DUALE HOCHSCHULE SH

in Trägerschaft der Wirtschaftsakademie Schleswig-Holstein

Zielgruppe: **Wirtschaftsinformatik**

Modul: **Software Engineering**

Foliensatz:

Anforderungsanalyse und Spezifikation

Prof. Dr. Malte Prieß

DUALE HOCHSCHULE SH

in Trägerschaft der Wirtschaftsakademie Schleswig-Holstein

Software Engineering

Anforderungsanalyse und Spezifikation

Prof. Dr. Malte Prieß

Agenda (Lernziele)

I. Stakeholder, Peopleware, Machbarkeitsstudie, Make or Buy, Ziele und Ergebnisse
II. Die Bedeutung im Entwicklungsprozess
III. Die Anforderungsanalyse
IV. Begriffslexikon und Begriffsmodell
V. Die Anforderungsspezifikation
VI. Die Darstellung der Spezifikation
    a. Anforderungsspezifikation (z.B. Pflichtenheft oder Product Backlog)
    b. Prototypen
    c. Modelle (z.B. UML, EPK, BPMN)

DUALE HOCHSCHULE SH

in Trägerschaft der Wirtschaftsakademie Schleswig-Holstein

Software Engineering

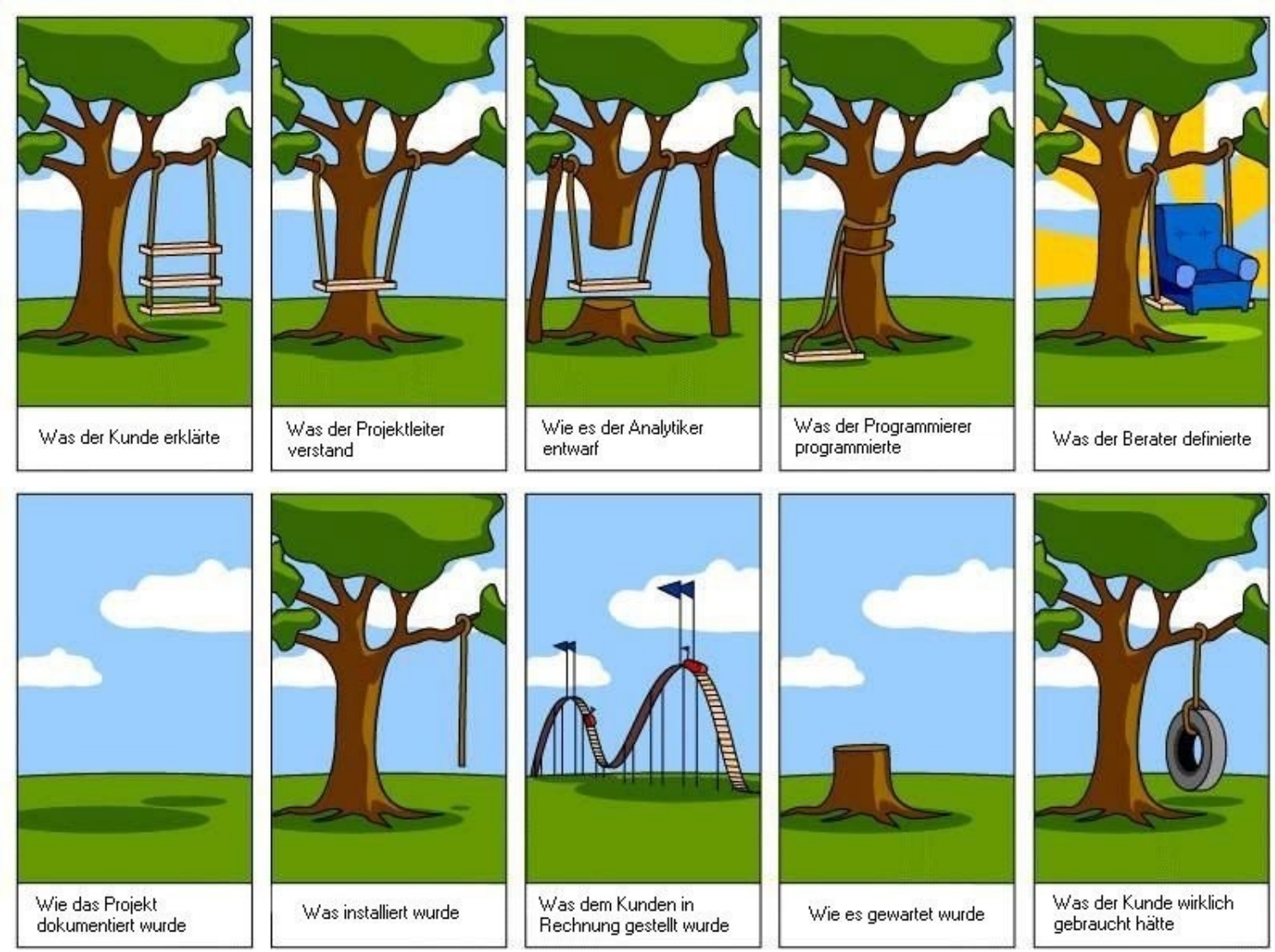
Stakeholder, Peopleware, Machbarkeitsstudie, Make or Buy, Ziele und Ergebnisse

Prof. Dr. Malte Prieß

Agenda (Lernziele)

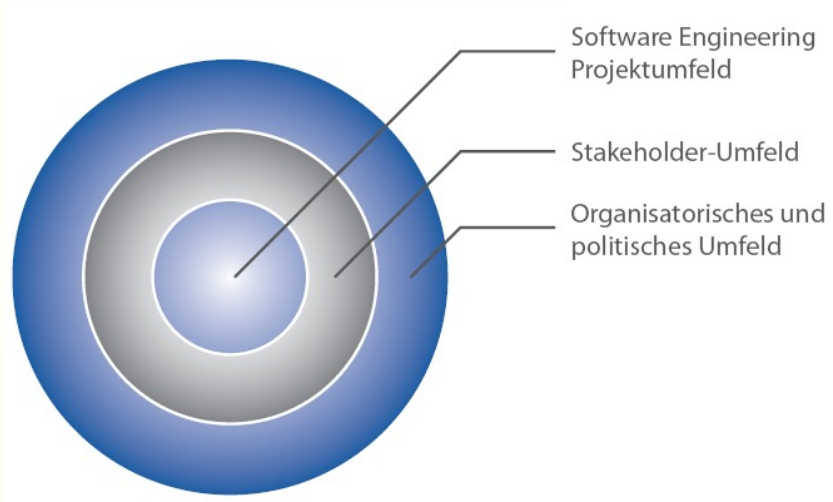
I. Stakeholder, Peopleware, Machbarkeitsstudie, Make or Buy, Ziele und Ergebnisse
II. Die Bedeutung im Entwicklungsprozess
III. Die Anforderungsanalyse
IV. Begriffslexikon und Begriffsmodell
V. Die Anforderungsspezifikation
VI. Die Darstellung der Spezifikation
    a. Anforderungsspezifikation (z.B. Pflichtenheft oder Product Backlog)
    b. Prototypen
    c. Modelle (z.B. UML, EPK, BPMN)

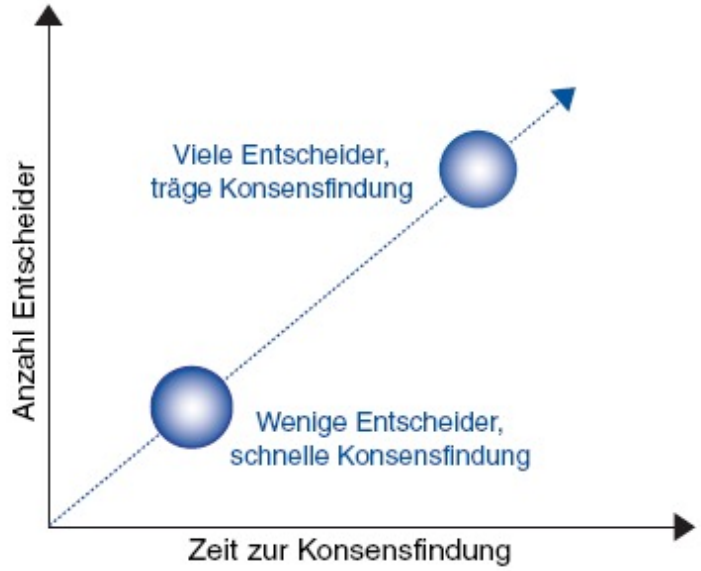
Stakeholder



Stakeholder

- Anteilhalter mit Interesse am Projekt → Anforderungsquelle
- Jede Stakeholder-Gruppe hat eigene Sicht auf das Projekt und eigene Fachsprache
- Interessenkonflikte erfordern Anforderungspriorisierung





Quelle: Grechenig et al. (2009)

Mögliche Stakeholder?

- Kunde, Auftraggeber, Sponsor
- Geschäftsführung, Abteilungsleitung
- Legacy Owner (Besitzer des Altsystems oder des Datenbestands)
- Betreiber von integrierten Schnittstellen
- Anwender
- Projektleiter, Projektsteuergruppe
- IT-Strategie, Architekt
- Entwickler
- Qualitätssicherung und Test
- Betrieb und Wartung

Weitere Anforderungsquellen

- Standards und Normen, Datenschutz, Gesetze
- Konkurrenten und deren Funktionalitäten

Quelle: Grechenig et al. (2009)

Softwareprojekte hängen von den engagierten Menschen ab

Studie über Erfolgsfaktoren für ein Softwareprojekt*

Citations of three engineering vice presidents of major technology companies:

I guess if you had to pick one thing out that is most important in our environment,
*I'd say **it's not the tools that we use, it's the people**. (VP 1)*

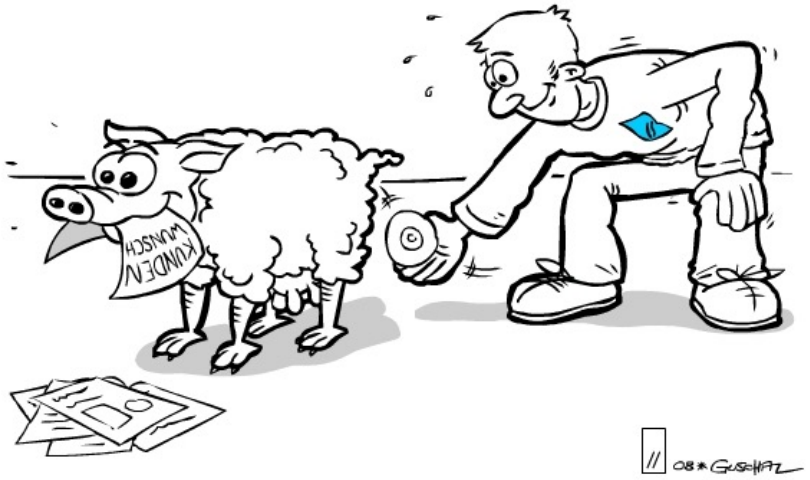
*The most important ingredient that was successful on this project was **having smart people**. … very little else matters in my opinion. ...*
*The most important thing you do for a project is selecting the staff. The success of the software development organization is very, very much associated with **the ability to recruit good people**. (VP 2)*

*The only rule I have in management is to ensure I have good people – **real good people** – and that I **grow good people** – and that I provide an environment in which good people can produce. (VP 3)*

* Curtis et al.: A Field Study of the Software Design Process for Large Systems, IEEE Trans. Software Engineering, vol. SE-31, 1988.

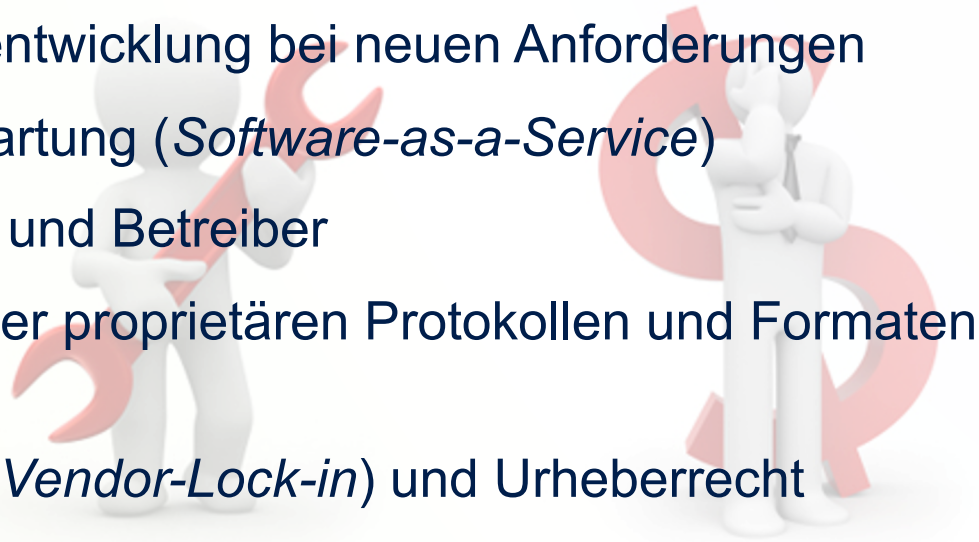
Untersuchung der Machbarkeit

- Identifikation des Inventionsanteils (Risiko, Forschungsförderung)
- Identifikation von ähnlichen Systemen (Konkurrenten)
- Technische Modellierung und Prototypen
- Rechtliche Prüfung (ggf. landesweit, europaweit, weltweit)
- Akzeptanzanalyse bei zukünftigen Anwendern
- Identifizierung der wesentlichen Kostentreiber
  (z.B. nicht-funktionale Anforderungen)
- Kostenanalyse



Entscheidungsfaktoren bezüglich Make or Buy?

- Individuelle Anpassung an Projektanforderungen (*Customizing*)
- Entwicklungskosten vs. Lizenzkosten (Anschaffung und laufender Betrieb)
- Anpassung und Weiterentwicklung bei neuen Anforderungen
- Betrieb, Support und Wartung (*Software-as-a-Service*)
- Schulung für Entwickler und Betreiber
- Nutzung von offenen oder proprietären Protokollen und Formaten (*Open Source*)
- Herstellerabhängigkeit (*Vendor-Lock-in*) und Urheberrecht
- IT-Strategie, z.B. Weiterverkauf einer Neuentwicklung



Ziele der Anforderungsanalyse

- **Erkennen der tatsächlichen Anforderungen** der unterschiedlichen Stakeholder an ein Softwaresystem
- **Erfassen der Anforderungen in Dokumenten und Modellen**, die ihre korrekte Erfüllung in der zu konstruierenden Software zusichern

Ergebnisse der Anforderungsspezifikation

- Anforderungsspezifikation (z.B. Pflichtenheft oder Product Backlog)
- Prototypen
- Objektorientierte Modelle (z.B. UML, EPK, BPMN)

DUALE HOCHSCHULE SH

in Trägerschaft der Wirtschaftsakademie Schleswig-Holstein

Anforderungsanalyse und Spezifikation

Die Bedeutung im Entwicklungsprozess

Prof. Dr. Malte Prieß

Agenda (Lernziele)

I. Stakeholder, Peopleware, Machbarkeitsstudie, Make or Buy, Ziele und Ergebnisse
II. Die Bedeutung im Entwicklungsprozess
III. Die Anforderungsanalyse
IV. Begriffslexikon und Begriffsmodell
V. Die Anforderungsspezifikation
VI. Die Darstellung der Spezifikation
    a. Anforderungsspezifikation (z.B. Pflichtenheft oder Product Backlog)
    b. Prototypen
    c. Modelle (z.B. UML, EPK, BPMN)

*„If you don‘t know where you‘re going,
you‘re unlikely to end up there.“*

Forrest Gump

Stellenwert der Anforderungen

The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements … No other part of the work so cripples the resulting system if done wrong. No other part is as difficult to rectify later.

Fred Brooks, 1987

Die Anforderungen des Kunden an die Software sind **die wichtigsten Informationen in einem Software-Projekt**.

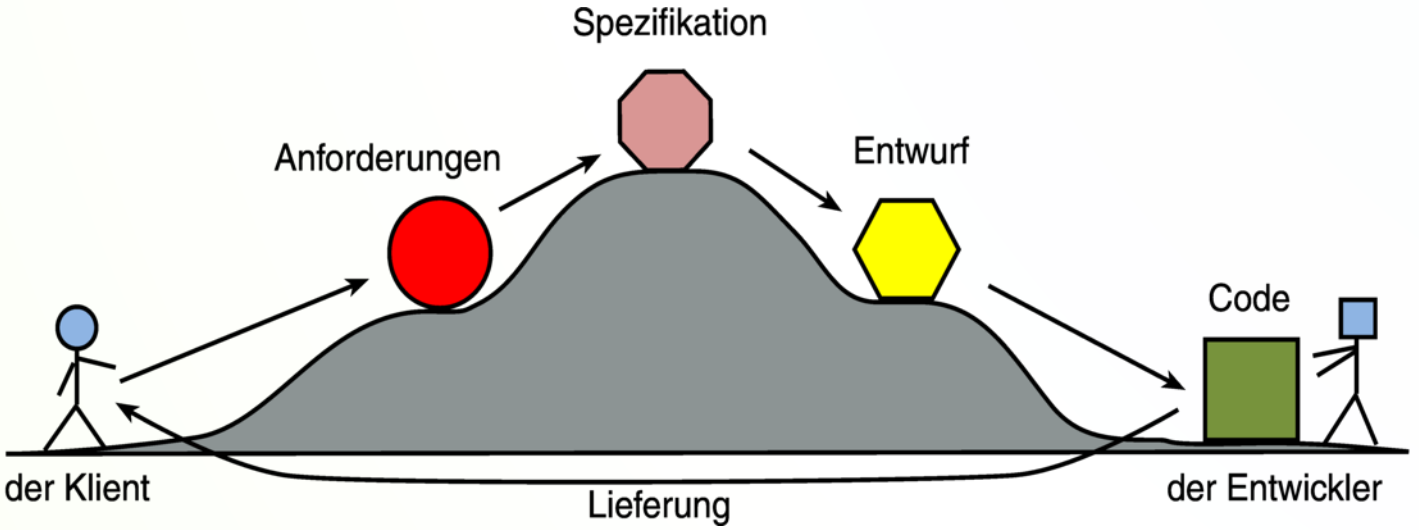
Quelle: Ludewig, Lichter, (2010), S.353ff

Kunde und Anforderungen

- Der Kunde erwartet, dass ihm die Software über einen gewissen Zeitraum hinweg als **williger und billiger Diener** zur Verfügung steht, ihm also
  - bestimmte **Leistungen erbringt**,
  - **ohne** von ihm umgekehrt **erhebliche Leistungen** (in Form von Kosten, Aufwand, Mühe, Arger) **zu fordern**. Die Software soll ihm dienen, nicht umgekehrt.
- Werden die Erwartungen, die Anforderungen des Kunden, nicht vollständig und präzise erfasst, ist damit zu rechnen, dass das entwickelte Produkt die Anforderungen nicht (vollständig) erfüllt
- Die **vollständige und präzise Erfassung der Anforderungen** ist die **allerwichtigste technische Voraussetzung** für eine erfolgreiche Software-Entwicklung

Quelle: Ludewig, Lichter, (2010)

Die Anforderungen werden **erhoben**, in der Spezifikation **formuliert**, **geprüft** und anschließend in den Entwurf **umgesetzt**. Schließlich wird **implementiert**, auf verschiedenen Ebenen **geprüft** und **korrigiert**. Das Resultat (**klassisch**: Fertige Software; **agil**: Inkrement) geht **zurück an den Klienten**.



Der Klient bekommt nur dann, was er haben will, wenn seine Anforderungen **sorgfältig** erhoben und unterwegs **nicht verfälscht** wurden.

Quelle: Ludewig, Lichter, (2010), S.354

Der Nutzen der Spezifikation

In der Praxis gibt es viele schlechte Spezifikationen; oft gibt es gar keine.

Die Spezifikation ist aber notwendig für

1. die **Abstimmung** mit dem **Kunden** bzw. mit dem Marketing,
2. den **Entwurf** und die **Implementierung**,
3. das **Benutzungshandbuch**,
4. die **Testvorbereitung**,
5. die **Abnahme**,
6. die **Wiederverwendung**,
7. die **Klärung** späterer Einwände, Regressansprüche usw.,
8. eine spätere **Re-Implementierung**.

Quelle: Ludewig, Lichter, (2010), S.355

Nachteile bei fehlender Spezifikation

1. Die Anforderungen bleiben **ungeklärt**, sie werden darum auch nicht erfüllt
2. Den Entwicklern **fehlt die Vorgabe**, darum fragen sie „auf dem kurzen Dienstweg“ Bekannte, die beim Kunden arbeiten, oder sie legen mangels Kontakten die eigenen Erfahrungen und Erwartungen zu Grunde
3. Die **Basis für das Handbuch fehlt**, es wird darum phänomenologisch, d.h. experimentell, verfasst
4. Ein **gutes Handbuch** ist ein umformulierter Auszug aus der Spezifikation! Darum **taugt** es auch **als Spezifikation**
5. Ein **systematischer Test** ist ohne Spezifikation unmöglich, denn es ist nicht definiert, welche Daten das System akzeptieren muss und welche Resultate es liefern soll

Quelle: Ludewig, Lichter, (2010), S.355, 366

Nachteile bei fehlender Spezifikation

6. Wenn bei der **Abnahme** nicht entschieden werden kann, ob das System richtig arbeitet, wird die Korrektheit zur **Glaubensfrage**
7. Oft zeigen sich **echte oder vermeintliche Mängel** der Software erst nach längerem Gebrauch. Ohne Spezifikation kann diese Unterscheidung aber nicht getroffen werden
8. Wer eine Software(-Komponente) **wiederverwenden** will, muss wissen, was sie leistet. Das ist in der Spezifikation dokumentiert
9. Wenn ein System ausgemustert und ersetzt wird, ist **Aufwärts-kompatibilität** gefordert (vgl. Heninger et al., 1978)

„Spezifikation im Kopf“ gibt es nicht!

Quelle: Ludewig, Lichter, (2010), S.355, 366

DUALE HOCHSCHULE SH

in Trägerschaft der Wirtschaftsakademie Schleswig-Holstein

Anforderungsanalyse und Spezifikation

Anforderungen

Prof. Dr. Malte Prieß

Typische Probleme bei Anforderungen

- **Verständlichkeit**
  Anforderungen werden in der Fachsprache der Domain repräsentiert,
  die von Softwareingenieuren nicht verstanden wird.
  Fachspezialisten sind ihrer Domain so vertraut, dass sie bestimmte Anforderungen als selbstverständlich erachten und nicht explizit kommunizieren.
- **Wenig Präzision**
  Es ist schwierig, Anforderungen präzise zu definieren, ohne diese schwer lesbar zu gestalten.
- **Vermischung**
  Verschiedene Funktionale und nicht-funktionale Anforderungen werden häufig vermischt, ohne diese genau zu identifizieren und voneinander abzugrenzen.
- **Mehrdeutigkeit**
  Spezifikationen in natürlicher Sprache sind oft mehrdeutig und zu flexibel, um exakte Beschreibungen zu erzeugen.

Was sind gute Anforderungen?

- **Vollständig**
  Alle Anforderungen sind explizit beschrieben → keine impliziten Annahmen eines Stakeholder über das zu entwickelnde System.
- **Eindeutig definiert/abgegrenzt**
  Präzise Definitionen helfen, Missverständnisse zwischen Stakeholdern (z.B. Entwickler und Auftraggeber) zu vermeiden.
  Keine Widersprüche in den Anforderungen untereinander.
- **Verständlich beschrieben**
  Alle Stakeholder können unter vertretbarem Aufwand die gesamte Anforderung lesen und verstehen.
- **Identifizierbar**
  Jede Anforderung soll über eine ID eindeutig identifizierbar sein.
- **Atomar**
  Es soll nur eine Anforderung pro ID beschrieben sein.

Was sind gute Anforderungen?

- **Einheitlich dokumentiert**
  Anforderungen und ihre Quellen sollen nicht in unterschiedlichen Dokumenten stehen oder unterschiedliche Strukturen haben.
- **Notwendig**
  Klären, ob Anforderungen unabdingbar sind.
- **Nachprüfbar**
  Die Anforderungen sollen mit Testfällen verknüpft werden, damit bei Abnahme auch geprüft werden kann, ob die Anforderungen erfüllt sind.
- **Rückwärts und vorwärts verfolgbar**
  Es ist erkennbar, ob jede Anforderung vollständig erfüllt wurde.
  Für die implementierten Funktionalitäten ist erkennbar, aus welchen Anforderungen sie resultieren.
- **Priorisiert**
  Prioritäten steuern die Reihenfolge für den Entwicklungsprozess.

