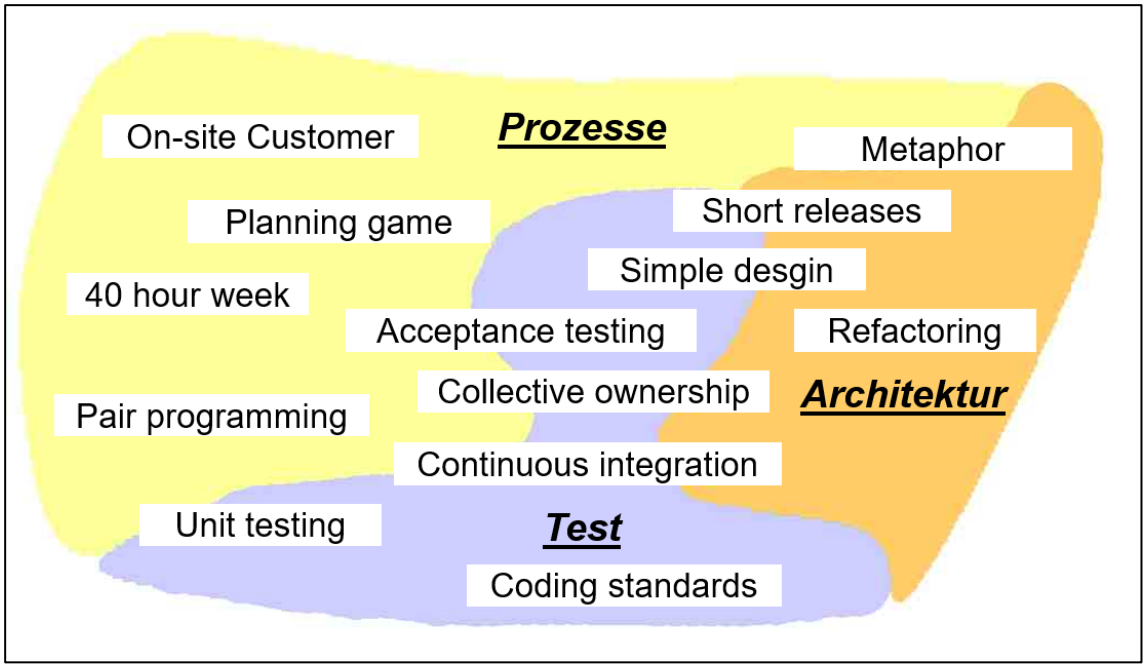


Abb. 103: eXtreme Programming: Prozesse, Architektur und Test

8.2.5 Die XP-Scrum-Kombi: Planungssitzung

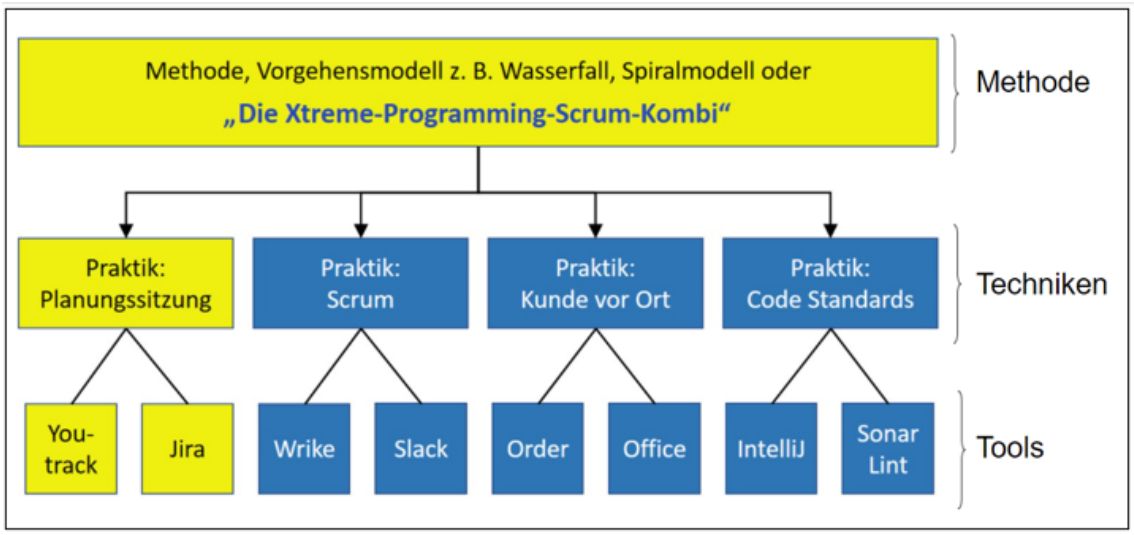


Abb. 104: Die XP-Scrum-Kombi: Planungssitzung

8.2.6 XP: Planungssitzung (Wdh.)

Anforderungen auf User Story Cards sammeln

- Kunde schreibt die Karten.
- Skizzen, Texte, auch Umgangssprache

Entwickler schätzen Kosten pro Karte

- Schätzung auf Erfahrungsbasis (Ähnlichkeit zwischen Karten)
- Abstrakte Bemaßung der Kosten (z. B. 1 bis 10 Punkte)
- Ab-/Rücksprache mit Kunden
- Fortschreibung der Karten mit Kostenvermerken (Story Points)

Auswahl von Karten für nächste Release durch Kunden

- Nach Wichtigkeit, Nutzen priorisiert
- Entwickler-Team wird limitiert durch seine Produktivität

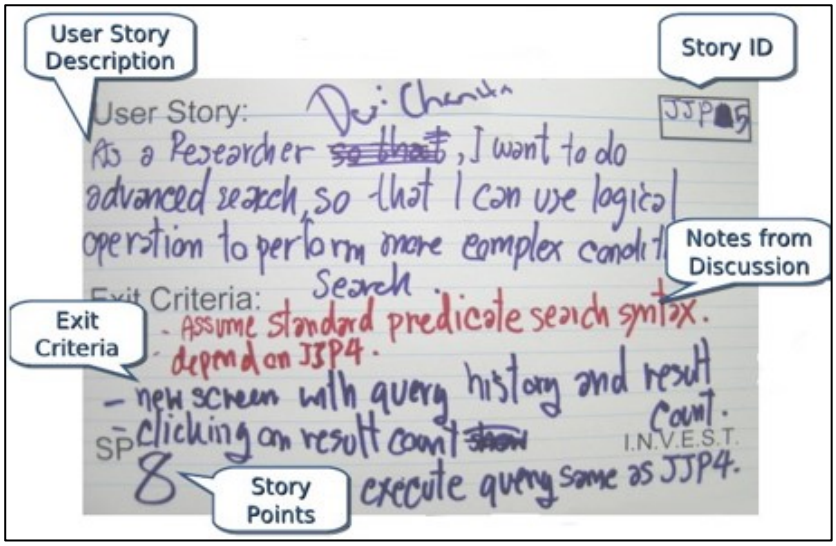


Abb. 105: eXtreme Programming User Story Cards (Wdh.)

8.2.7 Die XP-Scrum-Kombi: Kunde vor Ort

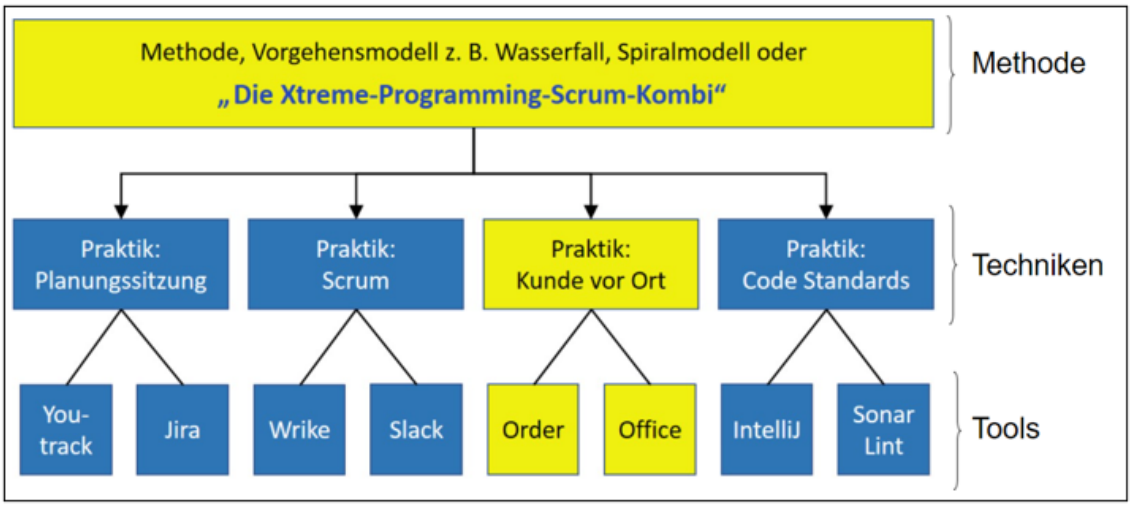


Abb. 106: Die XP-Scrum-Kombi: Kunde vor Ort

8.2.8 XP: Kunde vor Ort (Wdh.)

Kunde ist über die gesamte Projektlaufzeit verfügbar

- Idealerweise direkt vor Ort ...
- ... oder zumindest jederzeit und schnell erreichbar.

Kunde schreibt und überprüft User Stories

- Anforderungen (Karten) werden ständig weiterentwickelt.
- Fragen der Entwickler werden sofort beantwortet.

Kunde trifft Entscheidungen

- Zeitverluste vermeiden durch direkte Verfügbarkeit
- Keine Allein-(Fehl-)Entscheidungen durch Entwickler

8.2.9 Die XP-Scrum-Kombi: Code Verantwortung und Standards

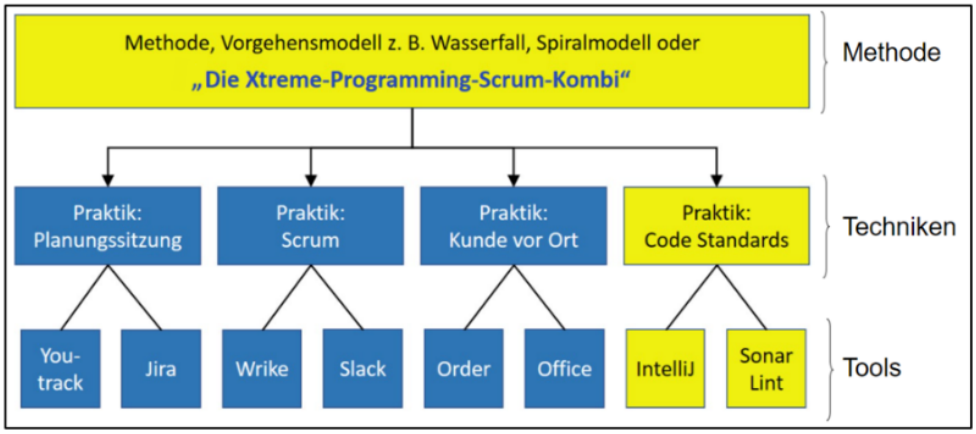


Abb. 107: Die XP-Scrum-Kombi: Code Verantwortung und Standard

8.2.10 XP: Code Verantwortung und Standards (Wdh.)

Gemeinsame Code-Verantwortung im Paar

- Jeder darf den Quellcode des anderen bearbeiten.
- Arbeitsteilung in Story Cards
- Kann temporäre personelle Ausfälle kompensieren

Code-Standards schaffen gemeinsames Verständnis

- Voraussetzung für gemeinsame Code-Verantwortung
- Leserlichkeit, „sprechender“ Quellcode
- Strukturstandards wie Klammerungen, Einrückungen, Namenskonventionen, Groß-/Kleinschreibung etc.

8.2.11 Die XP-Scrum-Kombi: Scrum

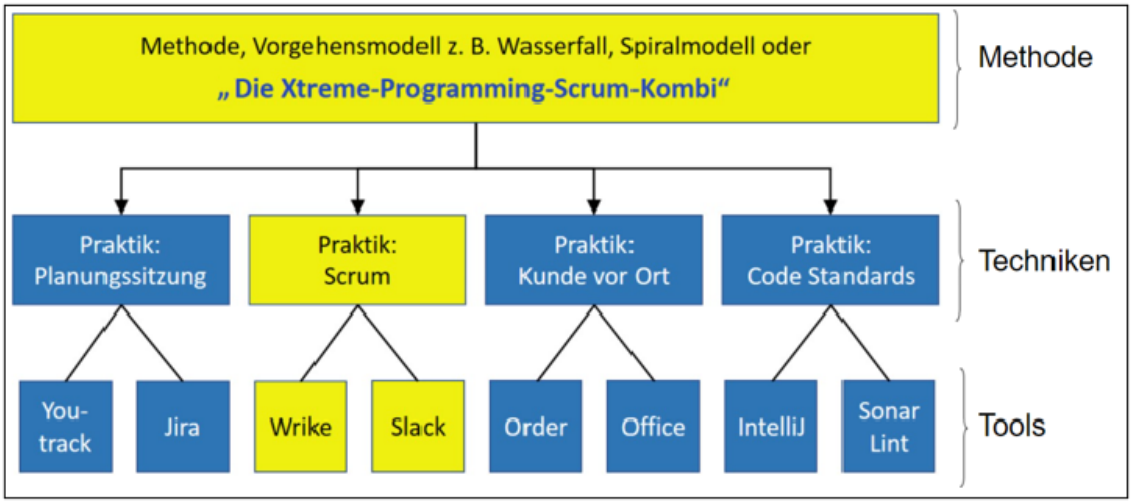


Abb. 108: Die XP-Scrum-Kombi: Scrum

8.2.12 Scrum – Das geordnete Gedränge (Wdh.)

Scrum – die englische Bezeichnung für das angeordnete Gedränge im Rugby-Sport. Der Bezug zu Scrum als Prozess für Projektmanagement und Entwicklung liegt darin, dass das Team im Mittelpunkt steht und jeweils vor dem nächsten Scrum den geplanten Spielzug bespricht. Ohne diese Selbstorganisation des Teams funktioniert Scrum nicht.

8.2.13 Scrum – klein, kreativ, schnell (Wdh.)

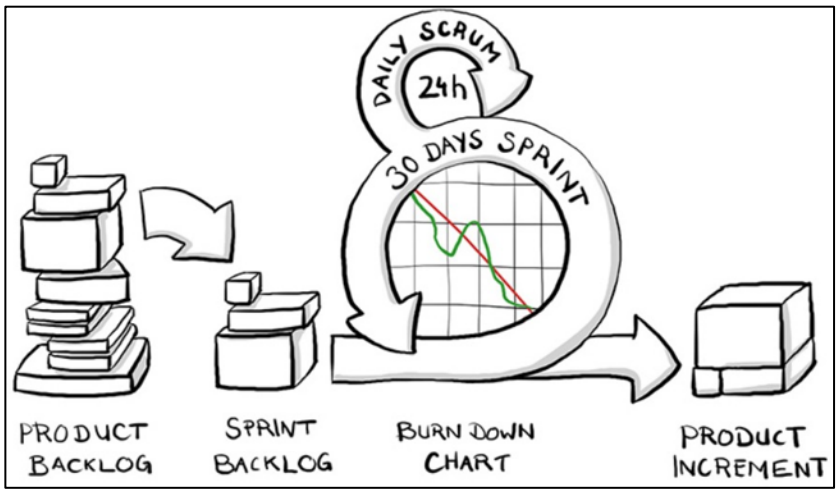


Abb. 109: Scrum Vorgehen (Wdh.)

8.2.14 Die Xtreme-Programming-Scrum-Kombi – Praktiken

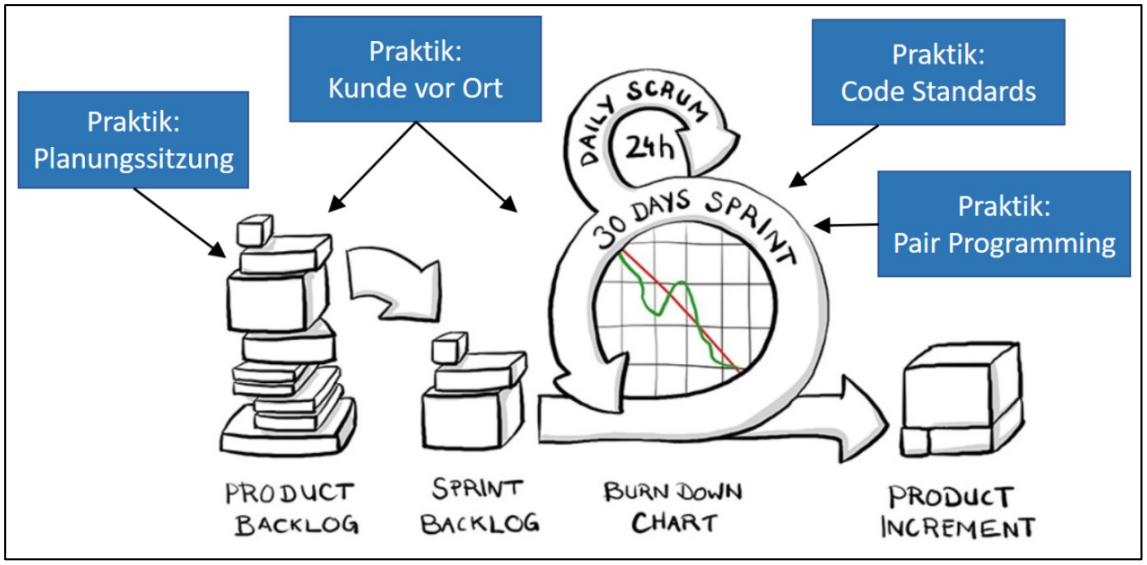


Abb. 110: Die Xtreme-Programming-Scrum-Kombi – Praktiken

8.2.15 Die Xtreme-Programming-Scrum-Kombi – XP-Praktiken in Scrum

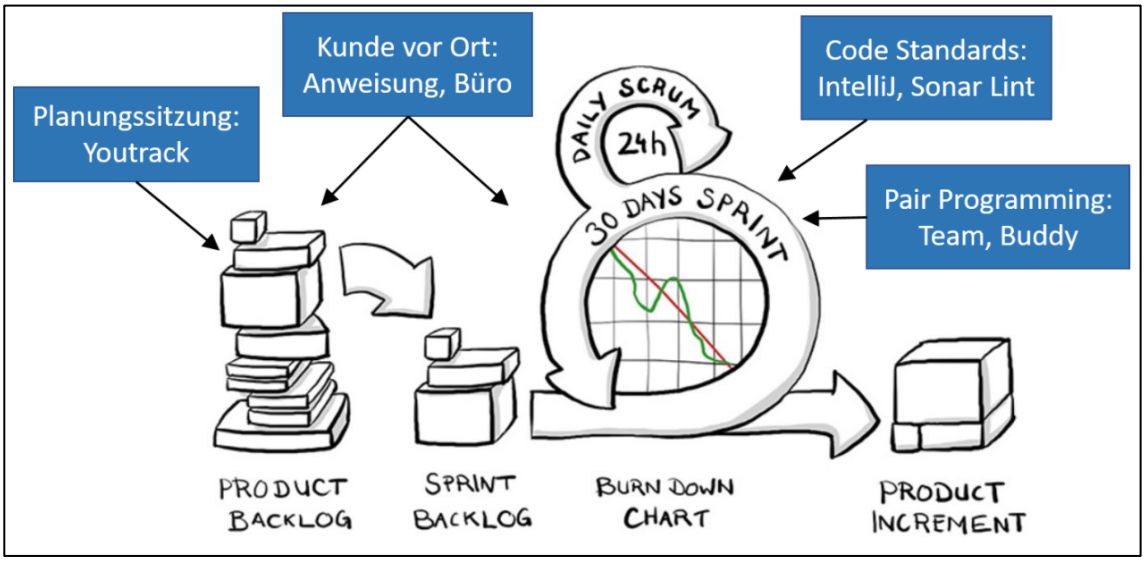


Abb. 111: Die Xtreme-Programming-Scrum-Kombi – XP-Praktiken in Scrum

8.2.16 Die Xtreme-Programming-Scrum-Kombi: Das Tool „Youtrack“

Abb. 112: Video zum Tool Youtrack

8.3 Abschlusstest – WBT08

8.3.1 Abschlusstest

Bitte beantworten Sie die folgenden Fragen durch das Ankreuzen der korrekten Antworten (Tab. 9). Bei einigen Fragen können auch mehrere Antworten richtig sein.

Nr.	Frage	Richtig	Falsch
1	Bei sequentiellen Vorgehensmodellen muss das zu erzeugende System vorausschauend vollständig spezifiziert werden.		
2	Bei agilen Vorgehensmodellen muss die Spezifikation vor der technischen Realisierung vollständig abgeschlossen sein.		
3	Zu den sequentiellen Vorgehensmodellen gehören...		
	das V-Modell		
	das Wasserfallmodell		
	SCRUM		

	die Crystal-Methodenfamilie		
	eXtreme Programming		
4	Die Regelungs- und Dokumentationsdichte sind bei schwergewichtigen Vorgehensmodellen besonders hoch im Vergleich zu leichgewichtigen Vorgehensmodellen.		
5	Zu den agilen Vorgehensmodellen gehören...		
	SCRUM		
	Adaptive Software Development		
	Concurrent Development		
	Wasserfallmethode		
	Crystal-Methodenfamilie		

Tab. 9: Abschlusstest – WBT08

8.4 Typische Aufgabenstellungen

Typische Aufgabenstellungen – Vorgehensmodelle: Historie und Trends

Zur Bearbeitung dieser Aufgabenstellungen beachten Sie bitte: Verlangt ist eine fachlich zutreffende, inhaltlich nachvollziehbare und kausal zusammenhängende Erörterung aus vollständigen Sätzen in lesbarer Handschrift. Für jede Aufgabe: Maximal zwei Seiten Text.

Aufgabe 1:

> Erläutern Sie vergleichend die Merkmale und Eignungen von schwergewichtigen-starren gegenüber leichtgewichtigen-flexiblen Vorgehensmodellen zur Entwicklung von betrieblichen Informations- und Kommunikationssystemen.

Aufgabe 2:

> Worauf beziehen sich die Metaphern „leichtgewichtig“ und „schwergewichtig“?

Aufgabe 3:

> Erläutern Sie die Strukturelemente und die Funktionsweise von Scrum.

Aufgabe 4:

> Erläutern Sie die wesentlichen Kritikpunkte an Scrum.

9 Grundlagen der Modelldarstellung

9.1 Die Systemtheorie

9.1.1 Wiederholung: Die Ergebnis-Sicht

Ergebnis-Sicht: Die fachliche Modellierung

Das „**Was**" der Entwicklung –

die Gestaltung und Darstellung des IT-Systems mit Modellierungsansätzen wie z. B. funktions-, datenfluss-, daten-, prozess- oder objektorientierter Modellierung.

Prozess-Sicht: Der dynamische Zeitablauf

Das „**Wie**" der Entwicklung –

die Vorgehensweise der Entwicklung –

die Prozessschritte zur Entwicklung des IT-Systems.

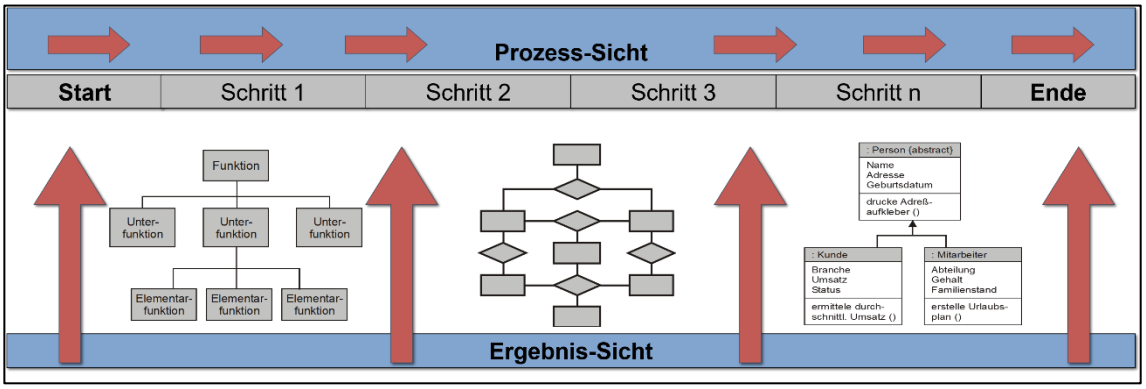


Abb. 113: Prozess- und Ergebnis-Sicht

9.1.2 Die fachlich-technische Perspektive

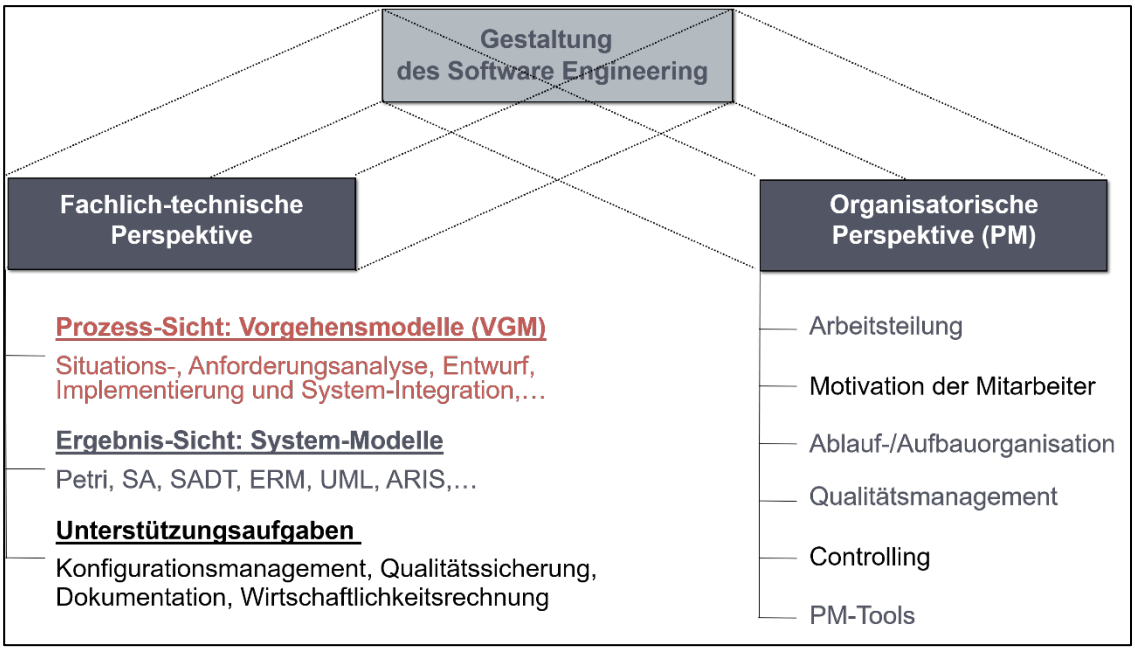


Abb. 114: Gestaltung des System Engineering

Die Ergebnis-Sicht ist auch unter fachlich-technischen Gesichtspunkten zu betrachten und bezieht i. d. R. keine organisatorischen Gesichtspunkte mit ein. Sie konzentriert sich auf Modelle, die die Komplexität von Software-Systemen beherrschbar machen.

9.1.3 Komplexe Sachverhalte verstehen

In der letzten Einheit haben Sie sich mit der Prozess-Sicht beschäftigt. Diese befasst sich mit dem Vorgehen in Projekten und deren Phasen.

Am Ende einer Projekt-Phase steht immer ein Ergebnis. Die Betrachtung dieser Ergebnisse wird als Ergebnis-Sicht bezeichnet.

Die Ergebnis-Sicht ist Teil dieser Lerneinheit. Hier liegt der Fokus auf Ergebnissen, wie beispielsweise ein Organigramm, ein Stück Programmier-Code, eine technische Zeichnung oder um auf unser Hausbau-Beispiel zurückzukommen: Die fertige Zeichnung des Architekten.

Um solche Ergebnisse liefern zu können, muss das gesamte Aufgaben-Konstrukt eines Projekts als System betrachtet werden. Die Systemtheorie hilft uns dabei, komplexe Sachverhalte auf ihren Kern herunterzubrechen.

Im Software Engineering hilft die System-Theorie dabei, komplexe Funktionen einer Software in kleinere Funktions-Teile zu zerlegen. Diese können dann leichter verstanden und entwickelt werden. Komplexe Software wird also beherrscht, indem sie mit Hilfe der System-Theorie in Einzelteile zerlegt wird.

9.1.4 Systemtheoretische Grundlagen

Gegenstand der allgemeinen Systemtheorie

- Anordnung von Elementen zu einem System und die Analyse der Beziehungen
- Begründer: Ludwig von Bertalanffy, 1951

Zum **Begriff** „System“

- Ein System wird durch eine abgegrenzte und geordnete Menge von Elementen gebildet, die untereinander in Beziehung stehen.
- Das System bildet als Ganzes eine Einheit und lässt sich deutlich von seiner Umwelt abgrenzen.
- System-Emergenz: Das Ganze ist mehr als die Summe seiner Teile.

Hauptaspekte der Analyse und Gestaltung von Systemen

- Wirkungsaspekte, Strukturaspekte
- Abstraktion, Dekomposition

9.1.5 Gegenstand der allgemeinen Systemtheorie

Unter einem System wird eine Menge von Elementen verstanden, zwischen denen Beziehungen und Wechselwirkungen bestehen und die gegenüber der Umwelt abgegrenzt sind. Begründet wurde die Systemtheorie durch den Biologen Ludwig van Bertalanffy um 1951.

Die Systemtheorie ist ähnlich wie die Vorgehensmodelle als Schablone auf jeden Lebensbereich anwendbar. Software-Komponenten lassen sich mit Hilfe der Systemtheorie genauso gut wie biologische Lebewesen in Zellen und Moleküle beschreiben.

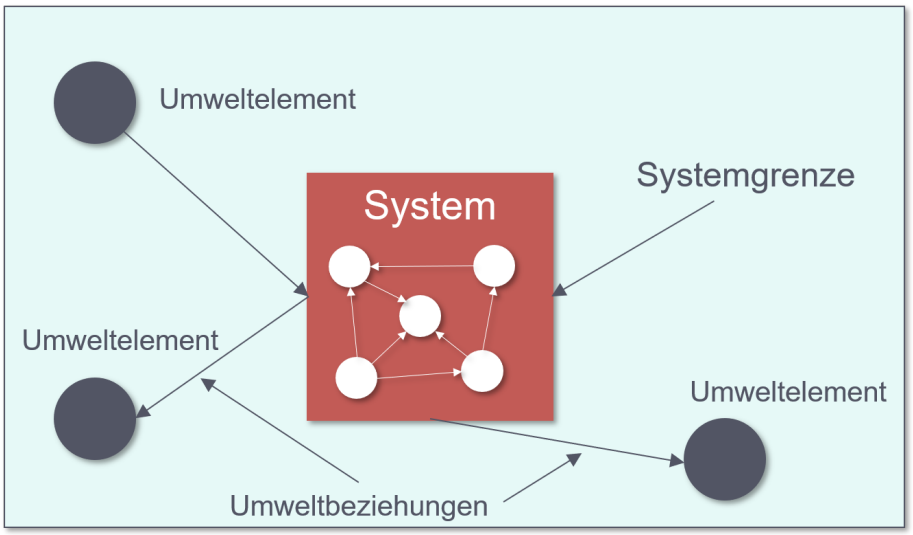


Abb. 115: Gegenstände der allgemeinen Systemtheorie

9.1.6 Zum Begriff „System“

Die definitorischen Elemente des Systembegriffs sind

1. die Elemente,
2. die Beziehungen und
3. die Abgrenzung des Systems zur Umwelt.

Die System-Emergenz bedeutet in diesem Sinne, dass das Ganze System immer mehr als die Summe seiner Teile ist. Mit einem betriebswirtschaftlichen Hintergrund wird dies als Synergie-Effekt beschrieben.

Elemente sind Teile eines Systems, die nicht sinnvoll weiter aufgebrochen werden können. Die Zerlegung ist dabei eine Frage der Zweckmäßigkeit. Elemente stellen die kleinsten betrachteten Einheiten dar, deren interne Struktur bei einem gegebenen Zweck nicht weiter interessiert.

Unter **Beziehung** versteht man konstante Verbindungen, die zwischen den Elementen existieren. Das gesamte Beziehungsgefüge der Systemelemente definiert formal die Ordnung eines Systems. Sie wird als Systemstruktur bezeichnet.

Unter **Umwelt** des Systems versteht man Elemente, die außerhalb der Grenzen des betrachteten Systems liegen. Ausschlaggebend für die Abgrenzung des Systems gegenüber der Umwelt ist die Intensität der Beziehungen zwischen den Elementen. Charakteristisch ist, dass innerhalb der Systemgrenzen ein größeres Maß an Beziehungen besteht bzw. wahrgenommen wird als zwischen System und Umwelt.

9.1.7 Systemabgrenzung und Wirkungsaspekte

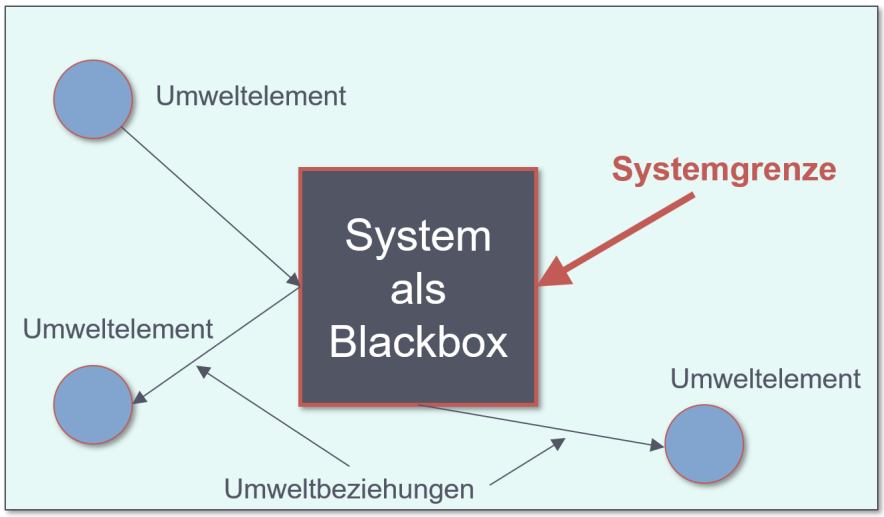


Abb. 116: Systemabgrenzung und Wirkungsaspekte

Zu den Hauptaspekten der Analyse und Gestaltung von Systemen gehören Wirkungsaspekte, Strukturaspekte, Abstraktion und Dekomposition.