Stabile Anforderungen

- Anforderungen, die sich nicht oder nur geringfügig ändern

Volatile Anforderungen

- Anforderungen, die sich voraussichtlich zur Entwurfs- oder Betriebszeit des Systems ändern

Auswirkungen auf die Entwurfsphase

- Stabile Anforderungen können früh und detailliert modelliert und *„hard-wired“* implementiert werden
- Volatile Anforderungen erfordern ein adaptiven Entwurf, wobei bewährte **Entwurfsmuster** helfen, zukünftige Änderungen und Erweiterungen einfacher integrieren zu können (*„soft-wired“*)

Nachverfolgbarkeit der Quelle

- Zusammenhang zum Stakeholder, der eine Anforderung wann und wie eingebracht hat

Nachverfolgbarkeit zu anderen Anforderungen

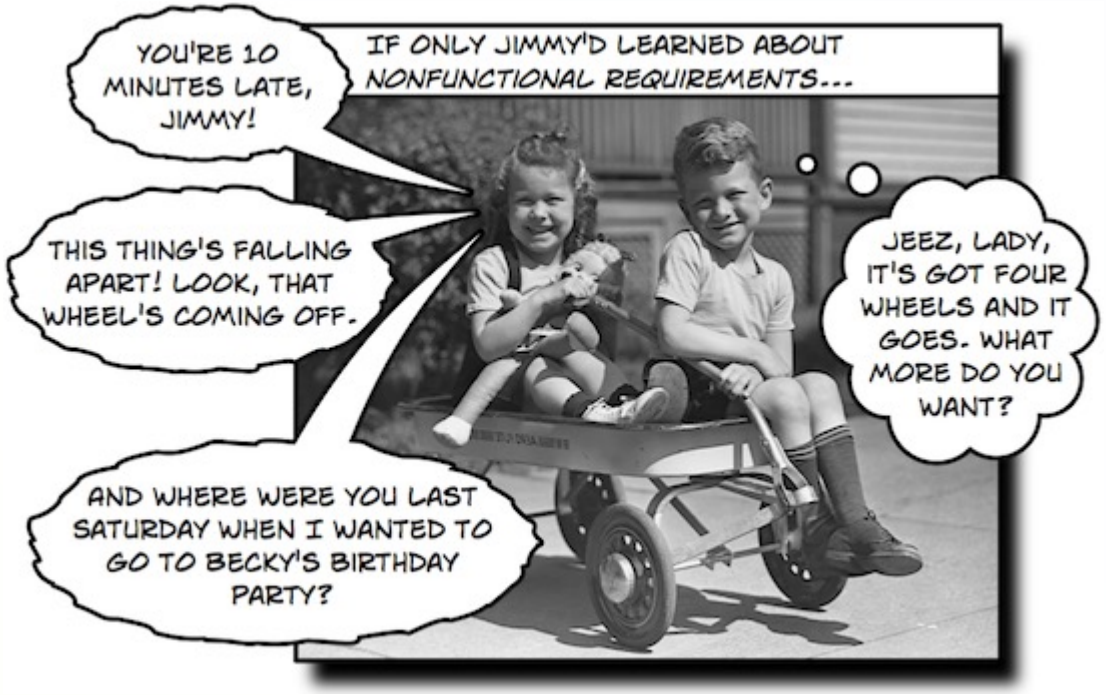
- Zusammenhänge zwischen voneinander abhängigen Anforderungen

Nachverfolgbarkeit zu abhängigen Artefakten

- Zusammenhänge der Anforderungen zu Artefakten nachgelagerter Phasen, die eine Anforderung umsetzen
  - Modelle des Entwurfs
  - Quelltext der Implementierung
  - Testfälle des Tests

Funktionale Anforderungen

- Beschreibung der **fachlichen** Funktionen/Services, die das System anbietet
- Spezifikation, **wie** das **System** auf spezifische Eingaben und Situationen **reagiert**

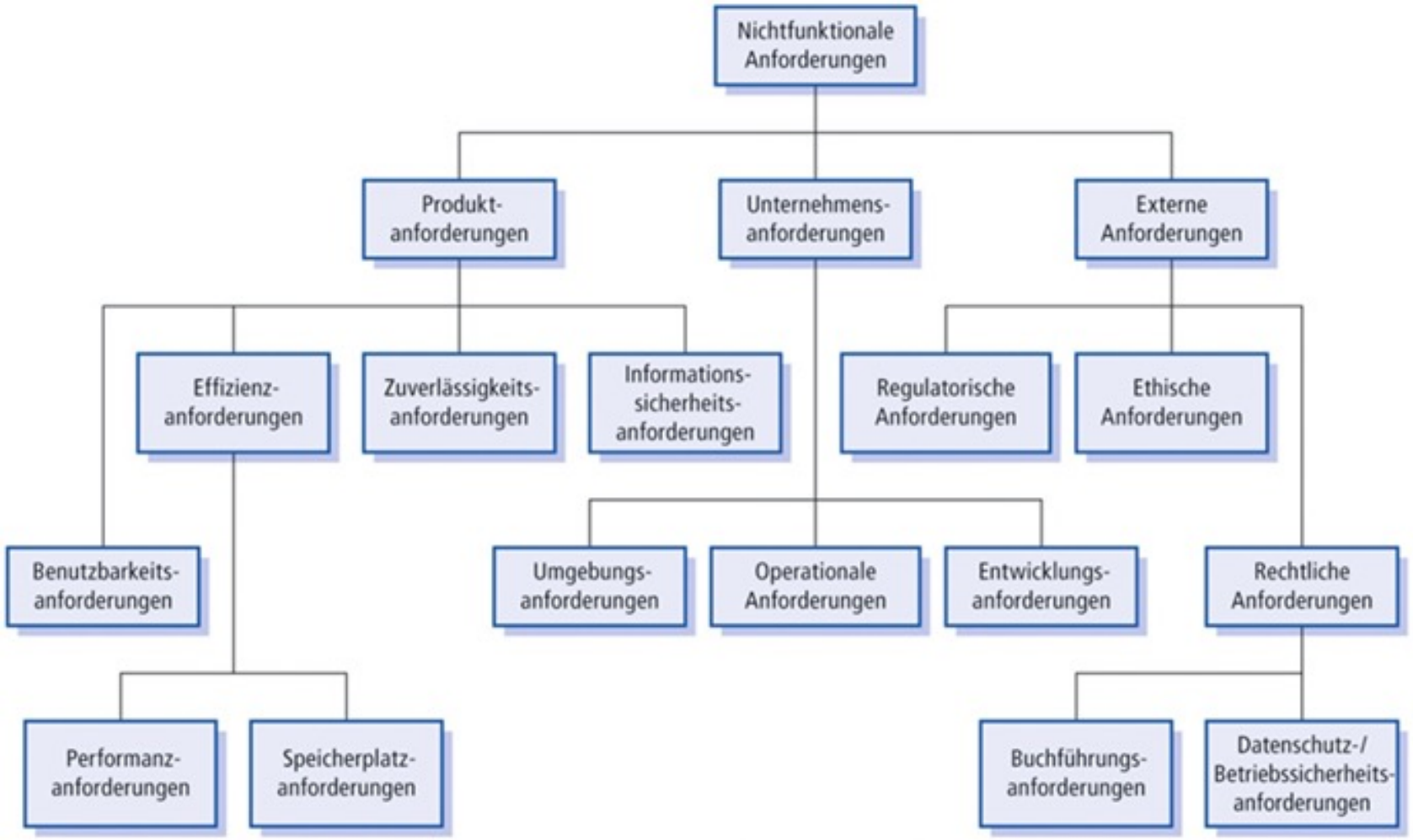


Nicht-funktionale Anforderungen (NFA)

» Anforderungen an die Funktionen/Services des Systems,
die **nicht deren fachliche Funktionen/Services betreffen**, sondern

- Qualitätseigenschaften (z.B. nach der ISO/IEC 25000),
- den organisatorischen Entwicklungsprozess,
- einzuhaltende Standards, Normen und gesetzliche Rahmen-bedingungen

Nicht-funktionale Anforderungen



Quelle: Sommerville (2012)

Funkt. versus nichtfunktionale Anforderungen

- Die Kurzbeschreibung eines Systems, also eine auf wenige Worte reduzierte Spezifikation, ist stets eine grobe Charakterisierung der Funktion.

  Beispiel: **„Das System sichert nachts die Inhalte aller Festplatten.“**

- Die **Abgrenzung** funktional – nichtfunktional ist **unscharf**:

  Anforderungen, die zwar die Funktion betreffen, aber nicht präzise formuliert werden können, werden vielfach als nichtfunktional eingeordnet

  - Beispiele: **Robustheit**, **Bedienbarkeit**, **Bedienoberfläche**

    „Das Kontoführungsprogramm soll den Kontostand auf irgendeine Weise in gut lesbarer, übersichtlicher Form anzeigen.“

  - Auch das Zeitverhalten wird meist als NFA behandelt

- Funktionale **und** nichtfunktionale Anforderungen **werden gebraucht**
- Praktisch alle Aussagen zur Wartbarkeit sind nichtfunktional

  „Das System muss leicht portierbar sein.“

Quelle: Ludewig, Lichter, (2010), S.369

Nichtfunktionale Anforderungen

- Die Formulierung nichtfunktionaler Anforderungen bereitet uns große Probleme
- Die NFRs werden entsprechend stiefmütterlich behandelt, meist ganz weggelassen oder nur durch Schlagwörter ausgedrückt

  Das ist aber ihrer großen Bedeutung nicht angemessen
- NFRs lassen sich am besten mit Hilfe von Normen spezifizieren
- Vgl. **DIN EN ISO 9241**
  - Teile 1 bis 9: Ergonomie, Technik der Dialoggestaltung
  - Teil 10: Grundsätze der Dialoggestaltung
  - Teile 11 bis 17: Details der Dialoggestaltung
- Allerdings bietet **keiner** der Standards **harte Vorschriften**, deren Einhaltung einfach und objektiv zu prüfen wäre
- In keinem Falle sollte man auf einen Versuch zur Präzision verzichten

Quelle: Ludewig, Lichter, (2010), S.370

Beispiele für NFR

Schlagwort	bessere Anforderung
einfach bedienbar	Das Programm soll auch von Laien ohne weitere Einweisung benutzt werden können.
robust	Eine Bedienung über die Tastatur darf unter keinen Umständen zu einem irregulären Abbruch des Programms führen.
portabel	Das Programm muss von einem Entwickler in höchstens 6h von Windows XP auf Linux portiert werden können.
übersichtlich kommentiert	Die Module des Programms müssen einen Kopfkommentar nach Std. xyz enthalten.
plattformunabhängig	Alle prozessorspezifischen Teile des Programms müssen in einem speziellen Modul liegen.
nicht zu große Module	Module dürfen max. 300 Zeilen ausführbaren Code enthalten.
angemessenes Handbuch	Das Benutzungshandbuch wird gemäß Richtlinie ABC aufgebaut. Es wird vom Gutachter N.N. geprüft.

Quelle: Ludewig, Lichter, (2010), S.370

Anforderungen zur Benutzbarkeit

Aspekt	Erklärung	Beispiel
Metapher, Benutzungs-modell	Das System sollte dem Benutzer eine bekannte Metapher anbieten, die es gestattet, die Operationen am System und Änderungen im System nachzuvollziehen. Die Operationen müssen so realisiert werden, dass sie mit der Metapher konsistent sind.	Informationen, die der Nutzer verwaltet, werden als logische Objekte behandelt, die eingegeben, angezeigt und gelöscht werden können (Prinzip des Zettelkastens). Der Nutzer hat jederzeit eine Vorstellung, wo sich die Information befindet, z. B. im Arbeitsbereich oder in der Datenbank.
Übersicht-lichkeit	Die Benutzungsoberfläche sollte so gestaltet sein, dass der Nutzer alle relevanten Informationen leicht erkennt.	Irrelevante Informationen werden nicht angezeigt; alle angezeigten Informationen sind logisch gruppiert.
Konsistenz	Gleiche oder ähnliche Operationen sollten auch mit gleichen oder ähnlichen Eingaben ausgelöst werden; ähnliche Inhalte sollten ähnlich dargestellt werden.	Eine bestimmte Tastenkombination hat stets die gleiche Bedeutung, z. B. Löschen der markierten Information. Die geschätzte Ausführungszeit längerer Operationen wird stets mit den gleichen Symbolen angezeigt.

Quelle: Ludewig, Lichter, (2010), S.371

Anforderungen zur Benutzbarkeit

Aspekt	Erklärung	Beispiel
Sicherheit	Die Bedienung sollte so angelegt sein, dass der Nutzer irrtümlich keinen Schaden anrichtet.	Operationen, die den internen Zustand verändern, werden erst wirksam, wenn der Nutzer die Änderung bestätigt hat.
Ästhetik	Die Benutzungsoberfläche sollte nach dem Empfinden des Nutzers schön (angenehm) sein.	Unangenehme Kontraste (z. B. durch gelbe Schrift auf blauem Grund) werden nicht erzeugt, es sei denn, sie signalisieren eine besonders kritische Situation.
Effizienz	Der Aufwand für die Bedienung sollte so gering wie möglich sein.	Es genügt, wenn der Nutzer einmal sein Passwort eingibt, um eine Folge von Operationen auszulösen, die jeweils ein Passwort erfordern.
Erlernbarkeit	Dem Nutzer sollte es leicht fallen, die Bedienung des Systems zu erlernen.	Das System verhält sich ähnlich wie andere Systeme, die dem Nutzer bereits vertraut sind. Alle Unklarheiten des Nutzers werden durch Hilfe-Funktionen geklärt.

Quelle: Ludewig, Lichter, (2010), S.371

Anforderungen an Werkzeuge für das Anforderungsmanagement?

- Verwaltung der Anforderungen in einer Datenbank
- Kollektives Editieren der Anforderungen
- Unterstützung für Reviews und Kommentare
- Versionierung der Anforderungen
- Import und Export verschiedener Dokumentenformate
- Verknüpfung von Entwurfsmodellen und anderen Dokumenten
- Integration mit weiteren Werkzeugen, z.B. Versionsverwaltung der Implementierung, Test- und Buildmanagement, Ticketsysteme
- Selektive Notifikationen der Stakeholder bei Anforderungsänderungen
- Nachverfolgbarkeit (*Traceability*) von Anforderungen
- Verwaltung von Eigenschaften (z.B. Prioritäten, Aufwandsschätzungen)
- Projektglossar

DUALE HOCHSCHULE SH

in Trägerschaft der Wirtschaftsakademie Schleswig-Holstein

Anforderungsanalyse und Spezifikation

Die Anforderungsanalyse

Prof. Dr. Malte Prieß

Agenda (Lernziele)

I. Stakeholder, Peopleware, Machbarkeitsstudie, Make or Buy, Ziele und Ergebnisse
II. Die Bedeutung im Entwicklungsprozess
III. Die Anforderungsanalyse
IV. Begriffslexikon und Begriffsmodell
V. Die Anforderungsspezifikation
VI. Die Darstellung der Spezifikation
    a. Anforderungsspezifikation (z.B. Pflichtenheft oder Product Backlog)
    b. Prototypen
    c. Modelle (z.B. UML, EPK, BPMN)

requirement — *(1) A condition or capability needed by a user to solve a problem or achieve an objective.*
(2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
(3) A documented representation of a condition or capability as in (1) or (2).

requirements analysis — *(1) The process of studying user needs to arrive at a definition of system, hardware, or software requirements.*
(2) The process of studying and refining system, hardware, or software requirements.

IEEE Std 610.12-1990

Lastenheft /Anforderungssammlung

- Die Analyse ist die Vorarbeit und Voraussetzung der Spezifikation; ihr Ergebnis wird im klassischen Vorgehen auch als **Lastenheft** bezeichnet
- Obwohl es Definitionen für diesen Begriff gibt (z.B. in DIN 69905), bleibt er in der Praxis unscharf, weil der **Inhalt des Lastenhefts** von Firma zu Firma **stark variiert**
- Lastenheft (oder **Anforderungssammlung)**
  - Beschreibt die **fachlichen Anforderungen aus Klientensicht**
  - Lücken, Unklarheiten und Widersprüche sind darin normal
- Erst die Anforderungsspezifikation sollte **vollständig, klar und konsistent** sein

Das Begriffslexikon

- Bei der Analyse wird bei Bedarf auch das **Begriffslexikon** angelegt; dieses wird während der gesamten Software-Entwicklung verwendet und ergänzt
- Begriffslexikon: **Häufig verwendete Begriffe und deren Bedeutung**
- Oft kommt während der Entwicklung oder bei der Abnahme die Frage auf, **woher** eine Anforderung gekommen ist

  → Konsequent **dokumentieren**, welcher Klient hinter welcher Anforderung steht
- Diese Zuordnung bei Besprechungen **regelmäßig überprüfen**

Quelle: Ludewig, Lichter, (2010), S.357, 358

Die Ist-Analyse

- Ziel der Analyse: **Soll-Zustand** feststellen
- Es sollte also genügen, die Klienten nach ihren Wünschen zu fragen
- **Aber**: Die Klienten konzentrieren sich auf das, was ihnen derzeit **nicht gefällt**
- Wir sind also bei jeder Umstellung auf die **Schwachpunkte des bestehenden Systems** fixiert
- Seine Stärken nehmen wir erst wahr, wenn sie uns fehlen

Implizit erwarten die Klienten, dass alles, was bisher akzeptabel oder gut war, unverändert bleibt oder besser wird. Die Anforderungen an das neue System bestehen also nur zum kleinsten Teil aus Änderungswünschen, **die allermeisten Anforderungen gehen in Richtung Kontinuität**.

Quelle: Ludewig, Lichter, (2010), S.358, 359

Die Ist-Analyse

- Darum hat die Ist-Analyse, also die Feststellung des bestehenden Zustands, **große Bedeutung** für die erfolgreiche Projektdurchführung
- Bei **Widersprüchen** ist der Ist-Zustand zudem ein **sicherer Halt**
- Der Analytiker muss sich also **gut einfühlen** und **genau beobachten**, um vom Kunden zu erfahren, welche Aspekte des Ist-Zustands für ihn wichtig sind und beibehalten werden sollten.
- Die Ist-Analyse ist eine **undankbare Tätigkeit**

Quelle: Ludewig, Lichter, (2010), S.358, 359

Die Ist-Analyse ist eine undankbare Tätigkeit

Sie öffnen also morgens das Schloss am Haupteingang?

Ja, habe ich Ihnen doch gesagt.

Jeden Morgen?

Natürlich.

Auch am Wochenende?

Nein, am Wochenende bleibt der Eingang zu.

Und während der Betriebsferien?

Da bleibt er natürlich auch zu.

Und wenn Sie krank sind oder Urlaub haben?

Dann macht das Herr X.

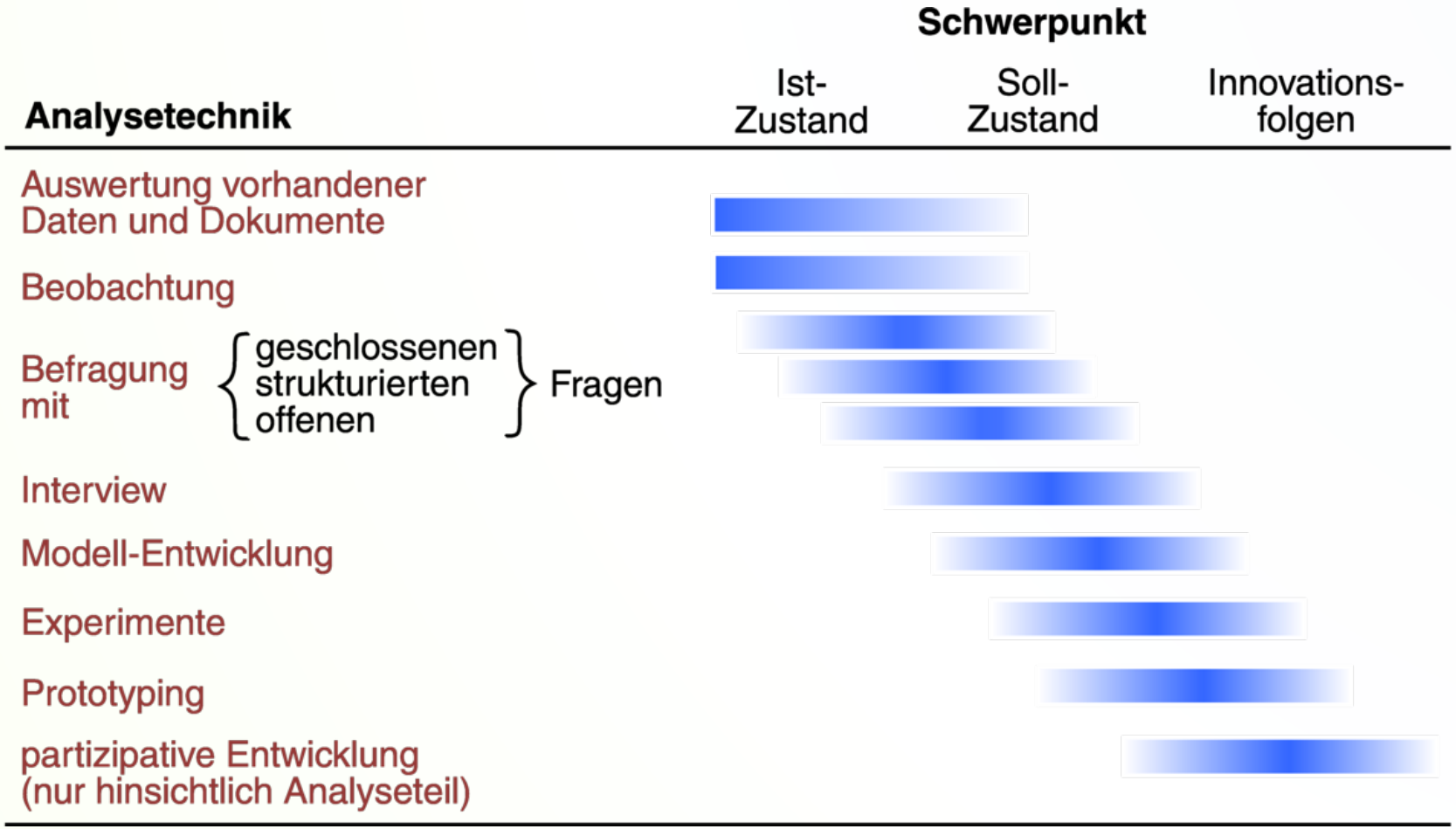
Und wenn auch Herr X ausfällt?

Dann klopft irgendwann ein Kunde ans Fenster, weil er nicht reinkommt.

Was bedeutet „morgens“? ...

Quelle: Ludewig, Lichter, (2010), S.359

Techniken der Analyse



Quelle: Ludewig, Lichter, (2010), S.361

DUALE HOCHSCHULE SH

in Trägerschaft der Wirtschaftsakademie Schleswig-Holstein

Anforderungsanalyse

Begriffslexikon und Begriffsmodell

Prof. Dr. Malte Prieß

Agenda (Lernziele)

I. Stakeholder, Peopleware, Machbarkeitsstudie, Make or Buy, Ziele und Ergebnisse
II. Die Bedeutung im Entwicklungsprozess
III. Die Anforderungsanalyse
IV. Begriffslexikon und Begriffsmodell
V. Die Anforderungsspezifikation
VI. Die Darstellung der Spezifikation
    a. Anforderungsspezifikation (z.B. Pflichtenheft oder Product Backlog)
    b. Prototypen
    c. Modelle (z.B. UML, EPK, BPMN)

Das Begriffslexikon

- In der Analyse kann bei Bedarf ein **Begriffslexikon** angelegt werden
- In den frühen Phasen wird dieses Dokument **aufgebaut und weiterentwickelt**
- Das Begriffslexikon enthält solche Begriffe, die
  - **wichtig** sind und
  - von verschiedenen Leuten, v.a. von Klienten, Analytikern und Entwicklern, **unterschiedlich ausgelegt werden** könnten.
- Dazu gehören häufig **Begriffe, die auf den ersten Blick völlig klar zu sein scheinen**
- Beispiel: Informationssystem für die Prüfungsdaten von Studenten

  → Begriffe wie etwa „Student“, „Prüfung“, „Note“, „Erstprüfung“

Quelle: Ludewig, Lichter, (2010), S.364

Informationen im Begriffslexikon

a) Begriff und Synonyma (im Sinne der Spezifikation)
b) Bedeutung (Definition, Erklärung)
c) Abgrenzung (wo ist dieser Begriff nicht anzuwenden?)
d) Gültigkeit (zeitlich, räumlich, sonst)
e) Fragen der Bezeichnung, Eindeutigkeit u. A.
f) Unklarheiten, die noch nicht beseitigt werden konnten
g) verwandte Begriffe (Querverweise)

» Die Angaben werden **aus den Gesprächen und Interviews abgeleitet** oder, wenn sie von den Analytikern kommen, **sorgfältig mit den Klienten überprüft**
» Typische Quellen (Projekt Universität): Angestellten in der Verwaltung, die Juristen der Uni, die Gesetze und Ordnungen der Universität

Quelle: Ludewig, Lichter, (2010), S.364

Beispiel

Begriff	**Student**, synonym Studentin, Studierender, Studierende
Bedeutung	Eine Person, die an der Universität Stuttgart immatrikuliert ist und noch nicht exmatrikuliert wurde, die folglich legal einen Studentenausweis der Universität Stuttgart hat oder haben könnte.
Abgrenzung	Gasthörer und Studierende anderer Hochschulen sind im Sinne dieses Systems keine Studenten.
Gültigkeit	Mit der Immatrikulation an der Universität Stuttgart entsteht ein neuer Student; er existiert bis zur Exmatrikulation, gleichgültig, wie sie zustande kommt. Ein Fachwechsel oder eine Namensänderung implizieren keine Exmatrikulation. Hat sich eine Person im Laufe ihres Lebens mehrfach an der Universität Stutt-gart immatrikuliert, so handelt es sich um mehrere, nicht identische Studenten.
Bezeichnung	Ein Student ist durch die Matrikelnummer und einen Zeitpunkt (zu dem die Matrikelnummer gültig war oder ist) eindeutig bestimmt, alle anderen Attribute, insbesondere der Name, können mehrfach vorkommen.
Unklarheiten	Es ist noch ungeklärt, wie Namen aus anderen Schriftsystemen (z. B. Russisch, Arabisch, Chinesisch) dargestellt werden.
Querverweise	Gasthörer, Matrikelnummer, Studentenausweis

Quelle: Ludewig, Lichter, (2010), S.365

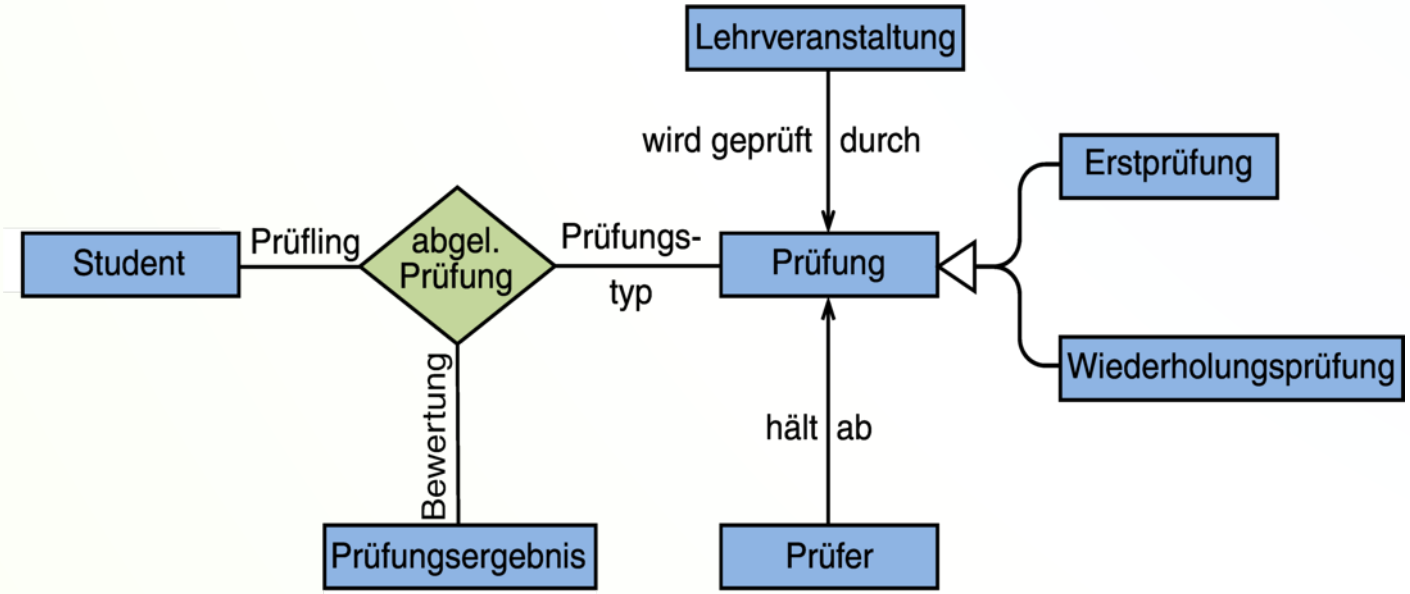
Hinweis

- Im Begriffslexikon klar unterscheiden zwischen
  - Objekten der **realen Welt**,
  - **Rollen** oder **Typen** der realen Welt und
  - **Daten** der Software
- Oft keineswegs deutlich
- Student kann meinen ...
  - ein Individuum, also einen speziellen Studenten
  - den Typ/Begriff „Student“
  - die Gruppe von Studenten
  - die Datensätze, die die Studenten repräsentieren

Quelle: Ludewig, Lichter, (2010), S.365

Begriffsmodell

- Durch Vernetzung der Begriffe entsteht ein **anwendungsfachliches Begriffsmodell**. Darin gibt es allgemeine Assoziationen, Generalisierungs- und Kompositionsbeziehungen.
- Begriffsmodelle helfen, den **Anwendungsbereich** zu erfassen und die Begriffe im Kontext zu verstehen
- Beispiel: Begriffsmodell (Ausschnitt) eines Prüfungsdaten-Verwaltungssystems



Quelle: Ludewig, Lichter, (2010), S.366

DUALE HOCHSCHULE SH

in Trägerschaft der Wirtschaftsakademie Schleswig-Holstein

Anforderungsanalyse und Spezifikation

Die Anforderungsspezifikation

Prof. Dr. Malte Prieß

Agenda (Lernziele)

I. Stakeholder, Peopleware, Machbarkeitsstudie, Make or Buy, Ziele und Ergebnisse
II. Die Bedeutung im Entwicklungsprozess
III. Die Anforderungsanalyse
IV. Begriffslexikon und Begriffsmodell
V. Die Anforderungsspezifikation
VI. Die Darstellung der Spezifikation
    a. Anforderungsspezifikation (z.B. Pflichtenheft oder Product Backlog)
    b. Prototypen
    c. Modelle (z.B. UML, EPK, BPMN)

specification — *A document that specifies, in a complete, precise, verifiable manner, the requirements, design, behavior, or other characteristics of a system or component, and, often, the procedures for determining whether these provisions have been satisfied.*

specification language — *A language, often a machine-processible combination of natural and formal language, used to express the requirements, design, behavior, or other characteristics of a system or component. For example, a design language or requirements specification language.*
Contrast with: programming language; query language.