Versucht man ein System abzugrenzen, betrachtet man das System zunächst als „Blackbox“, ohne seine Elemente. Zunächst werden hier durch die Wirkung der Umweltbeziehungen und -elemente auf das System die Systemgrenzen definiert. Dies wird auch als Systemabgrenzung unter Wirkungsaspekten verstanden.

Ein Beispiel: Bei der Hausbauplanung wird die Grenze des Systems beispielsweise am Grundstücksrand festgemacht.

Die Eigentümer werden zwar durch die Umweltelemente wie Nachbarschaft, Stadt und Bauamt beeinflusst, sie entscheiden jedoch selbst über ihr System „Grundstück“.

9.1.8 Beispiel: Wirkungsaspekte im IuK-System

Stellen Sie sich vor, Sie arbeiten in der IT-Abteilung eines größeren Unternehmens. Hier wird neben verschiedener Hardware mehrere Software-Produkte für die einzelnen Fachabteilungen des Unternehmens betrieben: Die Finanzabteilung benutzt ein Produkt von „SAP“, das Projekt Management Office benutzt „Microsoft Project Service Automation“ und die Vertriebsabteilung benutzt ein Produkt von „Salesforce“.

Jedes dieser einzelnen Software-Produkte stellt in sich ein System dar. So wie auch die Fachabteilungen zusammenarbeiten, müssen auch die Systeme miteinander in Beziehung stehen. So kann die Finanzabteilung keine Rechnungen schreiben, wenn sie nicht weiß, was verkauft wurde. Die Daten, die in der Finanzabteilung verwaltet und bearbeitet wer-

den, stammen folglich nicht aus der manuellen Eingabe, sondern aus einem anderen Software-System – in unserem Fall z. B. aus den Stundenbuchungen aus Microsoft Project Service Automation.

Aus dem Beispiel lässt sich erkennen, dass die Systeme sich durch ihren Zweck, die jeweilige Fachabteilung zu unterstützen, abgrenzen lassen. Jedoch sind sie „Umwelteinflüssen“ ausgesetzt – hier die Informationsweitergabe von einer Abteilung in die andere.

9.1.9 Systemabgrenzung und Strukturaspekte

Ist das System von der Außenwelt abgegrenzt und die Systemgrenzen festgelegt, wird das System selbst genauer untersucht. Unter den sogenannten Strukturaspekten werden die einzelnen zweckgebundenen Elemente der Blackbox und die Beziehungen dieser Elemente untereinander sichtbar. Das System erhält somit eine innere und äußere Struktur.

Ein Beispiel: Bei einem Hausbau wird die Struktur durch das Zusammenspiel der Mitwirkenden bestimmt. Der Architekt nimmt unter anderem die Raumeinteilung vor. Bei seiner Planung muss er aber beispielsweise auf den Verlauf von Strom und Wasserleitungen sowie auf eine korrekte Statik des Hauses achten. Die einzelnen Elemente des Systems „Haus“ sind miteinander verbunden und ergeben eine komplexe Struktur.

9.1.10 Beispiel: Strukturaspekte im IuK-System

Ein jedes Unternehmen braucht neben einer Finanzabteilung auch einen Vertrieb. Der Vertrieb ist dazu da, Produkte zu verkaufen. Unser imaginäres Unternehmen hat sich zur Software-gesteuerten Unterstützung durch „Salesforce“ entschieden. „Salesforce“ bietet Customer Relationship Management (CRM) Applikationen an, die den gesamten Verkaufsprozess von der Verkaufsanbahnung bis zur Nachvertragsphase unterstützen.

Der Prozess wird durch verschiedene Elemente definiert. Zwei Elemente möchten wir im Zusammenhang beschreiben. Das erste Element im System „Salesforce“ ist die Lead-Generierung. Hier werden interessierte Kunden ausfindig gemacht und kontaktiert. In diesem Zusammenhang entstehen Daten, die in das System „Salesforce“ eingetragen werden. Das Element Lead steht in Beziehung mit dem Element Opportunity. Opportunities sind Verkaufschancen. Hier hat ein potenzieller Kunde Interesse an Produkten bekundet und erste Vertragsverhandlungen können beginnen.

Zwischen den Elementen Lead und Opportunity besteht eine direkte Verbindung. Es zeichnet sich eine Struktur ab. Die Daten, die in der Lead-Generierung aufgenommen wurden, können in der Opportunity weiterverarbeitet werden. So entsteht im Zeitverlauf eine verlinkte Matrix, die die Elemente untereinander verbindet.

Ein **Lead** ist jeder, der an den Produkten oder Services eines Unternehmens Interesse gezeigt hat, aber möglicherweise noch keine Kaufentscheidung treffen kann. Der Begriff bezeichnet potenzielle Kunden, die bisher noch keine Geschäfte mit einem Unternehmen getätigt, aber ihre grundsätzliche Bereitschaft hierzu signalisiert haben.

Wird ein vielversprechender Lead an den Vertrieb weitergegeben, dann beginnt das Opportunity Management. Eine **Opportunity** stellt eine überprüfte und qualifizierte Verkaufsmöglichkeit dar.

9.1.11 Abstraktion und schrittweise Verfeinerung

- Hauptsachverhalte
- Vom Allgemeinen zum Speziellen
- Top-down, hierarchisch

Abstraktion ist die Kernzentralisation auf das Wesentliche. Man abstrahiert auf den oberen Ebenen von Einzelheiten.

Ein IuK-System hat immer mehrere Abstraktionsebenen. Auf der obersten Ebene steht das Gesamtsystem. Dieses wird vor dem Hintergrund der Komplexitätsreduktion auf mehrere Subsysteme aufgeteilt. Diese Aufteilung und Ausarbeitung des Systems werden so lange fortgeführt, bis ganz unten einzelne Arbeitsschritte aufgelistet werden können. Diese Arbeitsschritte können dann von Mitarbeitern oder Maschinen ausgeführt werden. Es geht also um eine schrittweise, hierarchische Verfeinerung der Hauptsachverhalte.

Wichtig bei diesem Vorgehen ist, dass die Zerlegung immer detaillierter wird, je weiter das System zerlegt wird. Außerdem ist die Zerlegung nur dann hilfreich, wenn scheinbare Subsysteme auch wirklich zusammengehörige Systeme sind, die sich von der Umwelt abgrenzen lassen. Hier ein Beispiel.

Schauen Sie wieder auf den Hausbau:

Der Bauplan des gesamten Hauses stellt das Gesamtsystem dar. Mögliche Subsysteme könnten beispielsweise der Keller, der Garten oder das Dach sein. Eine Teilaufgabe könnte das Zuschneiden der richtigen Balkenlängen für Holzbalken im Dach sein.

In einem IuK-System könnten Sie sich folgendes Szenario vorstellen:

Ein Unternehmen beauftragt IT-Spezialisten mit dem Aufsetzen eines kompletten Web-Auftritts. Dieser verkörpert das Gesamt-System. Intranet und Extranet könnten Subsysteme sein und das Anlegen aller Mitarbeiter-Daten im Intranet könnte eine Teilaufgabe sein.

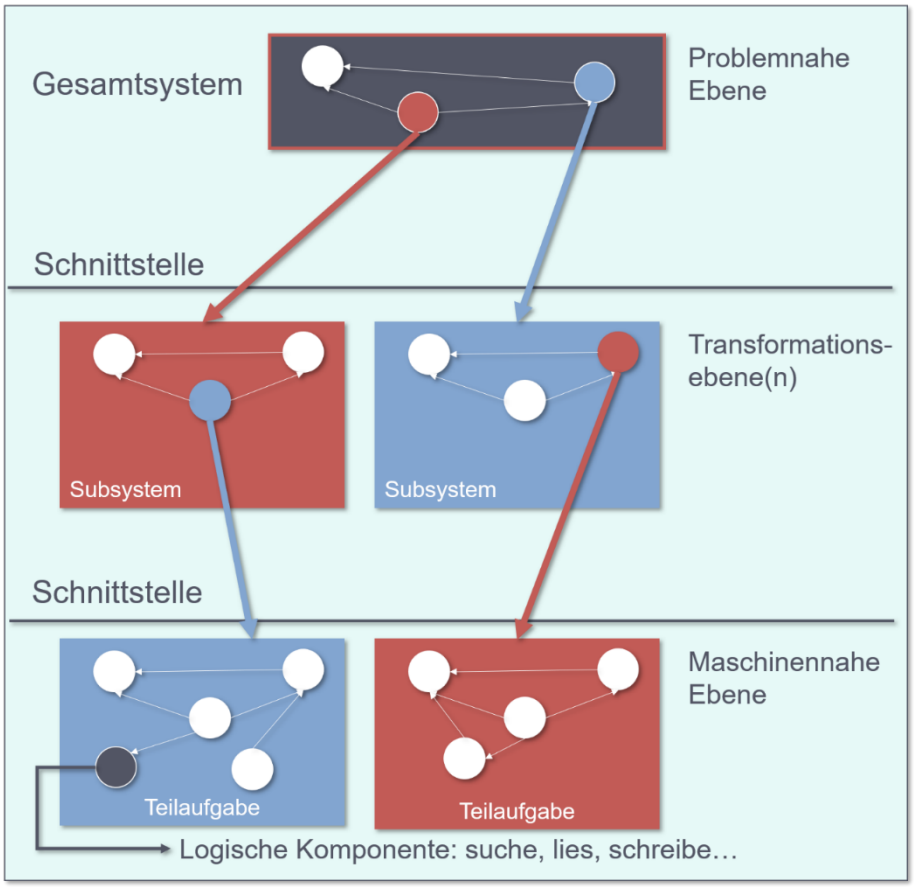


Abb. 117: Abstraktion des Gesamtsystems

9.1.12 Blockkonzept und Modularisierung – Teil 1

- Vollständige Schachtelung
- Logische Komponenten
- Bausteine, Moduln

Das Blockkonzept bietet die Möglichkeit, ein Gesamtsystem wie z. B. ein ERP-System schrittweise in seine Subsysteme und Moduln aufzubrechen. Ein ERP-System in einem Unternehmen kann aus verschiedenen Subsystemen, wie beispielsweise einem System für die Abteilung „Einkauf“ und einem für die Abteilung „Lager“, bestehen. Die Subsysteme sind logisch betrieblich miteinander verbunden, können aber einen getrennten Funktionsumfang liefern. Subsysteme werden folglich zweckgebunden betrachtet und können in logische Komponenten unterteilt werden. Subsysteme bestehen wiederum aus einer geordneten Sammlung von Modulen bzw. Bausteinen, die miteinander in Beziehung stehen. Diese Module können für mehrere Subsysteme gleichzeitig notwendig sein.

Ein Beispiel: Eine Mitarbeiterin aus der Abteilung Einkauf startet in ihrem ERP-System eine Anfrage, wie viele Artikel mit einer bestimmten Artikelnummer im Lager vorrätig sind. Das Subsystem der Abteilung „Einkauf“ greift bei dieser Anfrage auf die Datenhaltung des zum Subsystem „Lager“ zugeordneten Moduls „Warenbestand“ zurück.

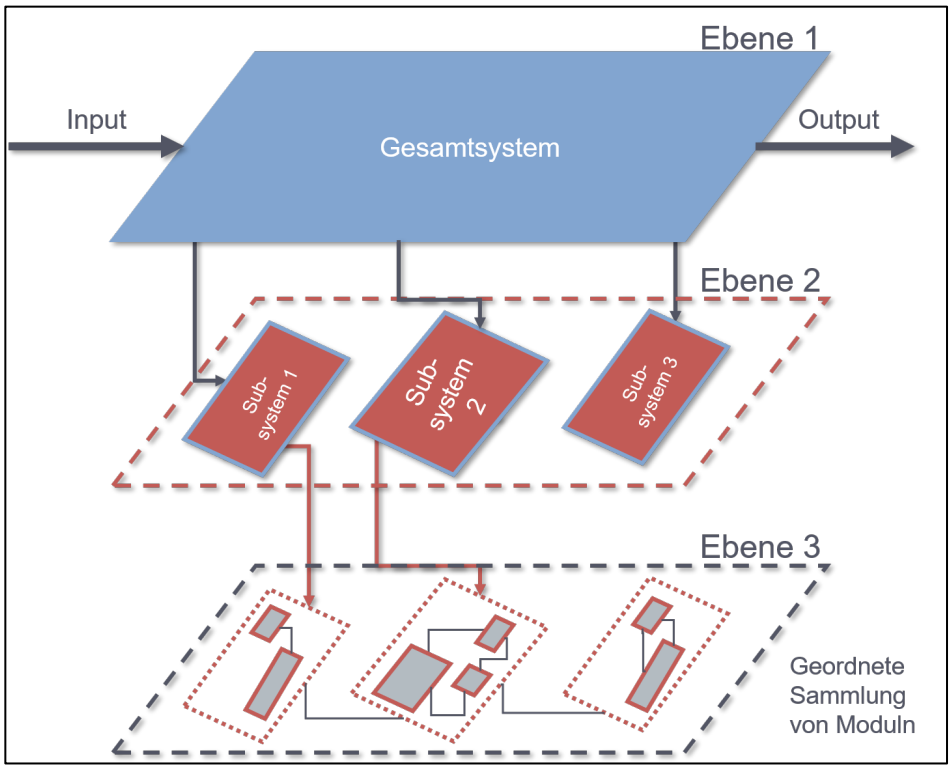


Abb. 118: Systemtheorie und Blockkonzept

9.1.13 Blockkonzept und Modularisierung – Teil 2

- Mehrfach verwendbare Moduln mit definierten Schnittstellen
- Keine Nebeneffekte bei Moduländerung oder Modulaustausch

Module entstehen durch Modularisierung. Module sind die kleinstmögliche Ebene einer Systemzerlegung. Der Vorteil von Modulen ist, dass diese mehrfach verwendet werden können.

Lediglich die Anpassung der Input- und Output-Schnittstellen muss durch die Veränderung neu definiert werden.

Module bilden darüber hinaus genau eine Funktion ab, wie zum Beispiel eine einfache Rechenoperation. Module sind mit Code-Snippets vergleichbar. In der Programmiersprache JavaScript gibt es bspw. Bibliotheken mit vorgefertigten Funktionen. Dieser fertig geschriebene Code lässt sich in den vorhandenen Code einfach einfügen und erweitert somit die Funktionen einer Web Site z. B. durch eine Bildgalerie.

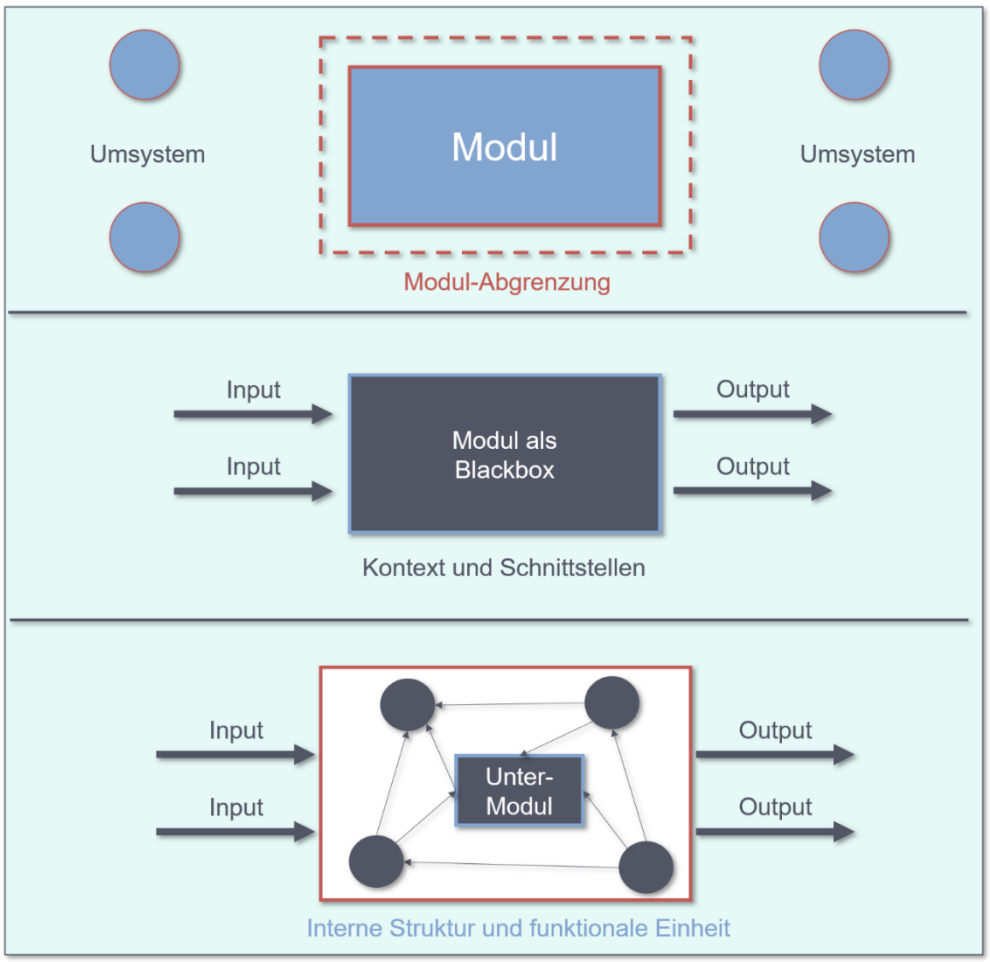


Abb. 119: Systemtheorie und Modularisierung

9.1.14 Strukturblockarten

- Beschränkung der Strukturblockarten
- Standardisierte Ablaufabbildungen
- Beherrschbare Dynamik

Die interne Struktur der Module folgt dabei einer standardisierten Ablauffolge, der Sequenz, Selektion oder Iteration – eben nicht einem Spaghetti Junction Design. Die Beschränkung auf diese drei Strukturblockarten macht die Dynamik eines Moduls beherrschbar.

Die funktionale Einheit eines Moduls ist somit auf einen Standard festgelegt, der einen Ablauf fixiert und somit in verschiedenen Modulen transparent einsetzbar ist.

Bei einer Sequenz folgen Elemente aufeinander. Bei einer Selektion wird auf Basis einer Entscheidungsregel ein Element von mehreren ausgewählt.

Bei einer Iteration wird auf Basis einer Entscheidungsregel definiert, wie oft der Vorgang wiederholt wird.

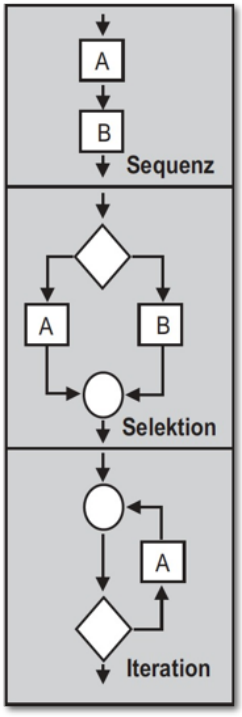


Abb. 120: Systemtheorie und Strukturblockarten

9.1.15 Beispiel: Strukturblockarten

Stellen Sie sich die Bearbeitung eines Vertrages mit Hilfe von Software vor.

Der Vertriebsmitarbeiter legt für das Angebot einen Datensatz in einer Datenbank an.

Dieses Angebot muss nun einen Genehmigungsprozess durchlaufen, da der Vertriebsmitarbeiter einen Rabatt gewährt hat, für den er die Zustimmung seines Managers benötigt.

Durch einen fest definierten Ablauf mit vordefinierten Strukturblockarten kann dieser Vorgang für jedes Angebot bearbeitet werden. Dabei spielt es keine Rolle, wer das Angebot erstellt hat oder wann und wo das Angebot erstellt wurde.

Dies funktioniert jedoch nur, wenn der Input für jedes Angebot das gleiche Format besitzt. Beispielsweise kann es nur ein Datumsformat geben, um das Ende der Angebotszeit zu errechnen.

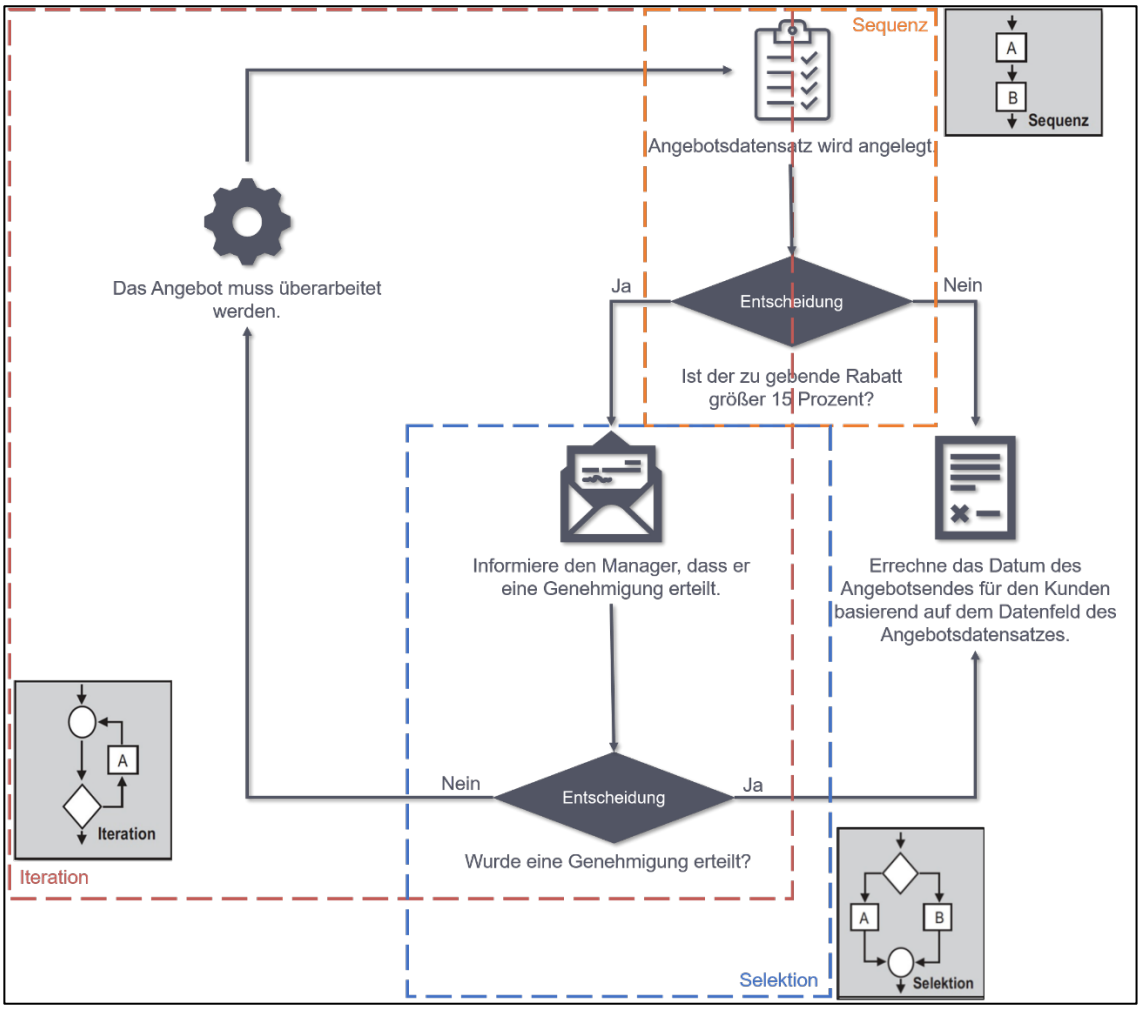


Abb. 121: Beispiel der Strukturblockarten anhand eines Prozesses

9.1.16 Beispiel: Modularisierung

Nehmen Sie an, der Manager hat entweder das Angebot genehmigt oder der Rabatt ist kleiner fünfzehn Prozent. Nun sollte basierend auf dem Datenfeld das Ende des Angebotes berechnet werden.

Der Input A, hier in blau dargestellt, ist definiert durch das Datumsformat DD.MM.YYYY und der Input B als Ganzzahl (integer) und Tagesbasis (türkis), nicht Wochen oder Monate.

Das Modul berechnet nun das Angebotsende, ohne den Kontext zu kennen und produziert den Output C (rot) wieder im Zahlenformat.

Das Modul ist nicht kontextbezogen, sondern kann auch in anderen Anwendungen verwendet werden. Zum Beispiel zur Errechnung des Vertragsendes der Mitarbeiter oder der Probezeit. Das Modul lässt sich ohne Nebeneffekte in verschiedenen Subsystemen verwenden.

Die Beispiele verdeutlichen bereits, dass es sich bei Software-Entwicklungen um komplexe Sachverhalte handelt. Deshalb ist die Systemtheorie ein wichtiges Werkzeug bei der Erstellung der Software-Produkte. Die Systemtheorie unterteilt einen komplexen Sachverhalt in einzelne Aufgabenkomplexe. Die kleinsten Einheiten – die Module – können dabei effizient mehrfach eingesetzt werden.

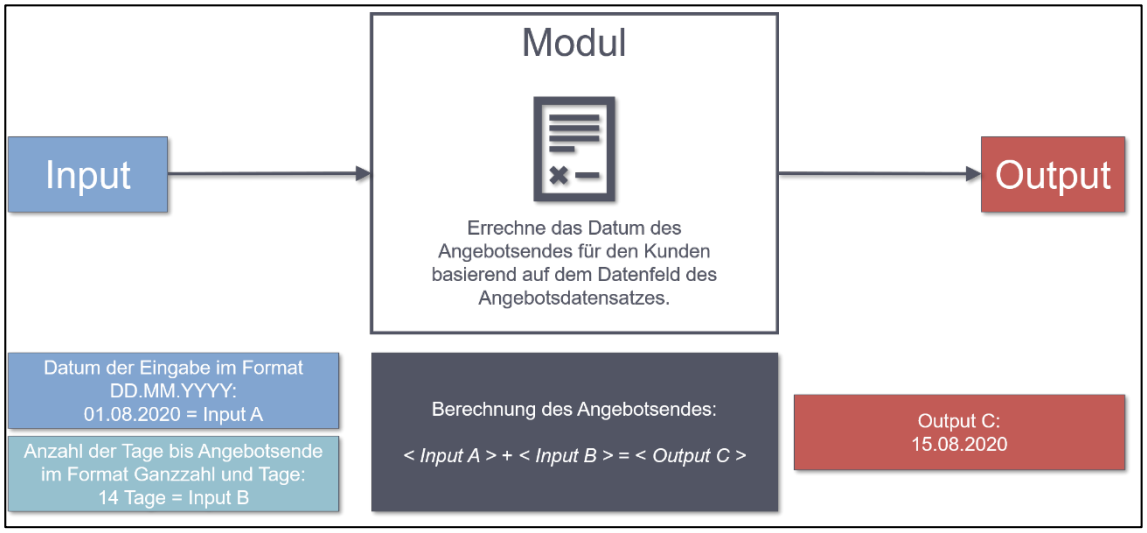


Abb. 122: Beispiel der Modularisierung anhand einer Rechenoperation

9.2 Die Graphentheorie

9.2.1 Komplexe Sachverhalte analysieren

Graphen sind mathematische Modelle für netzartige Strukturen in Natur und Technik. Nicht nur Computer-Netze, Projektpläne und Straßennetze können mit Hilfe der Graphentheorie optimiert werden, sondern auch chemische Moleküle werden durch Graphen dargestellt.

Im Software Engineering tauchen oftmals Optimierungsprobleme auf, die mit Hilfe von bestimmten Algorithmen gelöst werden können. Optimierungsprobleme können durch

Graphen dargestellt werden. Die Graphentheorie ist ein Werkzeug, das zur Lösung solcher Probleme beitragen kann.

Die Graphentheorie findet in vielen verschiedenen Forschungsgebieten Anwendung und ist, wie die Systemtheorie, eine Schablone zur Modellierung und Vereinfachung einzelner Aspekte unserer Welt.

Beispielsweise lässt sich durch die Graphentheorie ermitteln, ob gewisse Abläufe in einem Projekt noch Optimierungspotenzial bieten.

9.2.2 Graphentheorie

- Formale Theorie, die Systeme, Vorgänge und Abfolgen untersucht.
- Konzentriert sich auf Beziehungen zwischen Knoten und Kanten.

Zum Begriff „Graph“

- Unter einem Graph versteht man eine Menge von Knoten, ...
- ... die untereinander mit Kanten verbunden sind (siehe „System“).
- Gerichtete Graphen zeigen auch die Art der Verbindung (Richtung der Einflussnahme) zwischen den Knoten an.

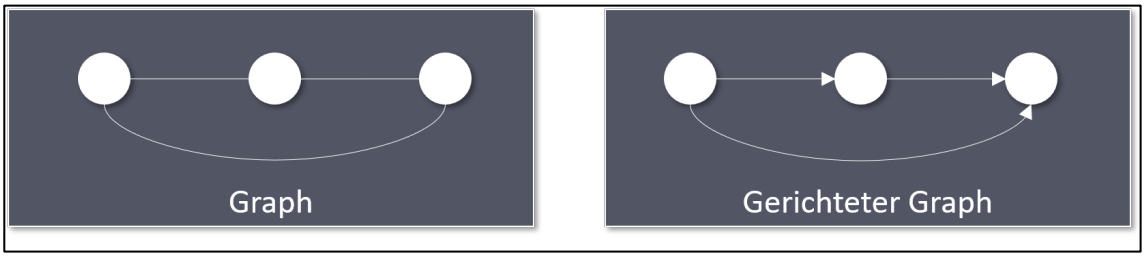


Abb. 123: Graph und gerichteter Graph

9.2.3 Zum Begriff „Graph“

Unter einem Graph versteht man eine Menge von Knoten, die untereinander mit Kanten verbunden sind (siehe „System“). Gerichtete Graphen zeigen auch die Art der Verbindung (Richtung der Einflussnahme) zwischen den Knoten an.

9.2.4 Anwendungsbereiche der Graphentheorie

Anwendungsbereiche der Graphentheorie

- Operations Research: Quantitative Methoden der Planung und Entscheidungsvorbereitung
- Ingenieur- und wirtschaftswissenschaftliche Bereiche

Eulersches Brückenproblem

- Euler 1736 in Königsberg: Gibt es einen Spazierweg über das Brückensystem der Pregel, bei dem jede Brücke genau einmal passiert wird?

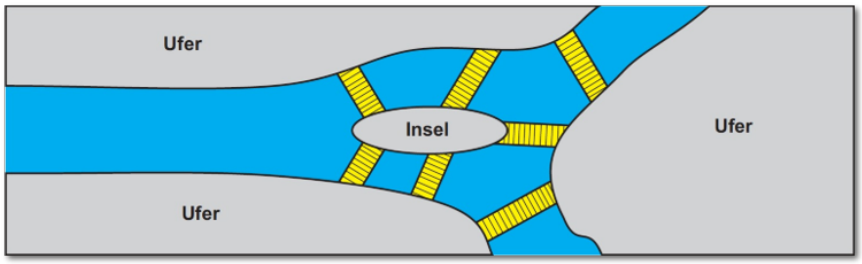


Abb. 124: Eulersches Brückenproblem

9.2.5 Beispiel: Travelling Salesman

Ein Vertriebsbeauftragter will eine bestimmte Menge von Kunden auf einer Rundreise mit minimaler Länge besuchen.

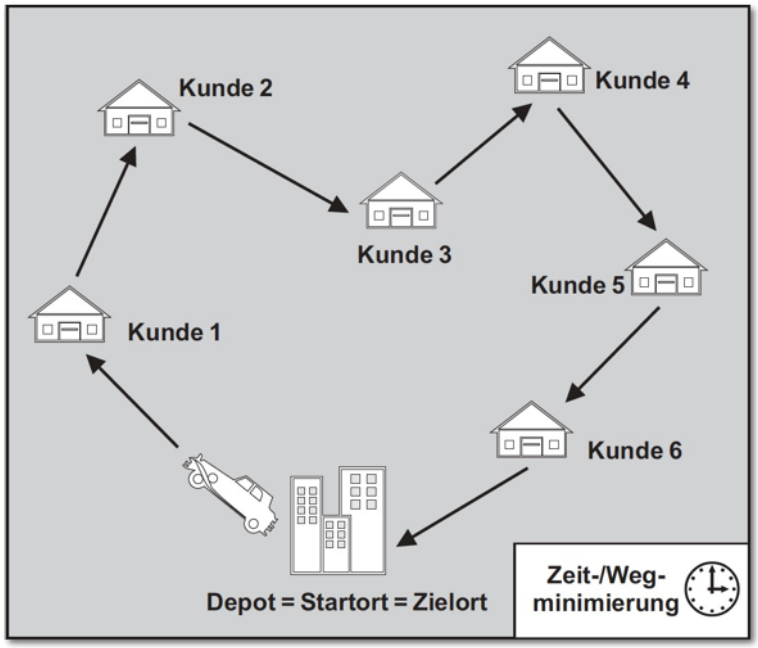


Abb. 125: Graphentheorie: Travelling Salesman

9.2.6 Beispiel: Touren-Problem

Innerhalb eines bestimmten Zeitraums ist eine bestimmte Menge von Kunden mit einer bestimmten Menge Lieferwagen mit bestimmten Produktmengen möglichst effizient zu beliefern.

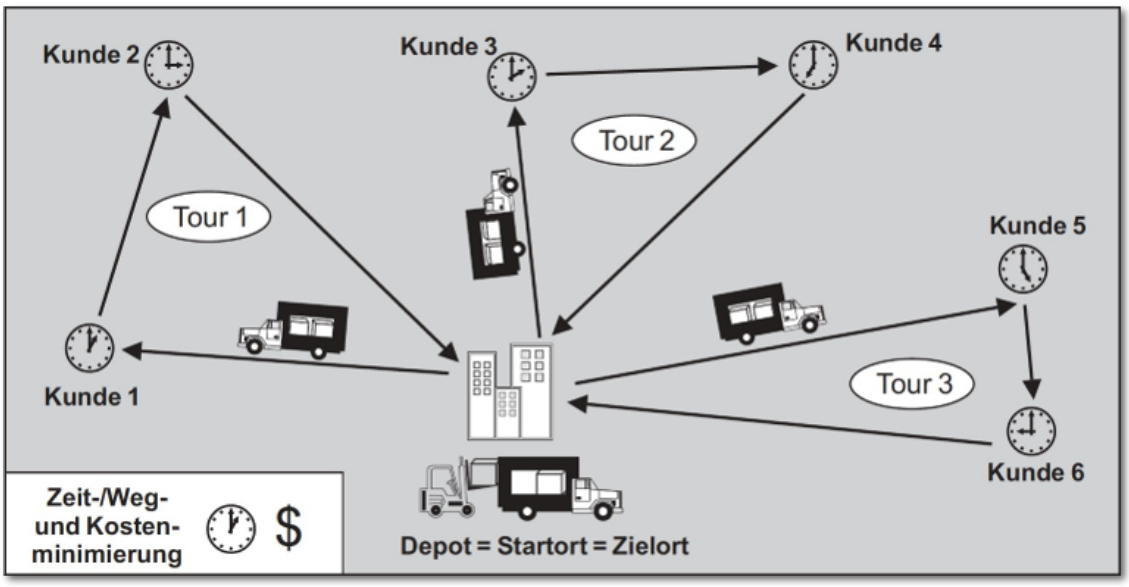


Abb. 126: Graphentheorie: Touren-Problem

9.2.7 Beispiel: Transport-Problem

Von einer bestimmten Menge an Depots aus ist eine bestimmte Menge von Kunden mit einer bestimmten Menge Lieferwagen mit bestimmten Produktmengen möglichst effizient zu beliefern ohne Transporte zwischen Kunden oder zwischen Depots.

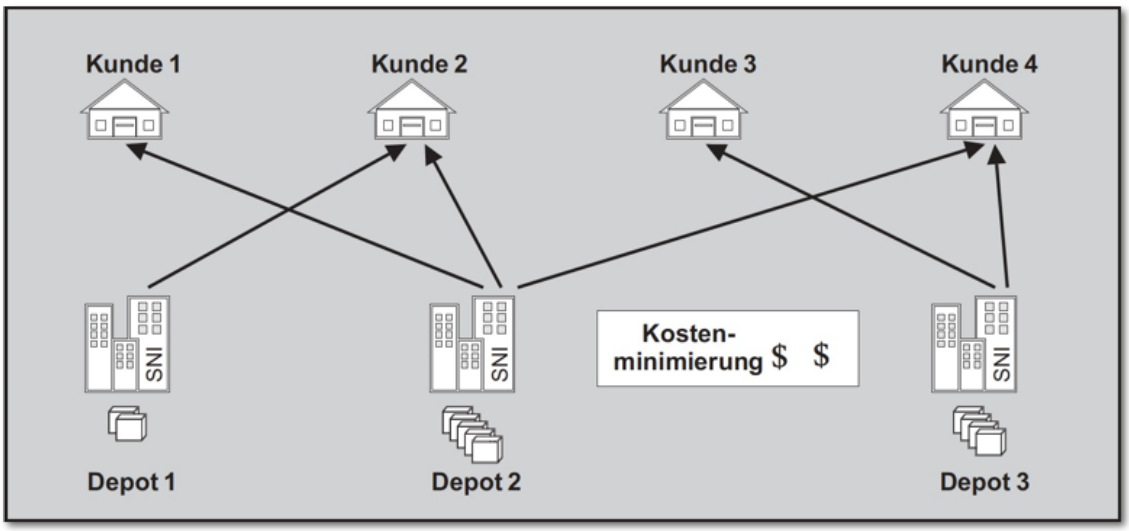


Abb. 127: Graphentheorie: Transport-Problem

9.2.8 Netzplantechnik

- Große, komplexe Projekte planen, analysieren, steuern, kontrollieren, optimieren.
- Ablaufstrukturplan mit zeitbeanspruchenden und zeitpunktbezogenen Elementen muss vorher erstellt worden sein.
- Zeitdauern und Vorgänger-/Nachfolgerbeziehungen sind bekannt.
- Netzplantechnik als Anwendungsmethode der Graphentheorie

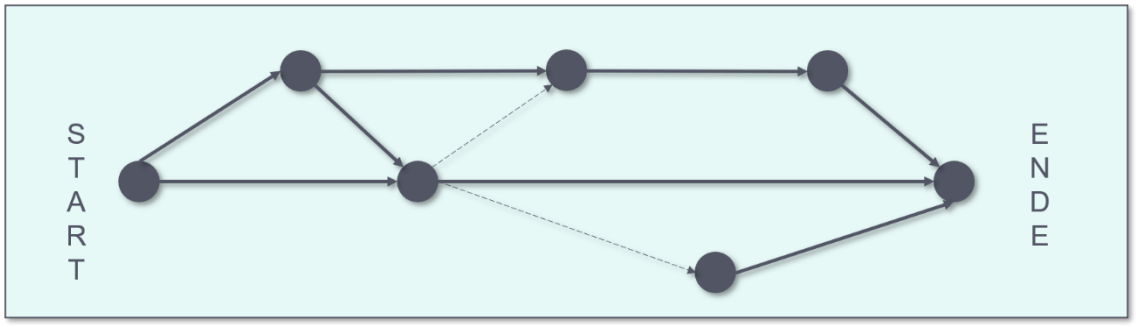


Abb. 128: Schema der Netzplantechnik

9.2.9 Beispiel: Die Netzplantechnik beim Hausbau

Die Netzplantechnik kann dabei helfen, vor allem unternehmerische Prozesse mit mehreren Teilschritten übersichtlich abzubilden und Abläufe zu optimieren. Dies kommt natürlich vor allem großen Projekten zugute.

Denken Sie noch einmal an unser Hausbau-Beispiel: Wann können beim Bau des Hauses Arbeiten gleichzeitig ausgeführt werden? Kann der Boden verlegt und das Dach gleichzeitig gebaut werden oder sollte das Dach vorher fertig sein, da es eventuell regnen könnte?

Schon bei dem vermeintlich einfachen Beispiel wird klar, dass die Elemente des Hausbaus genau überlegt werden müssen. Die vollständigen Arbeitsschritte, deren Dauern und Reihenfolgen müssen geplant und festgelegt werden, bevor Sie optimieren können.

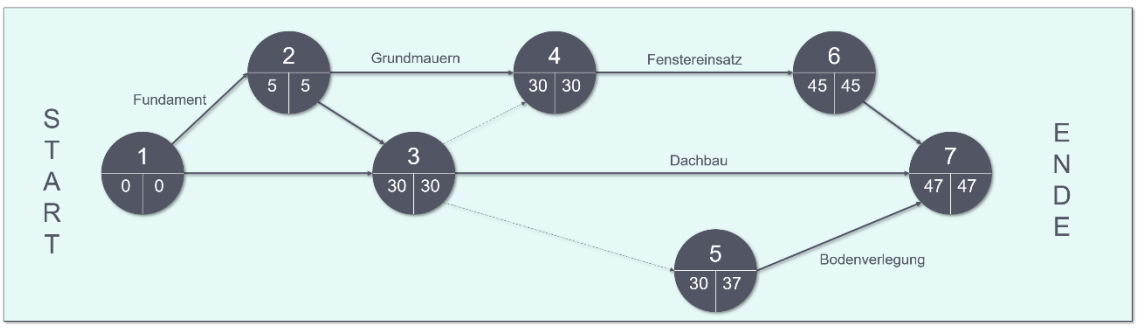


Abb. 129: Netzplantechnik anhand eines Hausbaus

9.2.10 Graphentheorie in der Praxis

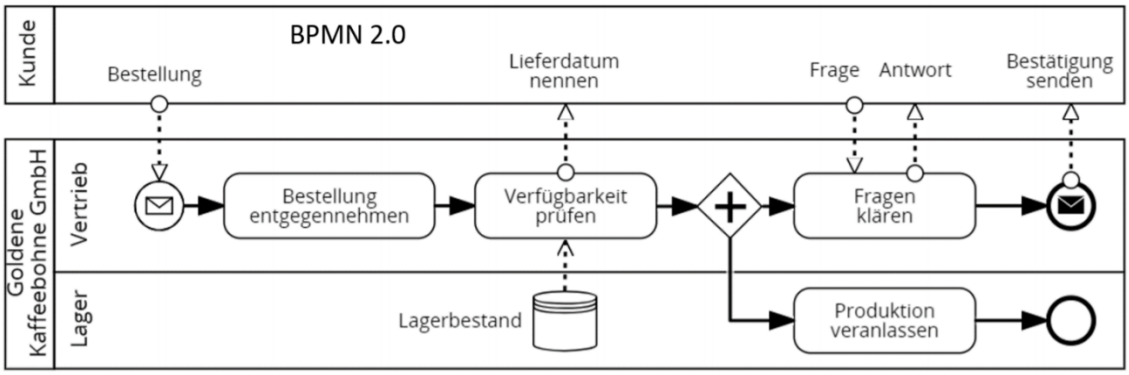


Abb. 130: Beispiel eines BPMN 2.0

Die Graphentheorie findet sich gerade in betriebswirtschaftlichen Prozessmodellierungen wieder. Da die Graphentheorie in vielen verschiedenen Einsatzgebieten etabliert ist, gibt es auch verschiedene Typen von Graphen.

Im Folgenden werden die Petri-Netze als Ausprägung von Graphen vorgestellt. Darüber hinaus finden Sie auf unserem E-Campus weitere Modellierungsmethoden für Geschäftsprozesse wie z. B. mit Hilfe von BPMN.

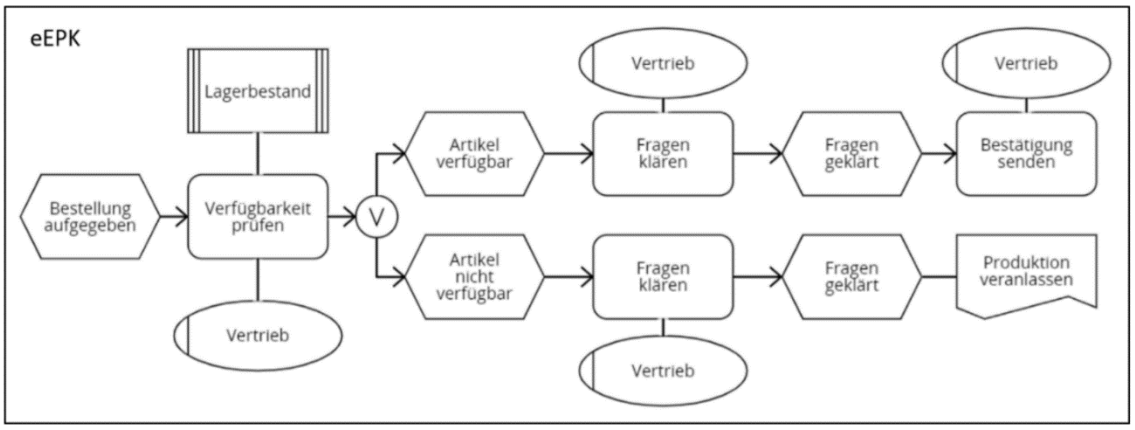


Abb. 131: Beispiel einer eEPK

9.3 Anwendung der Graphentheorie

9.3.1 Petri-Netze zur Modellierung

Die Petri-Netze sind eine Anwendung der Graphentheorie. Petri-Netze decken ein breites Anwendungsfeld ab und sind relativ einfach verständlich.

Petri-Netze definieren immer einen eintretenden Zustand und einen Zustandsübergang. Somit sind Petri-Netze sehr gut für die Modellierung und Analyse von Systemen und Prozessen geeignet. Gerade Computersysteme lassen sich durch IPO-Prozesse definieren (Input – Process – Output = Zustand – Zustandsübergang – Zustand).

Die graphische Darstellung von Prozessen hilft bei Verständnis und Analyse eines Modells.

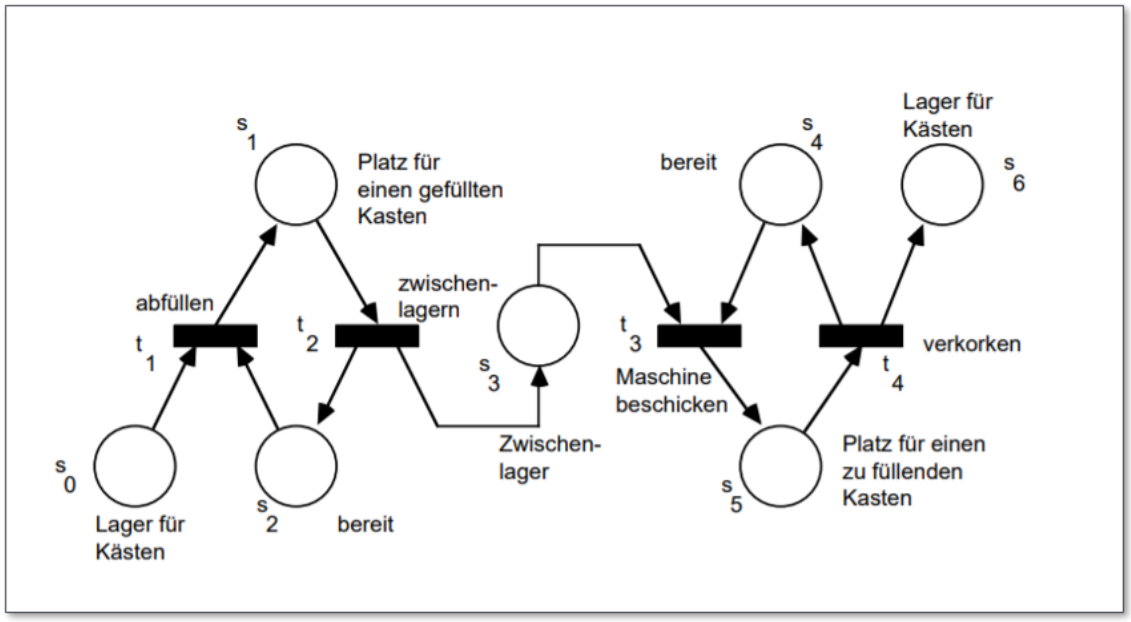


Abb. 132: Beispiel eines Petri-Netzes

9.3.2 Anwendungsbereiche der Petri-Netze

Petri-Netze: **Anwendungsbereiche**

- Anwendungsneutral: Mit Petri-Netzen lassen sich Systeme und Abläufe modellieren, die auf diskreten Aktionen und Abhängigkeitsbeziehungen zwischen diesen Aktionen basieren.
- Erlauben die Simulation der Modell-Dynamik mittels Verhaltensregeln.
- Bspw. Modellierung, Analyse und Simulation im Bereich der Automatisierungstechnik
- Wirtschaftswissenschaften: Modellierung von betrieblichen Abläufen als hochaktuelles Anwendungsgebiet – Geschäftsprozess-Modellierung
- U. a.: Geschäftsprozesse, Workflow-Management-Systeme, Customizing von Standard-Anwendungssoftware

Petri-Netze: **Elemente**

- Statische Knoten-Elemente: Zustände (passiv) und Ereignisse (aktiv)
- Kanten: Kausal-logische Zusammenhänge zw. Zuständen und Ereignissen
- Dynamische Elemente: Marken (für Zustände)

9.3.3 Elemente der Petri-Netze

Passive Elemente: Zustände (Kreise)

- Zustand (auch: Stelle, Platz, Bedingung)
- Momentane Lage, Situation eines Systems bzw. Stand eines Prozesses
- Bspw.: Lagerbestand, in Bearbeitung, offene Rechnung, warm, bereit
- Zustände werden als Kreise dargestellt.

Kausal-logische Zusammenhänge: Kanten (Pfeile)

- Gerichtete Kanten = Pfeile

Aktive Elemente: Ereignisse (Rechtecke)

- Ereignis (auch: Transition, Zustandsübergang, Aktion)
- Bewirken den Übergang von einem Zustand in einen anderen Zustand.
- Bspw.: Lagerbewegung, bearbeitet, Bezahlung, Erhitzung, fertigstellen
- Ereignisse werden als Rechtecke dargestellt.

9.3.4 Ereignisse und Zustände – Teil 1

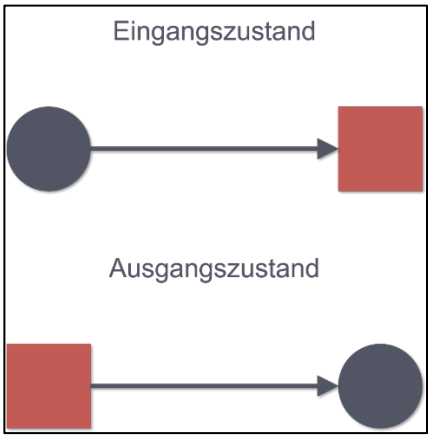


Abb. 133: Petri-Netze und Zustandsbeschreibungen

- Ein Ereignis beschreibt die Erzeugung, Veränderung, den Transport von z. B. Daten oder Rohstoffen bei der Realisierung von Prozessen.
- Ein Ereignis setzt genau definierte und erfüllte Zustände voraus und / oder führt zu einer genau definierten Menge von Zuständen.
- Die Beendigung eines Zustands erfolgt durch mindestens ein Ereignis.
- Der Beginn eines neuen Zustands wird von mindestens einem Ereignis angestoßen.
- Zustände und Ereignisse wechseln sich immer ab: bipartiter Graph

9.3.5 Ereignisse und Zustände – Teil 2

Ergebnisse und Zustände	Falsch	Richtig
Wege beginnen und enden nicht mit einer Kante.		
Es gibt keine alleinstehenden Knoten.		
Es gibt keine parallel verlaufenden Kanten.		
Es gibt keine bidirektionalen Kanten.		

Abb. 134: Mögliche Zustände und Ereignisse der Petri-Netze

9.3.6 Eingangs- und Ausgangszustand

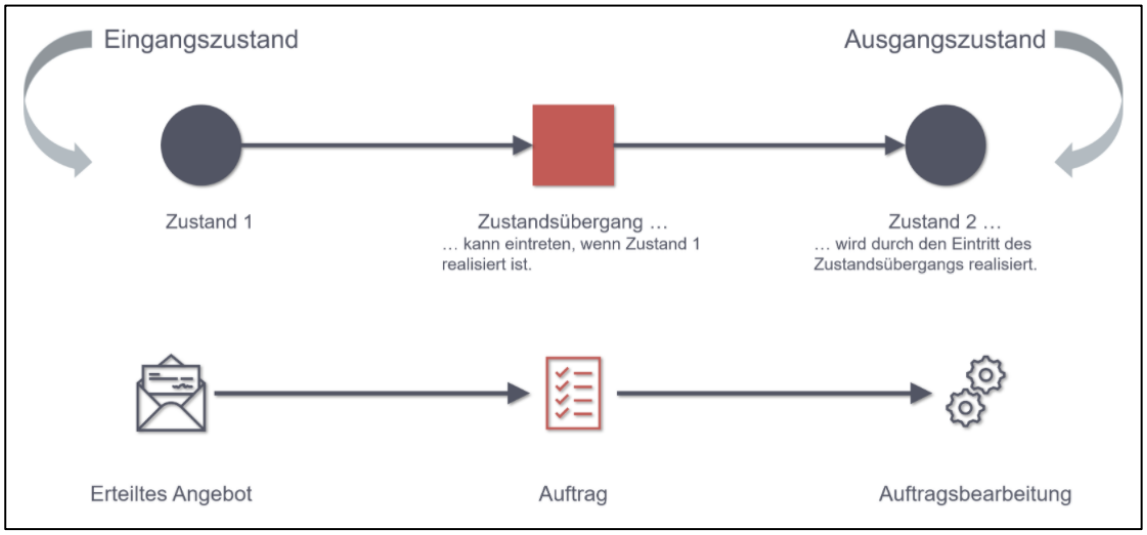


Abb. 135: Eingangs- und Ausgangszustände der Petri-Netze

Im oberen Beispiel sei nochmal der Unterschied zwischen den Zuständen mit betriebswirtschaftlichem Hintergrund verdeutlicht. Im Beispiel einer Auftragserstellung ist ein Angebot ein Zustand, welcher intern in einen Auftrag umgewandelt werden muss. Erst wenn aus der schriftlichen Bestätigung des Angebotes ein Auftrag entsteht, kann dieser von den Mitarbeitern bearbeitet werden.

9.3.7 Beispiel: Prädikats- / Transitions-Netz

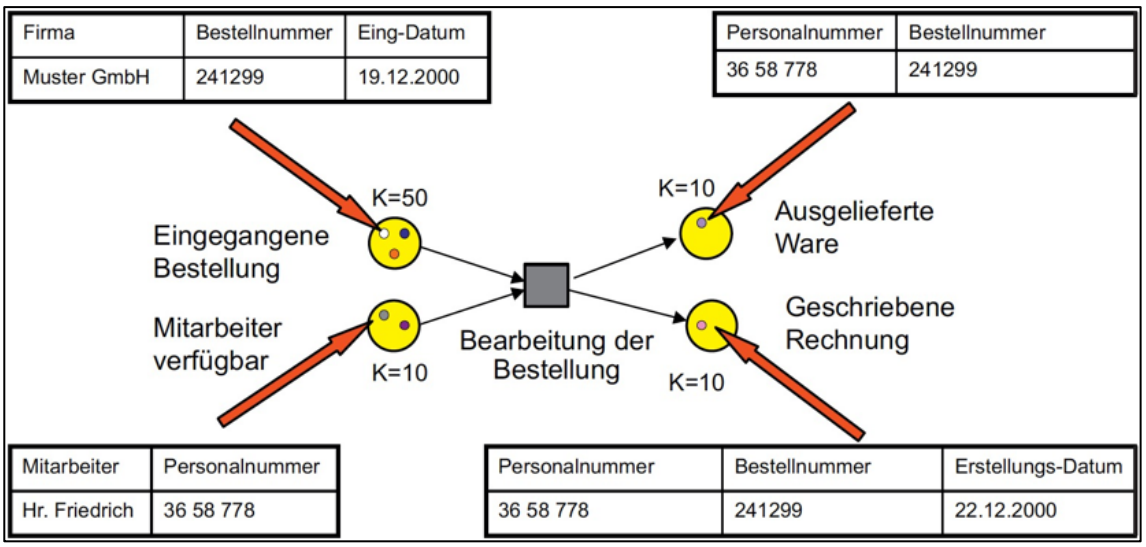


Abb. 136: Beispiel eines Prädikatsnetzes

In diesem Beispiel wird der Vorgang einer Auftragsbearbeitung verdeutlicht. Die eingegangene Bestellung muss von einem verfügbaren Mitarbeiter bearbeitet werden. Sowohl Bestellung als auch Mitarbeiter haben dabei bestimmte Ausprägungen, wie die Bestellnummer oder Personalnummer. Ist die Bestellung erfolgreich bearbeitet, kann die Ware ausgeliefert und die Rechnung erstellt werden. Auch diese beiden Zustände haben wiederum bestimmte Ausprägungen.

9.3.8 Vor- und Nachteile von Petri-Netzen

Vorteile Petri-Netze

- Klare Abbildung von Sequenzen, Nebenläufigkeiten und Verklemmungen in Prozessstrukturen
- Semantik der Petri-Netze lässt sich einfach identifizieren
- Verschiedene Abstraktionsebenen möglich (Vergröberung, Verfeinerung)
- Sowohl Statik als auch Dynamik von Prozessen modellierbar
- Relativ leichte Erlernbarkeit einfacher Petri-Netztypen
- Mathematische Verifizierbarkeit der Netze
- Die Petri-Netz-Grundlagen basieren auf den Grundlagen der System- und Graphentheorie und finden sich in diversen Modellierungsansätzen für betriebliche IT-Systeme wieder.

Nachteile Petri-Netze

- Fehlen von Systematiken zum Vorgehen bei der Erstellung von Petri-Netzen
- Modellierung subjektiv / keine Petri-Netz-orientierte Methoden verfügbar
- Erhöhter Lernaufwand höherer Petri-Netztypen

9.3.9 Anwendungsfelder der Petri-Netze

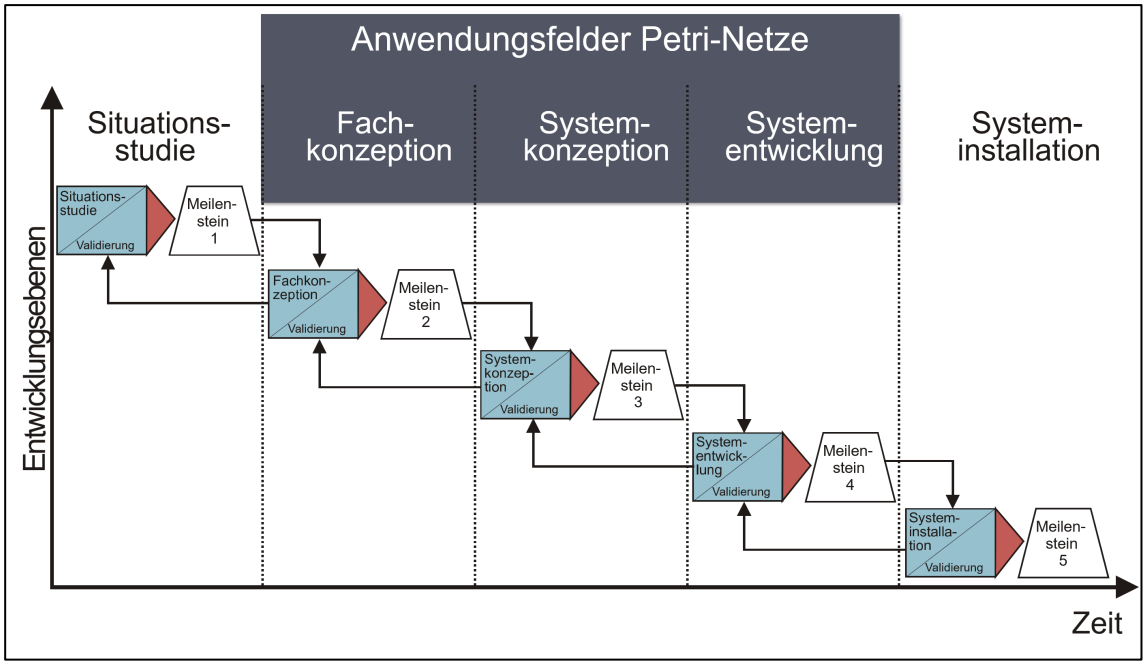


Abb. 137: Anwendungsfelder der Petri-Netze

Petri-Netze sind einfach anwendbar. Sie lassen sich dabei für verschiedene Zwecke nutzen. Dadurch sind Petri-Netze in den Phasen: Fachkonzeption, Systemkonzeption und Systementwicklung anzuwenden.

Bei der Fachkonzeption nimmt man Bezug auf betriebswirtschaftliche Prozesse, die mit Zuständen und Übergängen beschrieben werden.

In der Systemkonzeption könnte man diesen Prozessen systemrelevante Informationen hinzufügen, wie in unserem Beispiel des Prädikats- / Transitionsnetzes.

Letztendlich können Petri-Netze auch bei der System-Entwicklung verwendet werden. Die Erfahrung zeigt, dass die kleinste Einheit eines Systems, das Modul, einen definierten Input (Zustand) benötigt, der verarbeitet wird (Zustandsübergang) und letztendlich in einem Output (Zustand) mündet.

Petri-Netze lassen sich folglich in diesen verschiedenen Phasen vielfältig einsetzen und bieten eine methodische Durchgängigkeit.

9.4 Abschlusstest – WBT09

9.4.1 Abschlusstest

Bitte beantworten Sie die folgenden Fragen durch das Ankreuzen der korrekten Antworten (Tab. 10). Bei einigen Fragen können auch mehrere Antworten richtig sein.

Nr.	Frage	Richtig	Falsch
1	Das Prinzip der Dekomposition gehört nicht zur Systemtheorie.		
2	Das Blockkonzept besagt, dass in einer Manier von bottom-up ein System erbaut werden kann.		
3	Zu den Hauptaspekten der Systemtheorie gehören...		
	Wirkungsaspekt		
	Strukturaspekt		
	Blackbox-Aspekt		
	Abstraktion		
	Dekomposition		
4	Das Blockkonzept ist beispielsweise dazu da ein ERP-System in seine Subsysteme und Module aufzuteilen.		
5	Folgende Aussagen über Petri-Netze sind richtig:		
	Petri-Netze sind sehr spezifische Graphen.		
	Auf einen Zustand folgt immer ein Ereignis.		
	Ein Petri-Netz kann auf ein Ereignis enden.		
	Petri-Netze können nicht in der Netzplantechnik angewandt werden.		
	Petri-Netze helfen Abläufe zu optimieren.		
6	Ein Graph besteht aus verschiedenen _____. Zu diesen gehören _____ und _____.		

Tab. 10: Abschlusstest – WBT09

9.5 Typische Aufgabenstellungen

Typische Aufgabenstellungen – Grundlagen der Modelldarstellung

Zur Bearbeitung dieser Aufgabenstellungen beachten Sie bitte: Verlangt ist eine fachlich zutreffende, inhaltlich nachvollziehbare und kausal zusammenhängende Erörterung aus vollständigen Sätzen in lesbarer Handschrift. Für jede Aufgabe: Maximal zwei Seiten Text.

Aufgabe 1:

> Was ist ein System? Woraus besteht ein System?

Aufgabe 2:

> Erläutern Sie die systemtheoretischen Grundlagen (Gegenstand, Begriff) der Modellierung von IuK-Systemen und was unter Wirkungsaspekt, Strukturaspekt, Abstraktion und Dekomposition zu verstehen ist.

Aufgabe 3:

> Erläutern Sie den systemtheoretischen Aspekt „Blockkonzept und Modularisierung“ bei der Konstruktion von betrieblichen IuK-Systemen.

Aufgabe 4:

> Wenden Sie die 4 Systemaspekte auf das Modul „Systems Engineering“ an. Wenden Sie die 4 Systemaspekte auf Ihren Studiengang an.

Aufgabe 5:

> Nennen und erläutern Sie stichwortartig die „Prinzipien für die Entwicklung von betrieblichen Informations- und Kommunikationssystemen“.

10 Aufgaben- und funktionsorientierte Modellierung

10.1 Prinzipien und Objekte der Modellierung

10.1.1 Die Prinzipien der Modellierung

Im vorigen WBT wurden die Grundlagen der Modelldarstellung behandelt. WBT 10 erläutert nun die Prinzipien und Objekte der Modellierung.

Die Objekte der Modellierung erläutern das „Was“ der Modellierung. Die Objekte der Modellierung geben dabei eine unterschiedliche Sicht auf das zu modellierende System.

Die Prinzipien dienen als Leitfaden bei der Modellierung und beantworten die Frage, „wie“ bzw. nach welchen Grundsätzen modelliert wird. Die folgenden sechs Prinzipien helfen dabei, ein System zu modellieren:

1. Prinzip der hierarchischen Dekomposition
2. Prinzip der Strukturierung und Modularisierung
3. Prinzip der konstruktiven Voraussicht
4. Prinzip der perspektivischen Betrachtung
5. Prinzip der methodischen Standardisierung
6. Prinzip der Trennung der Essenz von der Inkarnation

10.1.2 Prinzip der hierarchischen Dekomposition

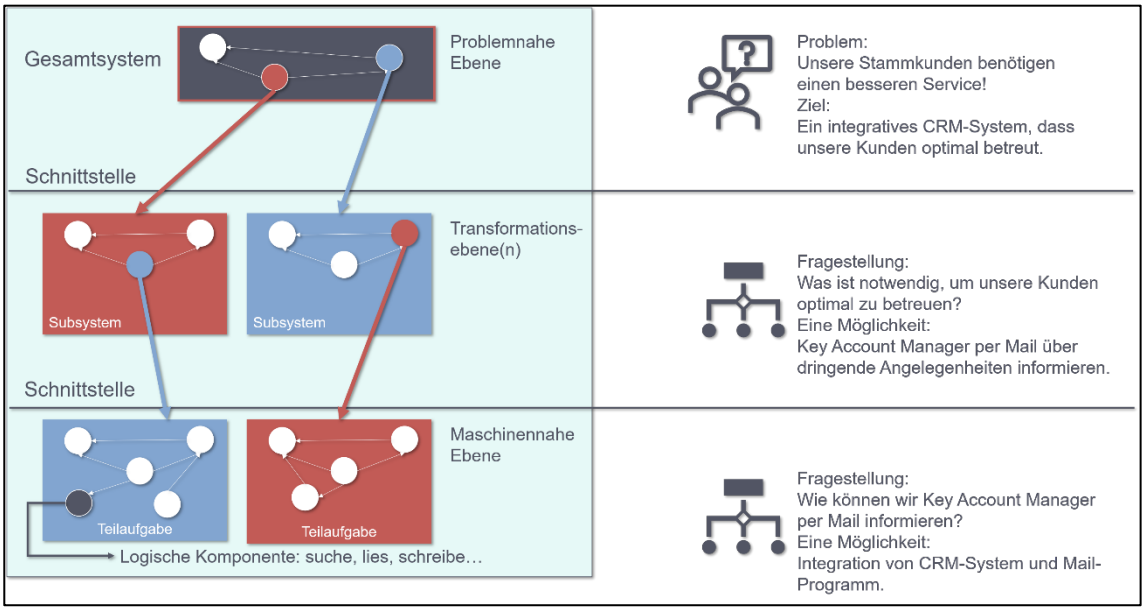


Abb. 138: Beispiel hierarchische Dekomposition

- Nach den systemtheoretischen Grundlagen der Abstraktion und schrittweisen Verfeinerung
- Top-down
- Deduktiv!
- Induktiv?