IEEE Std 610.12-1990

Zum Vergleich: Wiederholung Anforderung

requirement — *(1) A condition or capability needed by a user to solve a problem or achieve an objective.*
(2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
(3) A documented representation of a condition or capability as in (1) or (2).

requirements analysis — *(1) The process of studying user needs to arrive at a definition of system, hardware, or software requirements.*
(2) The process of studying and refining system, hardware, or software requirements.

IEEE Std 610.12-1990

requirements specification language — *A specification language with special constructs and, sometimes, verification protocols, used to develop, analyze, and document hardware or software requirements.*

software requirements specification (SRS) — *Documentation of the essential requirements (functions, performance, design constraints, and attributes) of the software and its external interfaces.*

IEEE Std 610.12-1990

Aus den Definitionen zu **specification** und **SRS** folgt:

Die Anforderungsspezifikation dokumentiert die wesentlichen Anforderungen an eine Software und ihre Schnittstellen, und zwar präzise, vollständig und überprüfbar.

Quelle: Ludewig, Lichter, (2010), S.375

Die Spezifikation ist im Idealfall **inhaltlich**

1. **zutreffend**: Sie gibt die Vorstellungen des Kunden richtig wieder.
2. **vollständig**: Jede (in einem Kopf oder in einem Dokument) vorhandene Anforderung ist in der Spezifikation enthalten.
3. **widerspruchsfrei (oder konsistent)**: Jede Anforderung ist mit allen anderen Anforderungen vereinbar. Eine inkonsistente Spezifikation ist nicht realisierbar, denn kein System kann widersprüchliche Anforderungen erfüllen.
4. **neutral (oder abstrakt)**: Die Spezifikation schränkt die Realisierung nicht über die wirklichen Anforderungen hinaus ein.
5. **nachvollziehbar**: Die Quellen der Anforderungen sind dokumentiert, die Anforderungen sind eindeutig identifizierbar.
6. **objektivierbar**: Das realisierte System kann gegen die Spezifikation geprüft werden, auch (missverständlich) „testbar“.

Quelle: Ludewig, Lichter, (2010), S.375, 376

Die **Darstellung** und **Form** der idealen Spezifikation ist

7. **leicht verständlich**: Alle Interessenten sind in der Lage, die Spezifikation zu verstehen.
8. **präzise**: Die Spezifikation schafft keine Unklarheiten und Interpretationsspielräume.
9. **leicht erstellbar**: Die Anfertigung und Nachführung der Spezifikation sind einfach und erfordern keinen nennenswerten Aufwand.
10. **leicht verwaltbar**: Die Speicherung der Spezifikation und der Zugriff darauf sind einfach und erfordern keinen nennenswerten Aufwand.

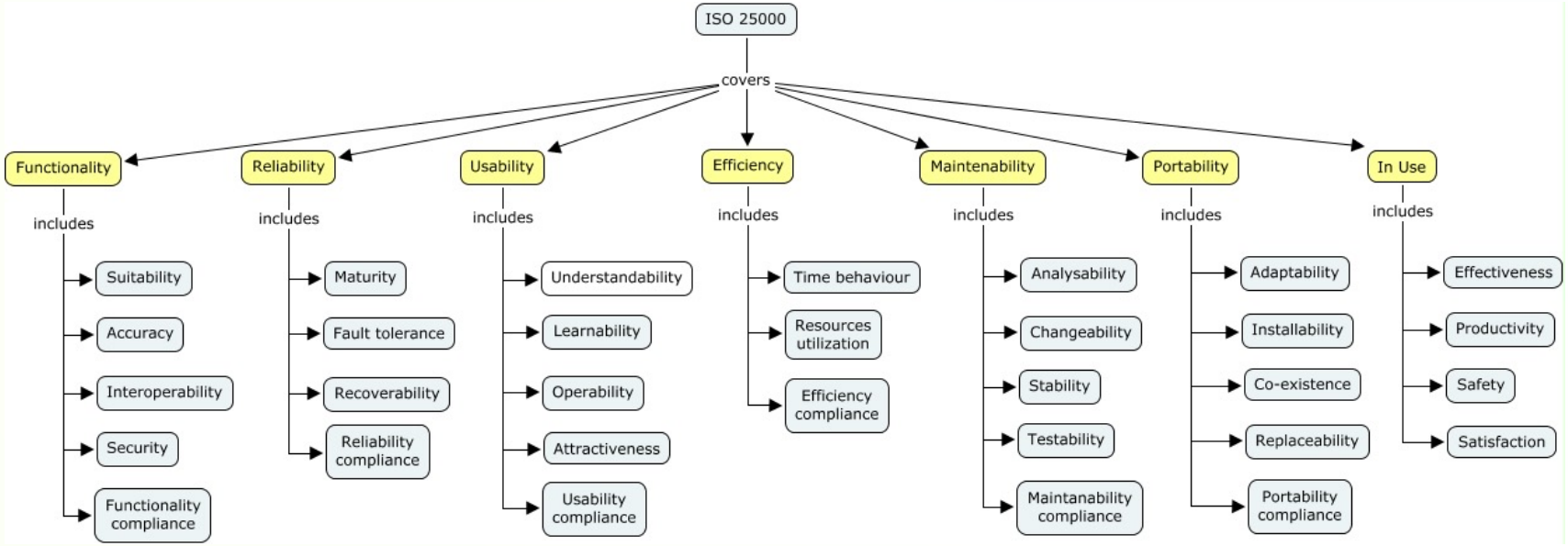
Diese Merkmale **konkurrieren**; wir suchen also einen **Kompromiss**.

Quelle: Ludewig, Lichter, (2010), S.376

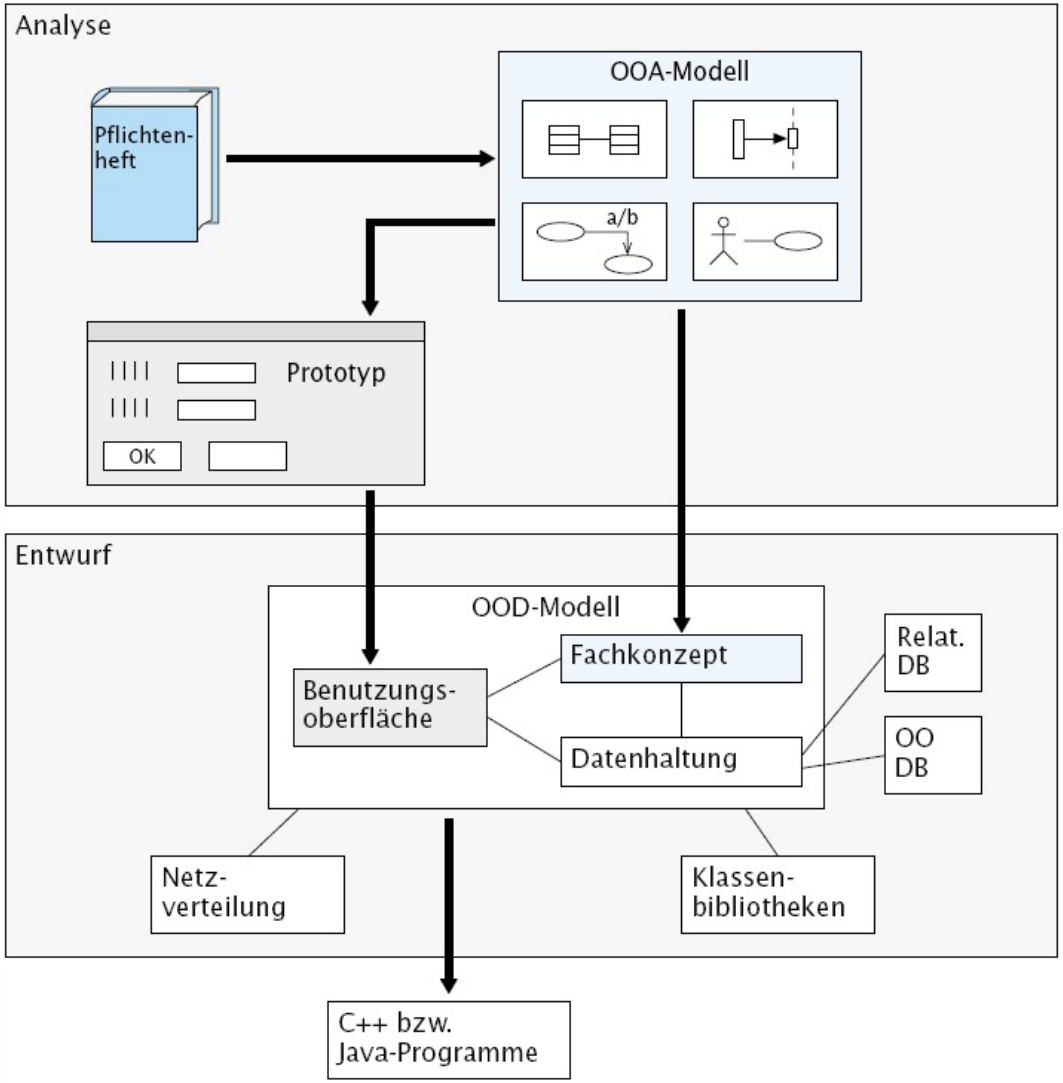
Qualität eines Softwaresystems ist nach ISO 9000 der **Grad der Anforderungserfüllung**

Qualitätsmerkmale eines Softwaresystems

ISO/IEC 25000
Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE)



Abgrenzung der Spezifikation zum Entwurf



Quelle: Balzert (2004)

Abgrenzung der Spezifikation zum Entwurf

- Die Begriffe **OOA** (objektorientierte Analyse) und **OOD** (objektorientiertes Design) sind nicht immer trennscharf zu verwenden. Als grundsätzlicher Konsens gilt:
  - OOA - **WAS** soll das Programm tun?
  - OOD - **WIE** soll das Programm es tun?
- In der **OOA** werden also **grundsätzliche Strukturen identifiziert** und nach Möglichkeit schon in Klassen-/Objektstrukturen (vor-)strukturiert. Dabei werden Fragen der technischen Implementierung VÖLLIG vernachlässigt. Als Ergebnis der OOA stehen bspw. **Use-Case-Diagramme**, **Aktivitätsdiagramme** oder (sehr **grob** gehaltene) **Klassendiagramme**.
- Beim **OOD** werden die Ergebnisse der vorigen Schritte **konkretisiert**: Wie werden einzelne Prozesse und Interaktionen realisiert? Wie wird die **Datenhaltung** realisiert, wie spielen die einzelnen Schichten (z.B. **MVC**) zusammen? Hier kommen auch Aspekte der zu verwendenden Programmiersprache ins Spiel (ist **Überladung**, **Mehrfachvererbung** möglich?). Als Ergebnis steht ein möglichst **konkretes Klassendiagramm**.

DUALE HOCHSCHULE SH

in Trägerschaft der Wirtschaftsakademie Schleswig-Holstein

Anforderungsanalyse und Spezifikation

Die Darstellung der Spezifikation

Prof. Dr. Malte Prieß

Agenda (Lernziele)

I. Stakeholder, Peopleware, Machbarkeitsstudie, Make or Buy, Ziele und Ergebnisse
II. Die Bedeutung im Entwicklungsprozess
III. Die Anforderungsanalyse
IV. Begriffslexikon und Begriffsmodell
V. Die Anforderungsspezifikation
VI. Die Darstellung der Spezifikation
    a. Anforderungsspezifikation (z.B. Pflichtenheft oder Product Backlog)
    b. Prototypen
    c. Modelle (z.B. UML, EPK, BPMN)

Ergebnisse der Anforderungsspezifikation

1. Anforderungsspezifikation

   (z.B. Pflichtenheft oder Product Backlog)

1. Prototypen
2. Modelle

   (z.B. UML, EPK, BPMN)

Lastenheft ☑

- Beschreibt die Anforderungen aus **Sicht des Auftraggebers**

Pflichtenheft ☐

- Beschreibt die Anforderungen aus **Sicht des Auftragnehmers**
- Umfasst die *„vom Auftragnehmer erarbeiteten Realisierungsvorgaben aufgrund der Umsetzung des vom Auftraggeber vorgegebenen Lastenhefts“* (DIN 69901-5)

Unterschiedliche Bezeichnungen für die Anforderungsspezifikation

- Software Requirements Specification (SRS)
- Lasten- und Pflichtenheft
- Product Backlog (Scrum)
- Fachkonzept/-dokumentation o.ä.

Unterschiedliche Vorschläge zur Gliederung

- **IEEE-Standard 830** – Software Requirements Specification
- Pflichtenheft nach dem Verband Deutscher Ingenieure (**VDI**) 3694
- **Gliederungsvorschläge** verschiedener **Lehrbuch-Autoren** (z.B. Balzert, Sommerville, Pressman)

Pflichtenheft-Gliederung nach Balzert

1. Zielbestimmung
   (Musskriterien, Wunschkriterien, Abgrenzungskriterien)
2. Produkteinsatz
   (Anwendungsbereiche, Zielgruppen, Betriebsbedingungen)
3. Produktübersicht
4. Produktfunktionen
5. Produktdaten
6. Produktleistungen
7. Qualitätsanforderungen
8. Benutzungsoberfläche
9. Nicht-funktionale Anforderungen
10. Technische Produktumgebung
    (Software und Hardware für Server und Clients, Produkt-Schnittstellen)
11. Anforderungen an die Entwicklungsumgebung
12. Gliederung in Teilprodukte

Quelle: Balzert (2004)

DUALE HOCHSCHULE SH

in Trägerschaft der Wirtschaftsakademie Schleswig-Holstein

Die Darstellung der Spezifikation

Anforderungsspezifikation – Pflichtenheft

Prof. Dr. Malte Prieß

Natürlichsprachliche Spezifikation

- Aufgrund von **Problemen** mit **formalen** und **grafischen** Spezifikationen sind **Texte** meist **das beste oder einzige Mittel zur Kommunikation** mit den Klienten
- Wie kann man die **Nachteile der Verwendung natürlicher Sprachen** vermindern oder beseitigen?
- Die Firma **SOPHIST** hat ein Regelwerk entwickelt, das diese Probleme angeht
  - Es definiert einige besonders wichtige Regeln
  - Dabei geht es immer darum, Lücken und Unschärfen zu vermeiden und deutlich zu machen, wer wann was tut

Quelle: Ludewig, Lichter, (2010), S.381, 382

Einige Sprachregeln

	Regel	Erläuterung, Beispiel
R1	Formulieren Sie jede Anforderung im Aktiv.	Der Akteur wird angegeben, und es wird sichtbar, ob das System oder der Benutzer etwas tut. Fordert man, dass etwas »*gelöscht wird*«, so bleibt das unklar.
R2	Drücken Sie Prozesse durch Vollverben aus.	Vollverben (wie »liest«, »erzeugt«, nicht »ist«, »hat«) verlangen weitere Informationen (Objekte, Ergänzungen), die den Prozess genauer beschreiben. Nicht: »*Wenn die Daten konsistent sind*«, sondern »*Wenn Programm ABC die Konsistenz der Daten geprüft hat*«. Und natürlich muss auch spezifiziert sein, was geschehen soll, wenn sie *nicht* konsistent sind (R4).
R3	Entdecken Sie unvollständig spezifizierte Prozesswörter (Verben).	Fehlen Angaben (Objekte), dann wird nach diesen Angaben gesucht, um vollständige Aussagen zu erhalten. Wenn eine Komponente *einen Fehler meldet*, fragt sich, wem.
R4	Ermitteln Sie unvollständig spezifizierte Bedingungen.	Für Bedingungen der Form »Wenn-dann-sonst« müssen sowohl der Dann- als auch der Sonst-Fall beschrieben sein (vgl. das Beispiel für R2).
R5	Bestimmen Sie die Universalquantoren.	Sind Sätze mit »nie«, »immer«, »jedes«, »kein«, »alle« wirklich universell gültig, oder gibt es Einschränkungen? Sind »*alle Personen*« wirklich alle Personen, oder nur die Anwesenden, die Mitarbeiter, die Besitzer einer Eintrittskarte?

Quelle: Ludewig, Lichter, (2010), S.382

Einige Sprachregeln

	Regel	Erläuterung, Beispiel
R6	Überprüfen Sie Nominalisierungen.	Nomen (z. B. »Generierung«, »Datenverlust«) weisen oft auf einen komplexen Prozess hin, der beschrieben werden muss. Verwendet man das Substantiv »*Anmeldung*«, dann fehlen meist die Ergänzungen, die beim Verb »*anmelden*« erwartet werden: Wer meldet sich wo und wofür an?
R7	Erkennen und präzisieren Sie unbestimmte Substantive.	Oft bleibt unklar, ob es sich um einen generischen Begriff oder um ein bestimmtes Objekt handelt. Ist vom »*Bediener*« die Rede, so fragt es sich, ob es nur einen gibt oder welcher gemeint ist. Ähnliches gilt für viele Begriffe (Gerät, Meldung).
R8	Klären Sie die Zuständigkeiten bei Möglichkeiten und Notwendigkeiten.	Ein Zwang muss realisiert werden. Steht in einer Anforderung, dass etwas *möglich* oder *unmöglich* ist, *passieren kann*, *darf* oder *muss*, so ist zu klären, wer dies erzwingt oder verhindert.
R9	Erkennen Sie implizite Annahmen.	Begriffe in den Anforderungen (»*die Firewall*«), die nicht erläutert sind, deuten oft auf implizite Annahmen (hier vermutlich auf die, dass es *eine Firewall gibt*).

Quelle: Ludewig, Lichter, (2010), S.382

Beispiel für Verletzung der Sprachregeln

»Es soll geprüft werden, dass neben Autor und Titel des Dokuments auch immer der Aufbewahrungsort eingegeben wird.«

- **R1**: Die Anforderung ist im Passiv formuliert (»... soll geprüft werden«, »... eingegeben wird«), die Akteure fehlen.
- **R7**: Es muss klar sein, welcher Autor gemeint ist. Ähnliche Fragen gibt es zum Aufbewahrungsort.
- **R3**: Die Prozesswörter »prüfen« und »eingeben« sind unvollständig spezifiziert. Wer prüft wann, wer gibt wann etwas ein?
- **R4**: In der Anforderung steckt eine unvollständig spezifizierte Bedingung. Was soll geschehen, wenn der Aufbewahrungsort nicht eingegeben wurde?
- **R5**: Es muss nachgefragt werden, ob der angegebene Sachverhalt auch wirklich immer gelten soll.

Quelle: Ludewig, Lichter, (2010), S.383

Beispiel für Verletzung der Sprachregeln

Eine verbesserte Fassung dieser Anforderung, die die Ergebnisse der sprachlichen Analyse berücksichtigt, könnte etwa so aussehen:

»*Nachdem der Administrator den Dokument-Datensatz eingegeben hat, prüft das System, ob der Datensatz neben dem (einzigen oder ersten) Autor und dem Titel auch den Aufbewahrungsort enthält. Sonst erzeugt das System eine Fehlermeldung und lässt den Dokument-Datensatz unverändert.*«

Quelle: Ludewig, Lichter, (2010), S.383

Die SOPHISTen geben dafür das **Schema A B C D E F** vor:

- **A**: klärt, wann und unter welchen Bedingungen die Aktivität stattfindet
- **B**: ist MUSS (Pflicht), SOLL (Wunsch) oder WIRD (Absicht)
- **C**: ist immer »das System« oder eine konkrete Nennung des Systems
- **D**: ist eine von drei Möglichkeiten: Die erste beschreibt eine selbständige Systemaktivität (»tut«), die zweite eine vom System angebotene Funktion (»bietet jemandem die Möglichkeit, zu tun«), die dritte die Inanspruchnahme einer von Dritten angebotenen Funktion (»ist fähig zu tun, wenn bestimmte Voraussetzungen gegeben sind«)
- **E**: enthält Ergänzungen, insbesondere ein Objekt
- **F**: ist das eigentliche Prozesswort (was passiert)

Quelle: Ludewig, Lichter, (2010), S.384

Schema A B C D E F – Beispiel

Nach Ende der Bürozeiten (**=A**) soll (**=B**) das System (**=C**) dem Operator die Möglichkeit bieten (**=D**), alle neuen Anmeldungen auf einem externen Datenträger (**=E**) zu sichern (**=F**).