Unter Abstraktion versteht man Verallgemeinerung, die Kernzentralisation auf das Wesentliche, das Absehen vom Besonderen und Einzelnen, das Loslösen vom Dringlichen. Abstraktion ist das Gegenteil von Konkretisierung.

Die hierarchische Dekomposition zerlegt die fachliche, übergeordnete Problemstellung, den Zweck eines Systems. Die Problemstellung wird dadurch redundanzfrei in seine Unterfunktionen zerlegt. Man folgt dabei der obersten Ebene, dem Problem, welches daraufhin aufgegliedert wird. Das Problem könnte bspw. darin bestehen, Kunden besser mit Hilfe eines Systems zu bedienen.

Daraufhin entwickelt man Subsysteme. Hier stellt man sich die Frage: Was ist notwendig, um Kunden besser durch ein System zu bedienen?

Zum einen könnte dies ein Service sein, der einkommende Mails automatisch an die Key Accounts weiterleitet, die für den Kunden zuständig sind. Zum anderen könnte es eine Funktion geben, die häufig gestellte Fragen automatisch auf einer Web Site anzeigt und erweitert.

Diese Beispiele lassen bereits erahnen, dass diese Funktionen noch weiter konkretisiert werden können. Genauso wird in der hierarchischen Dekomposition schrittweise verfeinert. Die beschriebene Vorgehensweise ist deduktiv. Das System wird von oben nach unten zerlegt. Aber auch ein induktives Vorgehen von unten nach oben (Buttom-up) ist denkbar.

10.1.3 Prinzip der Strukturierung und Modularisierung

- Nach den systemtheoretischen Grundlagen der Blockbildung und Modularisierung
- Vollständige Schachtelung
- Definierte Schnittstellen
- Strukturblockarten

In WBT 9 „Grundlagen der Modelldarstellung“ haben Sie sich bereits mit einigen Beispielen der Modularisierung beschäftigt.

Die Idee der Modularisierung ist, das Gesamtsystem nach dem Baukastenprinzip aus wiederverwendbaren Einzelbausteinen, den Modulen, zusammenzusetzen.

Das Gesamtsystem wird in Subsysteme mit eigenen funktionalen Eigenschaften aufgegliedert, welche wiederum aus Modulen bestehen.

Module haben definierte Schnittstellen, Inputs und Outputs und können ohne Nebeneffekte in anderen Systemen wiederverwendet werden.

Wie die Module die Inputs zu Outputs verarbeiten, ist standardisiert und wird durch sogenannte Strukturblockarten beschrieben.

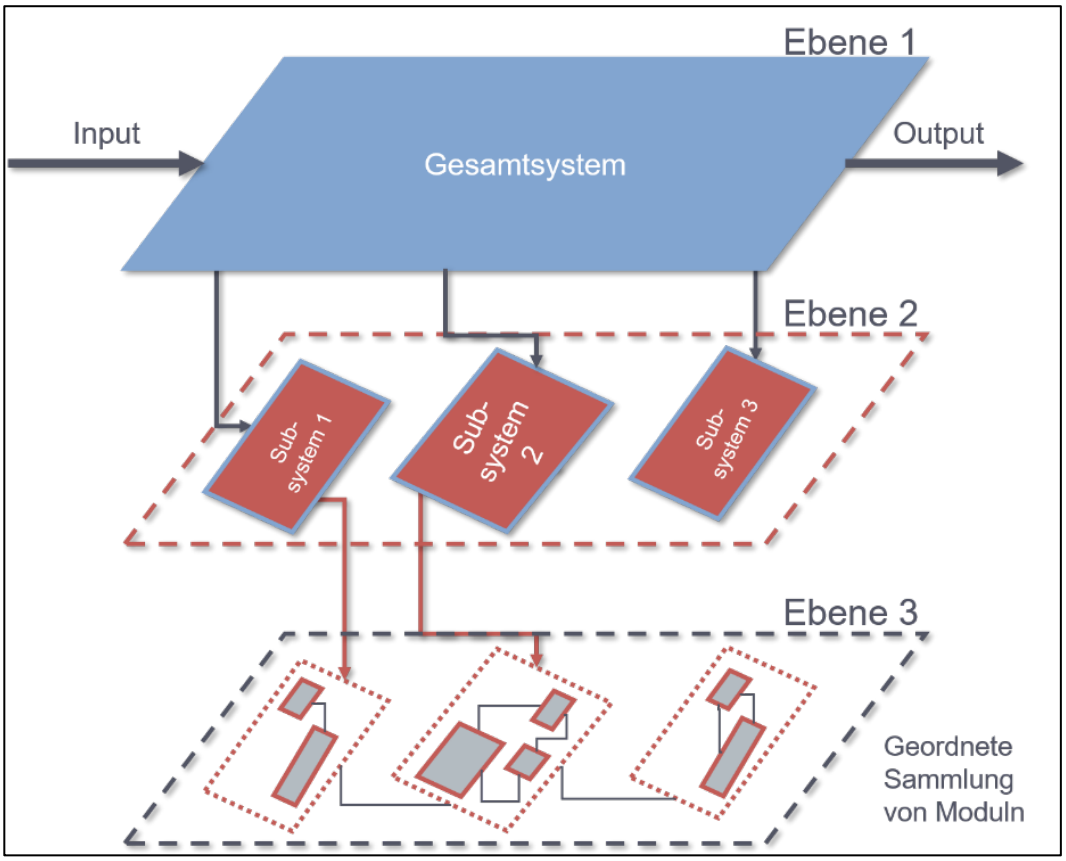


Abb. 139: Prinzip der Modularisierung

10.1.4 Prinzipien der konstruktiven Voraussicht und der perspektivischen Betrachtung

Das Prinzip der konstruktiven Voraussicht besagt, dass bereits bei den ersten Entwurfsschritten für das Software System auf dessen Änderbarkeit und Wartbarkeit zu achten ist. Schon bei der Planung eines Softwareproduktes soll durch vorausschauendes Handeln der Aufwand für Qualitätssicherung und Projektmanagement minimiert werden. Fehler, die nicht gemacht werden, brauchen auch nicht behoben werden.

An der Entwicklung eines Software-Produktes sind verschiedene Gruppen beteiligt, die das Software-Produkt aus jeweils eigenen Perspektiven betrachten – z. B. Benutzer, Ent-

wickler, Techniker, Unternehmensführung, -organisation, Kunden. Das Prinzip der perspektivischen Betrachtung besagt, dass abhängig vom Betrachter verschiedene Aspekte eines Systems zu beachten sind. Die Berücksichtigung und Integration der Perspektiven aller Beteiligten ist erforderlich.

Prinzip der konstruktiven Voraussicht

- Schon ab den ersten Schritten der Systementwicklung ist auf Qualitätssicherung, Änderbarkeit, Wartbarkeit zu achten.
- Erstellungsprozess durch praktikables Vorgehensmodell und methodische Standardisierung strukturieren.
- Integration aller Dokumentationsaktivitäten (in allen Phasen)

Prinzip der perspektivischen Betrachtung

- Abhängig vom Betrachter werden verschiedene Aspekte eines Systems herausgehoben.
- Die Integration der verschiedenen Sichtweisen ist erforderlich.
- System-Benutzer, Entwickler, Administratoren, Unternehmensführung, Kunden, Lieferanten ...

10.1.5 Prinzip der methodischen Standardisierung

- Für die Systementwicklung wird ein Bündel von Methoden, Konzepten, Techniken, Werkzeugen vom arbeitsteilig organisierten Entwicklungsteam eingesetzt.
- Alle Beteiligten sollen im Systementwicklungsprozess gemäß einem bestimmten Vorgehensmodell und mit Instrumenten arbeiten, die aufeinander abgestimmt sind und damit die Koordination von Ablauf und Entwicklungsergebnissen fördern.

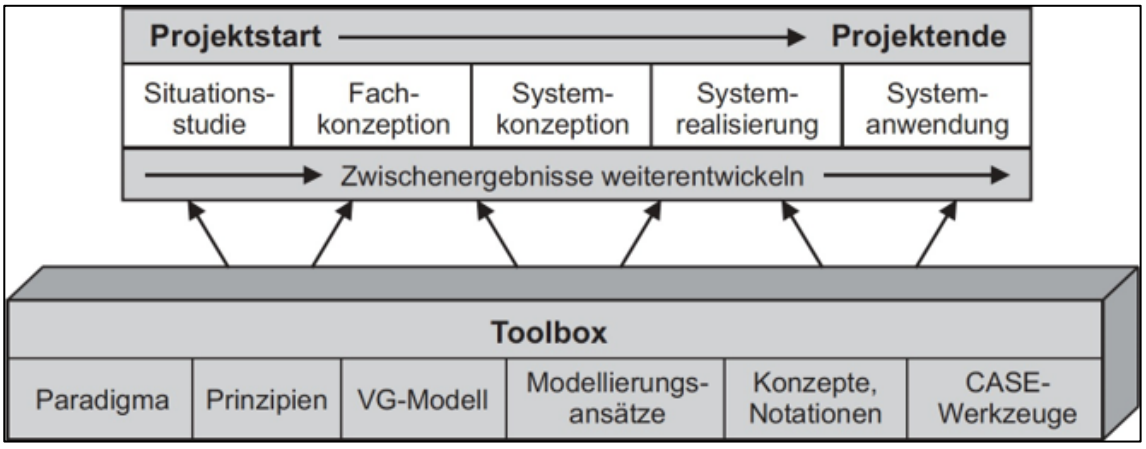


Abb. 140: Prinzip der methodischen Standardisierung

10.1.6 Prinzip der Trennung der Essenz von der Inkarnation

- „Essenz“ zuerst: Gut (fachlich) geplant, ist halb gewonnen.
- „Inkarnation“ danach: Technik und Programmierung strikt nach Plan
- Design first, code later

Abb. 141: Prinzip der Trennung der Essenz von der Inkarnation

Bei dem Prinzip der Trennung der Essenz von der Inkarnation geht es um die bewusste Trennung von Planung und Umsetzung.

Die Essenz wird dabei durch die fachlichen Anforderungen eines IT-Systems dargestellt. Die Problemstellung, die ein IT-System zu lösen hat, wird hier erst ohne technische Implikationen festgehalten.

Erst danach wird die fachliche Sicht in den Systementwurf überführt. Die Inkarnation ist dabei die konstruktive Wiedergeburt der fachlichen Sicht auf technischer Ebene.

Dementsprechend lassen sich auch die traditionellen Phasen der Vorgehensmodelle zuordnen: Die Essenz wird in der Situationsstudie und der Fachkonzeption gewonnen. Die Inkarnation erfolgt dann in den Phasen der Systemkonzeption, Systemrealisierung und Systemanwendung.

10.1.7 Objekte der Modellierung von betrieblichen IT-Systemen

Die Prinzipien der Modellierung sind anzuwenden, um die Modellierung eines IT-Systems bestmöglich zu gestalten. Damit ist die Frage nach dem „Wie“ der Modellierung beantwortet.

Nachdem Sie die Prinzipien der Modellierung kennen gelernt haben, folgt die Fragestellung:

- Auf welche Objekte werden die Prinzipien angewandt?

Objekte der Modellierung von betrieblichen IT-Systemen

- Ein IT-System wird entwickelt, um bestimmte fachliche Aufgaben zu erfüllen.
- Analyse und Entwurf des Aufgabensystems sind die wichtigsten Schritte für die Planung des IT-Systems.
- Das **Objekt** der Modellierung ist demnach das **durch das IT-System betroffene Aufgabengefüge im Unternehmen**.

Dimensionen des zu modellierenden Aufgabengefüges

- Das Aufgabengefüge im Unternehmen lässt sich gemäß den Merkmalen von Aufgaben in folgende Dimensionen unterteilen:
- **Struktur** von Aufgaben: Statische und dynamische Strukturaspekte
- **Ressourcen** von Aufgaben: Sachmittel und (vor allem) Daten
- **Einbindung** in die Organisationsstruktur: Einordnung in die Aufbau- und Ablauforganisation des Unternehmens

10.1.8 Beispiel: Aufgabenmodellierung und statische Funktionen

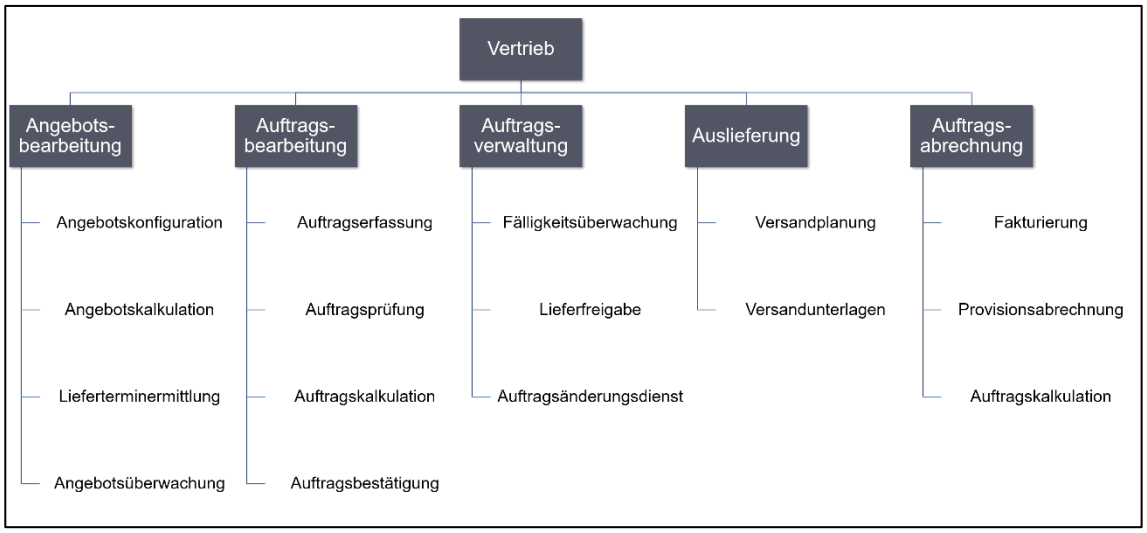


Abb. 142: Aufgabenmodellierung und statische Funktionen

Ein IT-System wird entwickelt, um bestimmte fachliche Aufgaben zu erfüllen. Eine Methode die fachlichen Aufgaben systematisch zu erfassen, ist ein statisches Funktionsdiagramm. Im obigen Beispiel werden die fünf Hauptfunktionen eines Vertriebs aufgeführt und jeweils in Teilaufgaben gegliedert. Das Prinzip der hierarchischen Dekomposition wurde angewendet.

10.1.9 Beispiel: Aufgabenmodellierung und dynamische Funktionen

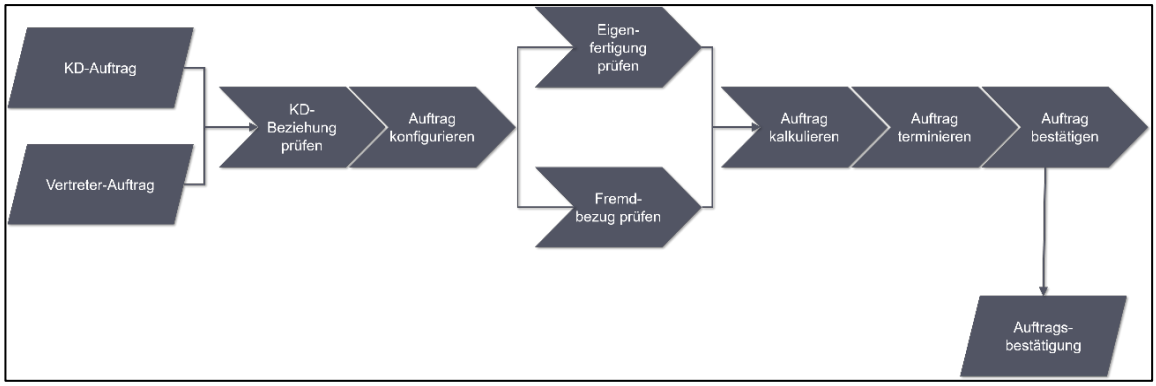


Abb. 143: Aufgabenmodellierung und dynamische Funktionen

Das Aufgabengefüge im Unternehmen lässt sich gemäß den Merkmalen von Aufgaben nicht nur in statischen, sondern auch in dynamischen Strukturen zeigen. Im oberen Beispiel sind die Aufgaben der Auftragsbearbeitung im Vertrieb anhand eines dynamischen Prozesses dargestellt. Diese Dynamik-Darstellung ergänzt die Statik des Vertriebs. Es werden Abläufe sichtbar gemacht.

10.1.10 Aufgabenmodellierung: Dynamische Prozessketten

Ein Beispiel für eine komplexere Prozesskette bietet das nebenstehende Beispiel. Es zeigt eine dynamische Prozesskette mit **Selektion**, **Sequenz** und **Iteration** bestimmter Aufgaben.

Diese drei Strukturblockarten sind in fast jeder Prozessdarstellungs-Methodik vorzufinden.

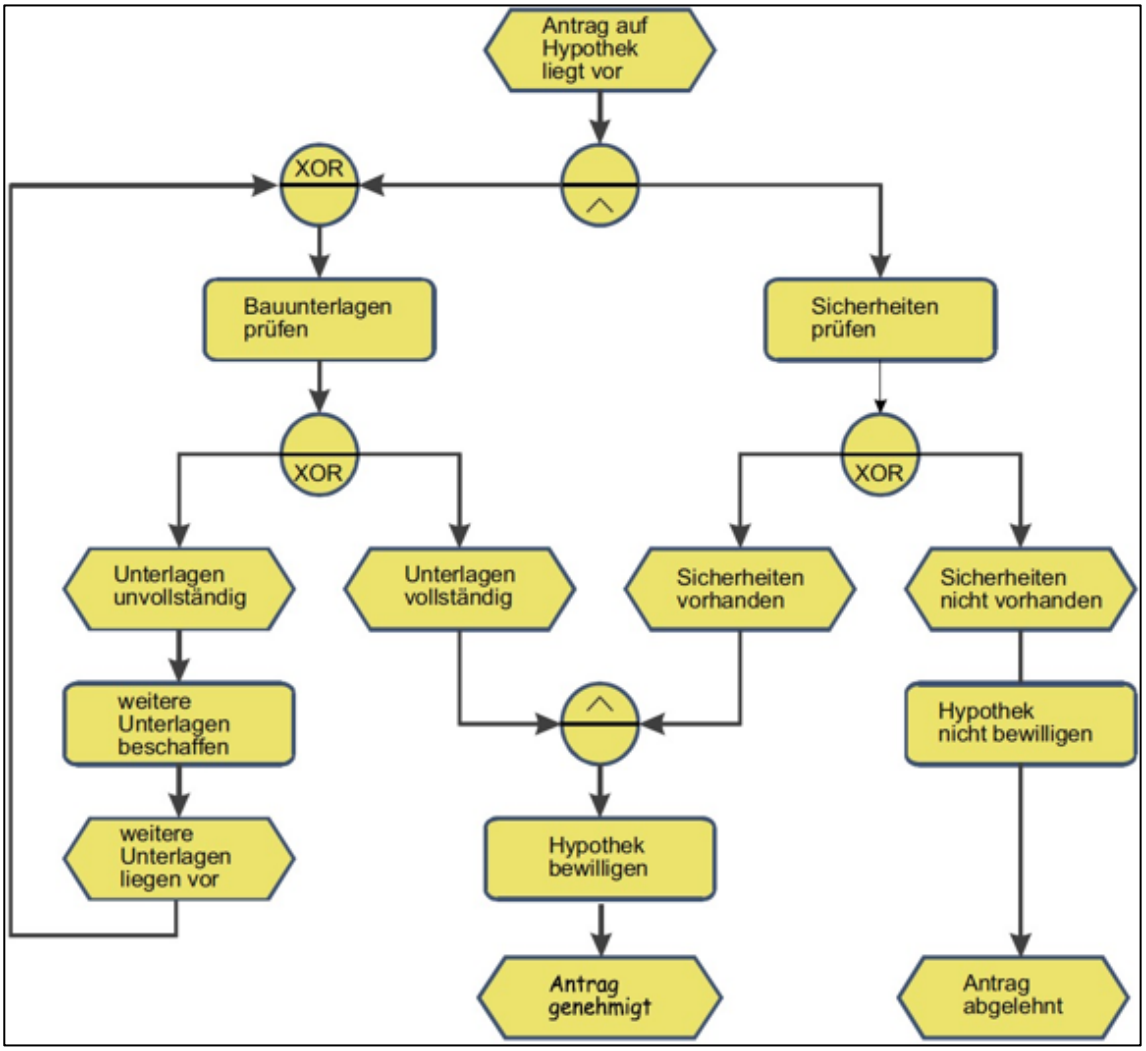


Abb. 144: Aufgabenmodellierung mit Prozesskette

10.1.11 Beispiel: Aufgabenmodellierung und Daten

Ein IT-System wird entwickelt, um bestimmte fachliche Aufgaben zu erfüllen. Das Objekt der Modellierung ist demnach das betroffene Aufgabengefüge. Eine wesentliche Dimension für das fachliche Aufgabengefüge ist die Ressource. Es gilt die Ressourcen zu

modellieren, die einer fachlichen Aufgabe zugeteilt sind. Was beschreibt die benötigte Ressource und welche Eigenschaften hat die Ressource?

Im nebenstehenden Beispiel werden die Daten (die Personen) erkennbar, die für die Aufgabe „Auftragsabwicklung“ von Bedeutung sind. Man benötigt nicht nur die Daten des Kunden, sondern auch die des Auftrages, der Artikel und des Herstellers von Produkten, die nicht im eigenen Haus produziert werden.

Datenstrukturen lassen sich sehr gut mit ER-Modellen (ERM – Entity Relationship Modelling) darstellen.

Im unteren Beispiel sehen Sie ein ER-Modell mit Objekten (Entitäten), Attributen und den Beziehungen zueinander. Fett markierte Attribute sind Schlüsselattribute zur eindeutigen Identifizierung einer Entität.

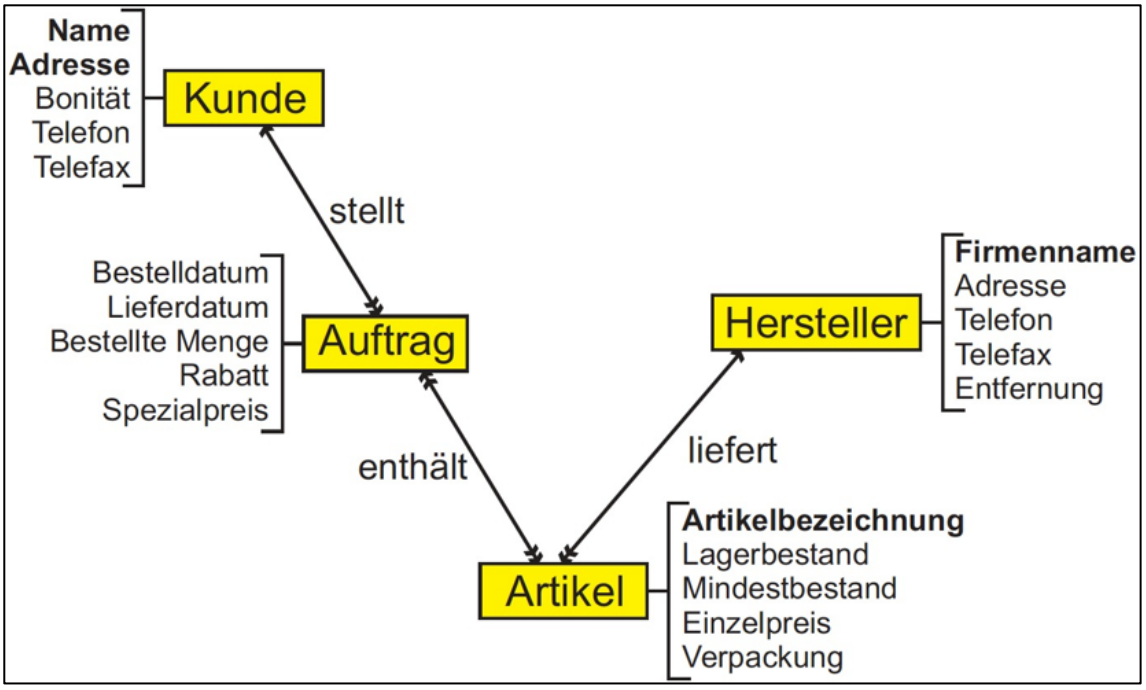


Abb. 145: Beispiel der Aufgabenmodellierung von Daten

10.1.12 Beispiel: Aufgabenmodellierung und Aufbauorganisation

Die zu erledigenden Aufgaben in einem Unternehmen sind mit ihrer Struktur und den Ressourcen, in die Unternehmensorganisation eingebunden.

Dies lässt sich sehr gut mit Organigrammen der statischen Aufbauorganisation eines Unternehmens darstellen.

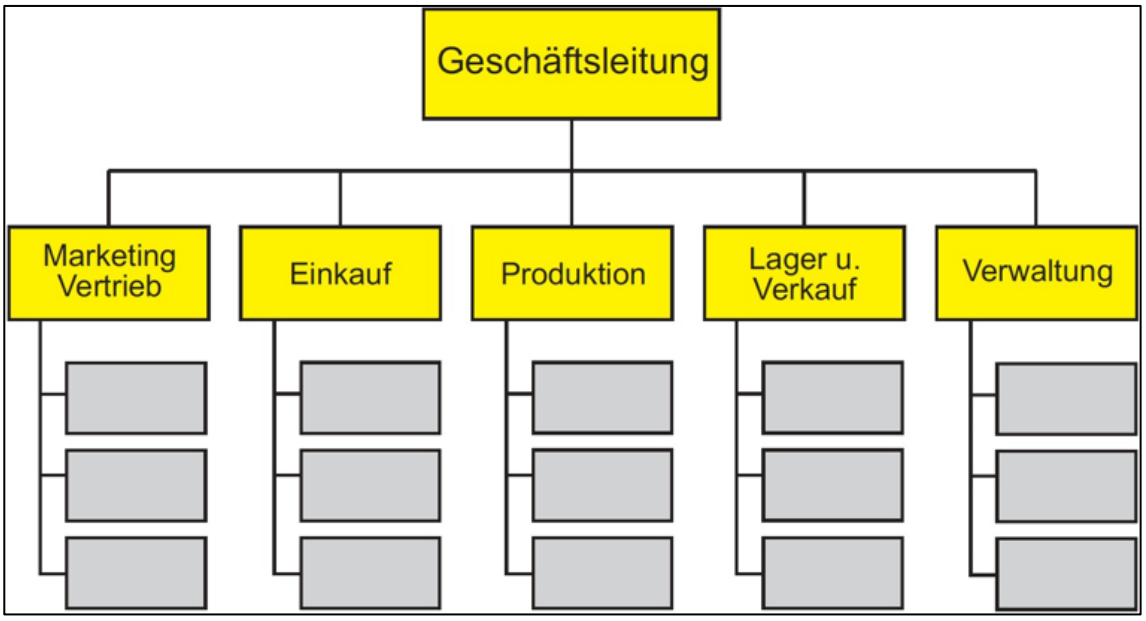


Abb. 146: Aufgabenmodellierung anhand der Aufbauorganisation

10.1.13 Beispiel: Aufgabenmodellierung und Ablauforganisation

Die Einbindung von Aufgaben in die Organisationsstruktur eines Unternehmens geschieht nicht nur über die statische Aufbauorganisation, sondern auch durch die dynamische Ablauforganisation. Die obige Matrix zeigt beispielhaft sowohl einen Ablauf von Aufgaben als auch die Zuordnung von Verantwortlichkeiten zu den Aufgaben.

	Abteilung Vertrieb	Abteilung Einkauf	Abteilung Produktion	Abteilung Lager/Vers.	Abteilung Verwaltung
Posteingang/-verteil.					X
Auftrag erfassen	X				
Auftrag prüfen	X				X
Auftrag konfigurieren	X		X	X	
Auftrag kalkulieren	X		X		
Auftrag terminieren	X	X	X	X	
Auftrag bestätigen	X				
.					
.					
.					

Abb. 147: Aufgabenmodellierung anhand der Ablauforganisation

10.1.14 Modellierungsansätze von betrieblichen IT-Systemen

Die vorangegangenen Beispiele verdeutlichen die **Dimensionen** des Aufgabengefüges von Unternehmen und die zugehörigen Modellierungsobjekte:

- Funktionen
- Prozesse
- Daten
- Organisationsstrukturen

Es existieren verschiedene „klassische“ Ansätze (Methoden mit Techniken) zur Modellierung von betrieblichen IT-Systemen, die jeweils auf bestimmte Modellierungsobjekte fokussieren:

- Funktionsorientierte Modellierung (z. B. mit HIPO)
- Datenorientierte Modellierung (z. B. mit ERM)
- Datenflussorientierte Modellierung (z. B. mit SA, SADT)

Weiterhin existieren neuere Modellierungsansätze, die die verschiedenen **Modellierungsobjekte** integrieren**,** wie z. B.:

- Sichtweisen-integrierende Modellierung (z. B. mit ARIS)
- Objektorientierte Modellierung (z. B. mit UML)

10.2 Funktionsorientierte Modellierung

10.2.1 Funktionsorientierte Modellierung

Die funktionsorientierte Modellierung bezieht sich auf funktionale Abläufe als Modellierungsobjekte.

Es werden betriebswirtschaftliche Funktionen in den Mittelpunkt gestellt. Eine betriebswirtschaftliche Hauptfunktion in einem Unternehmen ist bspw. der Vertrieb mit den dazugehörigen Aufgabengebieten.

Die funktionsorientierte Modellierung praktiziert insbesondere das Prinzip der Abstraktion und schrittweisen Verfeinerung. Zunächst gliedert man das betrachtete Gesamt-System in Teilfunktionen. In weiteren Schritten untersucht man für jede der Teilfunktionen

die innere Struktur. Für die funktionsorientierte Modellierung gibt es verschiedene Methoden und Techniken. Einige davon werden im nachfolgenden Kapitel aufgezeigt. Im unteren Beispiel wird der Datenfluss in einem Betrieb modelliert.

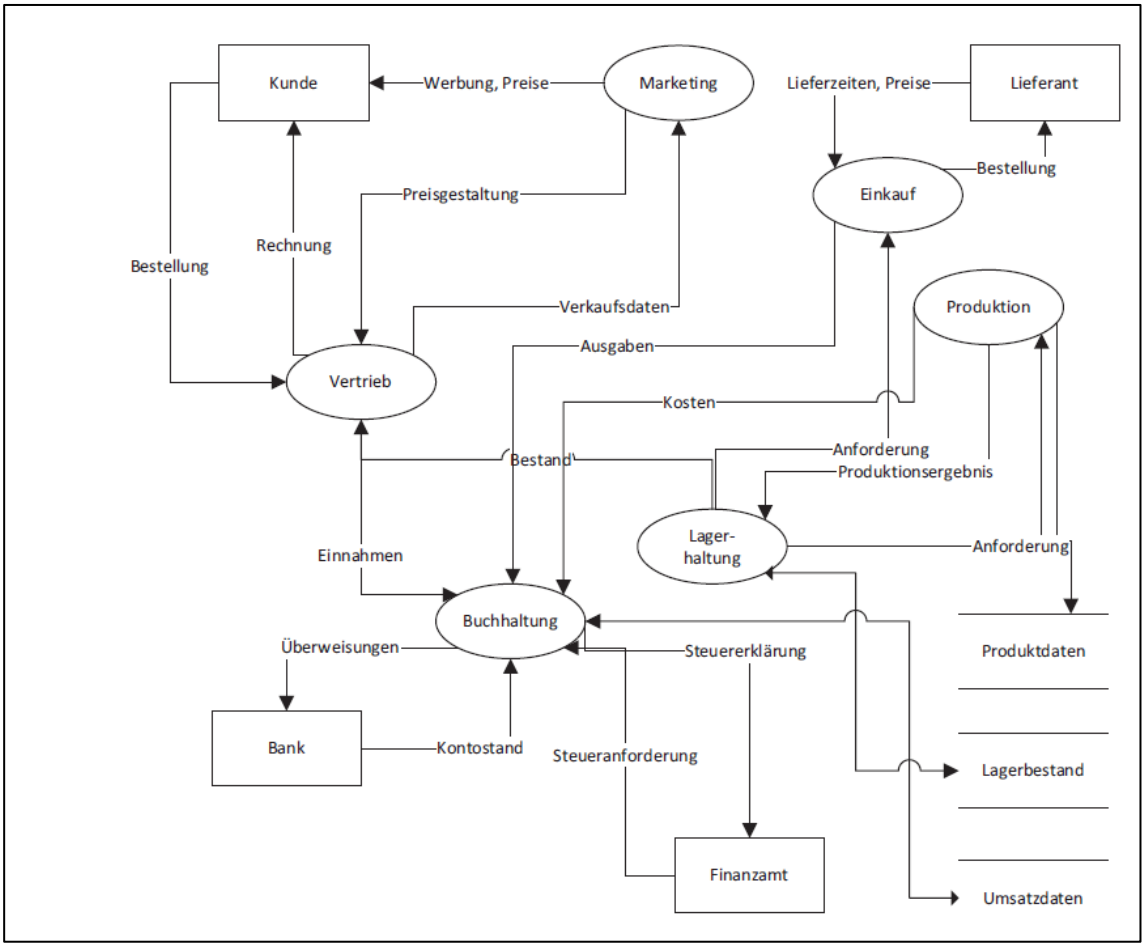


Abb. 148: Datenflussdiagramm eines Betriebs

10.2.2 Funktionsorientierte Aufbauorganisation des Unternehmens

- Die traditionellen Funktionalbereiche (Aufgabenbereiche) definieren die statischen Aufbauorganisationseinheiten des Unternehmens.
- Traditionell mit verrichtungsorientierter dynamischer Ablauforganisation

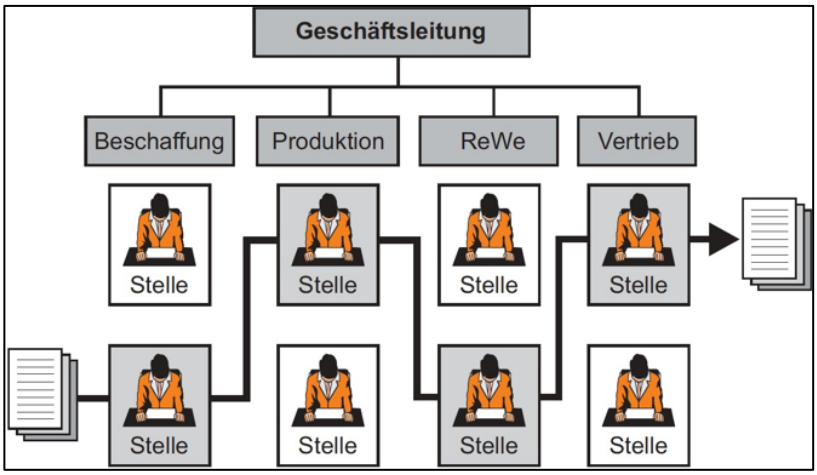


Abb. 149: Aufbauorganisation und Ablauforganisation

10.2.3 Beispiel: Auftragsbearbeitung und statische Funktionen

In diesem Beispiel sind die statischen Funktionen der Abteilung Vertrieb deutlich erkennbar. Die Aufgaben bzw. Funktionen des Vertriebs wurden hierarchisch heruntergebrochen. Das Prinzip der hierarchischen Dekomposition wurde angewendet. Die Dekomposition könnte auf der dritten Ebene weitergeführt werden, z. B. bis auf einzelne Handgriffe, die zur Aufgabenerfüllung notwendig sind.

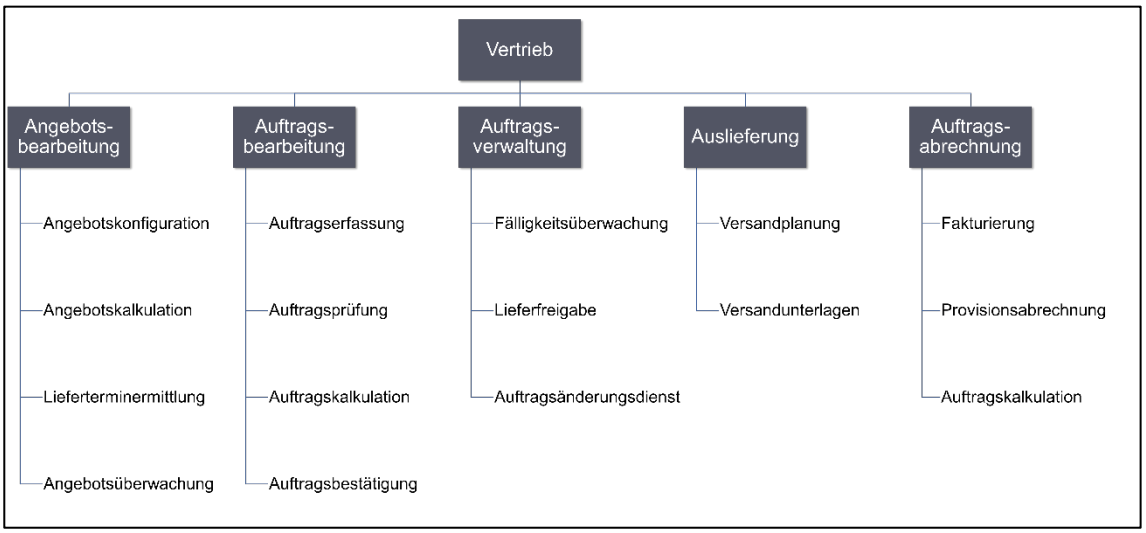


Abb. 150: Statische Funktionen des Vertriebs

10.2.4 Beispiel: Auftragsbearbeitung und Dekompositionsebenen

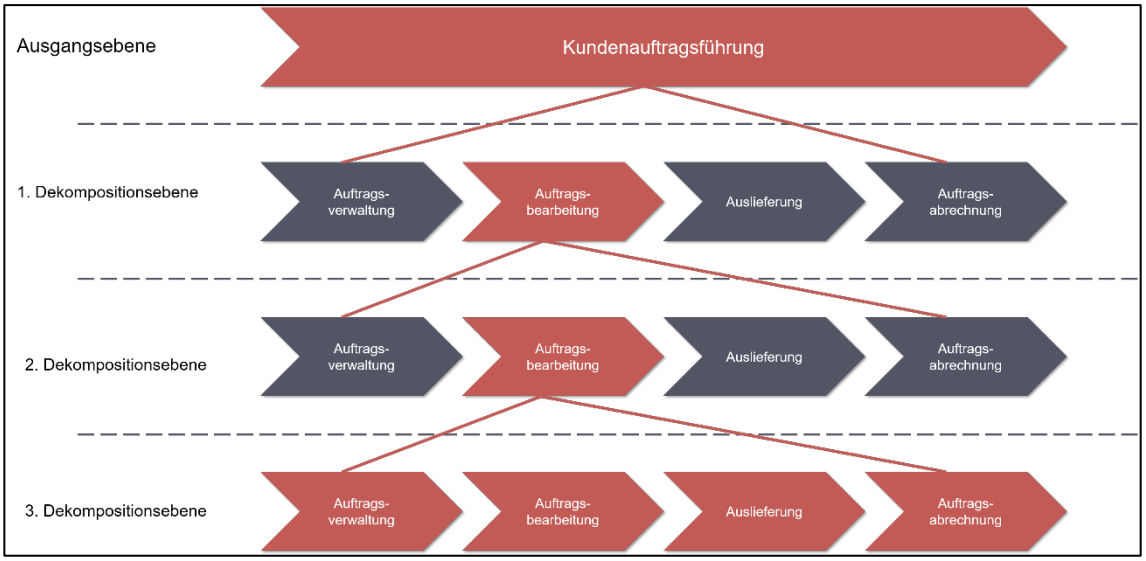


Abb. 151: Kundenauftragsführung und hierarchische Dekomposition

10.2.5 Funktionsorientierte Ablauforganisation des Unternehmens

- Verrichtungsorientierte dynamische Ablauforganisation
- Beispiel unten: Ersatzteilbeschaffung
- Anfrage Kunde an Vertriebsleiter – Sachbearbeiter A erstellt Angebot – Vertriebsleiter kontrolliert – Sachbearbeiter A verschickt – Kunde bestellt bei Sachbearbeiter C – Sachbearbeiter D erfasst Bestellung – Vertriebsleiter kontrolliert – Auftragsbestätigung an Kunde durch Sachbearbeiter A ...

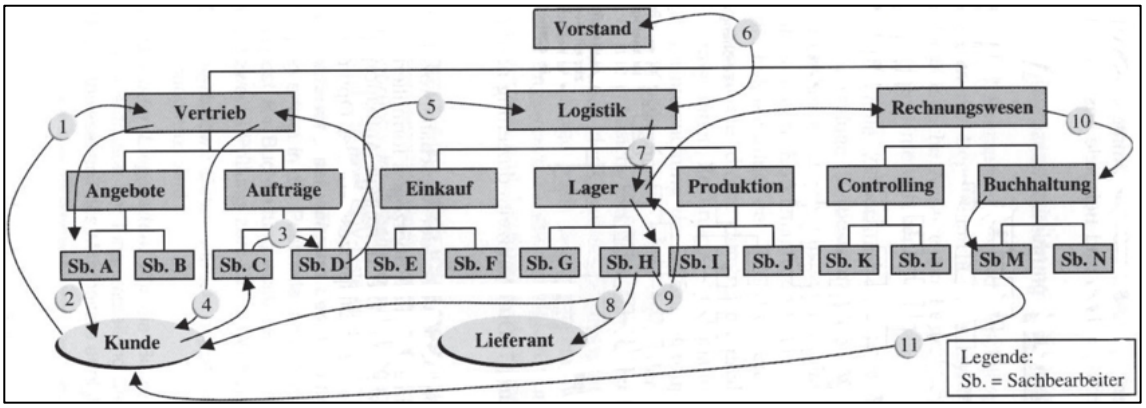


Abb. 152: Beispielhafter Ablauf einer Kundenanfrage in der Aufbauorganisation

10.2.6 Funktionsorientierte betriebliche IT-Systeme

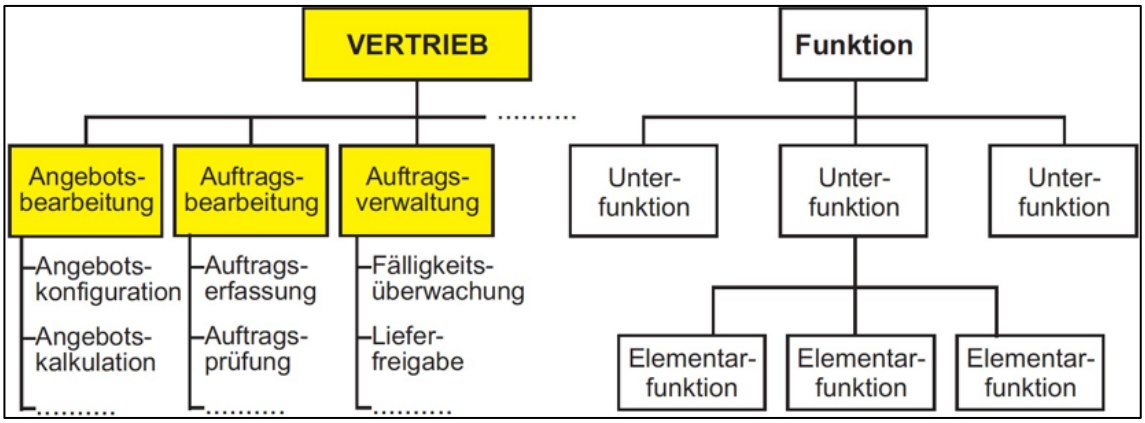


Abb. 153: Vertriebliche Funktionsebenen

- Häufig „Insel-Systeme“ mit vielen internen Schnittstellen
- Kein durchgehender Informationsfluss, evtl. Medienbrüche
- Geringe Flexibilität, hoher Koordinationsbedarf
- Geringe Kundennähe, viele Redundanzen

10.2.7 Funktionsstruktur, HIPO, Struktogramme und Präzedenzmatrix

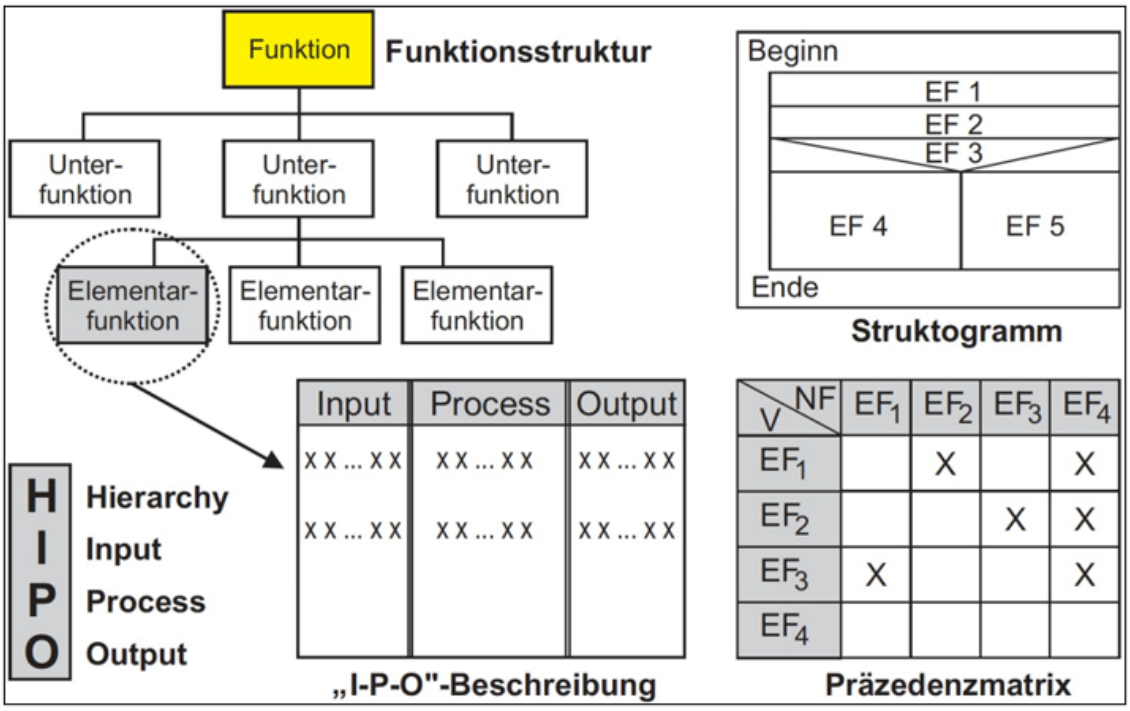


Abb. 154: Funktionsorientierte Modelltechniken

Die Funktionsbeispiele aus der statischen und dynamischen Organisation eines Unternehmens sind komplex. Mit den oben genannten Methoden wird versucht, das Funktionsgefüge in einem Unternehmen zu erschließen.

Auf den folgenden Seiten wird auf die obigen Methoden zur Beherrschung dieser Komplexität eingegangen.

10.2.8 Funktionsstruktur und HIPO

Die typisch hierarchische Funktionsstruktur eines Unternehmens ist meistens der Ausgangspunkt für die weitere Analyse der Teilfunktionen.

Sie betrachten also zunächst die „Blackbox“ und beschreiben, welche Informationen eine Funktion empfängt und welche Informationen die Funktion an andere Teilfunktionen weitergibt.

Dadurch ist die „Blackbox“ noch nicht geöffnet. Beispielsweise ist das System Unternehmen in die Teilfunktionen „Abteilungen“ aufgeteilt. Diese Aufteilung hört jedoch nicht bei den Abteilungen auf, sondern setzt sich bis zu den sogenannten Elementarfunktionen fort.

Erst in einem zweiten Schritt untersucht man die innere Struktur der „Blackbox“ bis hin zu den Elementarfunktionen. Dies geschieht beispielsweise durch ein HIPO-Diagramm. Das HIPO beschreibt, welche Information jede Funktion erhält, wie diese verarbeitet wird („Process“) und welches Ergebnis die Funktion produziert („Output“). Somit ist die „Blackbox“ in ihrer inneren Struktur beschrieben.

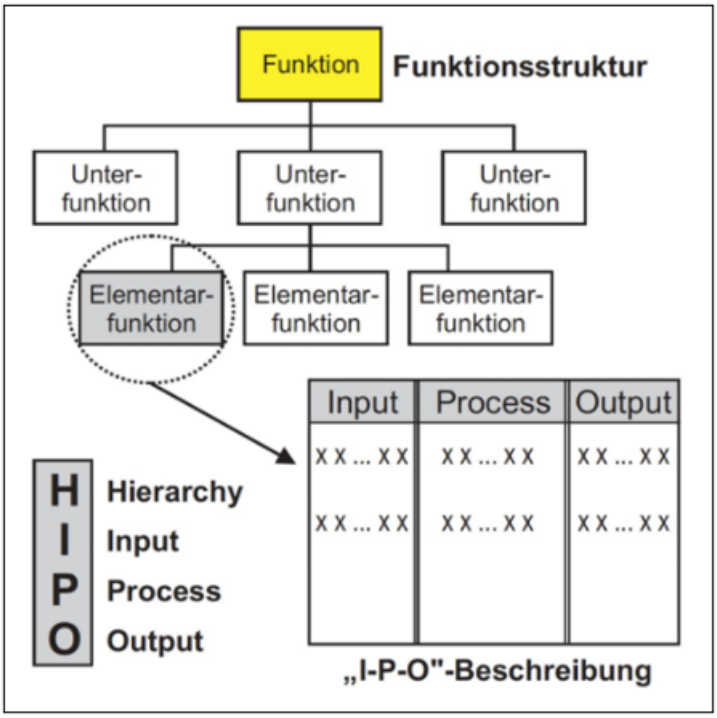


Abb. 155: Zusammenspiel von Funktionsstruktur und HIPO-Diagramm

10.2.9 Beispiel: Auftragsbearbeitung und HIPO-Diagramm

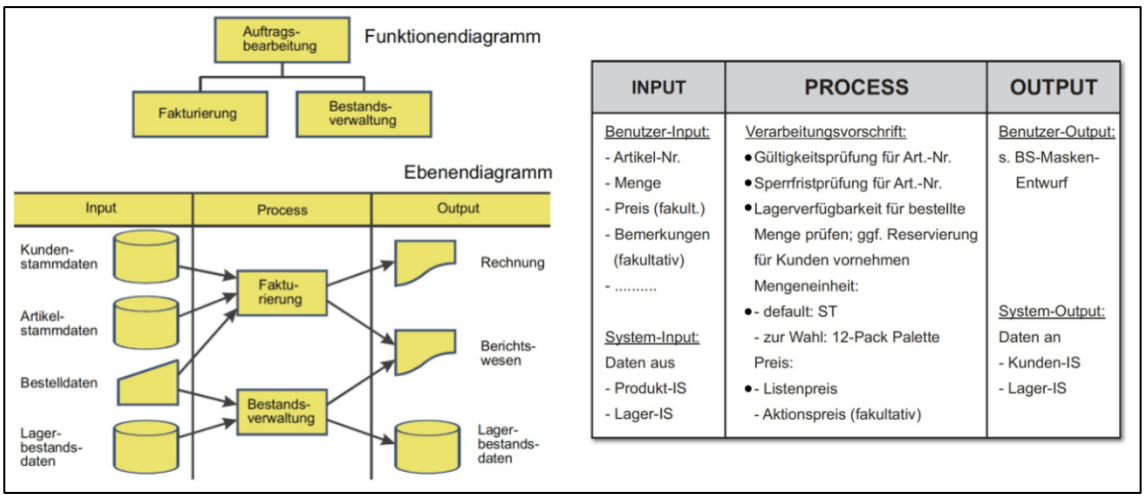


Abb. 156: Beispiel eines HIPO-Diagramms für die Auftragsbearbeitung

10.2.10 Beispiel: Datenmodell im Vertrieb

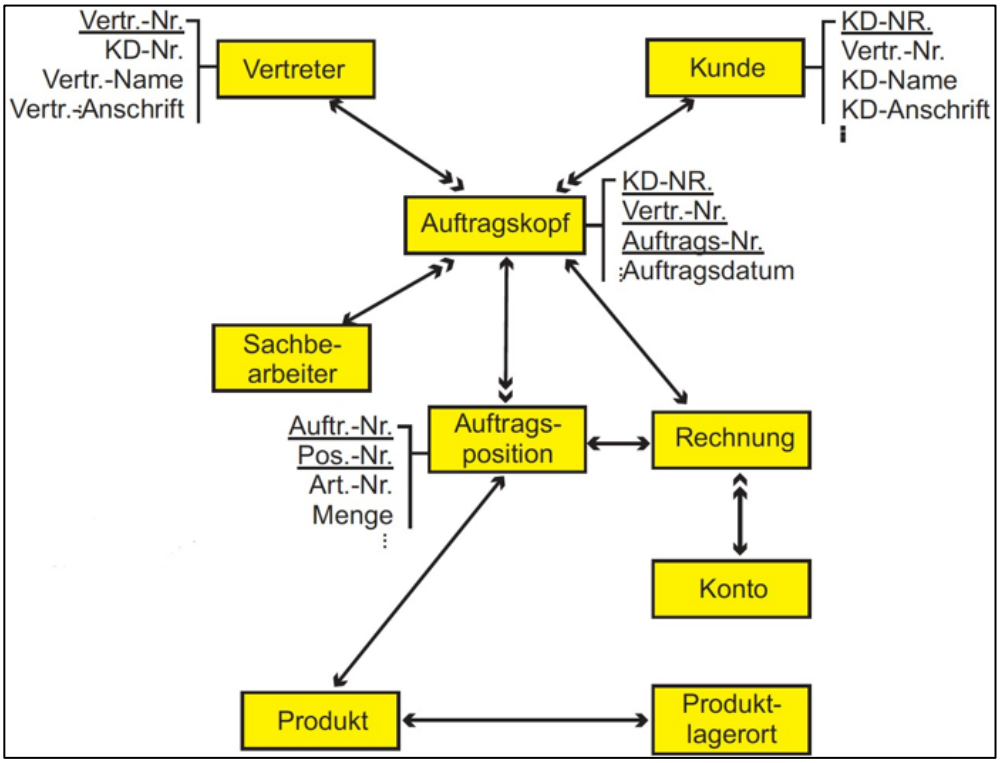


Abb. 157: Beispiel eines Datenmodells im Vertrieb

Das Datenmodell bietet eine weitere Sicht auf ein Unternehmen. Jede Abteilung produziert Daten. Daten gehören zu den Ressourcen jeder Abteilung.

Im Vertrieb können dies zum Beispiel die Adressdaten eines neuen Kunden sein, der eine Produktbestellung aufgegeben hat.

Alle diese Informationen können in ein Datenmodell aufgenommen werden, wie es die Grafik auf der rechten Seite beispielhaft zeigt.

Die Rechtecke stellen dabei Objekte dar, z. B. den Kunden, Sachbearbeiter oder Auftragspositionen. Die jeweils beigefügten zum Teil unterstrichenen Informationen sind Attribute.

Attribute beschreiben ein Objekt und seinen Inhalt. Die unterstrichenen Attribute sind Identifikatoren. Anhand der Kundennummer (KD-NR.) können Sie einen Kunden eindeutig identifizieren.

10.2.11 Beispiel: Auftragspositionen erfassen: Entwurf einer Bildschirm-Maske

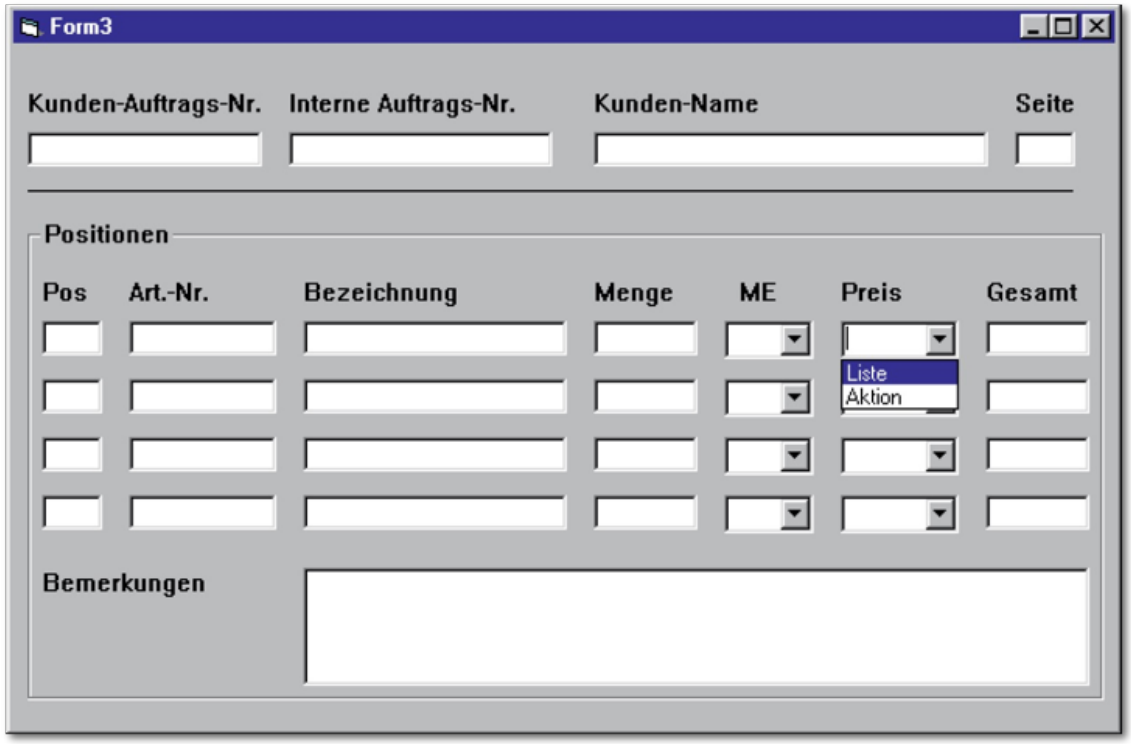


Abb. 158: Entwurf einer Bildschirmmaske zur Erfassung von Aufträgen

10.2.12 Steuerkonstrukte der Programmierung: PAP und Struktogramme

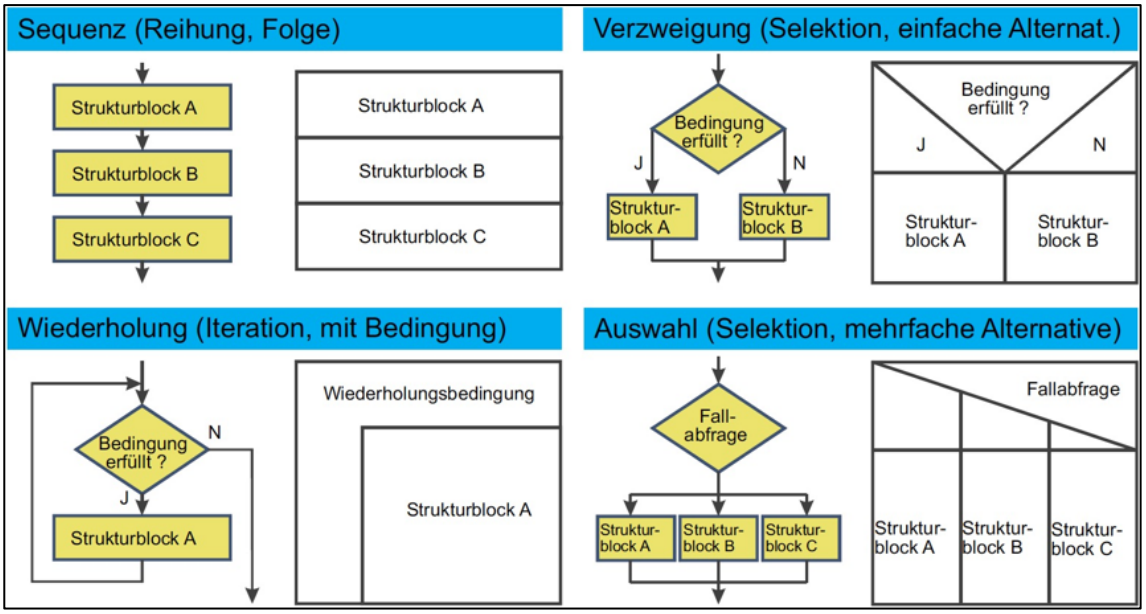


Abb. 159: Strukturblockarten zur Modellierung

10.2.13 Beispiel: Laufwegsteuerung als Präzedenzmatrix

Stelle / Aktion	Poststelle	Schadensregulierung		Buchhaltung
		Sach-bearbeiter	Leiter	
1	Schadens-meldung annehmen			
2		Meldung prüfen		
3		Zahlungs-betrag ermitteln		
4			Zahlungs-betrag genehmigen	
5	Original	Mitteilung erstellen	Kopie	
6	Mitteilung versenden			
7				Zahlungs-betrag anweisen

Abb. 160: Laufwegsteuerung eines Schadenfalls

Die untenstehenden Beispiele der Vorgangsbearbeitung einer Schadensregulierung und der Auftragsführung sind hier in einer Präzedenzmatrix ausgeführt.

Ebenso wie PAP und Struktogramme sind Präzedenzmatrizen Ablaufpläne. Sie zeigen jedoch den Zusatz der Verantwortlichkeiten.

Das Beispiel der Schadensregulierung durchläuft organisatorisch die Poststelle, die Abteilung Schadensregulierung und letztendlich die Buchhaltung. Es wird deutlich, wer wann was zu tun hat, um einen Schadensfall zu bearbeiten.

Ähnlich verhält es sich mit der Laufwegsteuerung der Auftragsführung. Linksseitig wird abgebildet, was zu tun ist, während die Spalten angeben, welche Abteilung miteinbezogen wird. Die Punkte stehen dabei für erbrachte Leistungen bzw. Ergebnisse. Die Knoten in Pfeil-Form stehen für Verarbeitungsschritte in der Abteilung selbst. Die gerichteten Pfeile geben an, in welche Richtung „gelaufen“ wird.

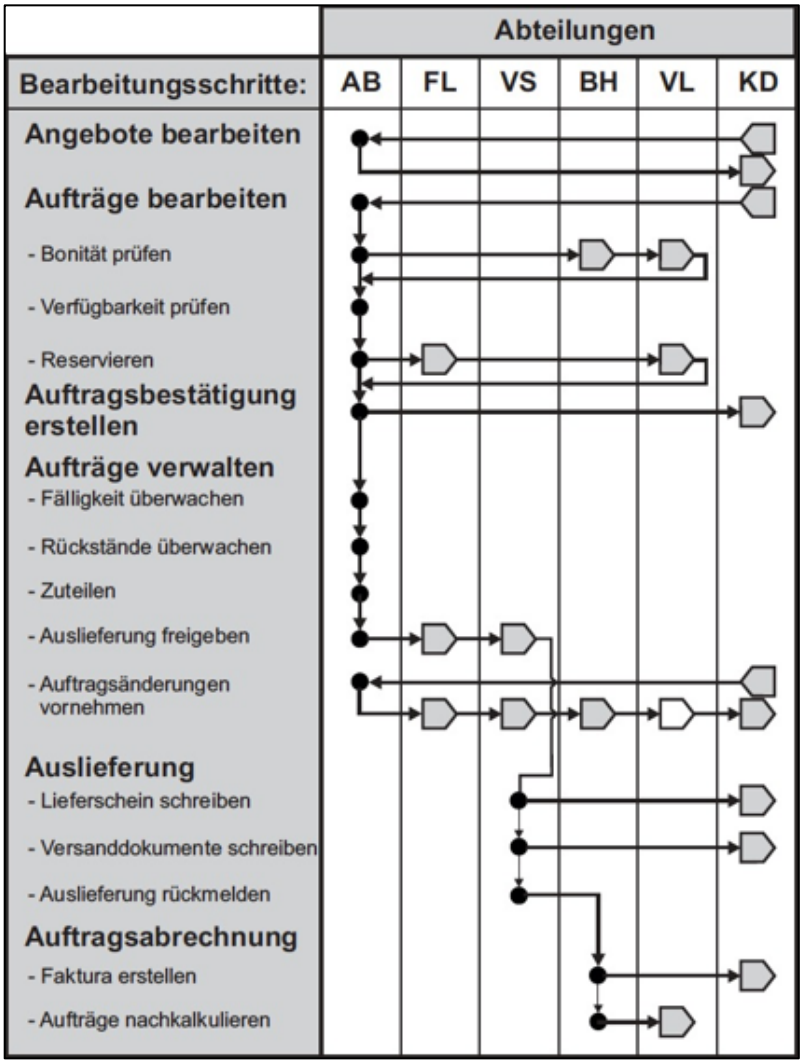


Abb. 161: Laufwegsteuerung der Auftragsbearbeitung

10.2.14 Fazit: Funktionsorientierte Modellierung von betrieblichen IT-Systemen

- Hierarchische Zerlegung in Funktionsbäumen: Teilungskriterien sind die Funktionen des IT-Systems gemäß den zu erfüllenden betrieblichen Aufgaben.
- Statische Aspekte: Aufbaustruktur des IT-Systems mit statischen Funktionsbäumen im Stil von Organigrammen
- Dynamische Aspekte: Ablaufstruktur des IT-Systems mit Funktionsreihenfolgen per Flussdiagrammen, PAP, Struktogrammen, Präzedenzmatrizen, IPO-Tabellen etc.
- Daten sind Ressourcen zur Erfüllung der Funktionen.
- Funktionsspezifische Datendefinitionen und Datenzuordnungen verursachen Redundanzen und Inkonsistenzen.
- In einem dynamischen Markt- / Unternehmensumfeld sind Definitionen von Datenstrukturen zeitstabiler als Funktionsstrukturen.
- Daher wird in Literatur und Praxis ein stärker datenorientierter Modellierungsansatz als zweckmäßiger angesehen.

10.3 Abschlusstest – WBT10

10.3.1 Abschlusstest

Bitte beantworten Sie die folgenden Fragen durch das Ankreuzen der korrekten Antworten (Tab. 11). Bei einigen Fragen können auch mehrere Antworten richtig sein.