- Selbständige Systemaktivität: Bei einem automatischen Backup fiele Teil D (und das Wort »zu«) weg
- Natürlich ist zu prüfen, ob im Satz noch implizite Annahmen stecken; beispielsweise scheint es definierte Bürozeiten zu geben
- Ebenso ist der Universalquantor »alle« zu prüfen

Quelle: Ludewig, Lichter, (2010), S.384

DUALE HOCHSCHULE SH

in Trägerschaft der Wirtschaftsakademie Schleswig-Holstein

Die Darstellung der Spezifikation

Anforderungsspezifikation – Product Backlog

Prof. Dr. Malte Prieß

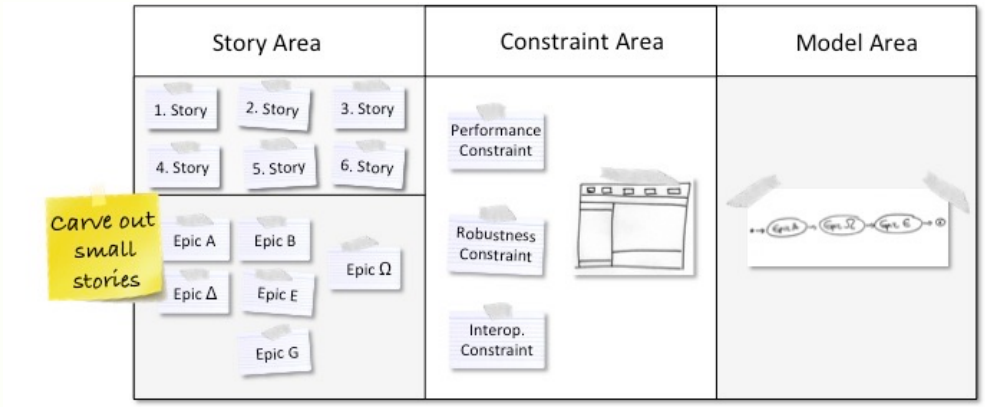
Product Backlog in Scrum

Anforderungen als **Epics**, **Constraints** und **User Stories** erfasst

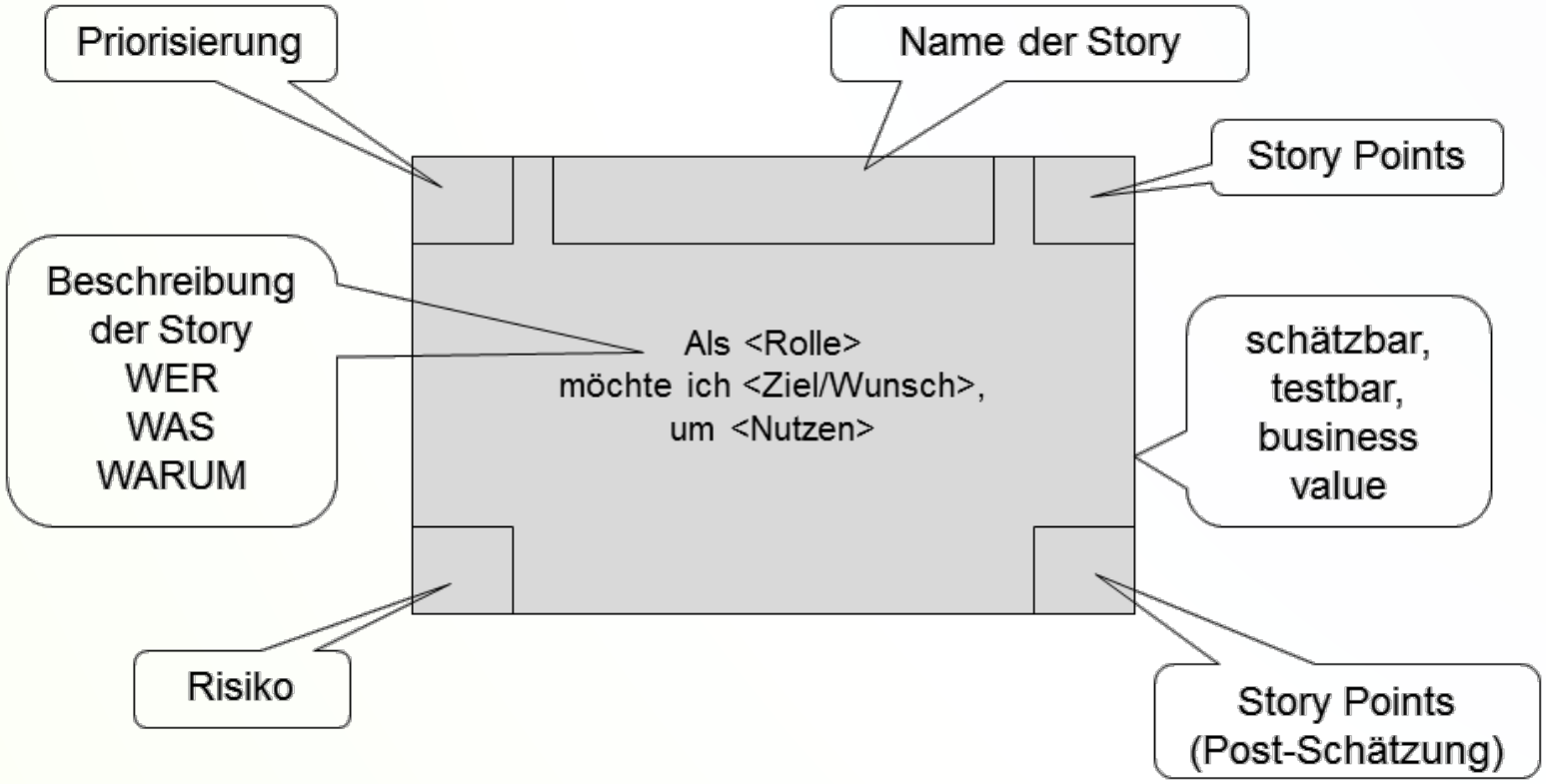
- Epic: Aggregierte, grobgranulare Sicht auf Anforderungen
- Constraint: Nicht-funktionale Anforderung
- User Story: Detaillierung des Epics auf eine in Software-Code umsetzbare Granularitätsstufe in Alltagssprache und nach folgendem Muster:

  As a <user type> I want to <do some action> so that I reach <a desired result>

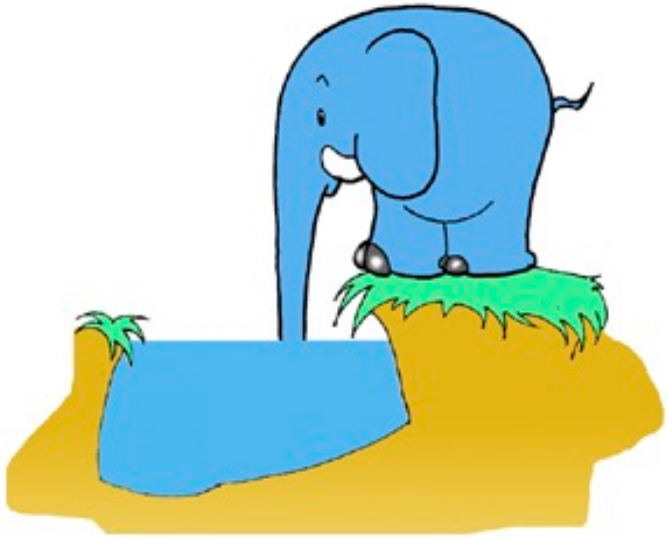
  oder kürzer: ***As a <user type> I want to reach <a desired result>***



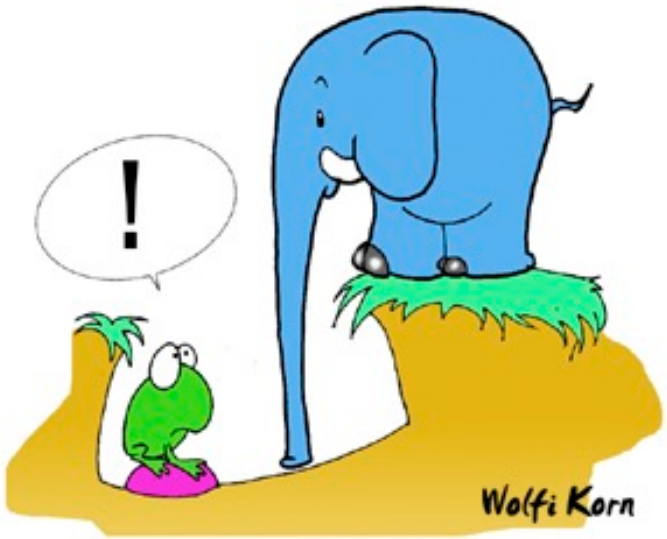
Schablone für Story Card (XP, Scrum)



1



2



DUALE HOCHSCHULE SH

in Trägerschaft der Wirtschaftsakademie Schleswig-Holstein

Die Darstellung der Spezifikation

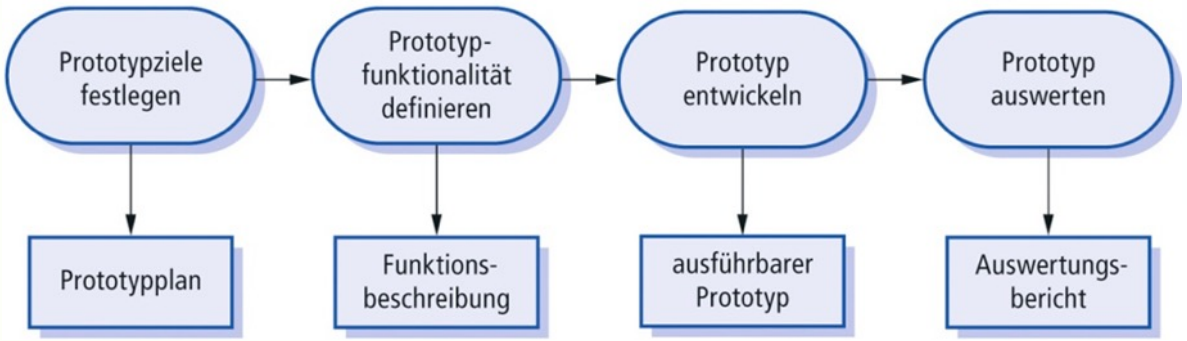
Prototypen

Prof. Dr. Malte Prieß

Unterschiedliche Ziele eines Prototyps

- Erstellung einer initialen Version
- Ermittlung und Validierung von Anforderungen
- Untersuchung des Entwurfs der Benutzeroberfläche (GUI-Mockup)
- Technische Machbarkeit einzelner Komponenten

Geplante Prototyp-Entwicklung



Quelle: Sommerville (2012)

Evolutionärer Prototyp

- Anwender werden einbezogen, um von **initialer Softwareversion** mittels Evolution zu einem **gebrauchstauglichen System** zu gelangen
- Prozess, um ein System den sich rasch verändernden Randbedingungen anzupassen
- Entwickler beginnen mit wichtigsten Funktionen, deren Anforderungen geklärt sind

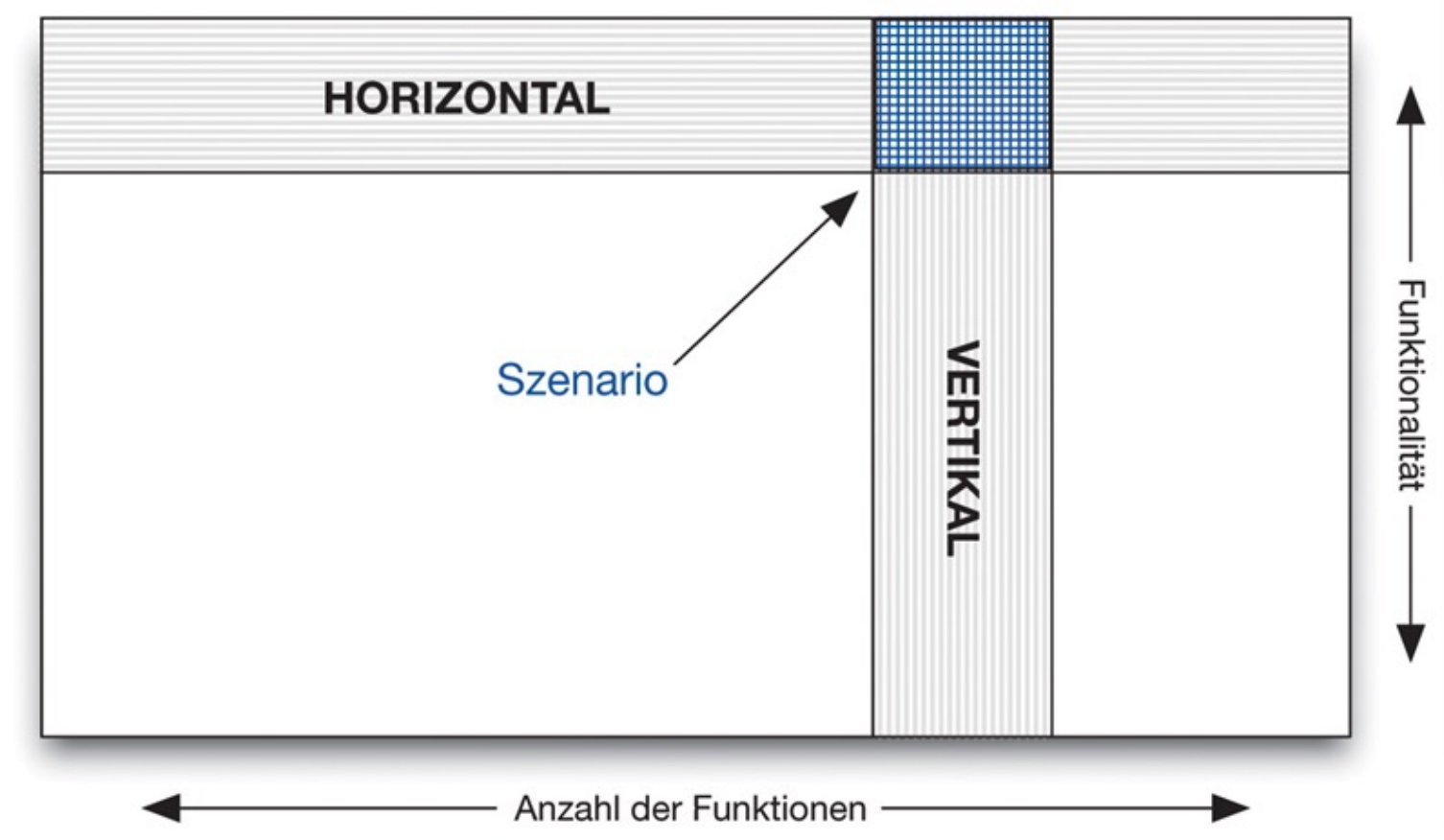
Explorativer Throw-Away Prototyp

- Anwender werden einbezogen, um **Anforderungen grundsätzlich zu klären**
- Ziel ist es, die Analyse zu unterstützen/zu ergänzen
- Prototyp wird **nicht** in evolutionärer Entwicklung weiterverwendet
- Häufige Bezeichnung: **Rapid Prototyping**
- Interne Struktur und Dokumentation nachrangig; Qualitätsstandards der Organisation müssen nicht eingehalten werden
- Eignet sich zur Integration in konventionelle Vorgehensmodelle

Experimenteller Prototyp

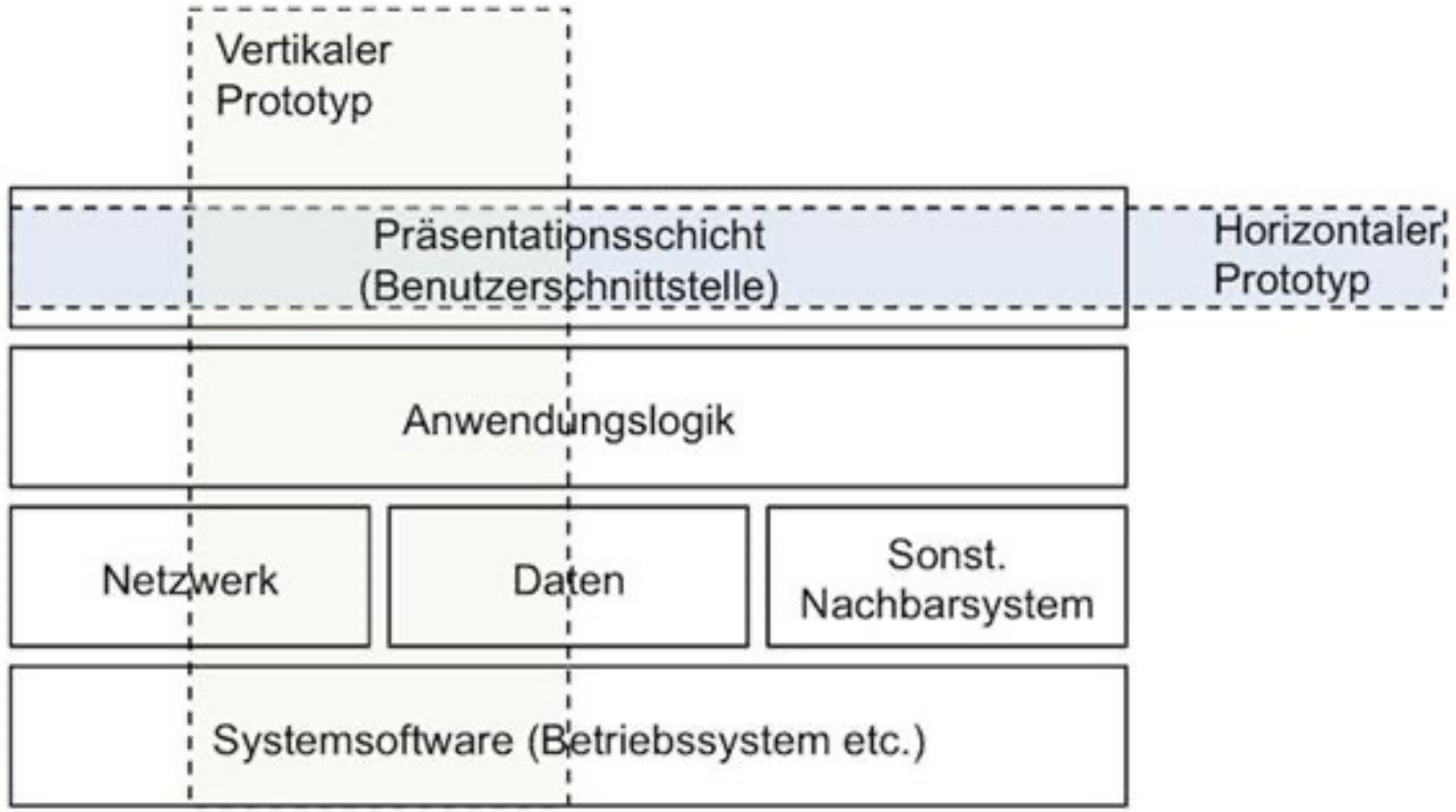
- **Technische Machbarkeit** soll untersucht werden (**Forschungsarbeit**)
- Vorstellungen vom Zielsystem sollen detailliert werden

Horizontaler vs. vertikaler Prototyp



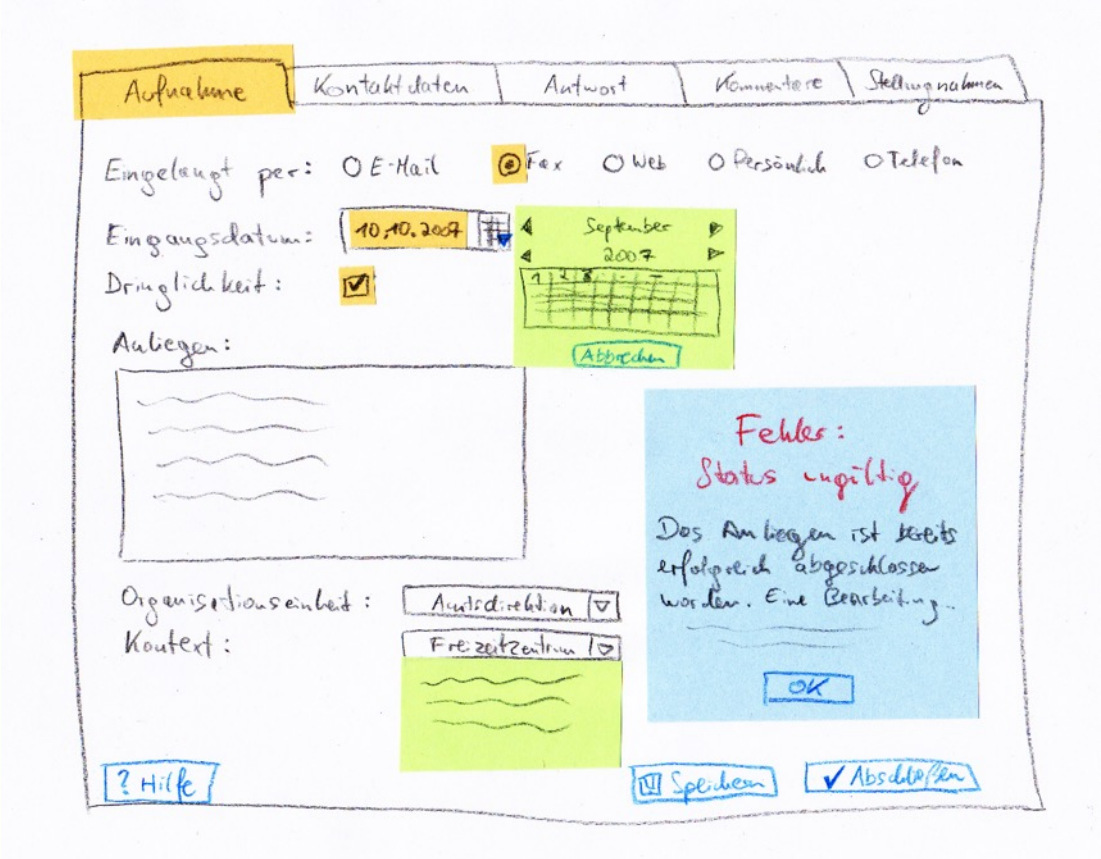
Quelle: Grechenig et al. (2009)

Horizontaler vs. vertikaler Prototyp



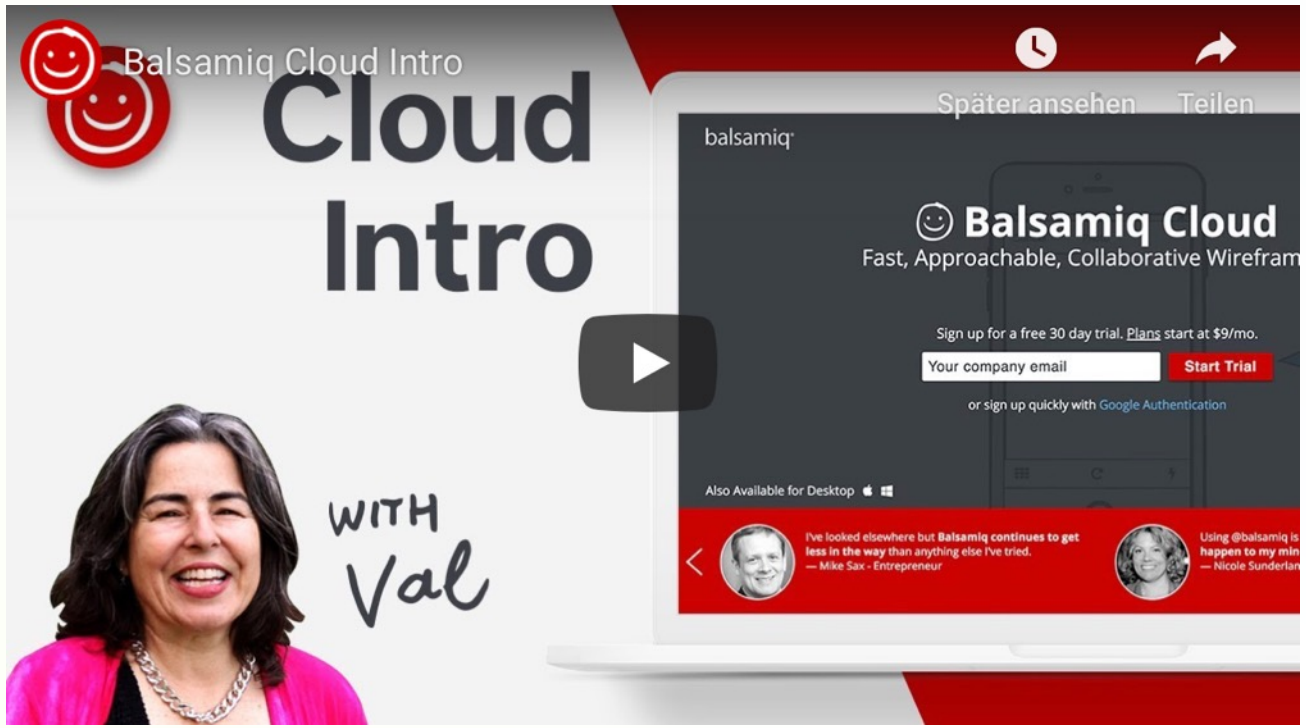
Paper Prototyping

- Low-Fidelity Prototyp → einfach zu erstellen und zu verändern
- Eingeschränkte Genauigkeit und Interaktivität



Balzamiq: https://balsamiq.com/

» *„Balsamiq Wireframes is the industry standard low-fidelity wireframing software. It makes work fun!“*



DUALE HOCHSCHULE SH

in Trägerschaft der Wirtschaftsakademie Schleswig-Holstein

Die Darstellung der Spezifikation

Modelle (z.B. UML, EPK, BPMN)

Prof. Dr. Malte Prieß

Modellbildung

- Modell ist abstrahiertes Abbild eines Originals zum Hervorheben des Wesentlichen und Verstecken des Unwesentlichen
- 3 Ziele: Abbildung, Verkürzung, Pragmatismus*
- Grafische Visualisierungen veranschaulichen Ausschnitte des modellierten Systems aus unterschiedlichen Sichten

Modellierungssprachen zur Notation

- Unified Modeling Language (UML)
- Business Process Modeling Notation (BPMN)
- Entity-Relationship-Modelle für Datenstrukturen
- Ereignisgesteuerte Prozessketten (EPK)
- Domain Specific Language (DSL)

* Stachowiak: Allgemeine Modelltheorie, Springer, 1973.