Nr.	Frage	Richtig	Falsch
1	Funktionsspezifische Datendefinitionen und Datenzuordnungen verursachen Redundanzen und Inkonsistenzen.		
2	Zu den dynamischen Aspekten der Aufgabenmodellierung gehören zum Beispiel PAP und Struktogramme.		
3	Zu den Dimensionen der Aufgabenmodellierung gehören...		
	Struktur von Aufgaben		
	Ressourcen von Aufgaben		
	Einbindung von Organisationsstrukturen		
4	Funktionsorientierte betriebliche IT-Systeme haben häufig viele interne Schnittstellen.		
5	Zu den Prinzipien der Modellierung gehören:		

	Die hierarchische Dekomposition, Strukturierung und Modularisierung, konstruktive Voraussicht, perspektivische Betrachtung, methodische Standardisierung und Trennung der Essenz von der Inkarnation.		
	Die hierarchische Dekomposition, Strukturierung und Modularisierung, konstruktive Voraussicht, perspektivische Betrachtung, methodische Standardisierung und Trennung der Essenz von der Reinkarnation.		
	Die hierarchische Komposition, Strukturierung und Modularisierung, konstruktive Voraussicht, perspektivische Betrachtung, methodische Standardisierung und Trennung der Essenz von der Inkarnation.		
6	Die traditionellen Funktionalbereiche (Aufgabenbereiche) definieren die statischen Aufbauorganisationseinheiten des Unternehmens.		

Tab. 11: Abschlusstest – WBT10

10.4 Typische Fragestellungen

Typische Aufgabenstellungen – Aufgaben- und funktionsorientierte Modellierung

Zur Bearbeitung dieser Aufgabenstellungen beachten Sie bitte: Verlangt ist eine fachlich zutreffende, inhaltlich nachvollziehbare und kausal zusammenhängende Erörterung aus vollständigen Sätzen in lesbarer Handschrift. Für jede Aufgabe: Maximal zwei Seiten Text.

Aufgabe 1:

> Erläutern Sie stichwortartig die „Prinzipien für die Entwicklung von betrieblichen Informations- und Kommunikationssystemen“.

Aufgabe 2:

> Erläutern Sie die „Funktionsorientierte Modellierung von IuK-Systemen“ im Hinblick auf deren Bezugsobjekte, statische und dynamische Aspekte und die Rolle der Daten.

Aufgabe 3:

> Erläutern Sie ein typisches funktionsorientiertes Set von Modellen zur Abbildung von statischen und dynamischen Aufgabenzusammenhängen im Unternehmen.

Aufgabe 4:

Beurteilen Sie das funktionsorientierte Set aus Aufgabe 3 im Hinblick auf die methodische Durchgängigkeit.

Aufgabe 5:

Erläutern Sie stichwortartig die „Prinzipien für die Entwicklung von betrieblichen Informations- und Kommunikationssystemen“.

Aufgabe 6:

Stellen Sie die Grundzüge der tradierten funktionsorientierten Unternehmensgestaltung einer modernen prozessorientierten Unternehmensgestaltung vergleichend gegenüber. Nehmen Sie dabei vorrangig Bezug auf die Aufbau- und Ablauforganisation eines Unternehmens.

11 Daten- und Prozessmodellierung

11.1 Datenmodellierung mit ERM

11.1.1 Wiederholung: Dimensionen der Modellierung

Sie haben bereits die Prinzipien und Objekte der Modellierung sowie die funktionsorientierte Modellierung kennen gelernt. Dabei konnten Sie feststellen, dass die Modellierung des Aufgabengefüges in einem Unternehmen sich als problematisch gestaltet. Denn die Aufgaben in einem Unternehmen ändern sich stetig.

In der vorliegenden Lerneinheit befassen Sie sich mit der Modellierung von Daten.

Daten spielen in jedem Unternehmen eine zentrale Rolle, denn für jeden Prozess werden bestimmte Daten benötigt. Beispielsweise benötigt man zur Abwicklung eines Kundenauftrags unter anderem Informationen über den Kunden, über die bestellten Produkte und die Anschrift des Kunden. Diese Daten spiegeln aber bei Weitem nicht alle benötigten Daten wider und es ist auch keine Beziehung zwischen den Daten ersichtlich. Dafür brauchen Sie die Datenmodellierung.

Ziel der Datenmodellierung ist also die Identifikation und Spezifikation aller Daten und deren Beziehungen untereinander in einem oder mehreren Prozessen in Unternehmen. Dieses Ziel erreichen Sie mit dem Entity Relationship Modeling (ERM).

11.1.2 Zeitstabilität von Modellen

In WBT 10 konnten Sie feststellen, dass Modelle auf Basis von Daten zeitstabiler erscheinen als funktionsorientierte Modelle. Funktionsorientierte Modelle ändern sich häufig, da sie auf variablen Organisationsstrukturen und volatilen Prozessen basieren. Ändert sich z. B. die Organisation eines Unternehmens, so muss sich das funktionsorientierte Modell anpassen.

Adressdaten eines Kunden können sich ebenso ändern, jedoch bleibt die Struktur des Adress-Datensatzes unverändert. Wie Datenstrukturen beschrieben werden, wollen wir anhand der Entity-Relationship-Modellierung zeigen.

Die Frage bei der Modellierung von Daten ist:

- Welche Daten sollen wo
- und wie erfasst und modelliert werden?

Um den Wert der Daten als Informations- und Wissensquelle noch etwas genauer herauszuarbeiten, wird zunächst mit dem Thema des Datenmanagements eingeleitet.

Darüber hinaus gelten auch hier wieder Modellierungsprinzipien. Die Abstraktion dient bspw. dazu, allgemeine Typen von Daten zu finden und deren relevante Attribute herauszuarbeiten.

11.1.3 Dimensionen des Datenmanagements

IT-Abteilung und **Datenmanagement**

- Aus Daten müssen Informationen werden.
- Informationen sind als wirtschaftliches Gut zu interpretieren.
- Aufgabe der IT-Abteilung: Nicht „Datenverarbeitung“, sondern Informationsversorgung

Aufgaben und **Ziele** des Datenmanagements

- Alle im Unternehmen verwendeten Daten planen, überwachen und steuern
- Dies unabhängig von den technischen Datenspeichern
- Ziele: Richtigkeit, Vollständigkeit, Aktualität, Konsistenz, Angemessenheit der Daten / Problem: „Unternehmensweites Datenmodell“ (UDM)

Konkrete **Aktivitätsbereiche** des Datenmanagements

- Entwicklung und Implementierung von Datenarchitekturen, Datenmodellen
- Organisation der Datenbeschaffung und Datennutzung („Big Data“?)
- Wartung und Pflege der Datenbestände

Daten sind gerade in der heutigen Zeit ein strategischer Wettbewerbsvorteil. Denn jede Information und das Wissen, das daraus generiert werden kann, helfen Unternehmen sich langfristig am Markt zu positionieren. Aus Daten werden Informationen und letztendlich Wissen.

11.1.4 Daten-Architektur: Bereich, Objekte und Zweck

Daten-Architektur: **Bereich und Objekte**

- Betrachtungsbereich: gesamtes Unternehmen
- Objekte der Daten-Architektur sind die Kern-Entitäten und Kern-Beziehungen des Unternehmens.
- „Informationsmodellierung“ auf hoher Abstraktionsstufe zur Komplexitätsreduktion
- Informationsmodellierung: Aufgabe des Unternehmensmanagements

Zweck der Daten-Architektur: **Umsetzung in Datenmodelle**

- Die „globale“ Daten-Architektur wird in ein semantisches Datenmodell überführt.
- Das semantische Datenmodell wird in ein Datenbankmodell umgesetzt und mit einem Datenbank-System realisiert.

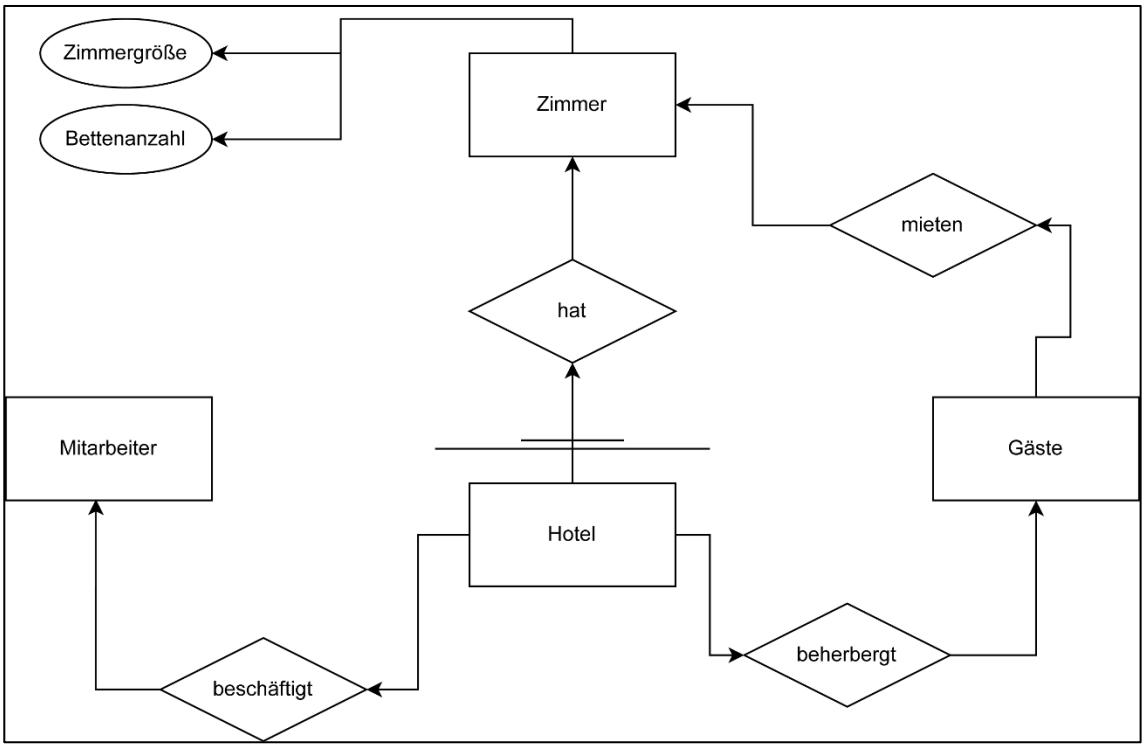


Abb. 162: Beispiel eines ER-Modells im Hotel

11.1.5 Daten-Architektur: Datenmodellierung Teil 1

- Datenmodell **allgemein**: Statische Darstellung eines Ausschnittes der Realität; Beschreibung von Gegenständen, Sachverhalten, Beziehungen
- Datenmodell **semantisch**: Strukturierte Darstellung der Semantik von Unternehmensdaten; Daten werden nach ihrer Sinnbedeutung, nicht unter technischen Aspekten dargestellt.
- Datenmodell **technisch**: Konzeption zur Datenstrukturierung und -verwaltung für verschiedene Datenbankmodelle
- **Methode** zur Datenmodellierung: ERM (Entity Relationship Modell / originär: Peter Chen, 1976)
- **ER-Modell**: Unabhängig vom später verwendeten Datenbankmodell und dem realisierenden Datenbank-System
- Aus dem ER-Modell wird die logische und technische **Haltung der Daten** in einem Datenbank-System abgeleitet.

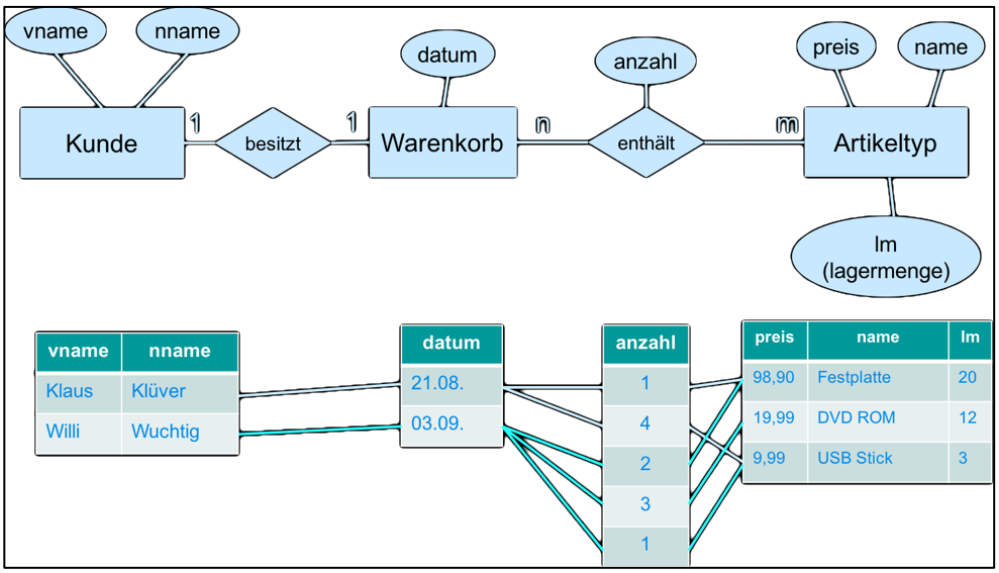


Abb. 163: ER- und Relationenmodell im Vergleich

11.1.6 Daten-Architektur: Datenmodellierung Teil 2

- Datenorientierte Modellierungsansätze für Informations- und Kommunikations-Systeme konzentrieren sich auf die betriebliche Datenstruktur, Datenrepräsentationsformen und die Datenmanipulation.
- Datenstruktur bspw. für ein IKS: Kunden, Artikel, Lager, Vertriebsbeauftragte, Aufträge, Lieferanten etc.
- Datenstruktur bspw. für ein IKS: Merkmale (Attribute) von Artikeln wie z. B. Preis, Bezeichnung, Menge etc. und Beziehungen z. B. zu Auftrag, Lieferant etc.
- Datenstrukturen sind i. d. R. zeitstabiler als Funktionen und eignen sich daher oft besser für eine längerfristig gültige Modellbasis eines IKS.
- ERM ist ein typisches Beispiel für einen datenorientierten Modellierungsansatz.

11.1.7 ERM – Entity Relationship Modeling

- Im Jahr 1976 von Peter Chen vorgestellt
- Semantische Datenmodelle
- In ER-Modellen werden permanent zu speichernde Daten und ihre Beziehungen modelliert.
- Keine Berücksichtigung von Datenflüssen, Organisationen, Funktionen

ERM – **Anwendungsbereiche**

- Allgemeiner Ansatz, um Datenmodelle zu entwerfen
- Unabhängig vom anvisierten Datenbank-System
- Das „Was“ eines Systems steht im Vordergrund, nicht das „Wie“.
- Phasen der System-Entwicklung: Fachkonzeption, Spezifikation

11.1.8 ERM: Anwendungsfelder

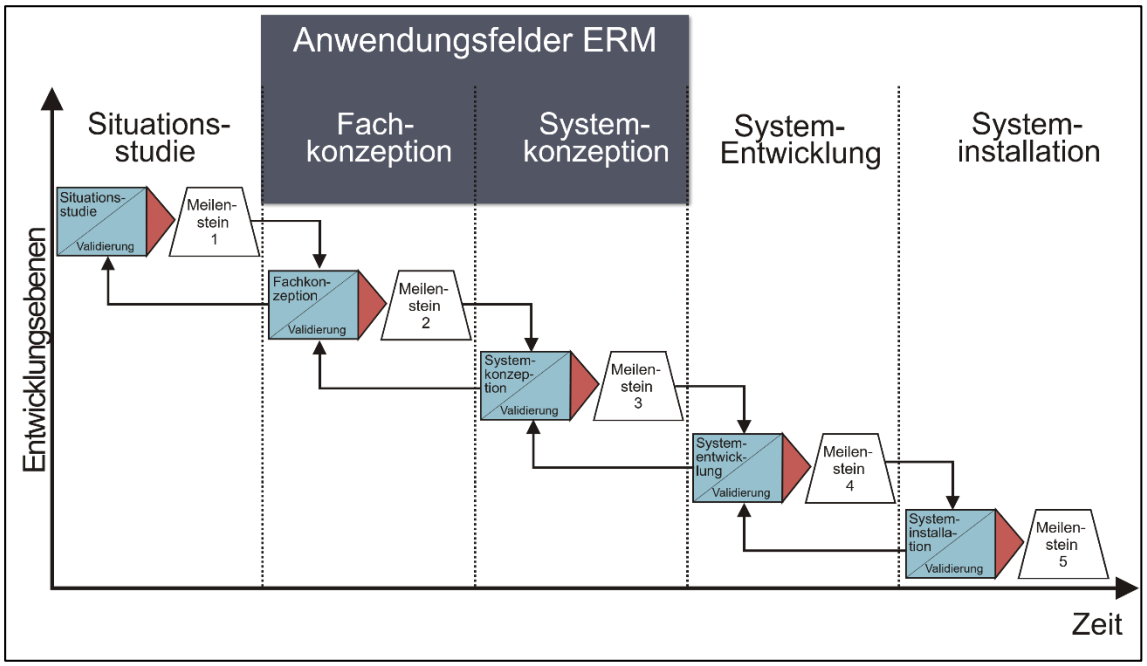


Abb. 164: Anwendungsfelder des ERM

ER-Modelle sind sinnvoll in der Fachkonzeption und in der Systemkonzeption einzusetzen. In einem Projekt wird in der Fachkonzeption ein semantisches ER-Modell herausgearbeitet. Dies bedeutet, das Anwendungsproblem der Fachabteilung wird mit Hilfe der Darstellungselemente visualisiert.

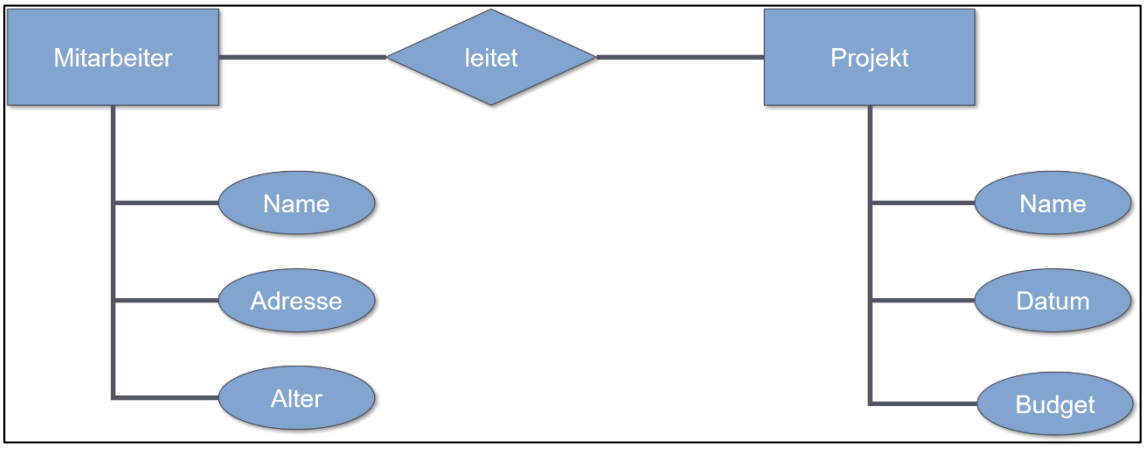


Abb. 165: Semantisches Beispiel des ERM

Die Fachkonzeption stellt dar, welche Daten mit welchen Beziehungen modelliert werden müssen. Darauf basierend erstellt die Systemkonzeption das relationale Datenmodell. Auf dem relationalen Datenmodell kann letztendlich über physische Datenbankkomponenten, wie Servermodelle, entschieden werden.

Personalschlüssel	Vorname	Nachname	Geburtstag	Straße	Hausnummer	PLZ	Ort
1	Antje	Musterfrau	05.06.1990	Gartenweg	3	36648	Musterstadt
2	Herrmann	Mustermann	02.01.1965	Licherweg	96	35548	Musterdorf

Projektschlüssel	Personalschlüssel (Projektleitung)	Startdatum	Enddatum	Budget	Währung
PJ-102	2	01.01.2021	31.12.2021	3.000.000	€
PJ-103	1	05.08.2020	01.11.2022	1.000.500	$

Abb. 166: Beispiel eines Relationenmodells

11.1.9 ERM: Darstellungselemente

Die klassischen Darstellungselemente zur ERM-Modellierung bestehen aus:

- Entitätstypen und
- Beziehungstypen.

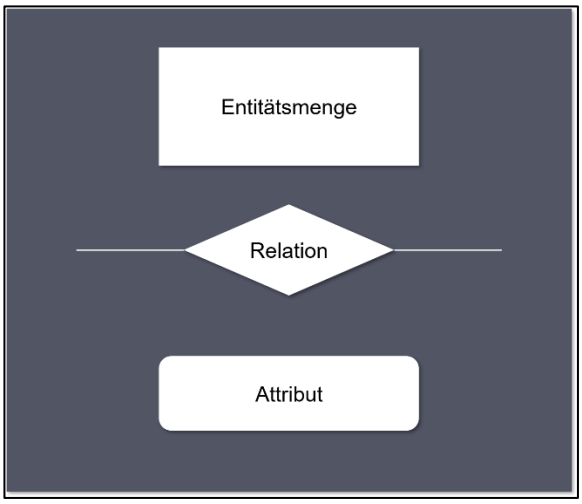


Abb. 167: Klassische Darstellungselemente des ERM

Entitäten sind Objekte der realen Welt. Dies könnte bspw. der Goldfisch Goldi sein. Entitätstypen hingegen sind die Abstraktion des Objektes. Der Entitätstyp für Goldi wäre somit ein Fisch. Entitätstypen bzw. Entitätsmengen werden im ER-Modell als Rechtecke dargestellt.

Durch die Abstraktion der Entitäten werden die Entitätstypen durch ihre wesentlichen Eigenschaften, die Attribute, beschrieben. Attribute werden durch Rechtecke mit abgerundeten Ecken dargestellt.

Kann eine Entität durch dieses Attribut eindeutig identifiziert werden, nennt man dieses Attribut Schlüsselattribut. Schlüsselattribute sind für die Identifikation und Verknüpfung von Objekten von großer Bedeutung. Durch ein Schlüsselattribut, z. B. das besondere Schuppenmuster von Goldi, ist dieser eindeutig identifizierbar und kann seinem Besitzer zugeordnet werden.

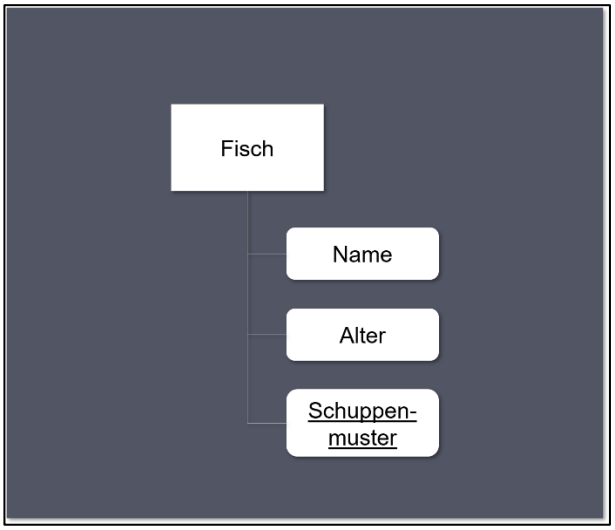


Abb. 168: Goldi als Fisch-Entität

Beziehungstypen bzw. Relationen beschreiben die Verbindungen zwischen Entitätstypen semantisch. Unser Goldfisch Goldi gehört einer Person Max. Relationen werden durch eine Raute dargestellt.

Relationen haben darüber hinaus zwei Charakteristika: Den Grad einer Relation und die Kardinalität. Der Grad visualisiert, wie viele Entitätstypen miteinander in Beziehung stehen. Die Kardinalität beschreibt, wie viele Entitäten über eine Beziehung verknüpft sind.

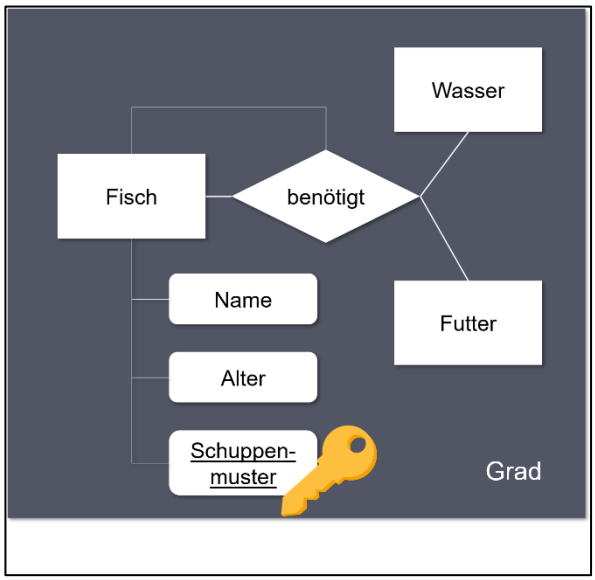


Abb. 169: Grad einer Entität

Ein Fisch benötigt z. B. zum Leben nicht nur Wasser, sondern auch Futter und Pflanzen sowie andere Fische. Dies drückt aus, dass ein Fisch mit anderen Entitätstypen in Verbindung steht.

Die Kardinalität hingegen beschreibt, zu wie vielen dieser Entitäten ein Fisch gehören kann. Ein Fisch kann mehreren Personen gehören, z. B. gehört Goldi sowohl Susi als auch Bernd. Eine oder mehrere Personen können allerdings auch mehrere Fische besitzen.

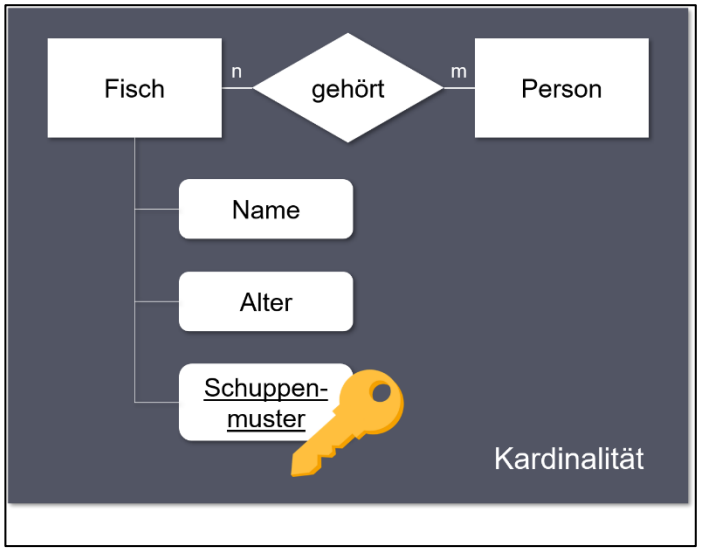


Abb. 170: Kardinalitäten im ERM

11.1.10 ERM: semantisches Beispiel

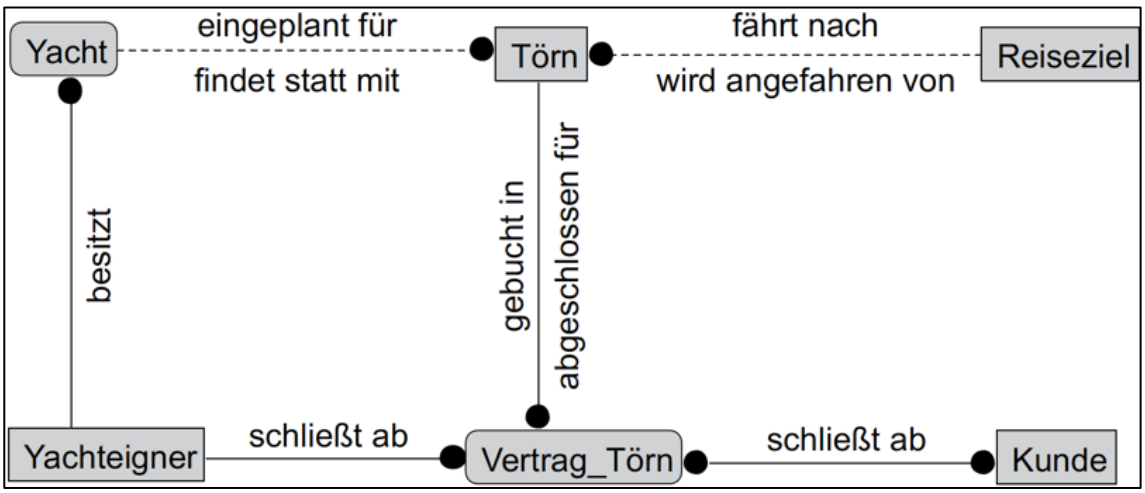


Abb. 171: Der Segeltörn als semantisches Beispiel

Semantische Modelle beziehen sich ausschließlich auf die Bedeutung der Daten. Technische Begebenheiten werden explizit ausgeschlossen und erst im Relationenmodell aufgegriffen. Dies ermöglich eine methodische Durchgängigkeit von der Fachkonzeption zur Systemkonzeption.

In diesem Beispiel wird ein Segeltörn modelliert. Die Entitätstypen sind dabei der Kunde, der Yachteigentümer, die Yacht, der gebuchte Segeltörn und das Reiseziel. Die Kanten stellen die Beziehungen zwischen den Entitätstypen dar.

11.1.11 ERM: Relationenmodell

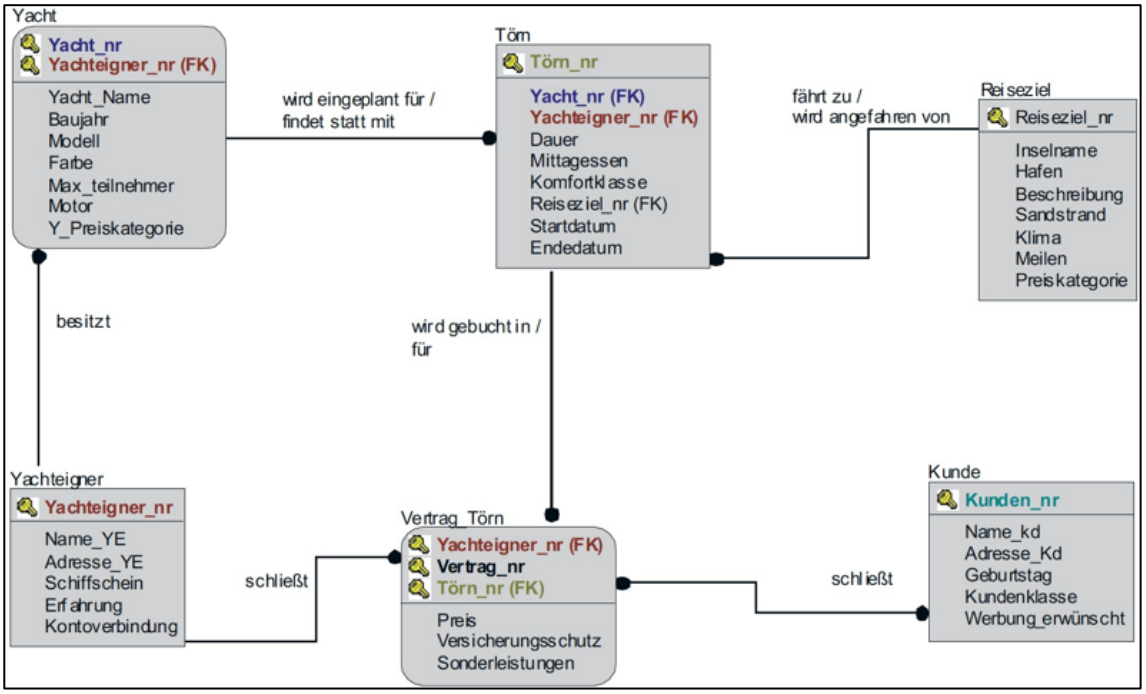


Abb. 172: Der Segeltörn als Relationenmodell

Das Relationenmodell geht einen Schritt weiter. Es zeigt nicht nur die Bedeutung der Daten eines Segeltörns, sondern auch die Eigenschaften bzw. Attribute der jeweiligen Entitätsmengen. Auch die Schlüsselattribute sind auf dem Beispiel durch einen Schlüssel gekennzeichnet.

Sie wissen nicht nur, dass ein Vertrag zwischen Yachteigner und Kunde abgeschlossen wird. Sie wissen auch, dass dieser Vertrag den Preis, den Versicherungsschutz und Sonderleistungen enthält. Den Kunden identifizieren Sie durch seine Kundennummer.

11.1.12 Datenstruktur entwerfen und implementieren

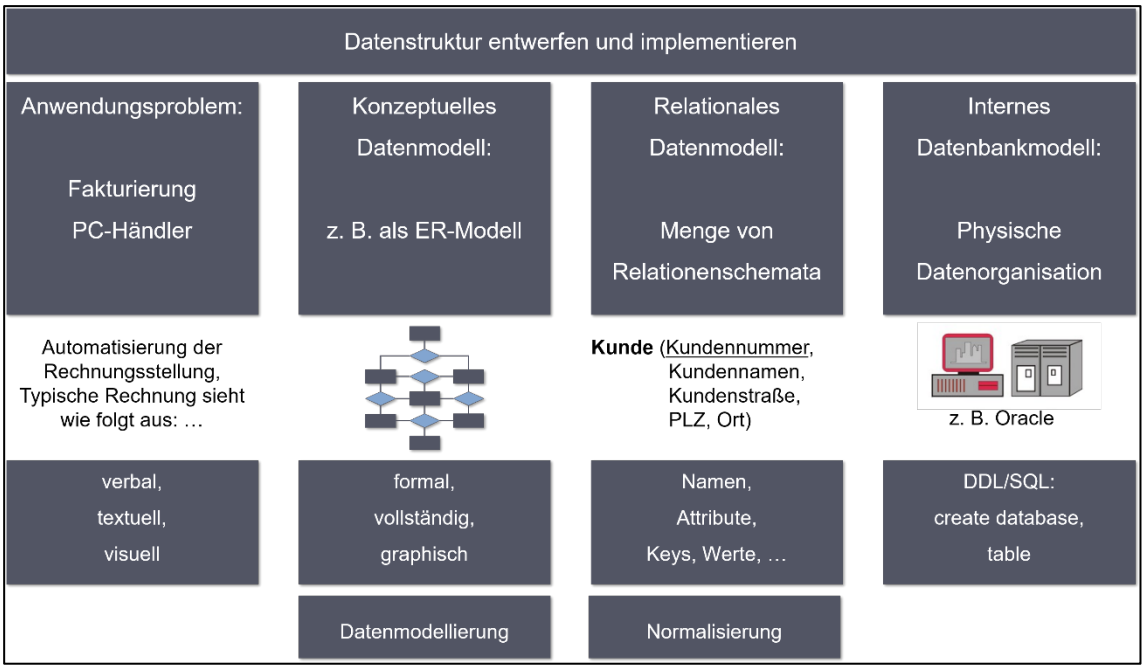


Abb. 173: Datenstrukturen entwerfen und implementieren

11.1.13 ER-Modellierung im Detail

Eine Vertiefung zur Datenmodellierung und zur Prozessmodellierung erhalten Sie in unserem E-Campus Wirtschaftsinformatik mit:

- „Datenmodellierung und Datenbank-Systeme" in der WBT-Kategorie „Software Engineering",
- „Geschäftsprozessmodellierung mit ARIS" in der WBT-Kategorie „Software Engineering" und
- „Geschäftsprozessmodellierung mit BPMN 2.0" in der WBT-Kategorie „Software Engineering".