Sprache zur Modellbildung für Softwaresysteme und Geschäftsprozesse

- Spezifikation
- Konstruktion
- Visualisierung
- Dokumentation

- OMG Standard seit 1998
- Aktuelle Version UML 2.5
- Sammlung von Diagrammen für objekt-orientierte Analyse und Entwurf
- UML-Diagramme teilen mittlerweile gemeinsames Metamodell

Quelle: Oesterreich (2012)

Es gibt eine **Vielzahl von UML Tools** mit großen Unterschieden in

- Funktionalität und Reife
- Konformität zur UML Spezifikation
- Preis

oose. Innovative Informatik

UML-Hersteller Vergleich
http://www.oose.de/nuetzliches/fachliches/uml-werkzeuge/

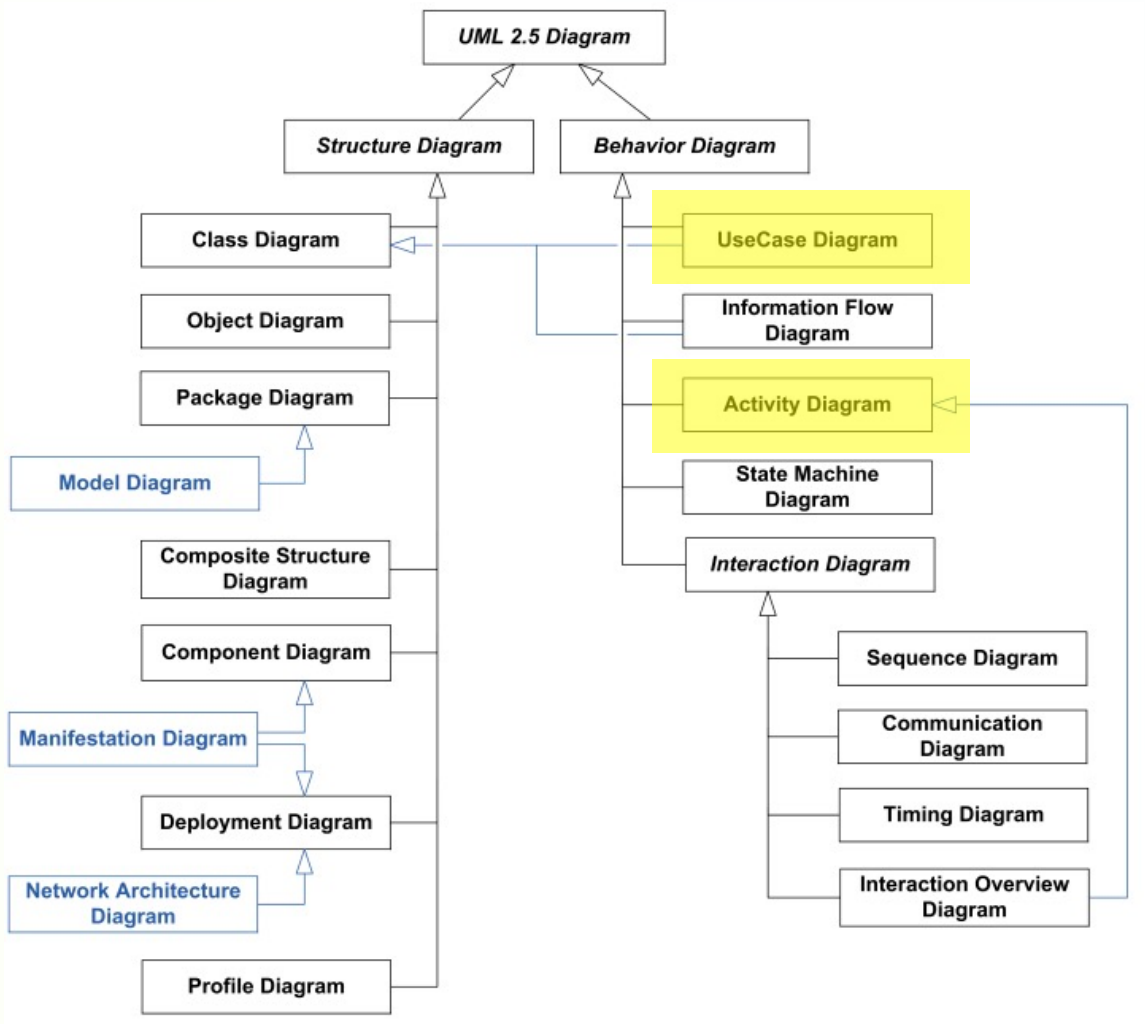
UML Tools - Empfehlungen

http://astah.net/student-license-request

http://www.visual-paradigm.com/download/community.jsp

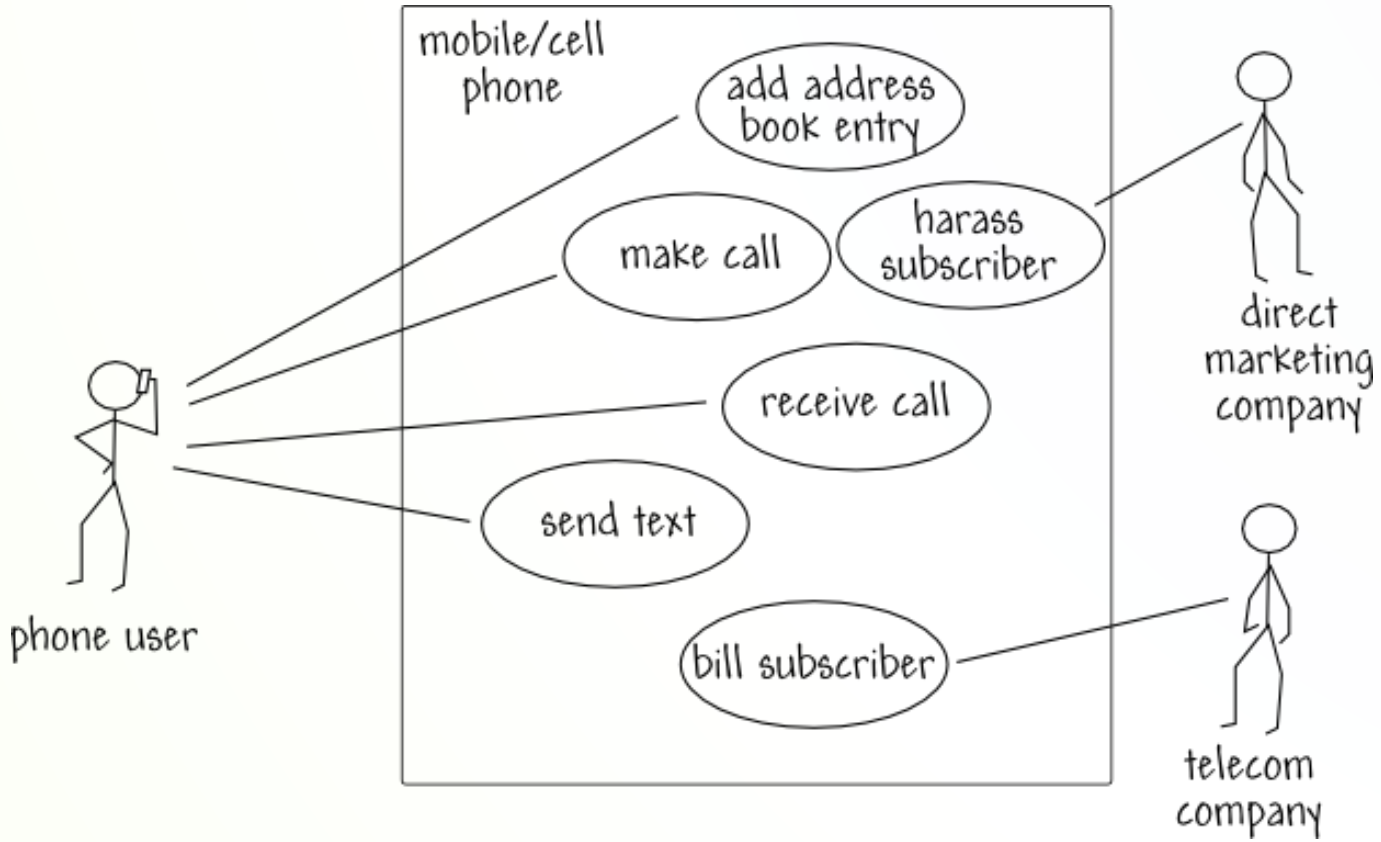
http://www.genmymodel.com/

Diagramme der UML 2.5



Anwendungsfall (Use Case)

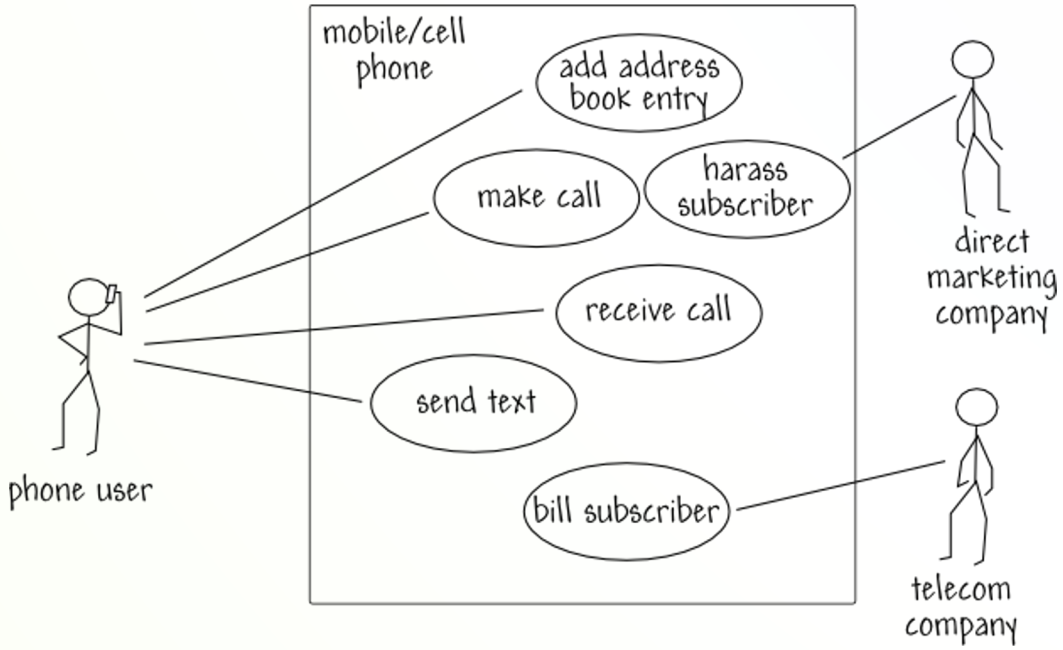
» Anwendungsfalldiagramm = **Verhaltensdiagramm** der UML



* Jacobson, Christerson, Jonsson, Overgaard: Object-Oriented Software Engineering, Addison-Wesley, 1992.

Anwendungsfall (Use Case)

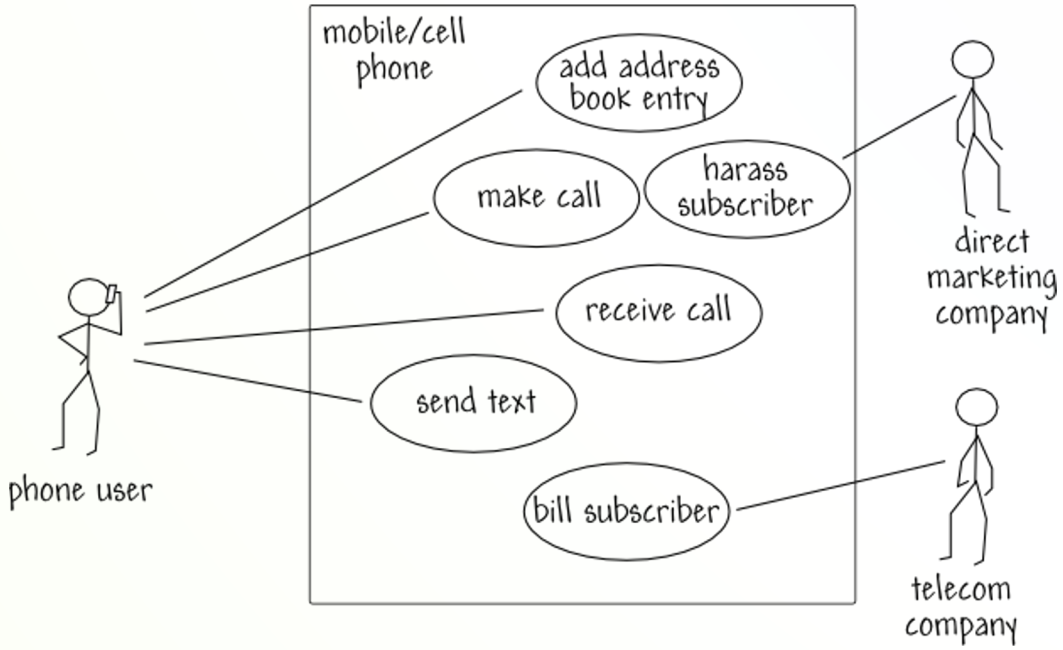
- Anwendungsfalldiagramm = **Verhaltensdiagramm** der UML
- Begriff **Use Case** eingeführt durch Jacobson 1992*
- Erfassung von **Akteuren** und **zugeordneten Anwendungsfällen** (= Use Cases) → Berührungspunkte der **Akteure** mit dem **System**



* Jacobson, Christerson, Jonsson, Overgaard: Object-Oriented Software Engineering, Addison-Wesley, 1992.

Anwendungsfall (Use Case)

- **Erster grober Überblick** über die Anforderungen an ein System → **hohes Abstraktionslevel**
- **Keine Information über technische Details**, wie z.B. Komponenten oder Datenstrukturen
- Überwiegend **während der Analyse** im frühen Stadium eines Softwareprojekts **eingesetzt**



* Jacobson, Christerson, Jonsson, Overgaard: Object-Oriented Software Engineering, Addison-Wesley, 1992.

Anwendungsfall (Use Case)

use case — *A sequence of interactions between an actor (or actors) and a system triggered by a specific actor, which produces a result for an actor.*

Jacobson et al. (1992)

Wesentliche Merkmale eines Anwendungsfalls (AF) sind:

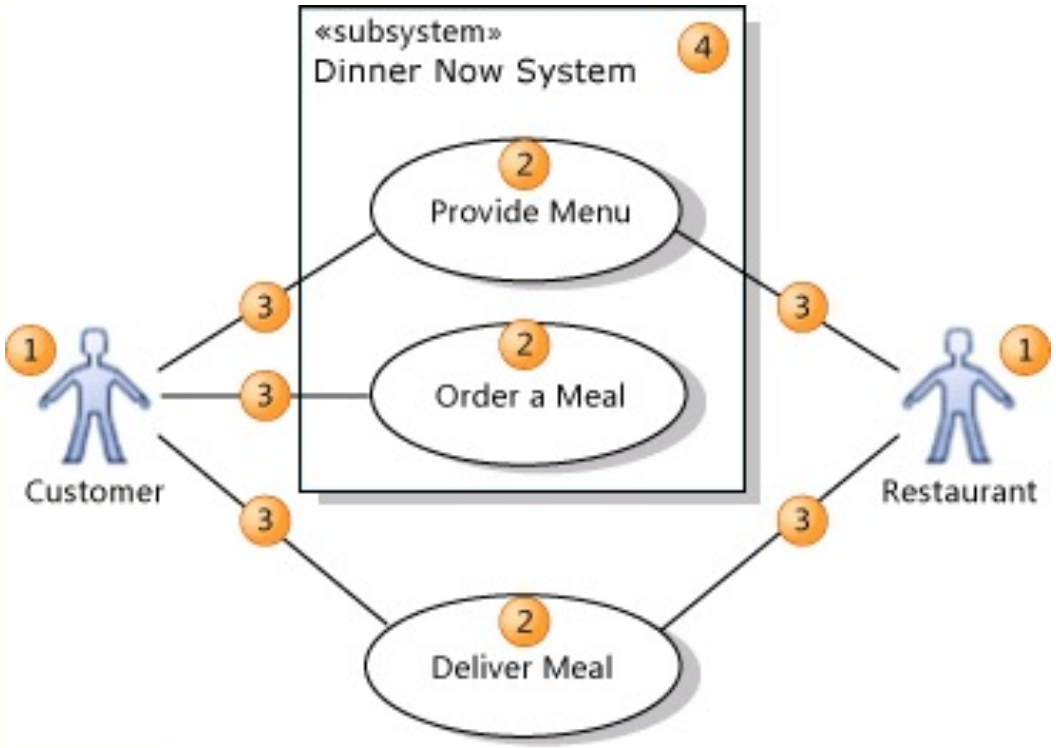
- Neben dem System ist immer **mind. ein Akteur** (actor) beteiligt

  (= Rolle eines mit dem System interagierenden Benutzers oder eines externen Systems)
- Anstoß durch ein **spezielles Ereignis** (einen **Trigger**), das ein Akteur – der Hauptakteur – auslöst
- Ein AF ist **zielorientiert**, der Akteur möchte das Ziel erreichen.
- Ein AF beschreibt **alle Interaktionen** zwischen dem System und den beteiligten Akteuren
- Ein AF **endet**, wenn das angestrebte Ziel erreicht ist oder wenn klar ist, dass es nicht erreicht werden kann

Quelle: Ludewig, Lichter, (2010), S.387

Anwendungsfalldiagramm (Use-Case-Diagramm)

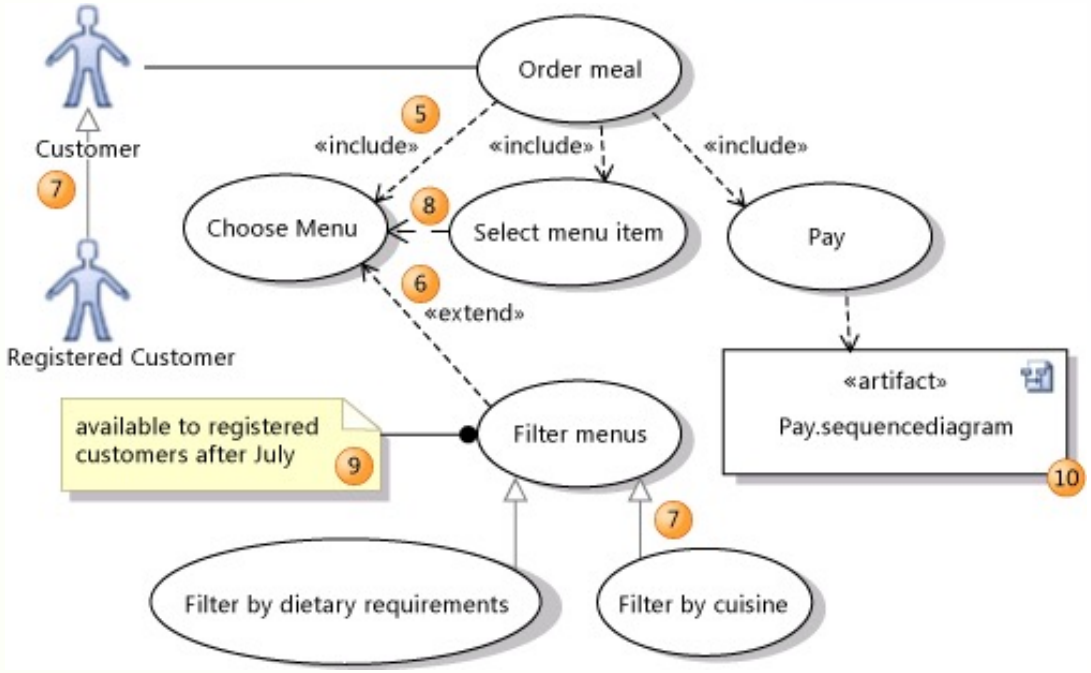
Nr.	Element	Beschreibung
1	Akteur	Repräsentiert Anwender, Gruppen oder externe Systeme, die mit dem System interagieren.
2	Use Case	Repräsentiert Aktion, die zur Erreichung eines bestimmten Ziels durchgeführt wird.
3	Assoziation	Beziehung zwischen Akteur und Use Case
4	System/Komponente	System oder Systemteil (Komponente) zur Gruppierung von Use Cases



Quelle: MSDN, UML Use Case Diagrams Reference

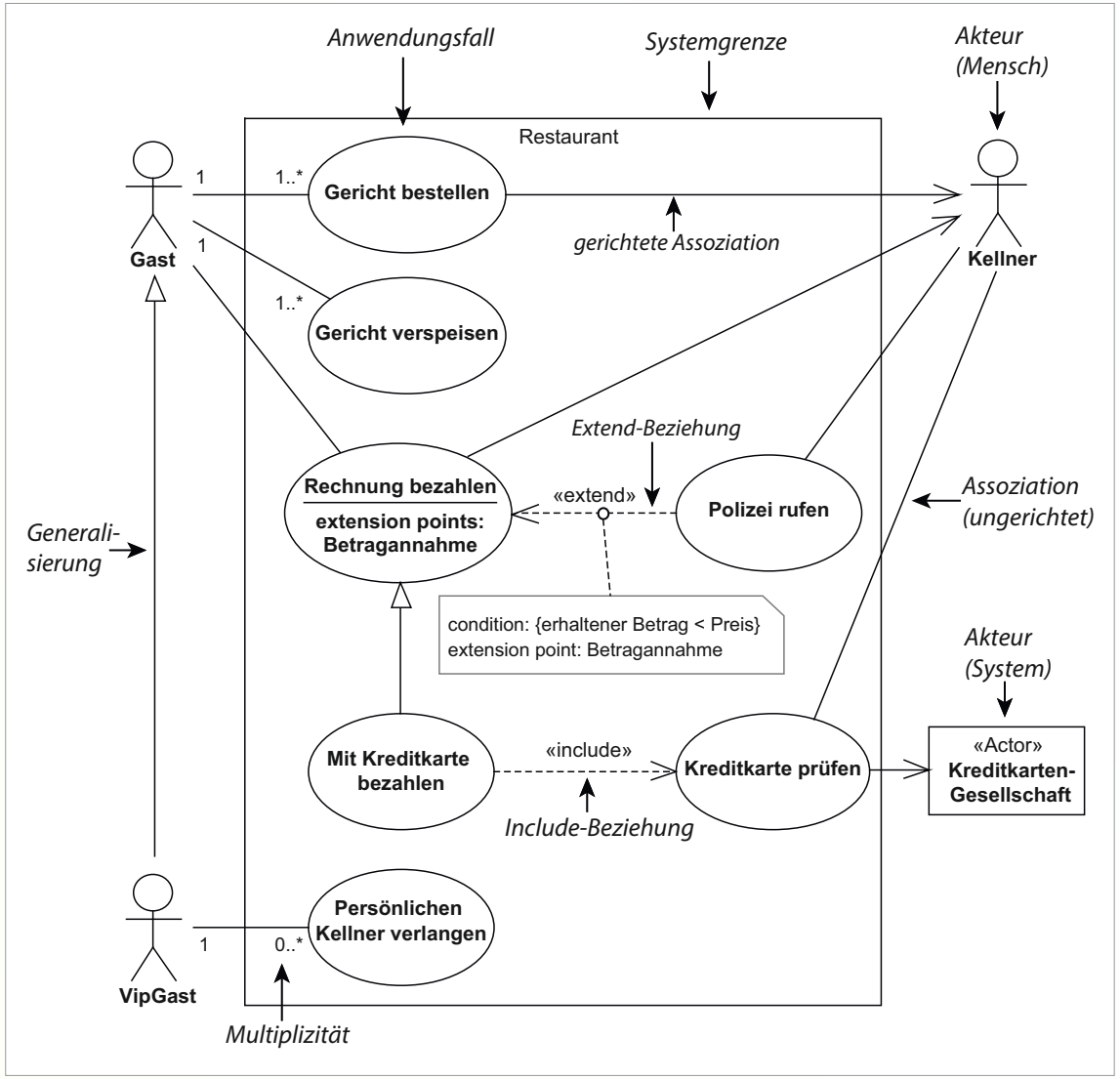
Anwendungsfalldiagramm (Use-Case-Diagramm)

Nr.	Element	Beschreibung
5	Include	Der inkludierte Use Case wird als Teil des inkludierenden Use Case aufgerufen. Dient der Unterteilung eines Use Case in feinere Teilaktionen
6	Extend	Der erweiternde Use Case fügt dem erweiterten Use Case **bedingt** Teilaktionen hinzu
7	Vererbung	Generalisierung bzw. Spezialisierung von Akteuren oder Use Cases
8	Abhängigkeit	Stellt logische Abhängigkeiten zwischen Use Cases dar
9	Kommentar	Beschreibender Kommentar
10	Artefakt	Verweis auf verwandtes (feiner spezifizierendes) Diagramm oder Dokument



Quelle: MSDN, UML Use Case Diagrams Reference

Anwendungsfalldiagramm (Use-Case-Diagramm)



Quelle: Kecher, Salvanos (2015), S.212

Ergänzende Beschreibung der Anwendungsfälle in einheitlicher Struktur (Templates, Schablonen)

Use Case-Template (*fully dressed*) nach Cockburn*

- Title
- Primary Actor
- Goal in Context
- Scope
- Level
- Stakeholders and Interests
- Precondition
- Minimal and Success Guarantees
- Trigger
- Main Success Scenario
- Extensions
- Technology and Data Variations List
- Related Information

* Cockburn: Writing Effective Use Cases, Addison-Wesley, 2000.

Ergänzende Beschreibung der Anwendungsfälle in einheitlicher Struktur (Templates, Schablonen)

Use Case-Template (*casual*) nach Cockburn*

- Title
- Primary Actor
- Scope
- Level
- Story (Informal Description)

* Cockburn: Writing Effective Use Cases, Addison-Wesley, 2000.

Templates für Anwendungsfälle

USE CASE	*Primär oder Sekundär (Kern- oder Nebenfunktion?)* UC-Name	**UC ID**	**1.1**
		Version	**0.1**
Ziel im Kontext	Was ist das Ziel dieses Anwendungsfalls und wie ordnet sich dieser in den Kontext der Anwendung ein?		
Akteure			
Primär	Auflistung der Akteure (Rollen, Systeme), die den UC aktiv ausführen, inkl. kurzer Beschreibung der jeweiligen Intention.		
Sekundär	Auflistung der Akteure (Rollen, Systeme), die in Zusammenhang mit dem UC stehen, inkl. kurzer Beschreibung der jeweiligen Intention.		
Vorbedingungen	Welche Bedingungen müssen erfüllt sein, damit der UC abgearbeitet werden kann?		
Auslösendes Ereignis	Welches Ereignis führt zur Auslösung des UC?		
Ablaufschritte			
Standard	Welche einzelnen Aktionen werden im Best-Case durchlaufen? (nummerierte Liste mit Aktionsbeschreibung; Wer tut was?)		
Alternativ	Welche Alternativen Aktionen können im Standard-Ablauf auftreten? (nummerierte Liste wie bei Standard-Ablauf erweitert um eine Stufe (a, b, c, ...)		
Nachbedingungen			
bei Erfolg	Was ist das Ergebnis einer erfolgreichen UC-Abarbeitung?		
bei Misserfolg	Was ist das Ergebnis einer erfolglosen UC-Abarbeitung? (Bei Fehler etc.)		
Referenzen	Verweise zu Diagrammen oder weiteren Dokumenten		
Bearbeitungskommentar	Platz für sonstige Bearbeitungskommentare, Änderungshinweise, Bearbeitungsstatus etc.		

Use Case für die Authentifizierung am Bankautomaten

Use Case für die Authentifizierung am Bankautomaten

Name	Authentifizieren
Ziel	Der Kunde möchte Zugang zu einem Bankautomaten BA42 erhalten
Vorbedingung	– Der Automat ist in Betrieb, die Willkommen-Botschaft wird angezeigt – Karte und PIN des Kunden sind verfügbar
Nachbedingung	– Der Kunde wurde akzeptiert – Die Leistungen des BA42 stehen dem Kunden zur Verfügung
Nachbedingung im Sonderfall	Der Zugang wird verweigert, die Karte wird entweder zurückgegeben oder einbehalten, die Willkommen-Botschaft wird angezeigt
Akteure	Kunde (Hauptakteur), Banksystem

Quelle: Ludewig, Lichter, (2010), S.388

Use Case für die Authentifizierung am Bankautomaten

Normalablauf	1. Der Kunde führt eine Karte ein 2. Der BA42 liest d. Karte und sendet d. Daten z. Prüfung ans Banksystem 3. Das Banksystem prüft, ob die Karte gültig ist 4. Der BA42 zeigt die Aufforderung zur PIN-Eingabe 5. Der Kunde gibt die PIN ein 6. Der BA42 liest die PIN und sendet sie zur Prüfung an das Banksystem 7. Das Banksystem prüft die PIN 8. Der BA42 akzeptiert den Kunden und zeigt das Hauptmenü

Quelle: Ludewig, Lichter, (2010), S.388

Use Case für die Authentifizierung am Bankautomaten

2. Der BA42 liest d. Karte und sendet d. Daten z. Prüfung ans Banksystem

3. Das Banksystem prüft, ob die Karte gültig ist

Sonderfall	**2a**	*Die Karte kann nicht gelesen werden* 2a.1 Der BA42 zeigt die Meldung »Karte nicht lesbar« (4 s) 2a.2 Der BA42 gibt die Karte zurück 2a.3 Der BA42 zeigt die Willkommen-Botschaft
Sonderfall	**2b**	*Die Karte ist lesbar, aber keine BA42-Karte* 2b.1 Der BA42 zeigt die Meldung »Karte nicht akzeptiert« (4 s) 2b.2 Der BA42 gibt die Karte zurück 2b.3 Der BA42 zeigt die Willkommen-Botschaft
Sonderfall	**2c**	*Das Banksystem ist nicht erreichbar* 2c.1 Der BA42 zeigt die Meldung »Banksystem nicht erreichbar« (4 s) 2c.2 Der BA42 gibt die Karte zurück 2c.3 Der BA42 zeigt die Willkommen-Botschaft
Sonderfall	**3a**	*Die Karte ist nicht gültig oder gesperrt* 3a.1 Der BA42 zeigt die Meldung »Karte ungültig oder gesperrt« (4 s) 3a.2 Der BA42 zeigt die Meldung »Karte wird eingezogen« (5 s) 3a.3 Der BA42 behält die Karte ein 3a.4 Der BA42 zeigt die Willkommen-Botschaft

Quelle: Ludewig, Lichter, (2010), S.388

Use Case für die Authentifizierung am Bankautomaten

5. Der Kunde gibt die PIN ein
6. Der BA42 liest die PIN und sendet sie zur Prüfung an das Banksystem
7. Das Banksystem prüft die PIN

Sonderfall	**5a**	*Der Kunde bricht den Vorgang ab* 5a.1 Der BA42 zeigt die Meldung »Vorgang wird abgebrochen« (2 s) 5a.2 Der BA42 gibt die Karte zurück 5a.3 Der BA42 zeigt die Willkommen-Botschaft
Sonderfall	**5b**	*Der Kunde reagiert nach 5 Sekunden nicht* 5b.1 Der BA42 zeigt die Meldung »Keine Aktivität, Abbruch« (2 s) 5b.2 Der BA42 gibt die Karte zurück 5b.3 Der BA42 zeigt die Willkommen-Botschaft
Sonderfall	**6a**	*Das Banksystem ist nicht erreichbar* → Schritt 2c.1 – 2c.3
Sonderfall	**7a**	*Die erste oder zweite eingegebene PIN ist falsch* 7a.1 Der BA42 zeigt die Meldung »Falsche PIN« (4 s) → Schritt 4 ff
Sonderfall	**7b**	*Die dritte eingegebene PIN ist falsch* 7b.1 Der BA42 zeigt die Meldung »PIN dreimal falsch« (5 s) → Schritt 3a.2 – 3a.4

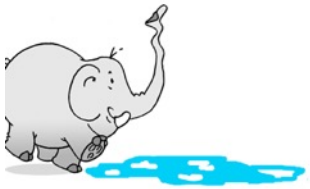
Quelle: Ludewig, Lichter, (2010), S.388

Andere UML-Diagramme bieten weitere geeignete Visualisierungen für die **Anforderungsspezifikation**, solange das Abstraktionslevel angemessen bleibt, d.h. auch **für Nicht-Softwareingenieure verständlich**.