In dem WBT zur Datenmodellierung erfahren Sie unter anderem auch, wie Sie Datenmodellierung praktizieren und was es mit dem Begriff Normalisierung auf sich hat.

11.2 Prozessmodellierung mit eEPK

11.2.1 Modellierung mit erweiterter Sichtweise

Bisher haben Sie Funktionen, Prozesse und Daten modelliert. In diesem WBT gehen Sie einen Schritt weiter und vereinigen diese verschiedenen Sichtweisen miteinander.

Mit Hilfe der erweiterten ereignisgesteuerten Prozesskette (eEPK), die auf Petri-Netzen beruht, werden Sie sich die Technik ARIS (Architektur integrierter Informationssysteme) anschauen.

ARIS zielt insbesondere auf betriebliche Prozesse ab und umfasst fünf Sichten: Funktionssicht, Organisationssicht, Datensicht, Steuerungssicht (heute Prozesssicht) und Leistungssicht. Diese fünf Sichten werden metaphorisch oft in einem Haus gezeigt, weshalb man auch vom „ARIS-Haus“ spricht.

Um betriebliche Abläufe richtig identifizieren und anschließend modellieren zu können, ist die Prozessorientierung äußerst hilfreich. Sie schafft die Grundlage der Modellierung und wird auf der folgenden Seite erläutert.

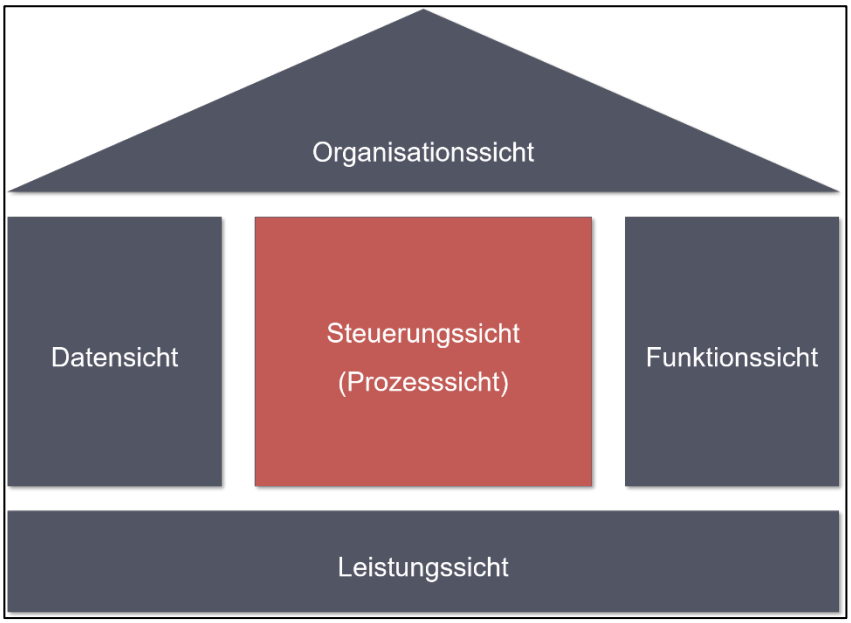


Abb. 174: Schematische Ansicht des ARIS-Haus

11.2.2 Merkmale der Prozessorientierung

- Die Organisationsstruktur orientiert sich nicht mehr an betrieblichen Funktionen, sondern an den Wertschöpfungsprozessen des Unternehmens.
- Prozesse leiten sich aus den Kernkompetenzen ab und sind auf genau definierte marktbezogene Leistungen ausgerichtet.
- Die Prozessleistung ist messbar und kontrollierbar.
- Im Gegensatz zu funktionsorientierten Arbeitsabläufen sind Prozesse stellen-, abteilungs-, funktionsbereichsübergreifend. Sie verlaufen quer zu funktionalen Organisationsstrukturen.
- Jeder Prozess bildet einen eigenständigen Verantwortungsbereich. Prozesse definieren somit die Organisationseinheiten des Unternehmens.
- Die Prozessarbeit wird von interdisziplinären Teams getragen.

11.2.3 Geschäftsfeld und Prozess-Team

Da sich ein Geschäftsprozess meist durch mehrere Fachabteilungen zieht, ist es von Nutzen, das „Silo-Verhalten“ in Unternehmensabteilungen aufzubrechen und die Mitarbeiter stattdessen abteilungsübergreifend in Prozess-Teams zusammen arbeiten zu lassen.

Die Verantwortung erhält das Prozess-Team, beispielsweise bei der Ausführung eines Kundenauftrags. Dies stellt sicher, dass der Bestellvorgang bis hin zur Auslieferung des Produkts einwandfrei ausgeführt wird. Das Team arbeitet dabei nicht in der jeweiligen Fachabteilung, sondern sitzt zusammen, um Kommunikationswege zu optimieren.

Die funktionalen Segmente eines Unternehmens werden folglich aufgebrochen und das Unternehmen organisiert sich nach den internen Prozessabläufen.

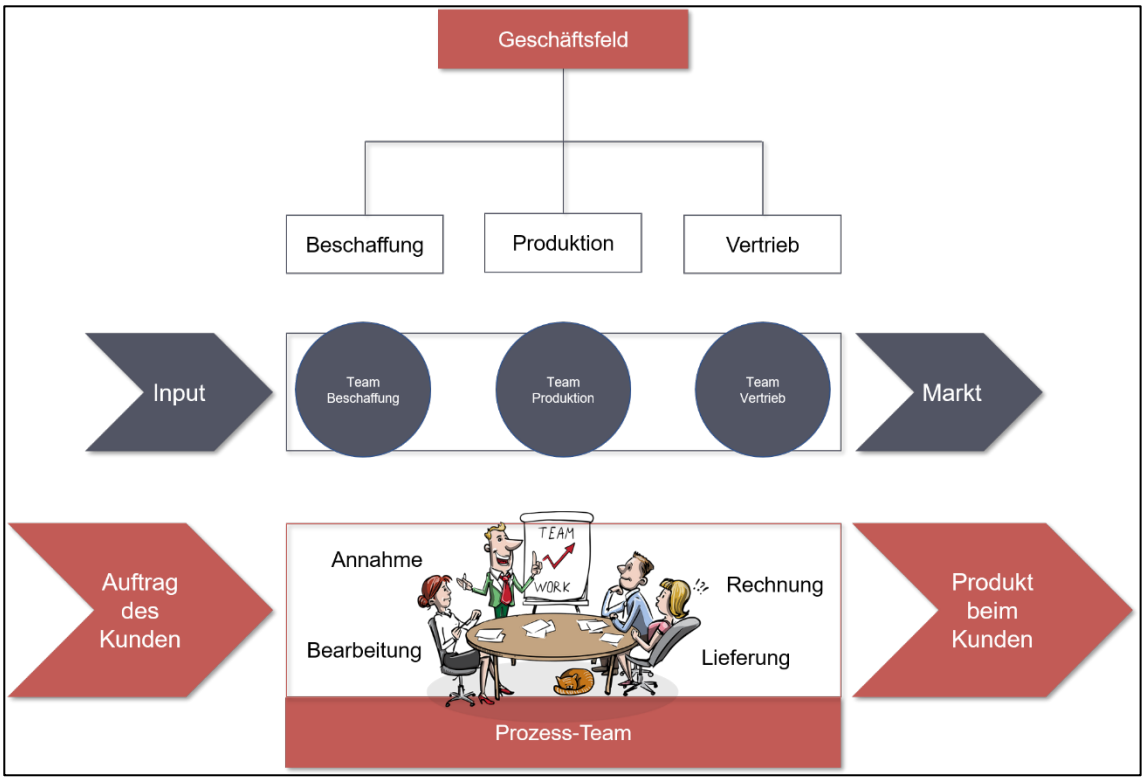


Abb. 175: Geschäftsumfeld und Prozess-Team

11.2.4 Prozesse im Unternehmen

In einem Unternehmen gibt es natürlich mehr als einen Prozess. Jede Prozessleistung wird durch einen Prozessverantwortlichen überwacht und sichergestellt.

Verschiedene Prozesse greifen ineinander. Dies wird durch die roten Verbindungen verdeutlicht. Die einwandfreie Auftragsbearbeitung hat zum Beispiel Einfluss auf die strategische Finanzplanung, welche wiederum Einfluss auf neue Produktinnovationen hat.

Wie man erkennen kann, hängen verschiedene Prozesse zusammen. Sie sind im Unternehmen miteinander verbunden und in ihrer Gesamtheit komplex. ARIS ist eine geeignete Methode, um erkennen zu können, welche Ressourcen die integrativen Geschäftsprozesse benötigen und um Verantwortungen zu klären.

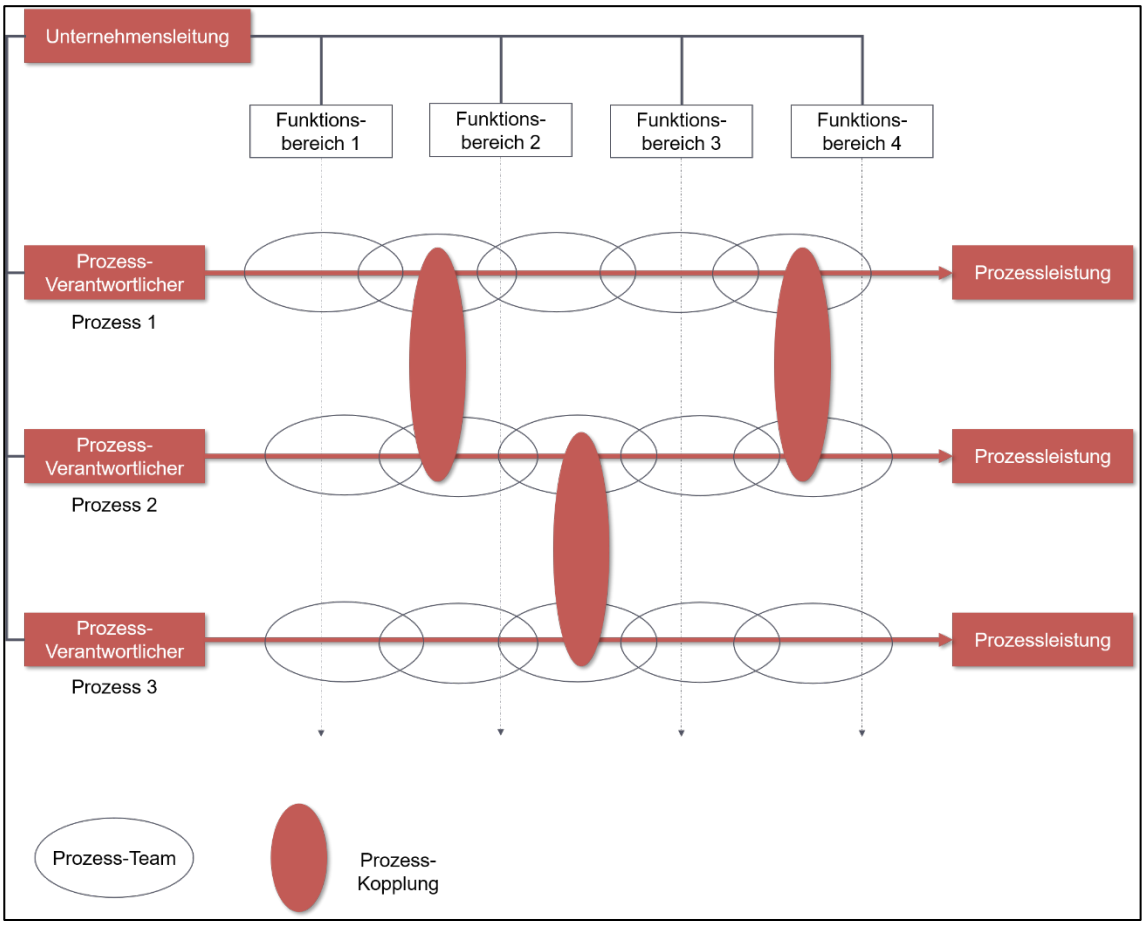


Abb. 176: Unternehmensprozesse und Prozessverantwortung

11.2.5 Architektur Integrierter Informationssysteme

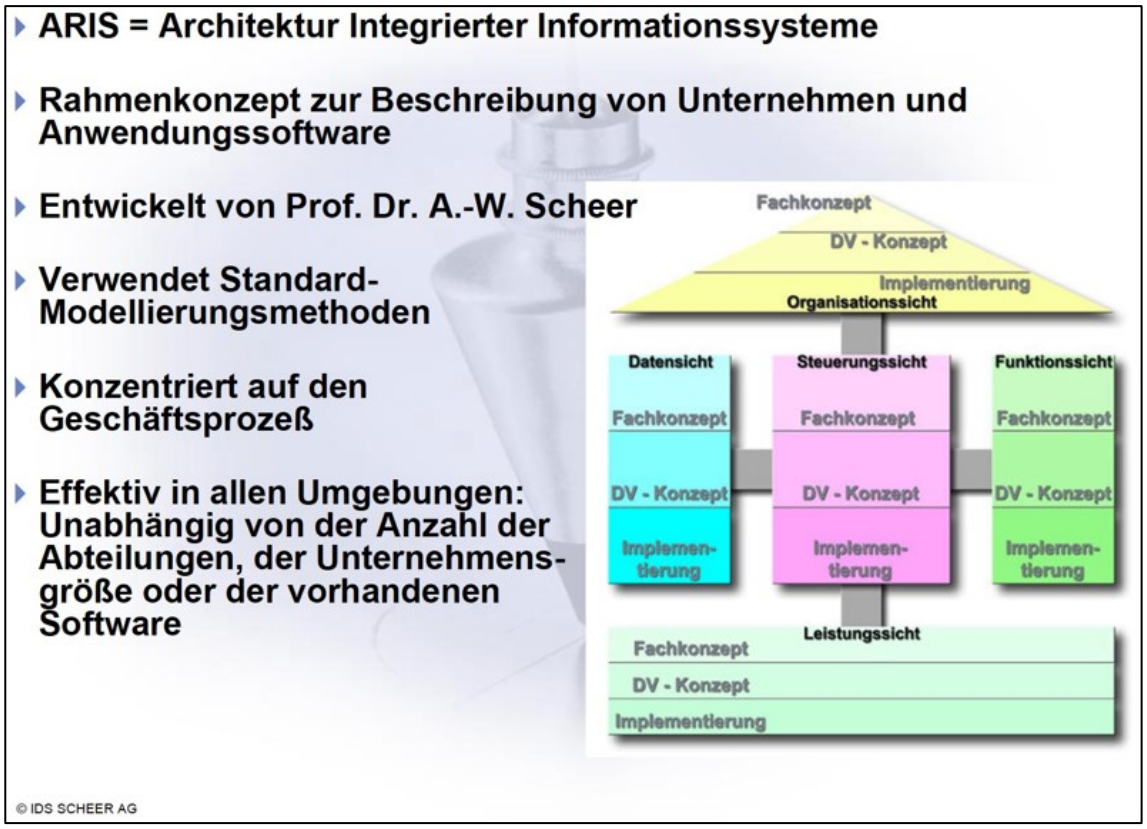


Abb. 177: ARIS Gesamtübersicht

11.2.6 Beispiel: Kundenauftragsbearbeitung

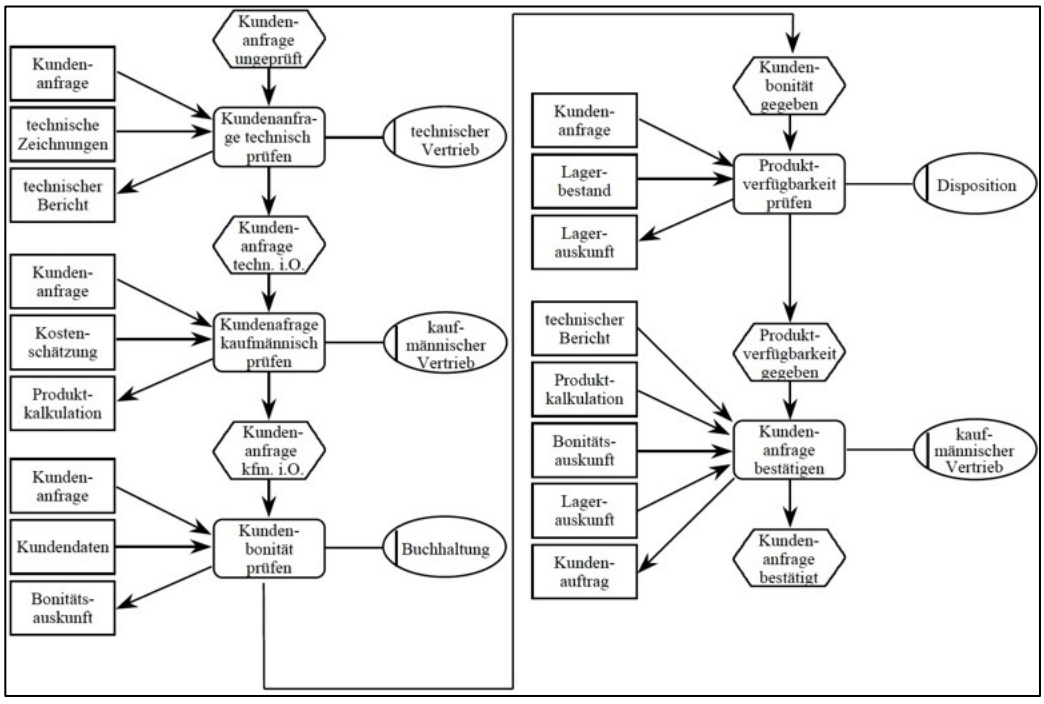


Abb. 178: eEPK am Beispiel eines Kundenauftrages

Am Beispiel des Prozesses Kundenauftragsbearbeitung wird auf die einzelnen Komponenten bzw. Sichtweisen von ARIS eingegangen.

Das Beispiel zeigt bereits eine erweiterte ereignisgesteuerte Prozesskette (eEPK). Basierend auf dem Petri-Netz wurden hier bereits Ereignisse (Sechsecke) und Funktionen (Rundeck) ergänzt.

Den Funktionen sind organisatorische Einheiten zugeordnet (Ovale). Darüber hinaus werden den Funktionen auch Ressourcen zugeordnet (Rechtecke).

In ARIS können alle Sichten einzeln modelliert und integriert werden.

11.2.7 ARIS: Ereignisse und Funktionen

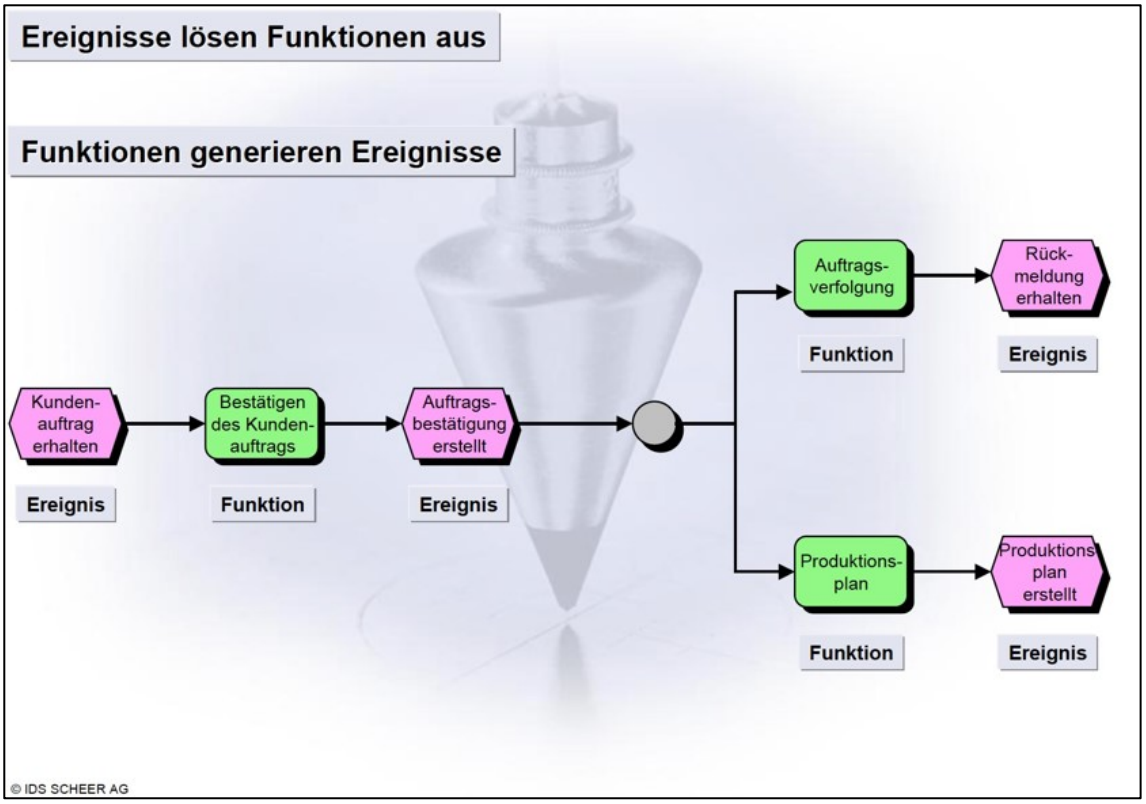


Abb. 179: ARIS – Ereignisse und Funktionen

11.2.8 ARIS: Funktionen und Daten

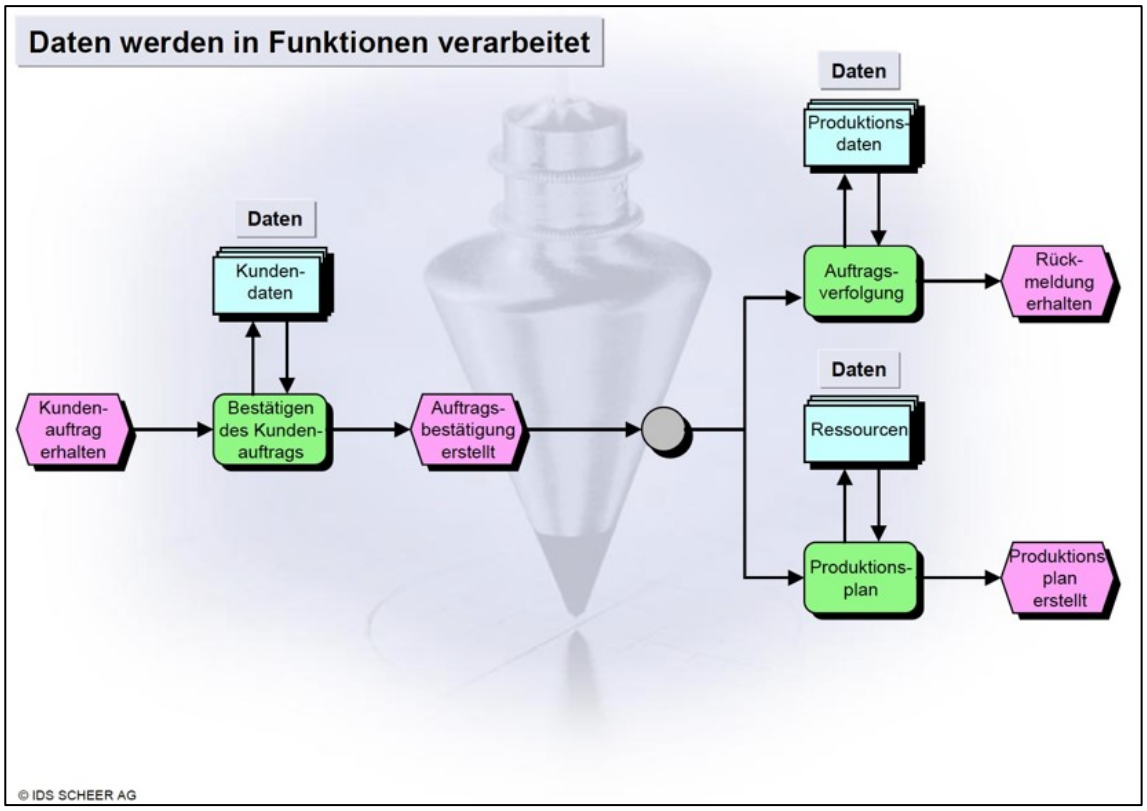


Abb. 180: ARIS – Funktionen und Daten

11.2.9 ARIS: Funktionen und Verantwortlichkeiten

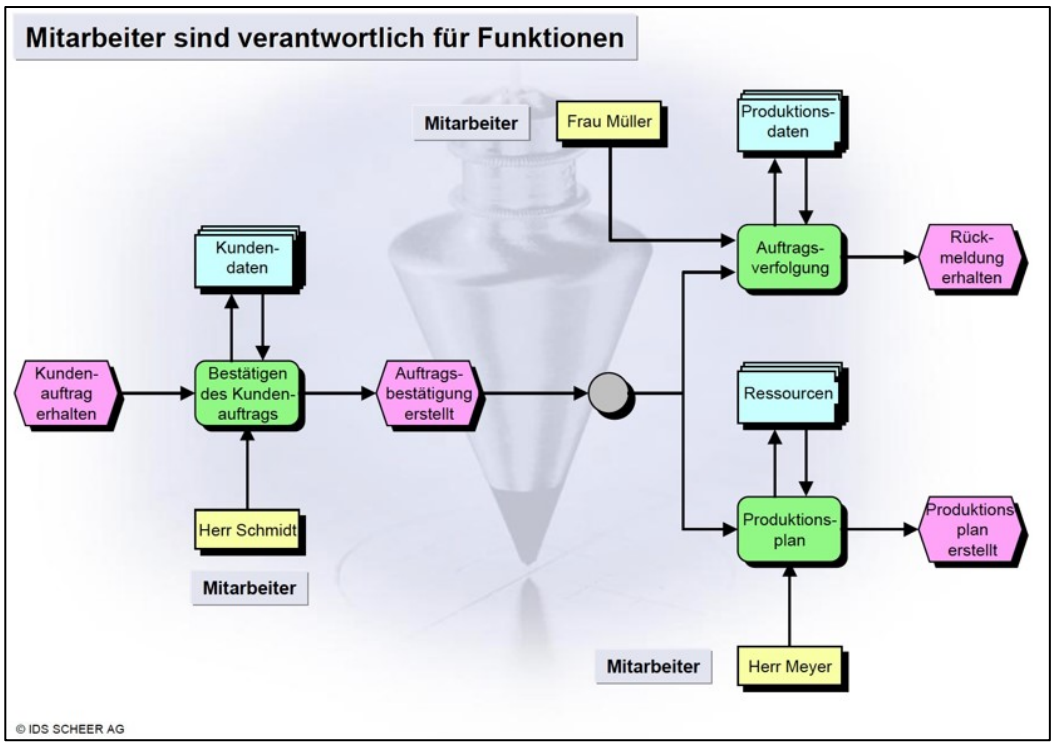


Abb. 181: ARIS – Funktionen und Verantwortlichkeiten

11.2.10 ARIS: Verantwortlichkeiten und Organisationseinheiten

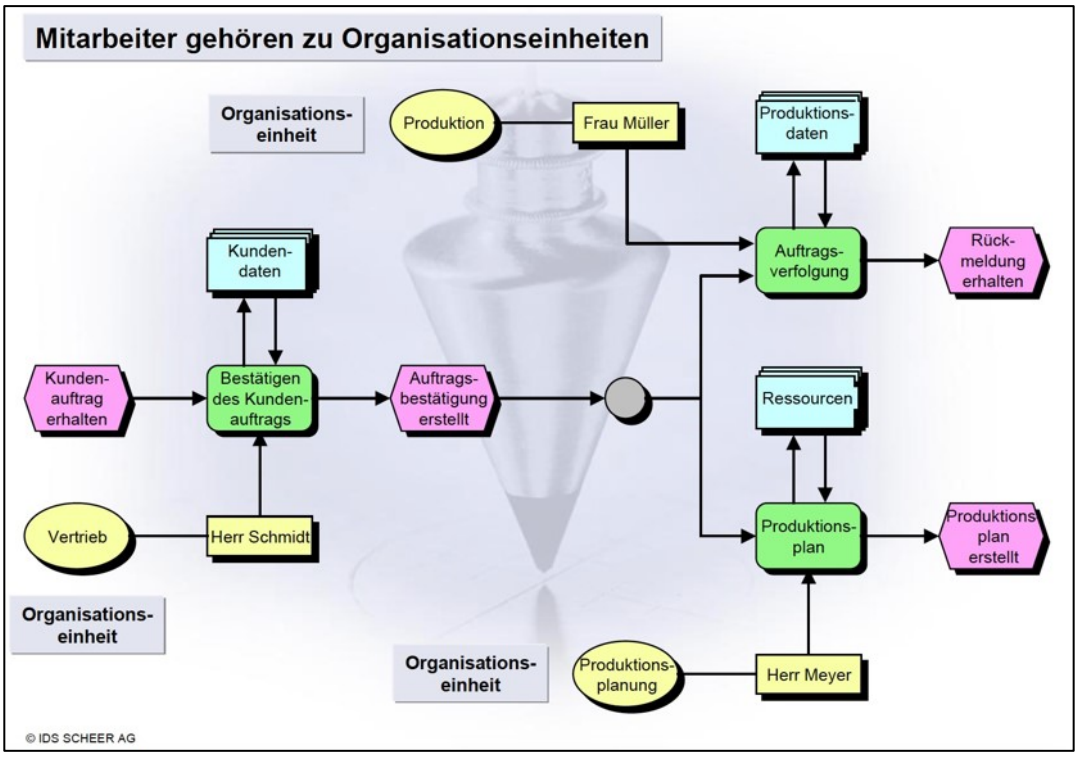


Abb. 182: ARIS – Verantwortlichkeiten und Organisationseinheiten

11.2.11 ARIS: Funktionen und Leistungen

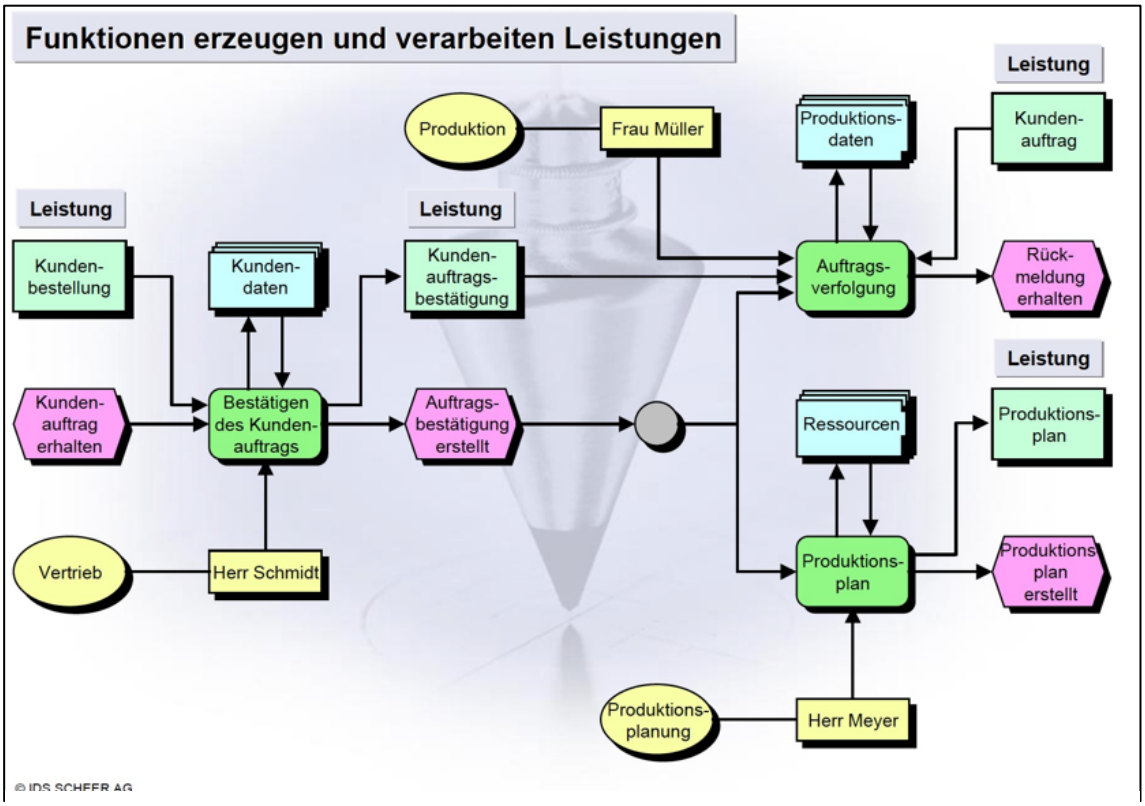


Abb. 183: ARIS – Funktionen und Leistungen

11.2.12 ARIS: Komplexitätsreduzierung

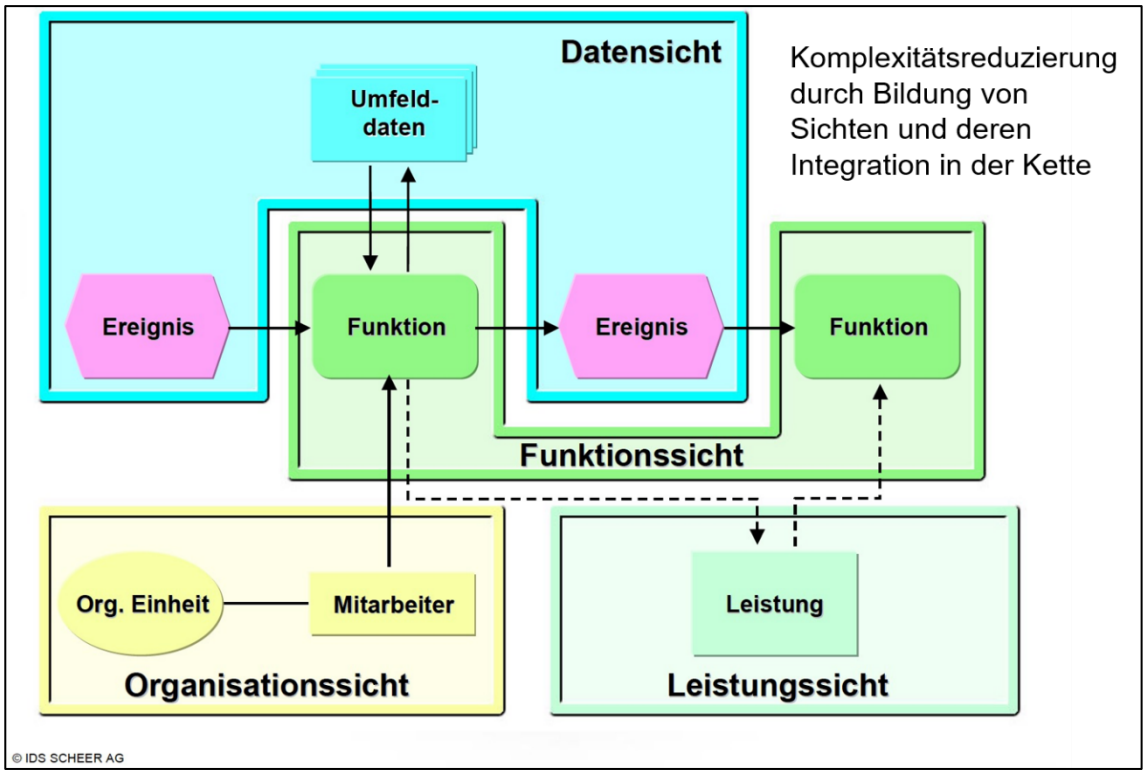


Abb. 184: ARIS – Kombination der Sichtweisen

ARIS wurde für die Modellierung von Geschäftsprozessen entwickelt. Um der vorherrschenden Komplexität von Geschäftsprozessen entgegen zu wirken, unterteilt man mit ARIS Geschäftsprozesse in fünf einzelne Sichten.

Diese werden zunächst einzeln für sich betrachtet und schließlich über die eEPK integriert. Daraus ergibt sich die vollständige Prozessdarstellung. Die fünf Sichten sind:

- die Datensicht,
- die Funktionssicht,
- die Organisationssicht,
- die Leistungssicht und
- die Steuerungssicht (Prozesssicht).

Diese fünf Sichten wurden metaphorisch in einem Haus zusammengefasst, weshalb man auch vom ARIS-Haus spricht.

11.2.13 ARIS: Die Sichtweisen

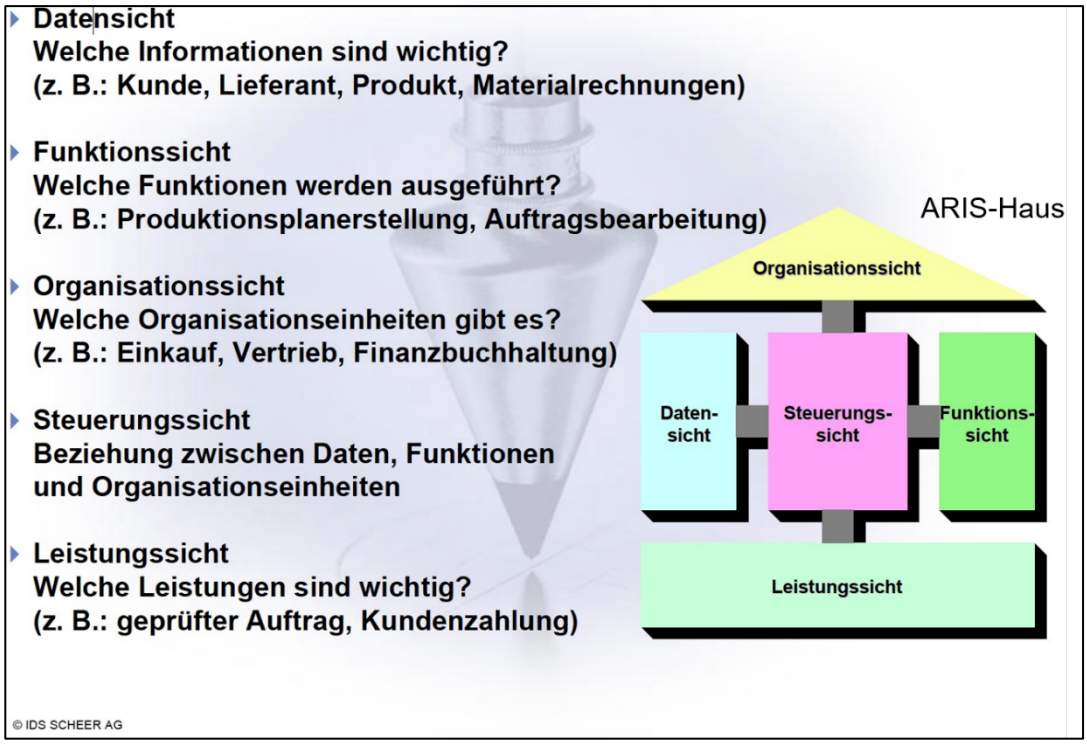


Abb. 185: ARIS – Die Sichtweisen

11.2.14 ARIS-Haus

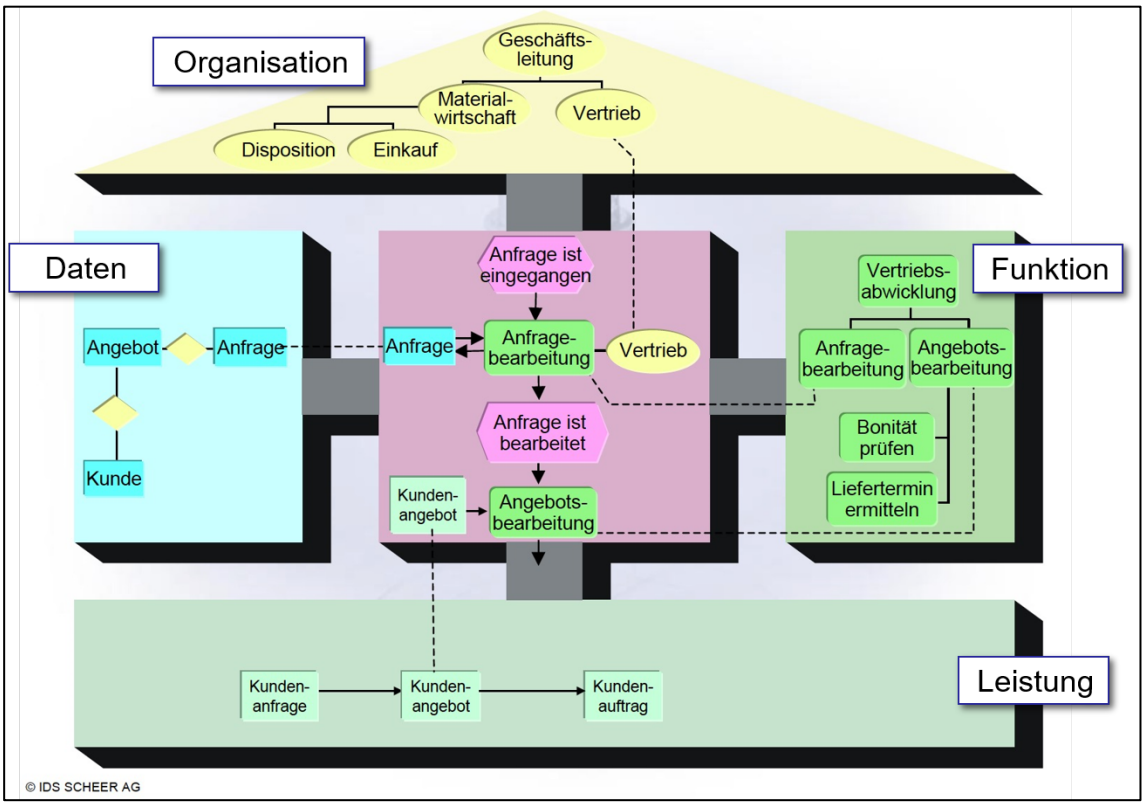


Abb. 186: ARIS – Das Haus am Beispiel einer Anfragebearbeitung

11.2.15 ARIS und Phaseneinteilung

Die Anwendung der Methode umschließt drei Phasen: Die Fachkonzeption, die Datenverarbeitungs-Konzeption und die Implementierung.

Bei der Anwendung der ARIS-Methode erfolgt bei jeder der fünf Sichten das gleiche Vorgehen: Die Fachkonzeption, die Datenverarbeitungs-Konzeption und die Implementierung. Die Fachkonzeption ist dabei eine fachliche Grob-Konzeption des Prozesses. Die DV-Konzeption baut auf dem Fachkonzept auf und umfasst alle technischen Aspekte dessen. (Auch hier wird also wieder das Prinzip der schrittweisen Verfeinerung und das Prinzip der Trennung der Essenz von der Inkarnation angewendet). In der letzten Phase werden die beiden vorigen Konzepte implementiert. Auf der nächsten Seite sehen Sie dieses Vorgehen noch einmal genauer.

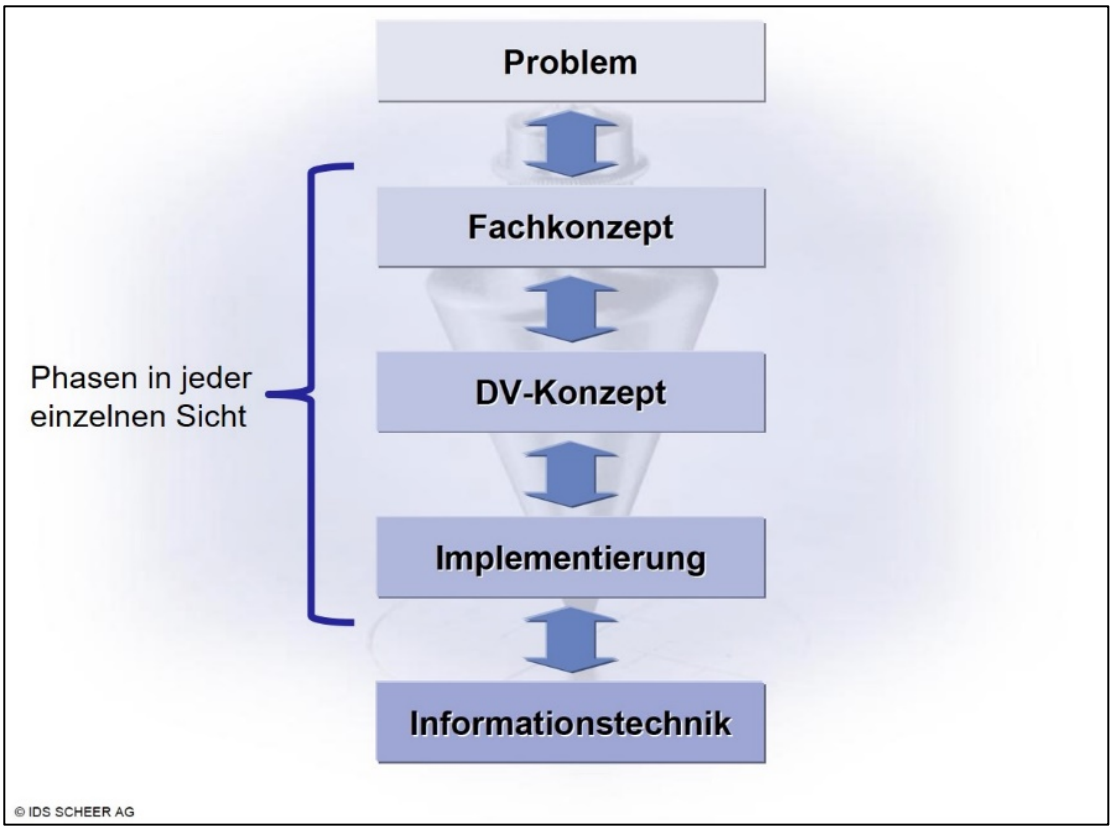


Abb. 187: ARIS – Vorgehensweise in den einzelnen Sichtweisen

11.2.16 ARIS in der Konzeption und Implementierung

Fachkonzept

- Standardisierte fachliche Beschreibung der Unternehmenskonzepte

DV-Konzept

- Übernahme der fachlichen Anforderungen in die Beschreibungssprache der DV-Technik
- Z. B. Struktogramme, Datenbankmodelle oder Netztopologien

Implementierung

- Beschreibung der konkreten Hard- und Software-Komponenten für die Umsetzung des Unternehmenskonzeptes
- Z. B. Datenbeschreibungssprachen, Netzwerkprotokolle oder Quellcode

Abb. 188: ARIS – Sichtweisen und Phasenvorgehen integriert

11.3 Abschlusstest – WBT11

11.3.1 Abschlusstest

Bitte beantworten Sie die folgenden Fragen durch das Ankreuzen der korrekten Antworten (Tab. 12). Bei einigen Fragen können auch mehrere Antworten richtig sein.

Nr.	Frage	Richtig	Falsch
1	ARIS basiert auf eEPK und bietet eine integrative Sicht auf einen Geschäftsprozess.		
2	Ein logisches Datenmodell wird auch Relationenmodell genannt.		
3	Welche Prozesse und Strukturansichten gibt es?		
	Organisationssicht		
	Datensicht		
	Steuerungssicht		
	Funktionssicht		
	Leistungssicht		
4	Ein ER-Modell kann nicht in der Fachkonzeption oder Systemkonzeption angewendet werden.		
5	Welche Phasen umfasst ARIS?		
	Fachkonzeption		
	DV-Konzeption		
	Technische Konzeption		
	Problem-Konzeption		
	Implementierung		
6	Ein ER-Modell bietet eine _____ Sicht auf die Daten als Ressourcen im Unternehmen.		

Tab. 12: Abschlusstest – WBT11

11.4 Typische Aufgabenstellungen

Typische Aufgabenstellungen – Daten- und Prozessmodellierung

Zur Bearbeitung dieser Aufgabenstellungen beachten Sie bitte: Verlangt ist eine fachlich zutreffende, inhaltlich nachvollziehbare und kausal zusammenhängende Erörterung aus vollständigen Sätzen in lesbarer Handschrift. Für jede Aufgabe: Maximal zwei Seiten Text.

Aufgabe 1:

> Erläutern Sie die Aufgaben, Ziele und konkreten Aktivitäten eines betrieblichen Datenmanagements. Nehmen Sie anschließend Stellung zur Entwicklung eines „unternehmensweiten Datenmodells“ (UDM).

Aufgabe 2:

> Erläutern Sie die Schritte der Informationsmodellierung beginnend auf der obersten Ebene des Unternehmens über semantische Datenmodelle und Relationen-Modelle hin zu technischen Datenbanksystemen.

Aufgabe 3:

> Aus welchen Komponenten besteht ARIS? (mit Sichten als „Ergebnismodelle“, Phasen als Vorgehensmodelle).

Aufgabe 4:

> Erläutern Sie die ARIS-Sichten.

Aufgabe 5:

> Erläutern Sie die ARIS-Phasen.

Aufgabe 6:

> Erläutern Sie, warum ARIS als „integrierender“ Modellierungsansatz verstanden wird. Erläutern Sie dabei, was ARIS „integriert“ und auf welche Weise dies geschieht.

Anhang

Lösung des Abschlusstests in WBT 01 (Tab. 2):

Nr.	Frage	Richtig	Falsch
1	Termine in der Entwicklung einer Software sind nicht planbar.		
	Richtig		
	Falsch	X	
2	Software ist ein komplexes, immaterielles und technisches Produkt.		
	Richtig	X	
	Falsch		
3	Was umfasst Software Engineering unter anderem?		
	Entwicklungstätigkeiten	X	
	Projektmanagement	X	
	Qualitätssicherung	X	
	Wirtschaftlichkeitsanalyse	X	
	Subjektive Bewertungsmethoden		
4	Modelle sind zweckorientiert.		
	Richtig	X	
	Falsch		
5	Das WAS der Planung und Entwicklung von IT-Systems wird durch die ERGEBNIS-SICHT dargestellt. Das WIE der Planung und Entwicklung von IT-Systems wird durch die PROZESS-SICHT dargestellt.		
6	Was sind die Hauptmerkmale des allgemeinen Modellbegriffs?		
	Abbildungsmerkmal	X	
	Standardisierungsmerkmal		
	Pragmatisches Merkmal	X	
	Messbarkeitsmerkmal		

	Verkürzungsmerkmal	X	
	Praktisches Merkmal		

Tab. 13: Lösungen des Abschlusstests WBT01

Lösung des Abschlusstests in WBT 02 (Tab. 3):

Nr.	Frage	Richtig	Falsch
1	Welche Erfolgsfaktoren sind in IT-Projekten wichtig?		
	Ein Projektleiter mit fachlicher Expertise	X	
	Unterstützung durch die Geschäftsführung	X	
	Einbindung von zukünftigen Nutzern in der Entwicklungsphase	X	
	Projektcontrolling	X	
	Projektreporting	X	
2	Das magische Dreieck ist bei IT-Projekten zu vernachlässigen.		
	Richtig		
	Falsch	X	
3	Die Projektgröße stellt keinen zu beachtenden Faktor bei der Projektplanung dar.		
	Richtig		
	Falsch	X	
4	Projekte werden durch den Technologiefortschritt bedingt.		
	Richtig	X	
	Falsch		
5	Projekte können neben dem Tagesgeschäft durchgeführt werden.		
	Richtig		
	Falsch	X	
6	Projektmanagement umfasst...		
	die Funktion	X	

	die Organisationsform	X	
	die technische Ausführung		
	die Methoden	X	
	die technische Fachkonzeption		
7	Projekte zeichnen sich durch die EINMALIGKEIT der Bedingungskonstellation aus. Die EXAKTE Zielvorgabe ist dabei besonders wichtig, um die Projektziele „in time, on budget, an on TIME." zu erreichen.		

Tab. 14: Lösungen des Abschlusstests WBT02

Lösung des Abschlusstests in WBT 03 (Tab. 4):

Nr.	Frage	Richtig	Falsch
1	Die Projekt-Organisation stellt keinen Einflussfaktor bei der Aufsetzung eines Projektes dar.		
	Richtig		
	Falsch	X	
2	Die Einfluss-Projekt-Organisation unterstellt der Projektleitung dediziertes Personal mit voller Entscheidungsbefugnis.		
	Richtig		
	Falsch	X	
3	Welche Projekt-Organisationsformen gibt es?		
	Einfluss-Projekt-Organisation	X	
	Reine Projekt-Organisation	X	
	Matrix-Projekt-Organisation	X	
	Arbeiterkreis		
4	Die Projektleitung muss keine besonderen Bedürfnisse der Mitarbeiter beachten.		
	Richtig		
	Falsch	X	
5	Die Projektleitung muss sich im Projekt durch Fachkenntnisse einbringen.		

	Richtig	X	
	Falsch		

Tab. 15: Lösungen des Abschlusstests WBT03

Lösung des Abschlusstests in WBT 04 (Tab. 5):

Nr.	Frage	Richtig	Falsch
1	Berichte arbeiten nur mit Schätzwerten, die zu Beginn des Projektes geplant wurden.		
	Richtig		
	Falsch	X	
2	Projektmanagement-Standards sind „Best Practices“, die für jedes Projekt ohne Anpassung gültig sind.		
	Richtig		
	Falsch	X	
3	Welche Schritte der Projektplanung können durch eine PM-Software abgebildet werden?		
	Vorgänge des Projektes erfassen	X	
	Dauer eines Vorganges graphisch abbilden	X	
	Ist- und Soll-Zustände vergleichen	X	
	Kapazitätsüberlastungen anzeigen	X	
	Einen Netzplan erstellen	X	
4	Das PMBoK und PRINCE2 sind Projektmanagement-Standards, die weit verbreitet als „Best Practices“ gelten.		
	Richtig	X	
	Falsch		
5	Projektmanagement-Software bieten automatisch generierte Eventualpläne bei Projektabweichung.		
	Richtig		
	Falsch	X	

6	Projektmanagement-Standards bieten...		
	Analysewerkzeuge	X	
	Vorgangserfassung		
	die Abbildung von Eventualplänen		
	Methoden, die auf das Projekt angepasst werden können	X	
	technische Fachkonzeptionen	X	
7	Das Hauptproblem bei Projekten ist, dass häufig SCHÄTZWERTE verwendet werden.		

Tab. 16: Lösungen des Abschlusstests WBT04

Lösung des Abschlusstests in WBT 05 (Tab. 6):

Nr.	Frage	Richtig	Falsch
1	Die Prozess-Sicht ist die organisatorische Projektmanagement Perspektive auf das Software Engineering.		
	Richtig		
	Falsch	X	
2	Das allgemeine Vorgehensmodell ist auch allgemein gültig und kann auf jedes Projekt angewendet werden.		
	Richtig		
	Falsch	X	
3	Das allgemeine Vorgehensmodell besteht aus folgenden Phasen…		
	Vorprojekt	X	
	Hauptprojekt	X	
	Detailprojekt	X	
	Systembau, Systemeinführung und -übergabe	X	
	Systemnutzung	X	

4	Durch die Vorgehensmodelle werden unstrukturierte Vorgehensweisen, die z. B. zu mangelnder Softwarequalität führen systematisch durchbrochen.		
	Richtig	X	
	Falsch		
5	Die Grundformen der Vorgehensmodellen sind...		
	die allgemeinen Vorgehensmodelle	X	
	die sequentiellen Vorgehensmodelle	X	
	die evolutionären Vorgehensmodelle	X	
	die agilen Vorgehensmodelle	X	
	die strukturierten Vorgehensmodelle		
6	Die allgemeinen Vorgehensmodelle liefern eine strukturierte Vorgehensweise und hält PHASEN, AKTIVITÄTEN und KEINE ERGEBNISSE fest.		

Tab. 17: Lösungen des Abschlusstests WBT05

Lösung des Abschlusstests in WBT 06 (Tab. 7):

Nr.	Frage	Richtig	Falsch
1	Sequentielle Vorgehensmodelle sind bei Standard-Software-Einführungen zu bevorzugen.		
	Richtig		
	Falsch	X	
2	Clustering ist unter die parallel-sequentiellen Vorgehensmodelle einzusortieren.		
	Richtig		
	Falsch	X	
3	Das allgemeine Vorgehensmodell besteht aus folgenden Phasen…		
	Vorprojekt	X	
	Hauptprojekt	X	
	Detailprojekt	X	

	Systembau, Systemeinführung und -übergabe	X	
	Systemnutzung	X	
4	Evolutionäre Vorgehensmodelle sind mit agilen Vorgehensmodellen identisch.		
	Richtig		
	Falsch	X	
5	Die Grundformen der Vorgehensmodellen sind...		
	die allgemeinen Vorgehensmodelle	X	
	die sequentiellen Vorgehensmodelle	X	
	die evolutionären Vorgehensmodelle	X	
	die agilen Vorgehensmodelle	X	
	die strukturierten Vorgehensmodelle		

Tab. 18: Lösungen des Abschlusstests WBT06

Lösung des Abschlusstests in WBT 07 (Tab. 8):

Nr.	Frage	Richtig	Falsch
1	Agile Vorgehensmodelle haben eine flexible Sichtweise und unterliegen einer hohen Dokumentationslast.		
	Richtig		
	Falsch		X
2	Die agilen Vorgehensmodellen besinnen sich zurück auf den Ansatz „Code and Fix“.		
	Richtig		
	Falsch		X
3	Die Vorgehensweise in agilen Projekten ist...		
	inkrementell.	X	
	als Verhaltensschablone für alle Projektbeteiligten gedacht.	X	
	unspezifisch.	X	

	mit Methoden und Praktiken angereichert.	X	
	für jedes Projekt übernehmbar.		
4	Durch die Vorgehensmodelle werden unstrukturierte Vorgehensweisen, die z. B. zu mangelnder Softwarequalität führen systematisch durchbrochen.		
	Richtig	X	
	Falsch		
5	Die Grundformen der Vorgehensmodellen sind...		
	die allgemeinen Vorgehensmodelle	X	
	die sequentiellen Vorgehensmodelle	X	
	die evolutionären Vorgehensmodelle	X	
	die agilen Vorgehensmodelle	X	
	die strukturierten Vorgehensmodelle		
6	Die Trendentwicklung seit den 90er Jahren zeigt deutlich, dass konventionelle Vorgehensmodelle auch weiterhin ihre Berechtigung haben.		

Tab. 19: Lösungen des Abschlusstests WBT07

Lösung des Abschlusstests in WBT 08 (Tab. 9):

Nr.	Frage	Richtig	Falsch
1	Agile Vorgehensmodelle haben eine flexible Sichtweise und unterliegen einer hohen Dokumentationslast.		
	Richtig		
	Falsch	X	
2	Die agilen Vorgehensmodellen besinnen sich zurück auf den Ansatz „Code and Fix“.		
	Richtig		
	Falsch	X	
3	Die Vorgehensweise in agilen Projekten ist…		
	inkrementell.	X	

	als Verhaltensschablone für alle Projektbeteiligten gedacht.	X	
	unspezifisch.	X	
	mit Methoden und Praktiken angereichert.	X	
	für jedes Projekt übernehmbar.		
4	Durch die Vorgehensmodelle werden unstrukturierte Vorgehensweisen, die z. B. zu mangelnder Softwarequalität führen systematisch durchbrochen.		
	Richtig	X	
	Falsch		
5	Die Grundformen der Vorgehensmodellen sind...		
	die allgemeinen Vorgehensmodelle	X	
	die sequentiellen Vorgehensmodelle	X	
	die evolutionären Vorgehensmodelle	X	
	die agilen Vorgehensmodelle	X	
	die strukturierten Vorgehensmodelle	X	
6	Die Trendentwicklung seit den 90er Jahren zeigt deutlich, dass konventionelle Vorgehensmodelle auch weiterhin ihre Berechtigung haben.		

Tab. 20: Lösungen des Abschlusstests WBT08

Lösung des Abschlusstests in WBT 09 (Tab. 10):

Nr.	Frage	Richtig	Falsch
1	Das Prinzip der Dekomposition gehört nicht zur Systemtheorie.		
	Richtig		
	Falsch	X	
2	Das Blockkonzept besagt, dass in einer Manier von bottom-up ein System erbaut werden kann.		
	Richtig		
	Falsch	X	
3	Zu den Hauptaspekten der Systemtheorie gehören...		

	Wirkungsaspekt	X	
	Strukturaspekt	X	
	Blackbox-Aspekt		
	Abstraktion	X	
	Dekomposition	X	
4	Das Blockkonzept ist beispielsweise dazu da ein ERP-System in seine Subsysteme und Moduln aufzuteilen.		
	Richtig	X	
	Falsch		
5	Folgende Aussagen über Petri-Netze sind richtig:		
	Petri-Netze sind sehr spezifische Graphen.	X	
	Auf einen Zustand folgt immer ein Ereignis.	X	
	Ein Petri-Netz kann auf ein Ereignis enden.		
	Petri-Netze können nicht in der Netzplantechnik angewandt werden.		
	Petri-Netze helfen Abläufe zu optimieren.	X	
6	Ein Graph besteht aus verschiedenen ELEMENTEN. Zu diesen gehören KNOTEN, KANTEN und BEZIEHUNGEN.		

Tab. 21: Lösungen des Abschlusstests WBT09

Lösung des Abschlusstests in WBT 10 (Tab. 11):

Nr.	Frage	Richtig	Falsch
1	Funktionsspezifische Datendefinitionen und Datenzuordnungen verursachen Redundanzen und Inkonsistenzen.		
	Richtig	X	
	Falsch		
2	Zu den dynamischen Aspekten der Aufgabenmodellierung gehören zum Beispiel PAP und Struktogramme.		
	Richtig	X	

	Falsch		
3	Zu den Dimensionen der Aufgabenmodellierung gehören...		
	Struktur von Aufgaben	X	
	Ressourcen von Aufgaben	X	
	Einbindung von Organisationsstrukturen	X	
4	Funktionsorientierte betriebliche IT-Systeme haben häufig viele interne Schnittstellen.		
	Richtig	X	
	Falsch		
5	Zu den Prinzipien der Modellierung gehören:		
	Die hierarchische Dekomposition, Strukturierung und Modularisierung, konstruktive Voraussicht, perspektivische Betrachtung, methodische Standardisierung und Trennung der Essenz von der Inkarnation.	X	
	Die hierarchische Dekomposition, Strukturierung und Modularisierung, konstruktive Voraussicht, perspektivische Betrachtung, methodische Standardisierung und Trennung der Essenz von der Reinkarnation.		
	Die hierarchische Komposition, Strukturierung und Modularisierung, konstruktive Voraussicht, perspektivische Betrachtung, methodische Standardisierung und Trennung der Essenz von der Inkarnation.		
6	Die traditionellen Funktionalbereiche (Aufgabenbereiche) definieren die statischen Aufbauorganisationseinheiten des Unternehmens.		
	Richtig	X	
	Falsch		

Tab. 22: Lösungen des Abschlusstests WBT10

Lösung des Abschlusstests in WBT 11 (Tab. 12):

Nr.	Frage	Richtig	Falsch
1	ARIS basiert auf eEPK und bietet eine integrative Sicht auf einen Geschäftsprozess.		
	Richtig	X	
	Falsch		
2	Ein logisches Datenmodell wird auch Relationenmodell genannt.		
	Richtig		
	Falsch		X
3	Welche Prozesse und Strukturansichten gibt es?		
	Organisationssicht	X	
	Datensicht	X	
	Steuerungssicht	X	
	Funktionssicht	X	
	Leistungssicht	X	
4	Ein ER-Modell kann nicht in der Fachkonzeption oder Systemkonzeption angewendet werden.		
	Richtig		
	Falsch		X
5	Welche Phasen umfasst ARIS?		
	Fachkonzeption	X	
	DV-Konzeption	X	
	Technische Konzeption		
	Problem-Konzeption		
	Implementierung	X	
6	Ein ER-Modell bietet eine STARRE Sicht auf die Daten als Ressourcen im Unternehmen.		

Tab. 23: Lösungen des Abschlusstests WBT11

Impressum

Reihe: **Arbeitspapiere Wirtschaftsinformatik** (ISSN 1613-6667)

Bezug: http://wi.uni-giessen.de

Herausgeber: Prof. Dr. Axel Schwickert
Prof. Dr. Bernhard Ostheimer

c/o Professur BWL – Wirtschaftsinformatik
Justus-Liebig-Universität Gießen
Fachbereich Wirtschaftswissenschaften
Licher Straße 70
D – 35394 Gießen
Telefon (0 64 1) 99-22611
Telefax (0 64 1) 99-22619
eMail: Axel.Schwickert@wirtschaft.uni-giessen.de
http://wi.uni-giessen.de

Ziele: Die Arbeitspapiere dieser Reihe sollen konsistente Überblicke zu den Grundlagen der Wirtschaftsinformatik geben und sich mit speziellen Themenbereichen tiefergehend befassen. Ziel ist die verständliche Vermittlung theoretischer Grundlagen und deren Transfer in praxisorientiertes Wissen.

Zielgruppen: Als Zielgruppen sehen wir Forschende, Lehrende und Lernende in der Disziplin Wirtschaftsinformatik sowie das IT-Management und Praktiker in Unternehmen.

Quellen: Die Arbeitspapiere entstehen aus Forschungs-, Abschluss-, Studien- und Projektarbeiten sowie Begleitmaterialien zu Lehr-, Vortrags- und Kolloquiumsveranstaltungen der Professur BWL – Wirtschaftsinformatik, Prof. Dr. Axel Schwickert, Justus-Liebig-Universität Gießen sowie der Professur für Wirtschaftsinformatik, insbes. medienorientierte Wirtschaftsinformatik, Prof. Dr. Bernhard Ostheimer, Fachbereich Wirtschaft, Hochschule Mainz.

Hinweise: Wir nehmen Ihre Anregungen zu den Arbeitspapieren aufmerksam zur Kenntnis und werden uns auf Wunsch mit Ihnen in Verbindung setzen.

Falls Sie selbst ein Arbeitspapier in der Reihe veröffentlichen möchten, nehmen Sie bitte mit einem der Herausgeber unter obiger Adresse Kontakt auf.

Informationen über die bisher erschienenen Arbeitspapiere dieser Reihe erhalten Sie unter der Web-Adresse http://wi.uni-giessen.de/