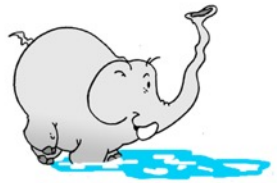
Komponenten-, Verteilungs- und Aktivitätsdiagramme werden häufig in Pflichtenheften o.ä. eingesetzt.

(Vorstellung weiterer UML-Diagramme erfolgt im Kapitel Entwurf)

1.



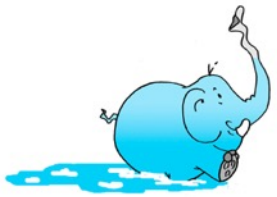
2.



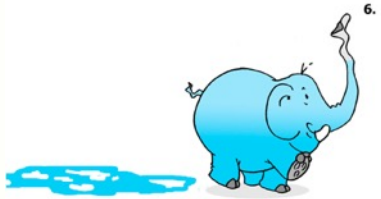
3.

4.

5.



6.



Geschäftsprozess

„Inhaltlich abgeschlossene, zeitlich und sachlogische **Folge von Aktivitäten**,
die zur **Bearbeitung eines betriebswirtschaftlich relevanten Objektes**
notwendig sind.“

Becker et al. 2005

„Sammlung von **Aktivitäten**, die einen **Input** benutzen,
um einen **Output** zu erzeugen,
der einen **Wert für den Kunden** darstellt.“

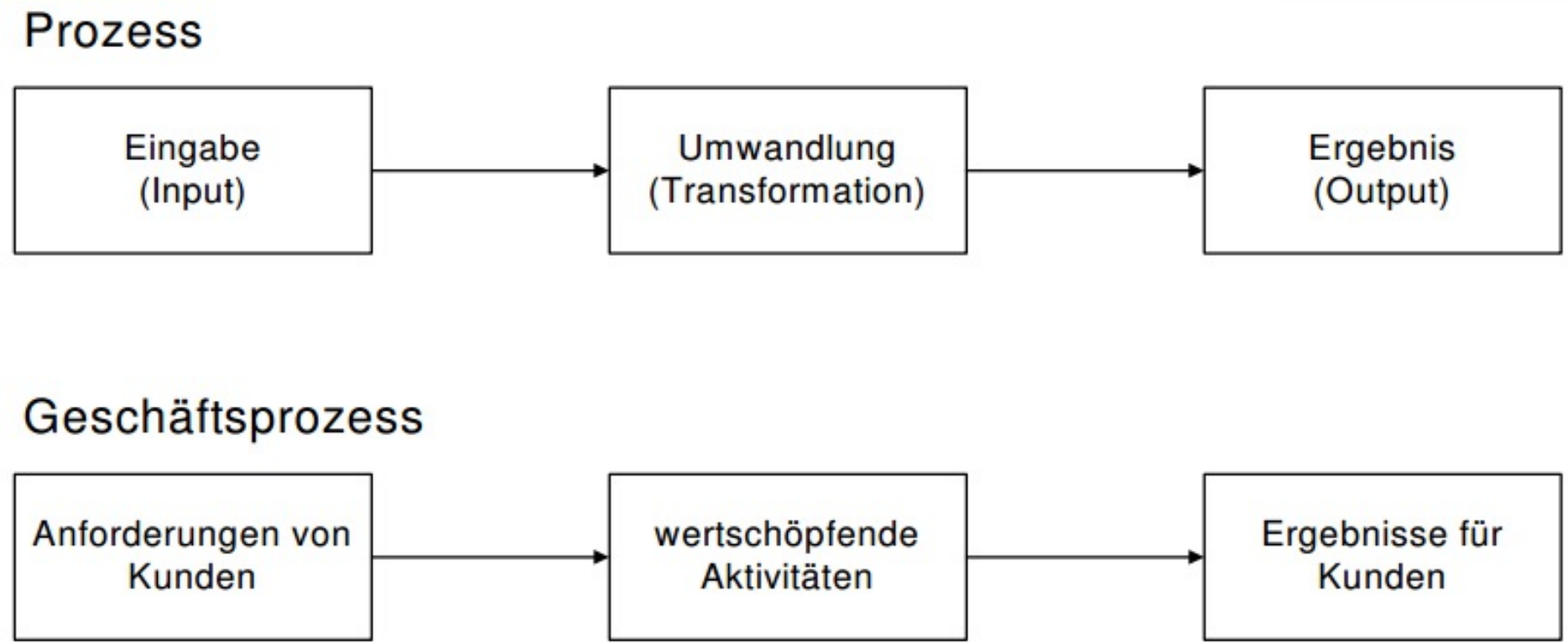
Hammer, Champy 1993

„Strukturierte, messbare Menge von **Aktivitäten**,
um einen **spezifizierten Output**
für einen **bestimmten Kunden oder Markt** zu erzeugen.“

Davenport 1992

Becker, Kugeler, Rosemann: Prozessmanagement, 5. Aufl., Springer, 2005.; Hammer, Champy: Reengineering the Corporation - A Manifesto for Business Revolution, 3rd ed., Harper Business, 1993.; Davenport: Process Innovation: Reengineering Work Through Information Technology, Harvard Business Review, 1992.

Geschäftsprozess



Kriterien für Einstufung als „bedeutsamer Geschäftsprozess“

» Wertschöpfende Aktivität	96,2 %
» Funktionsübergreifend	76,9 %
» Kundenorientiert	73,1 %
» Prozessverantwortlicher vorhanden	69,2 %
» Ziele und Messgrößen definiert	61,5 %
» Von strategischer Bedeutung für das Unternehmen	42,3 %

Quelle: Befragung deutscher Großunternehmen an der LMU München*

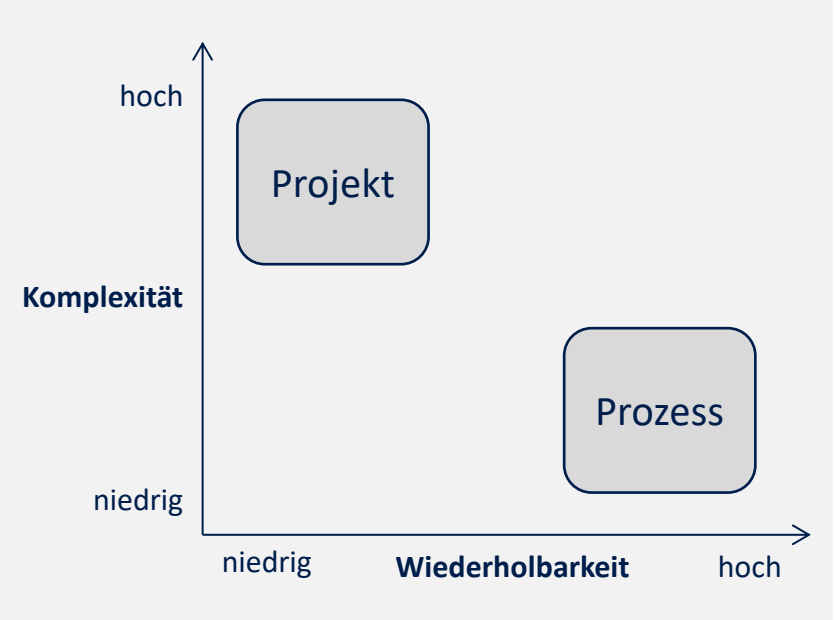
* Gadatsch: Grundkurs Geschäftsprozess-Management - Methoden und Werkzeuge für die IT-Praxis - Eine Einführung für Studenten und Praktiker, 5. Aufl., Vieweg + Teubner, 2008.

Geschäftsprozesse

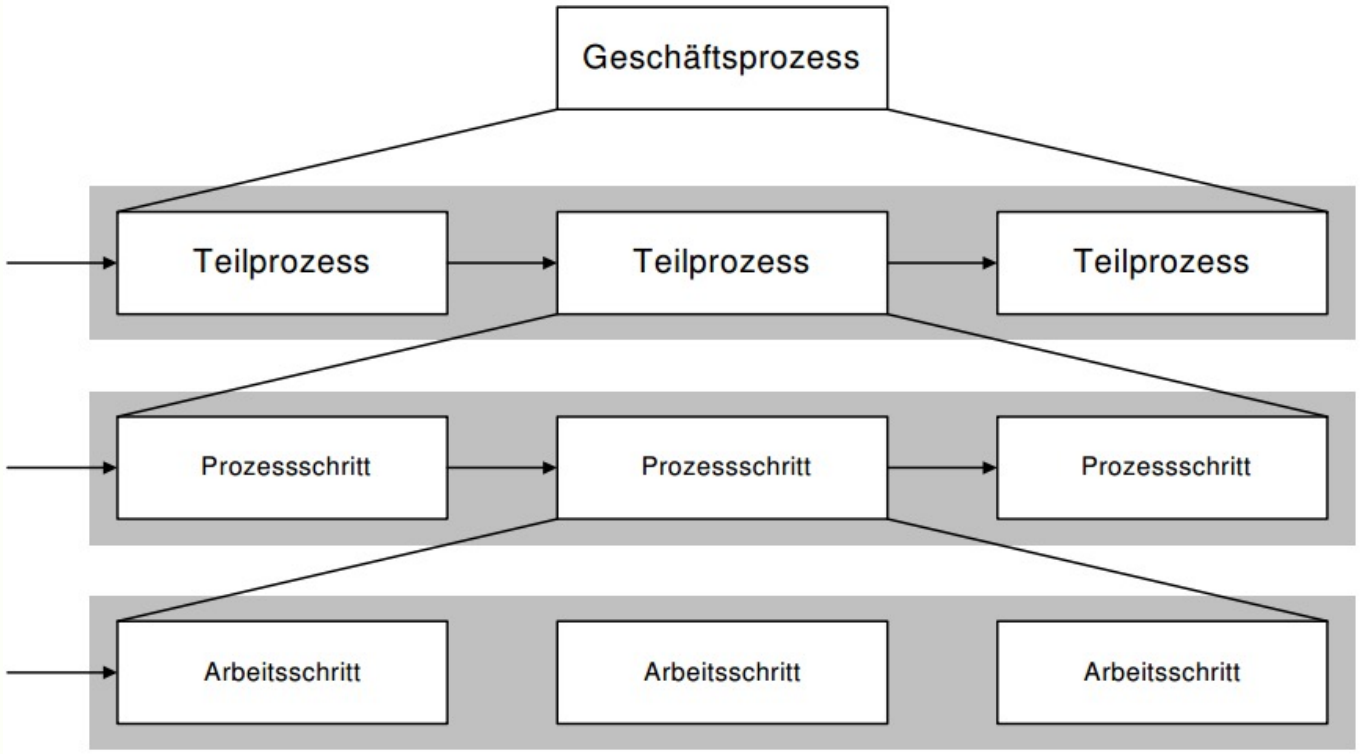
- laufen im Rahmen einer dauerhaften Prozessorganisation ab
- sind auf Dauer angelegt
- sind in ihrer Ausführung häufiger und standardisierter als Projekte

Projekte

- sind einmalige und zeitlich befristete Vorhaben von relativ hoher Komplexität und Neuartigkeit
- werden in einer temporären Projektorganisation ausgeführt



Geschäftsprozesse können hierarchisch untergliedert werden



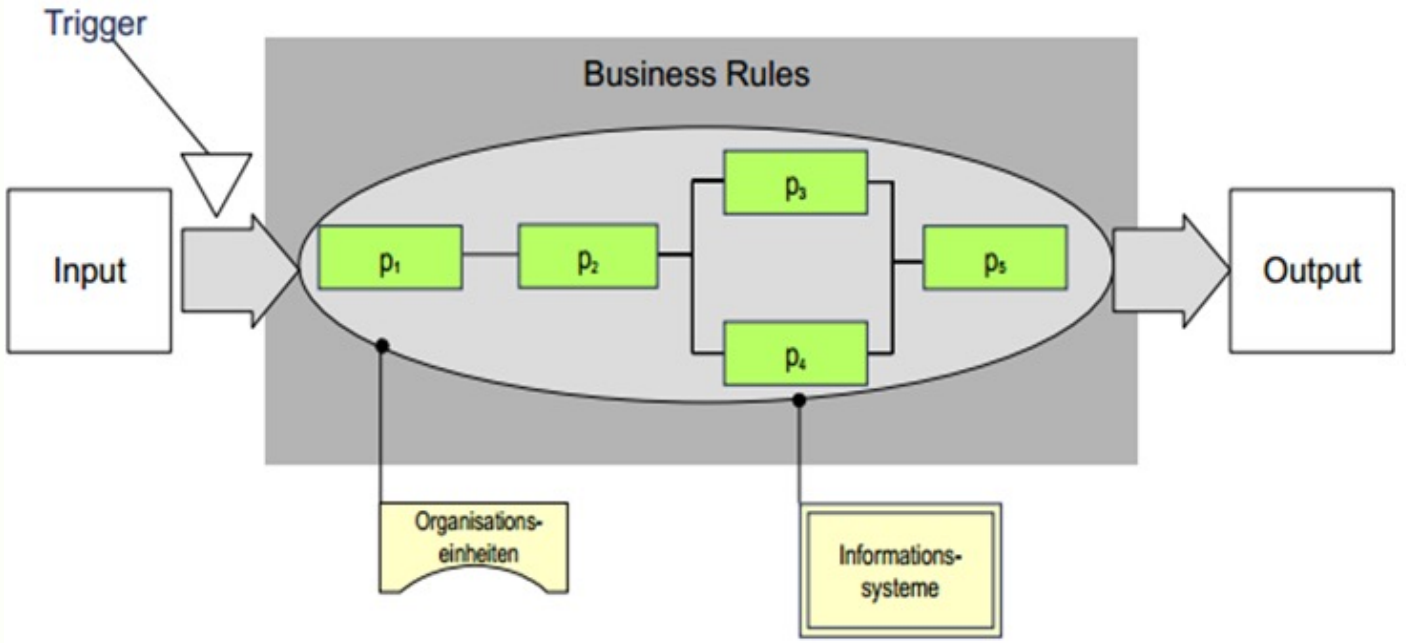
Schmelzer, Sesselmann: Geschäftsprozessmanagement in der Praxis - Kunden zufrieden stellen - Produktivität steigern - Wert erhöhen, 5. Aufl., Hanser, 2004.

Geschäftsprozesse können hierarchisch untergliedert werden

Schmelzer, Sesselmann	*Striening, Haist, Fromm (IBM)*	*Horvarth & Partners*	*REFA-Verband*	*...*
Geschäftsprozess	Prozess	Geschäftsprozess	Unternehmens-prozess	...
Teilprozess	Subprozess	Prozess	Hauptprozess	...
Prozessschritt	Aktivität	Teilprozess	Teilprozess	...
Arbeitsschritt	Aufgabe	Aktivität	Arbeitssystem-prozess	...

Schmelzer, Sesselmann: Geschäftsprozessmanagement in der Praxis - Kunden zufrieden stellen - Produktivität steigern - Wert erhöhen, 5. Aufl., Hanser, 2004.

Abstraktes Modell eines Geschäftsprozesses



Notationen zur Geschäftsprozessmodellierung

- Ereignisgesteuerte Prozessketten (**EPK**s)
- Business Process Modeling Notation (**BPMN**)
- Aktivitätsdiagramme der Unified Modeling Language (**UML**)

Ereignisgesteuerte Prozessketten (EPKs)

- Modellierungssprache für Geschäftsprozesse
- Entstanden 1992, Universität Saarbrücken/SAP
- Teil des ARIS-Konzept
  (*Architektur integrierter Informationssysteme*)
- Werkzeug: ARIS-Toolset
- Modellierung mit EPKs:
  https://www.peterjohann-consulting.de/prozessmanagement/

Scheer: ARIS - Vom Geschäftsprozess zum Anwendungssystem, Springer, 2002.

EPKs – Notationselemente

Ereignisse sind passiv und beschreiben einen Zustand, z.B. „Bonität negativ“. Sie lösen Funktionen aus und sind Ergebnis ausgeführter Funktionen.

Funktionen sind aktiv und beschreiben eine Aufgabe, z.B. „Bonität prüfen“.

Gerichtete Kontrollflusskante

Logische Operatoren:

UND-Verknüpfung (Konjunktion)

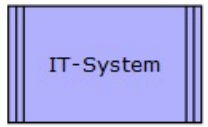
OR-Verknüpfung (Adjunktion)

XOR-Verknüpfung (Disjunktion)

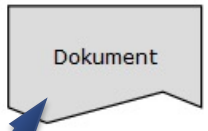
Informationsobjekt

Rollen werden an Funktionen angeheftet und beschreiben, wer eine Funktion ausführt.

Organisationseinheiten (OE) werden an Funktionen angeheftet und beschreiben, welcher OE eine Funktion zugeordnet ist.



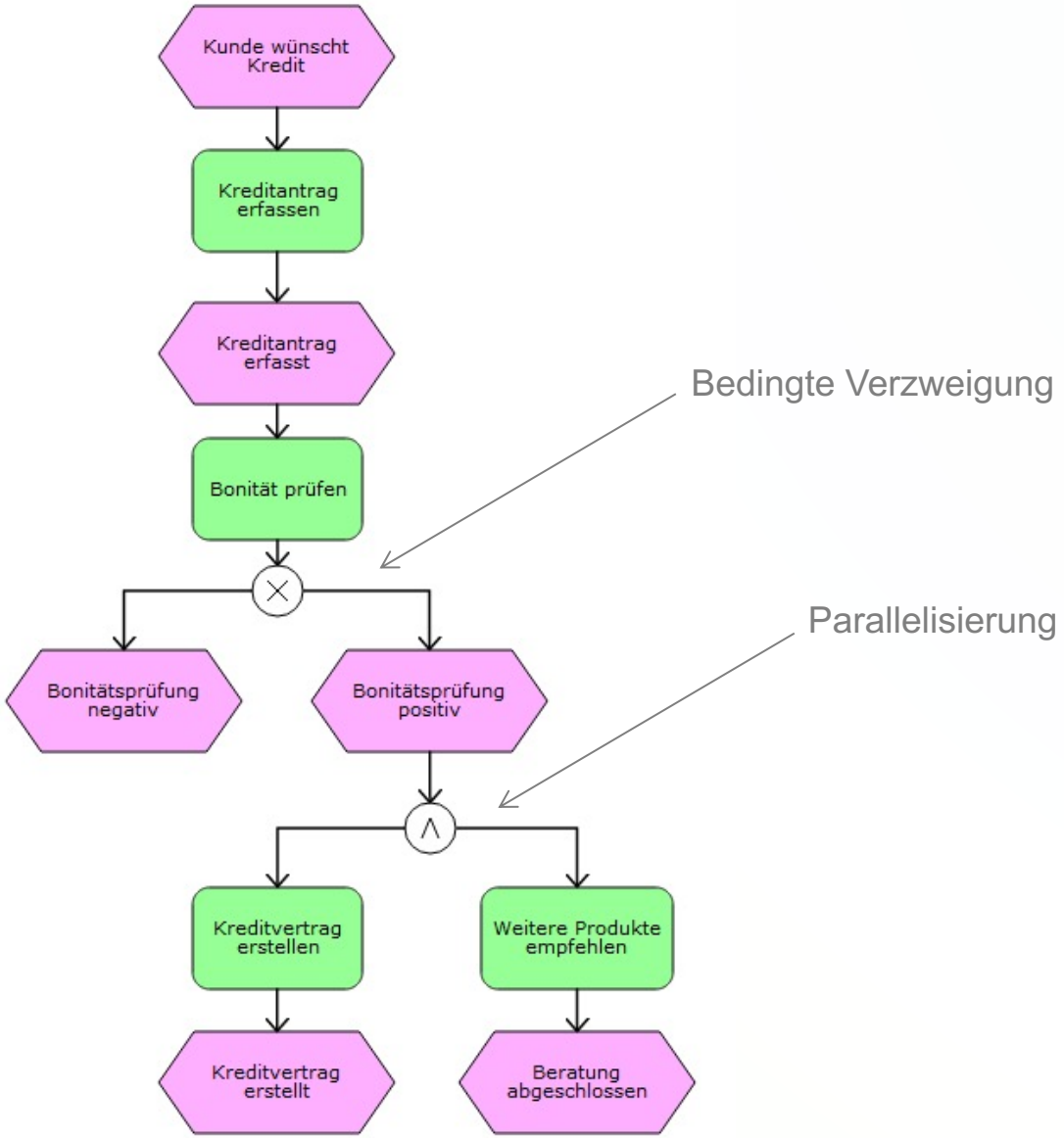
IT-Systeme werden an Funktionen angeheftet und beschreiben z.B. ein System oder eine Anwendung, das eine Funktion verwendet.



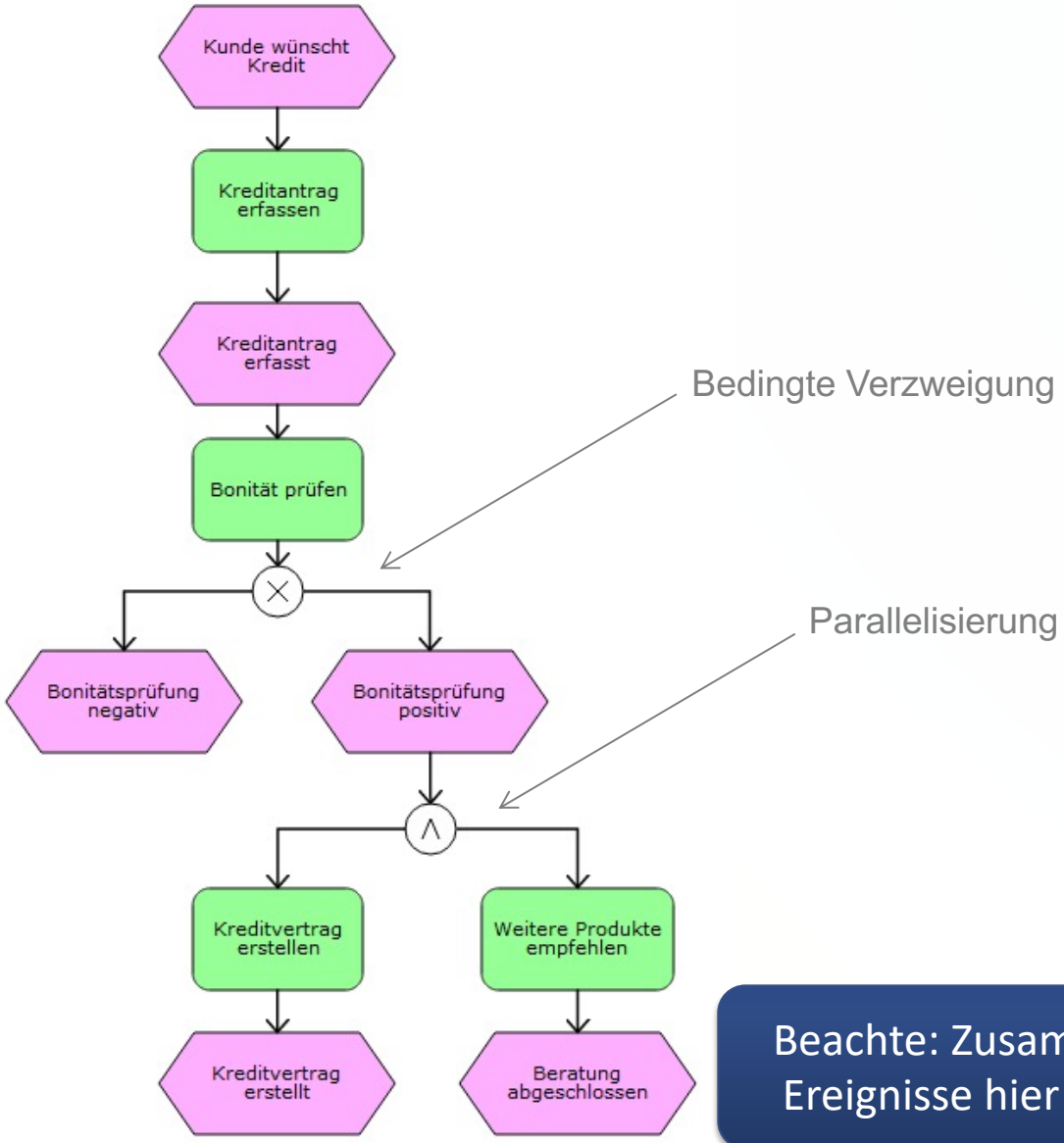
Dokumente werden an Funktionen angeheftet und beschreiben ein Dokument, das eine Funktion erzeugt und verwendet.

Prozessschnittstellen verweisen auf andere Prozesse oder Prozessschnittstellen.

EPK – Beispiel Kreditantrag (Ausschnitt)

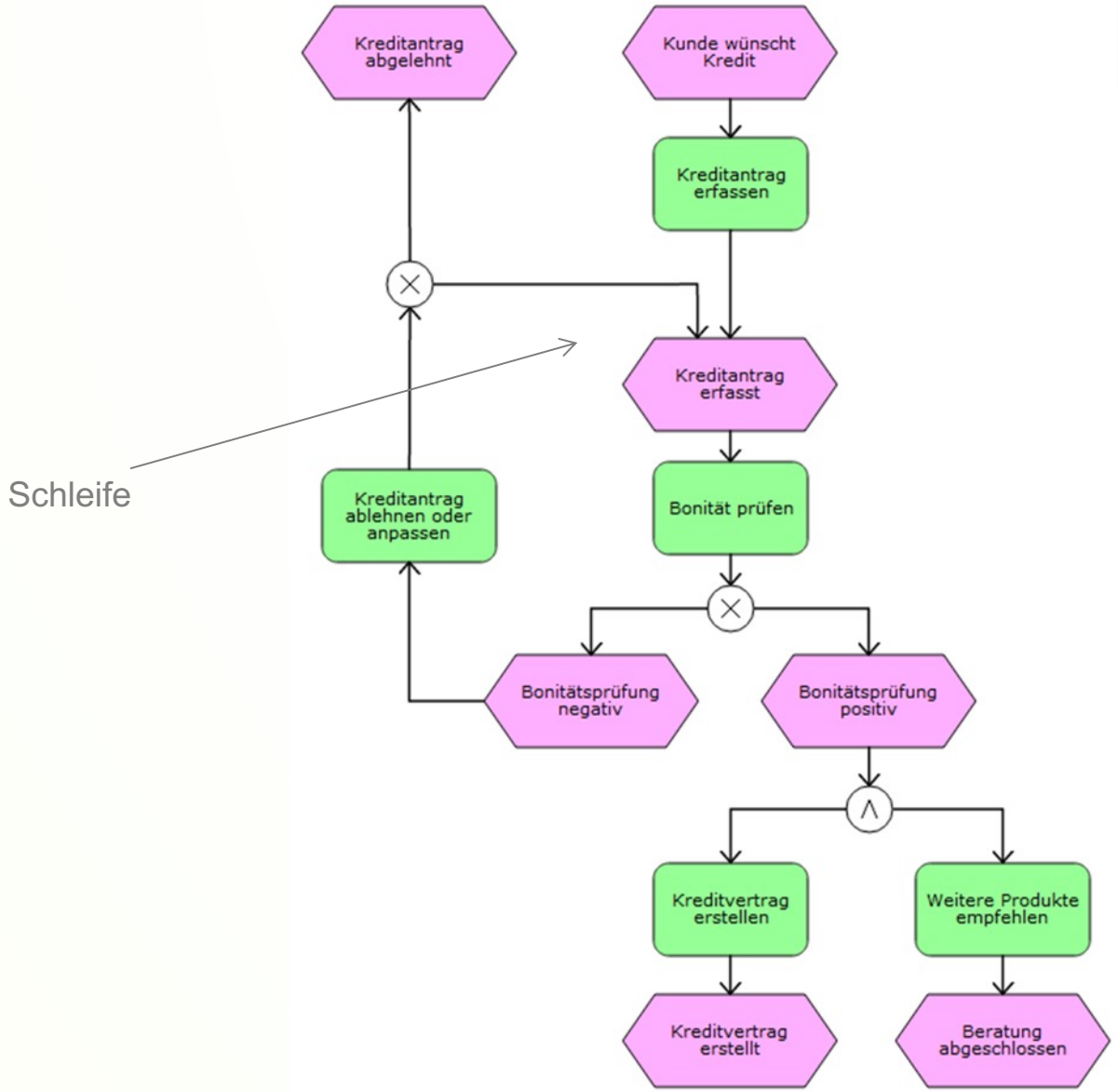


EPK – Beispiel Kreditantrag (Ausschnitt)

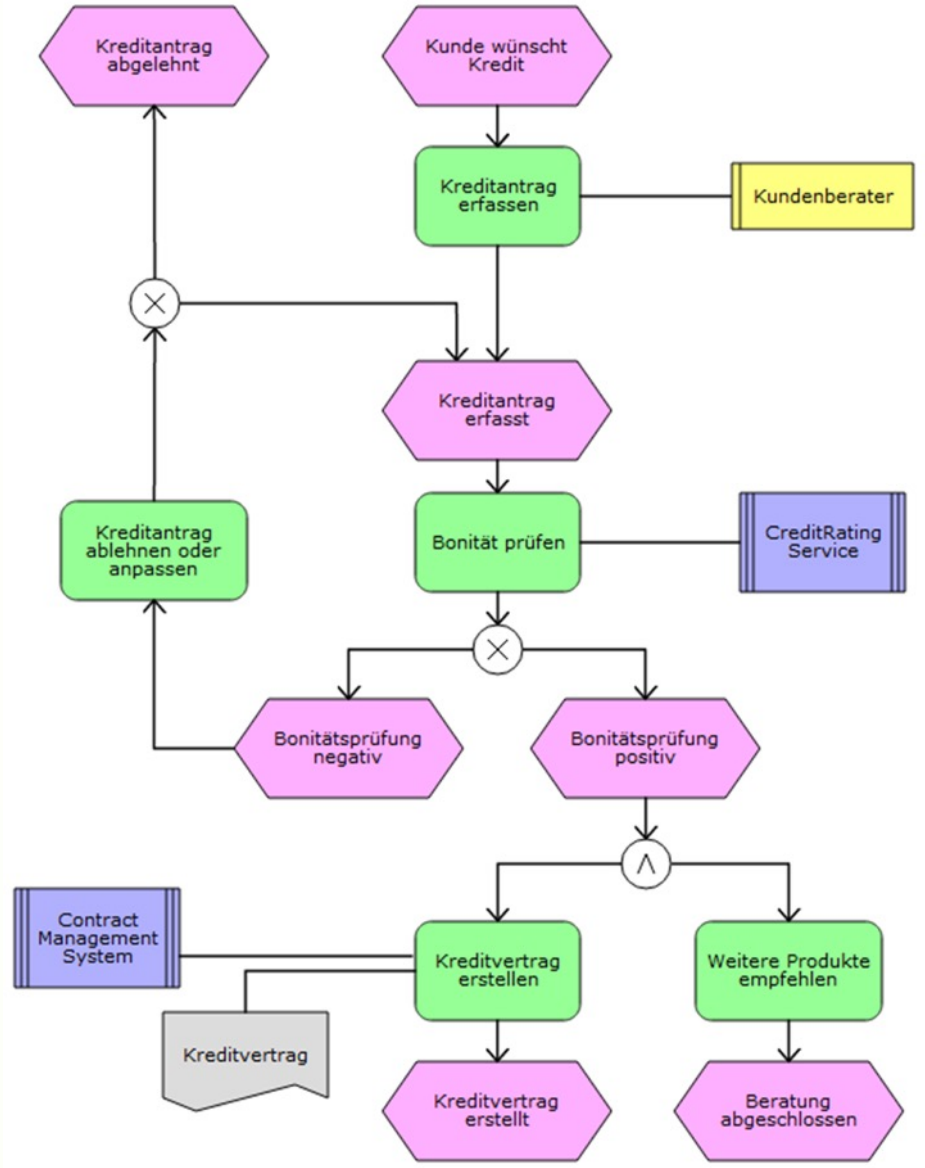


Beachte: Zusammenführung der Ereignisse hier noch notwendig

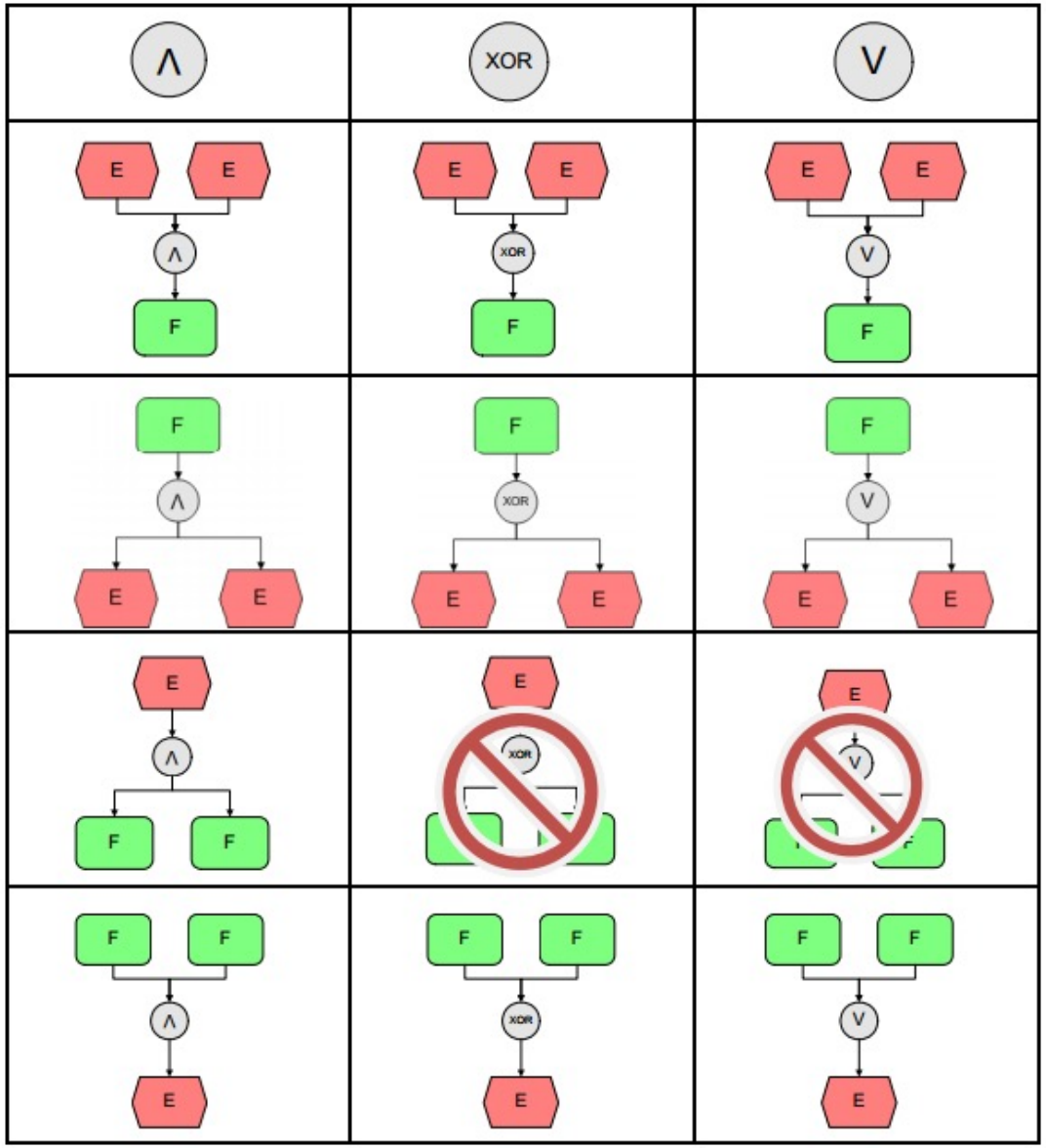
EPK – Beispiel Kreditantrag



EPK – Beispiel Kreditantrag



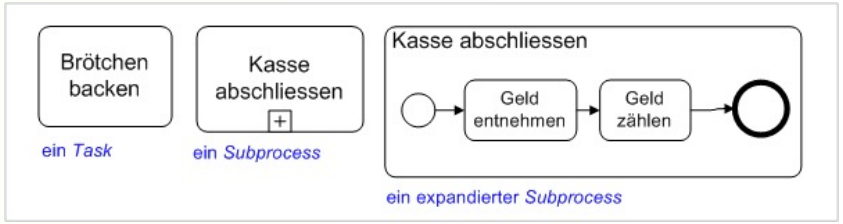
EPK – Verknüpfungsregeln

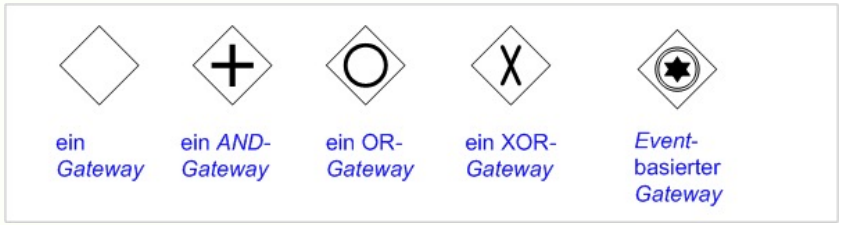


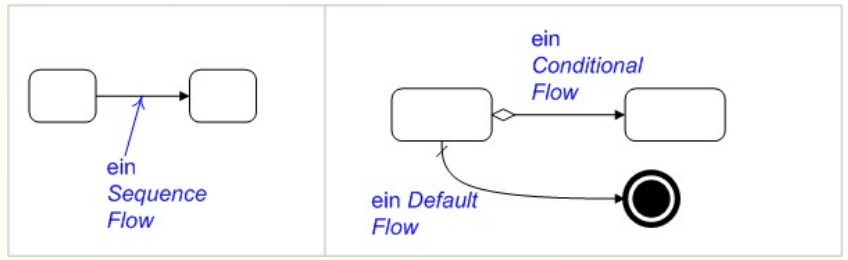
BPMN

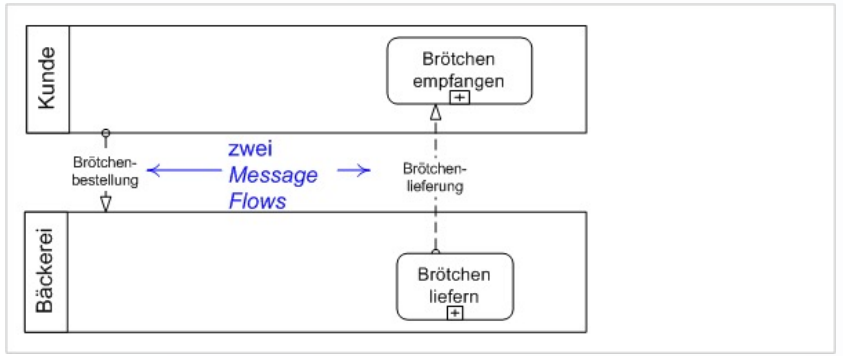
- Modellierungssprache für Geschäftsprozesse
- Entstanden 2001, IBM → seit 2005 OMG-Standard
- Aktuelle Version: BPMN 2.0

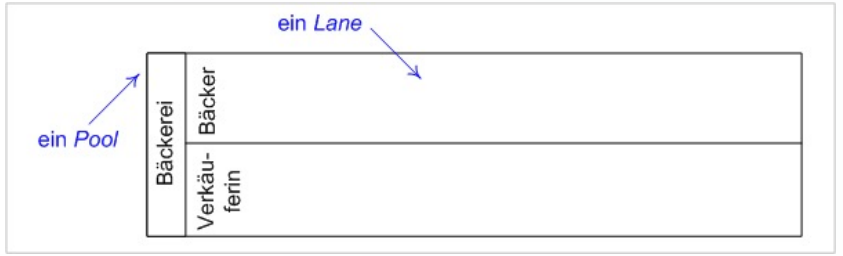
BPMN – Notationselemente

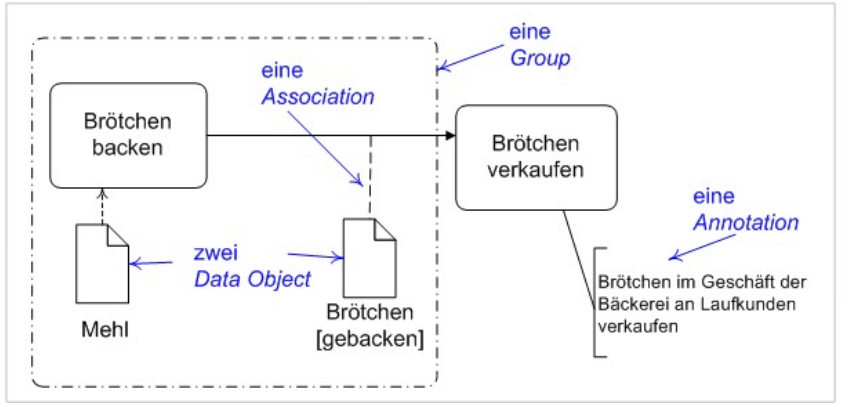






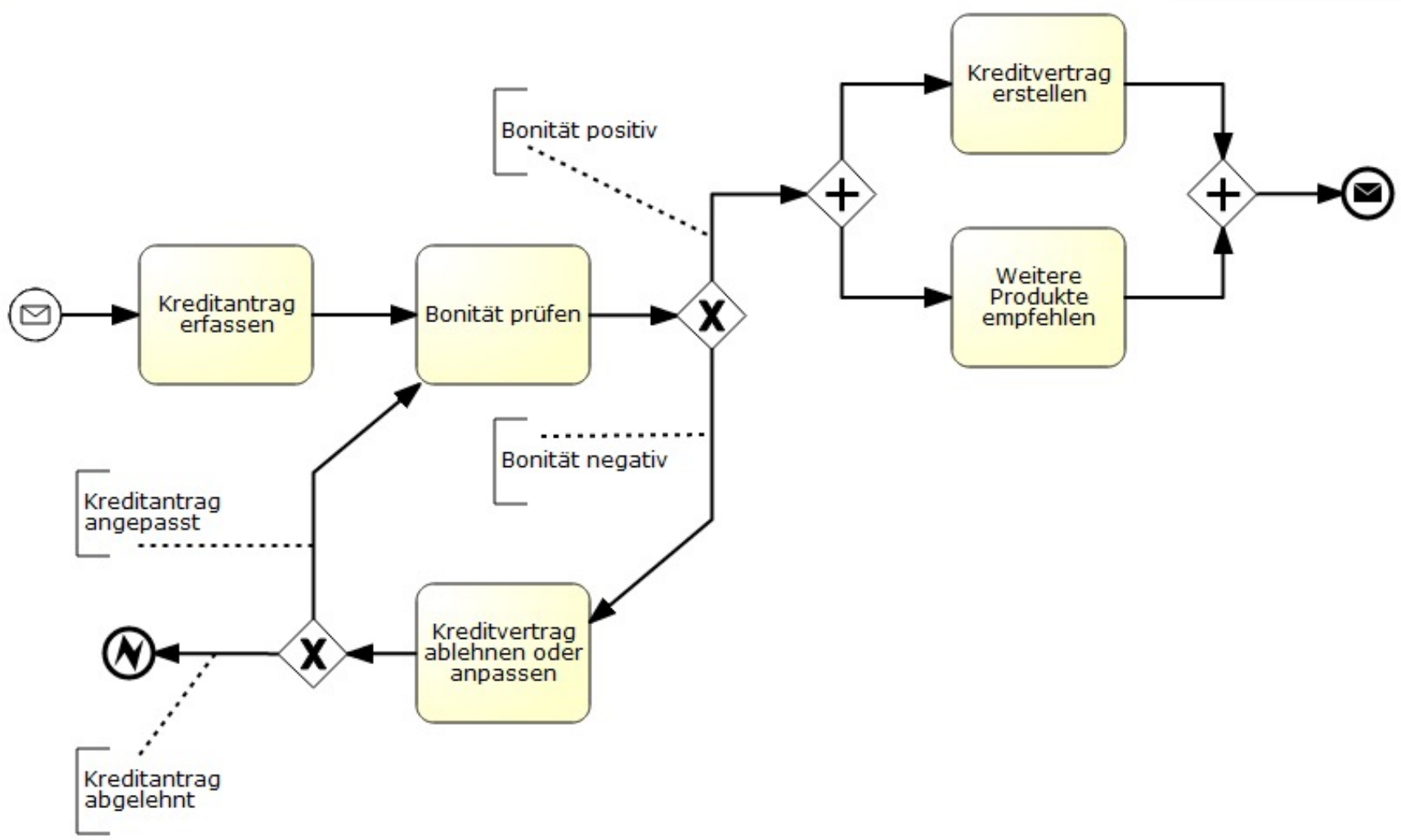




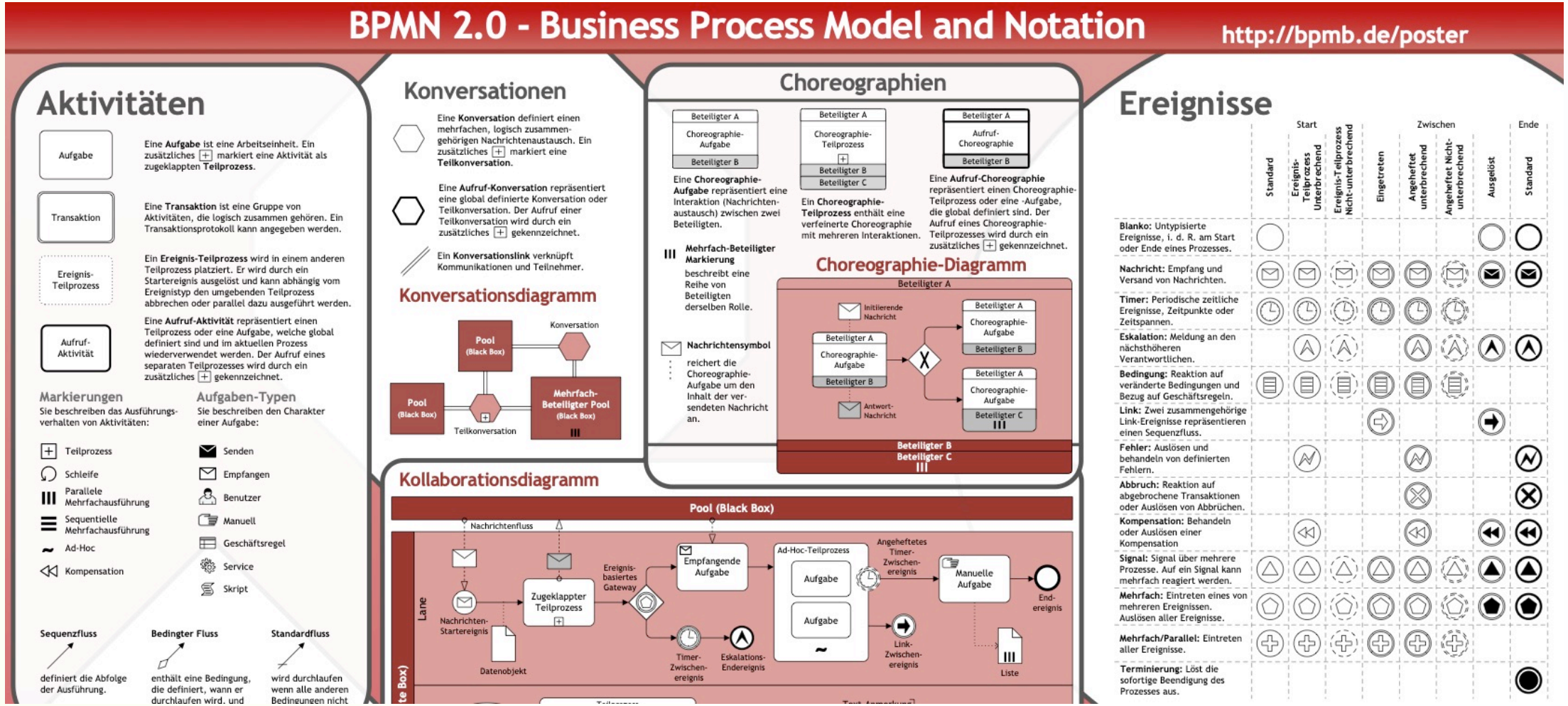


Vgl. BPMN2_0_Poster_DE.pdf

BPMN – Beispiel Kreditantrag

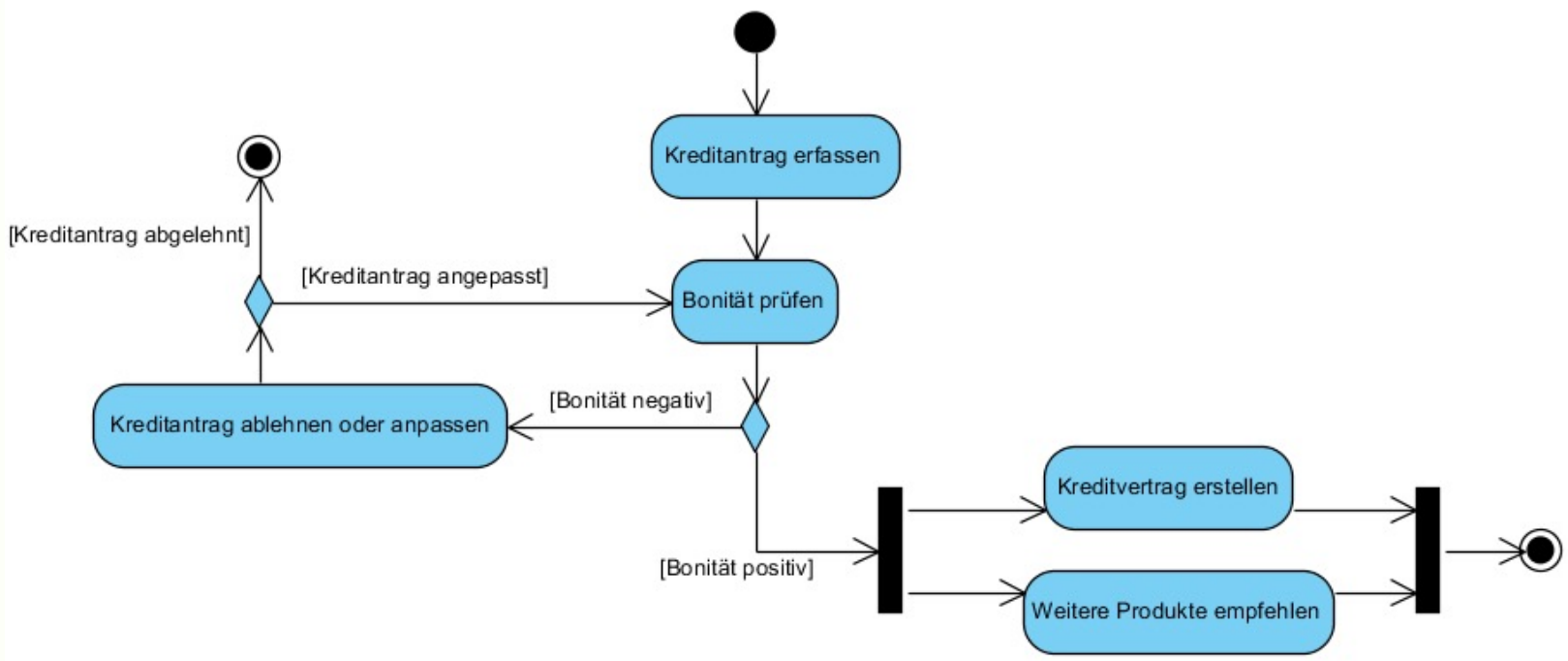


BPMN – BPMN2_0_Poster_DE.pdf (➔ ILIAS)



- Verhaltensdiagramm der UML
- Aktivitätsdiagramme können in allen Phasen der Softwareentwicklung eingesetzt werden:
  - Während der **Analyse/Definition-Phase** werden Aktivitätsdiagramme verwendet, um Geschäftsprozesse zu modellieren und zu analysieren. Sie zeigen die Reihenfolge der Prozesse und Tätigkeiten sowie mögliche Alternativabläufe.
  - In der **Entwurf/Design-Phase** bieten Aktivitätsdiagramme vielfältige Möglichkeiten zur Modellierung interner Systemprozesse.
- Darstellung des Kontroll- und Datenflusses in einer Aktivität; Fokus liegt auf den Aktionen eines Systems
- Aktivität untergliedert in elementare Aktionen/Aktivitätsschritte/Tasks, die logisch miteinander verbunden sind (inkl. Bedingungen, Schleifen und Parallelität)

UML Aktivitätsdiagramm – Beispiel Kreditantrag



Aktivitätsdiagramm – Beispiel Kreditantrag

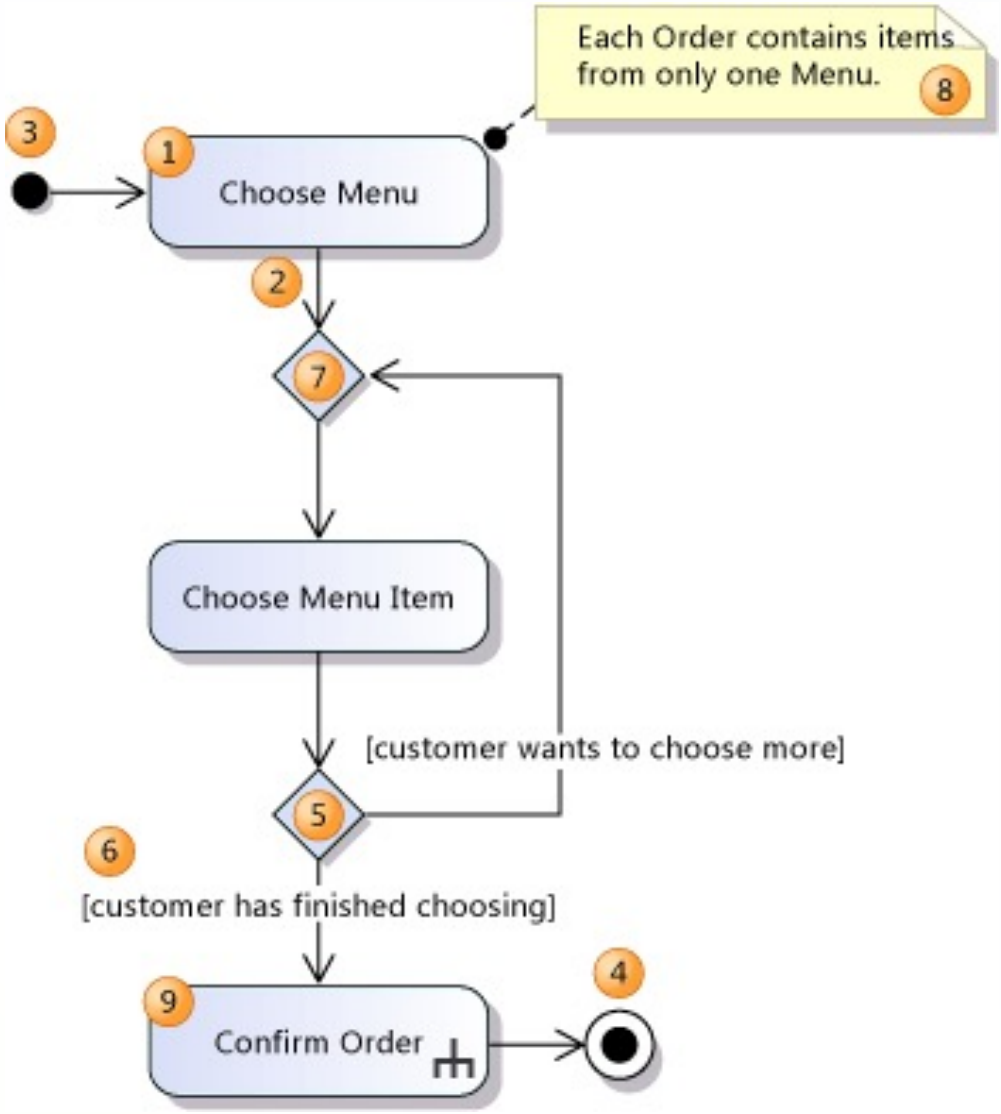
(Geschäfts)Prozess/ Unternehmensprozess	• ???
Prozessschritt/Aktivität/Teilprozess	• ???
Aktion/Task/Aufgabe/ Aktivitätsschritt/Arbeitsschritt	• ???

Aktivitätsdiagramm – Beispiel Kreditantrag

(Geschäfts)Prozess/ Unternehmensprozess	• „Kreditantrag und -vertrag erstellen“
Prozessschritt/Aktivität/Teilprozess	• „Kreditantrag stellen und prüfen“ • ...
Aktion/Task/Aufgabe/ Aktivitätsschritt/Arbeitsschritt	• „Kreditantrag erfassen“ • „Bonität prüfen“ • „Kreditantrag ablehnen oder anpassen“ • ...

Aktivitätsdiagramm – Notationselemente

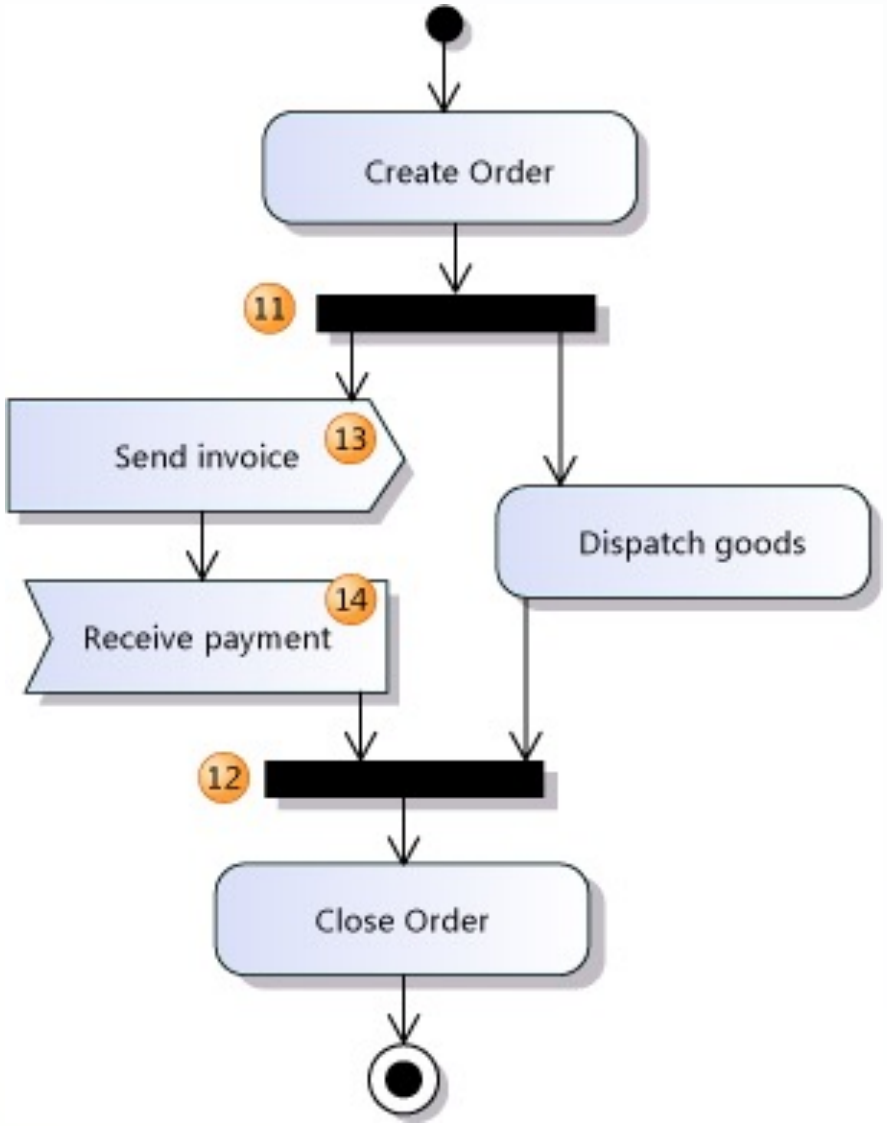
Nr.	Element
1	Aktion
2	Kontrollfluss
3	Startknoten
4	Endknoten
5	Entscheidungsknoten
6	Guard
7	Zusammenführungsknoten
8	Kommentar
9	Aktion → Aufruf einer Aktivität mit eigenem Modell



Quelle: MSDN, UML Activity Diagrams Reference

Aktivitätsdiagramm – Notationselemente

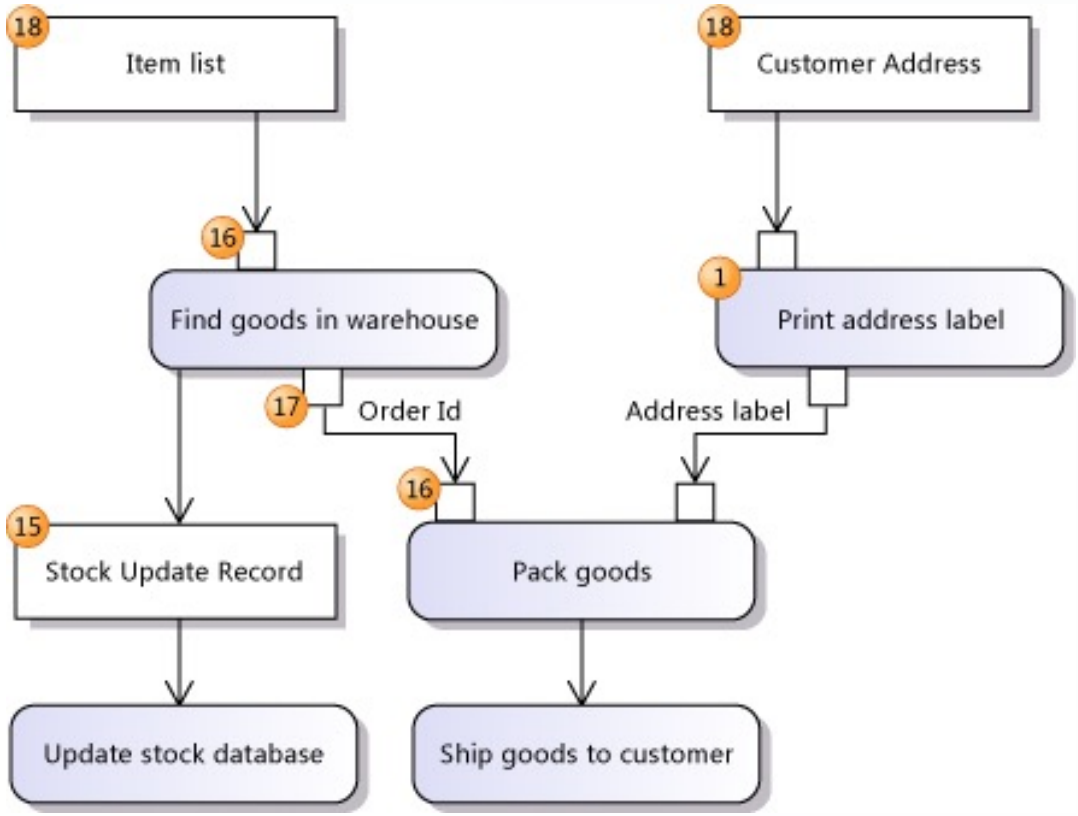
Nr.	Element
11	Fork-Node (Parallelisierung)
12	Join-Node (Synchronisation)
13	Aktion sendet Signal, z.B. an parallelen Thread
14	Aktion empfängt Signal



Quelle: MSDN, UML Activity Diagrams Reference

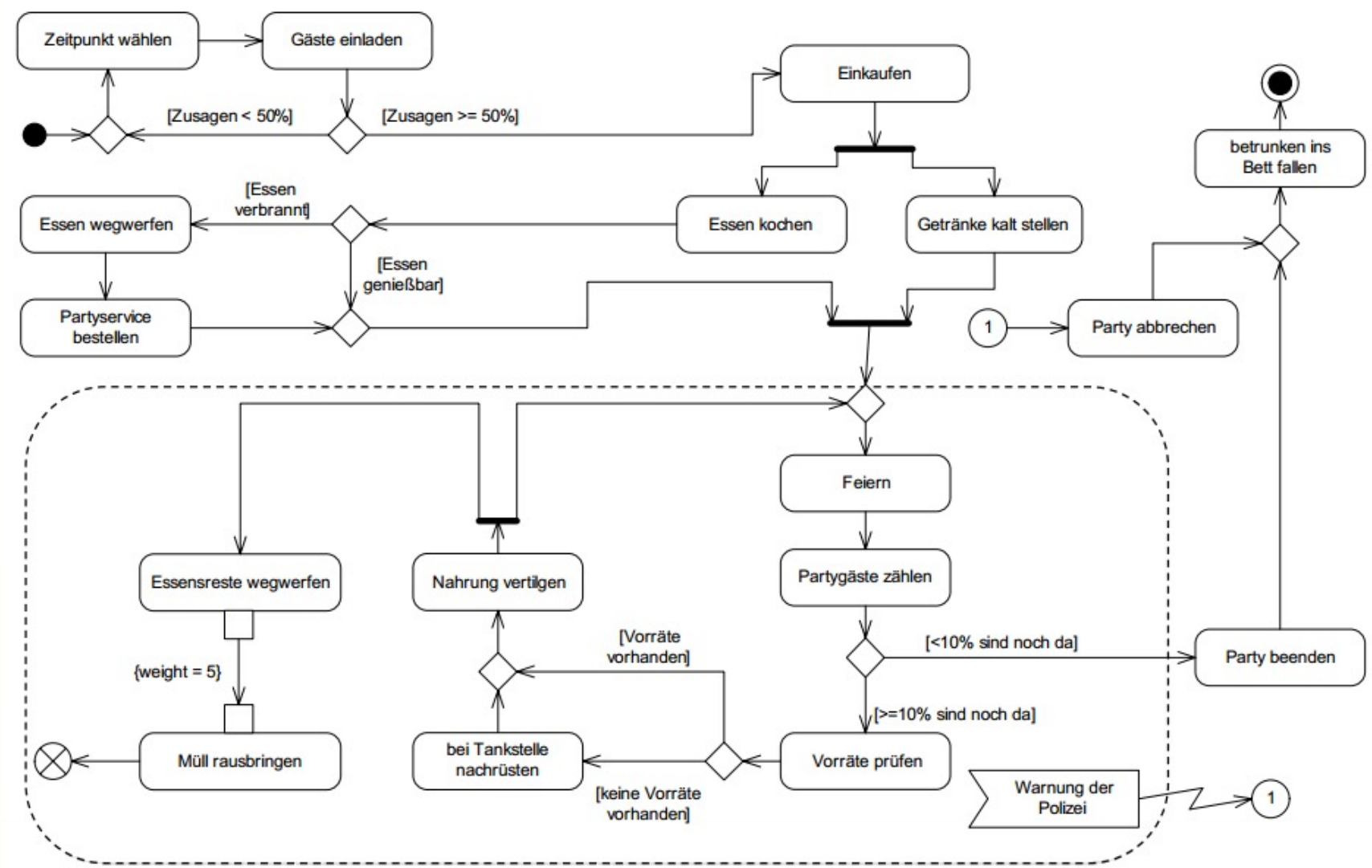
Aktivitätsdiagramm – Notationselemente

Nr.	Element
1	Aktion
15	Objekt
16	Input Pin (Input-Parameter) → Senke für Objektfluss
17	Output Pin → Quelle für Objektfluss
18	Objekt als Input-Parameter der Aktivität



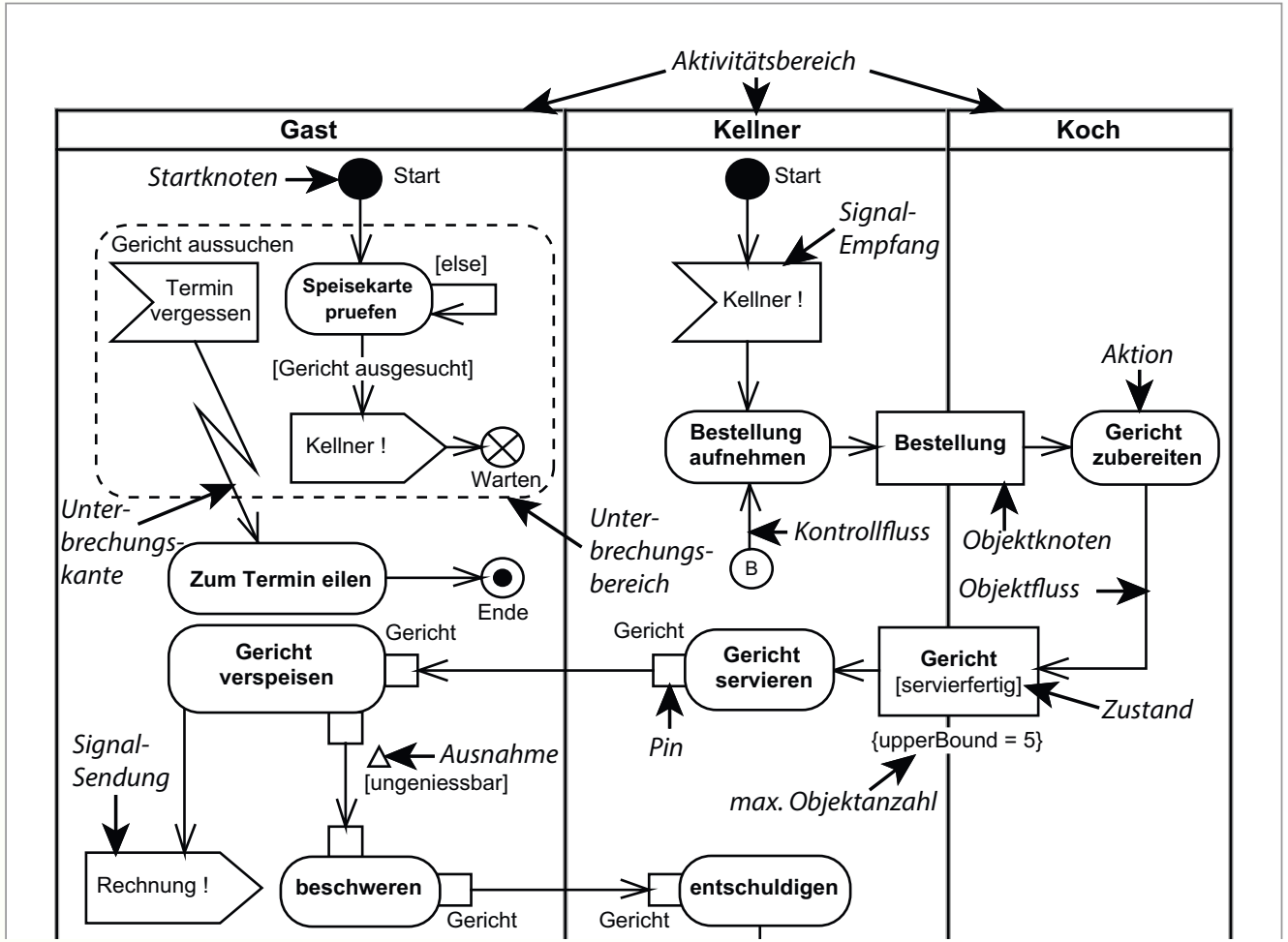
Quelle: MSDN, UML Activity Diagrams Reference

Aktivitätsdiagramm – Beispiel Party



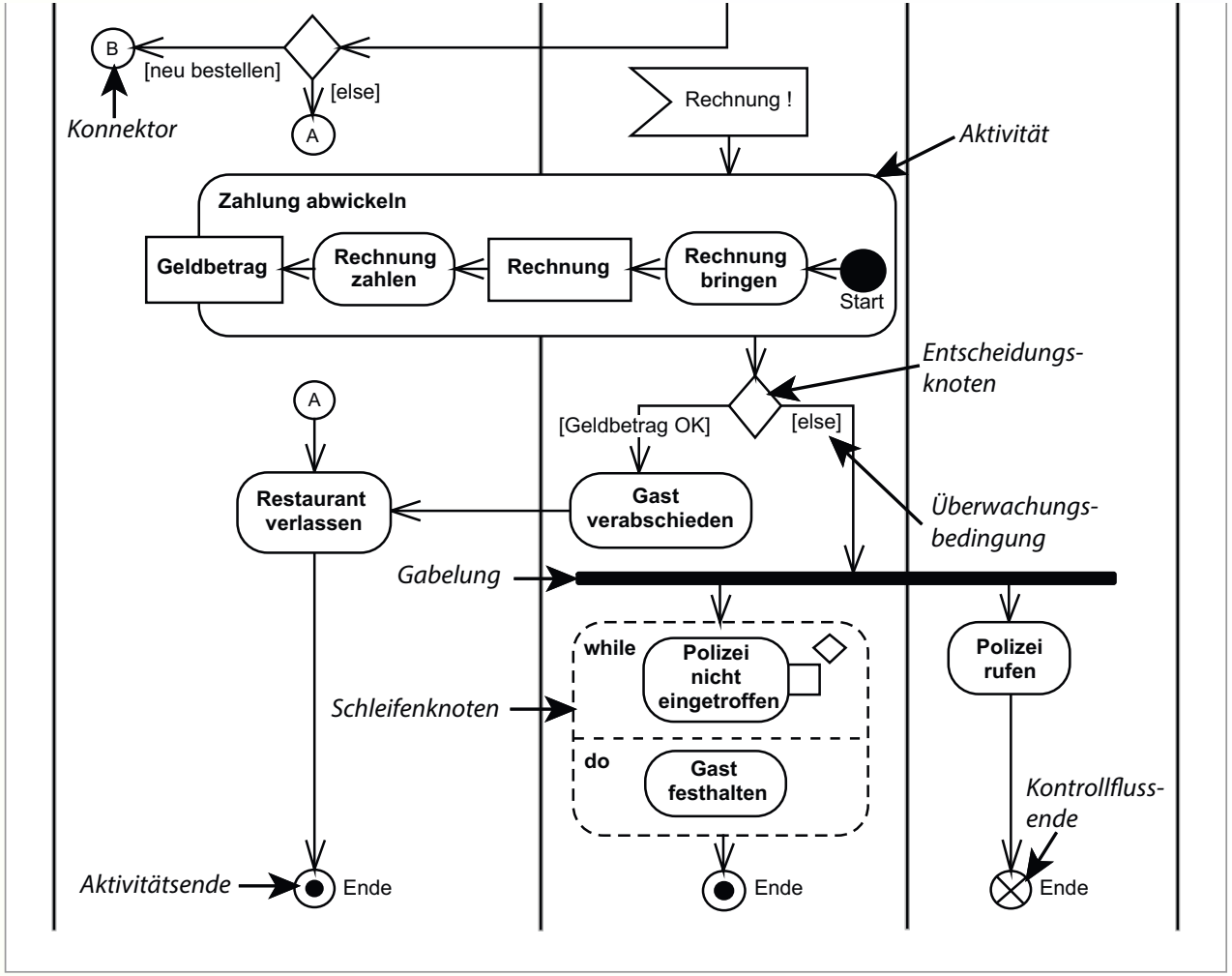
Quelle: Rupp, Queins, Zengler: UML 2 glasklar, 4. Aufl., Hanser, 2012.

Aktivitätsdiagramm – Beispiel Restaurant

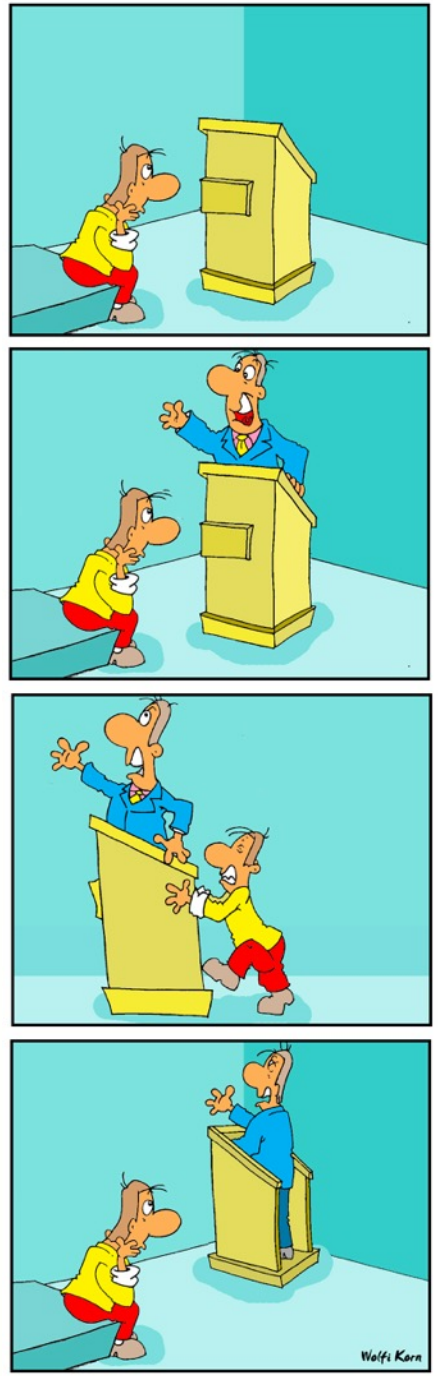


Quelle: Kecher, Salvanos (2015), S. 229

Aktivitätsdiagramm – Beispiel Restaurant



Quelle: Kecher, Salvanos (2015), S. 229



DUALE HOCHSCHULE SH

in Trägerschaft der Wirtschaftsakademie Schleswig-Holstein

★ Vielen Dank für die Aufmerksamkeit ★